Mejora en la asignación de recursos con Particle Swarm Optimization para el control de enfermedades transmitidas por mosquitos

Topiltzin Hernández Mares

1 de abril de 2022

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Estado del arte

Antes de hablar del problema y su solución, es necesario entrar en contexto. Por esto, en la primera sección de este capítulo se definirán conceptos básicos, tales como trampas para mosquitos, las cuales tienen un papel central en el problema. Enseguida, se hablará sobre términos de inteligencia artificial y métodos de optimización, los cuales ayudarán a entender mejor las problemática y su solución.

En la segunda parte de este capítulo se hablará de algunos trabajos recientes para poder entender el estado del arte en problemas de optimización. Sin embargo, por la poca investigación relacionada a la optimización de vigilancia de mosquitos, se analizarán trabajos similares enfocados en redes y monitoreo.

1.1. Conceptos básicos

1.1.1. Inteligencia artificial

En el libro Artificial Intelligence: A Modern Approach [1], se define la inteligencia artificial (IA) como máquinas con pensamiento parecido al humano y con acciones racionales.

Una definición más elaborada es la de [2]: IA es un término utilizado para etiquetar a computadoras que imitan funciones humanas cognitivas, tales como aprendizaje y resolución de problemas. De igual manera, se llama inteligencia artificial a algoritmos que trabajan con la misma complejidad que expertos humanos.

1.1.2. Optimización

Puede ser definida como el proceso de encontrar la mejor solución posible entre todas las disponibles. Por lo tanto, la tarea de optimización es modelar algún problema en términos de alguna función de evaluación y emplear algún algoritmo de búsqueda para minimizar (o maximizar) esa función objetivo. Sin embargo, no es posible asegurar que el resultado óptimo será encontrado, ya

que hay problemas demasiado grandes que es imposible encontrar la solución óptima [3].

1.1.3. Heurísticas

En el área computacional de optimización, se llaman heurísticas a los métodos usados para buscar soluciones de alta calidad, sin asegurar la óptima.

Una definición más elaborada la da [4], pues dice que una técnica heurística es un método que busca soluciones buenas (y casi óptimas) con un costo computacional bajo. Sin embargo, estos métodos no son capaces de indicar qué tan cerca de la optimización perfecta se encuentra la respuesta actual dada por una heurística.

1.1.4. Metaheurísticas

Se refiere a técnicas que coordinan, manipulan y guían el comportamiento y soluciones de métodos heurísticos de optimización. Tales métodos pueden ser desde procedimientos de alto nivel o pueden describir las alteraciones posibles para una heurística [5].

Existe una gran cantidad de metaheurísticas [4, 3], más adelante se hablará de algunos métodos, tal como algoritmos genéticos y optimización de enjambre de partículas.

1.1.5. Inteligencia de enjambre

La inteligencia de enjambre es un campo computacional que diseña y estudia algoritmos para buscar soluciones a problemas. Estos algoritmos están inspirados en el comportamiento complejo y, algunas veces, coordinado de enjambres de cualquier organismo natural [3].

Figura 1.1: Inspiración de algoritmos de inteligencia de enjambre.

1.1.6. Algoritmos evolutivos

Son un tipo de algoritmos que buscan soluciones a problemas, tomando inspiración de la selección natural y genética [6, 7]. Estos algoritmos pueden representar cromosomas como cadenas de texto, los cuales son las soluciones a

los problemas; los alfabetos representan genes; y más conceptos tomados de la biología.

1.1.7. Particle Swarm Optimization

Particle Swarm Optimization (PSO) fue inventada por Eberhart y Kennedy en 1995 [8], es un algoritmo estocástico inspirado en el comportamiento de una parvada de aves en busca de alimento. Cada partícula en el algoritmo, que representa una solución al problema, vuela en el espacio de búsqueda, actualizando su velocidad y su posición. La ecuación clásica para actualizar la velocidad de una partícula es 1.1 y para actualizar su posición es 1.2.

$$v_{ij} = v_{ij} + c_1 rand()(p_{ij} - x_{ij}) + c_2 rand()(p_{nj} - x_{ij}) \tag{1.1}$$

$$x_{ij} = x_{ij} + v_{ij} \tag{1.2}$$

Actualmente, el algoritmo PSO canónico (CPSO) o PSO con peso inercial (PSO-iw) es considerado como el PSO estándar [9, 10]. En esta forma de PSO, se introduce una nueva variable w, modificando 1.1 a 1.3. Esta nueva variable equilibra la búsqueda local y global de la solución.

$$v_{ij} = wv_{ij} + c_1 rand()(p_{ij} - x_{ij}) + c_2 rand()(p_{nj} - x_{ij}) \tag{1.3}$$

1.2. Trabajos relevantes en el área

1.2.1. Optimización de ubicación de una red de sensores inalámbricos en un taller inteligente

En 2018, Li et al [11] desarrollaron un nuevo algoritmo con el objetivo de optimizar la ubicación de nodos de una red de sensores inalámbricos (WSN) para minimizar el error de detección de objetos 3D en un taller de manufactura inteligente.

Los autores tomaron inspiración del comportamiento de enjambres de moscas, e implementaron en algoritmo de optimización de moscas (FOA). Sin embargo, este algoritmo presenta un comportamiento 2D, y su problema necesita un análisis en tres dimensiones. Por esto, introdujeron una nueva dimensión en la búsqueda de soluciones, resultando en el algoritmo es 3D-FOA. Como una alternativa, decidieron añadir una variable más a su algoritmo, un peso inercial variable, proveniente de PSO-iw, resultando en un nuevo algoritmo: 3D-AFOA.

Al tener dos algoritmos capaces de optimizar los nodos de la red MSN, realizaron diversos experimentos para comparar dichos métodos. Inicialmente, establecieron la posición fija de 3 nodos de la red y un solo nodo objetivo a detectar en un taller simulado. Establecieron una población de 50 y 500 como número máximo de generaciones en ambos algoritmos. En la figura 1.2 se muestran los resultados obtenidos.

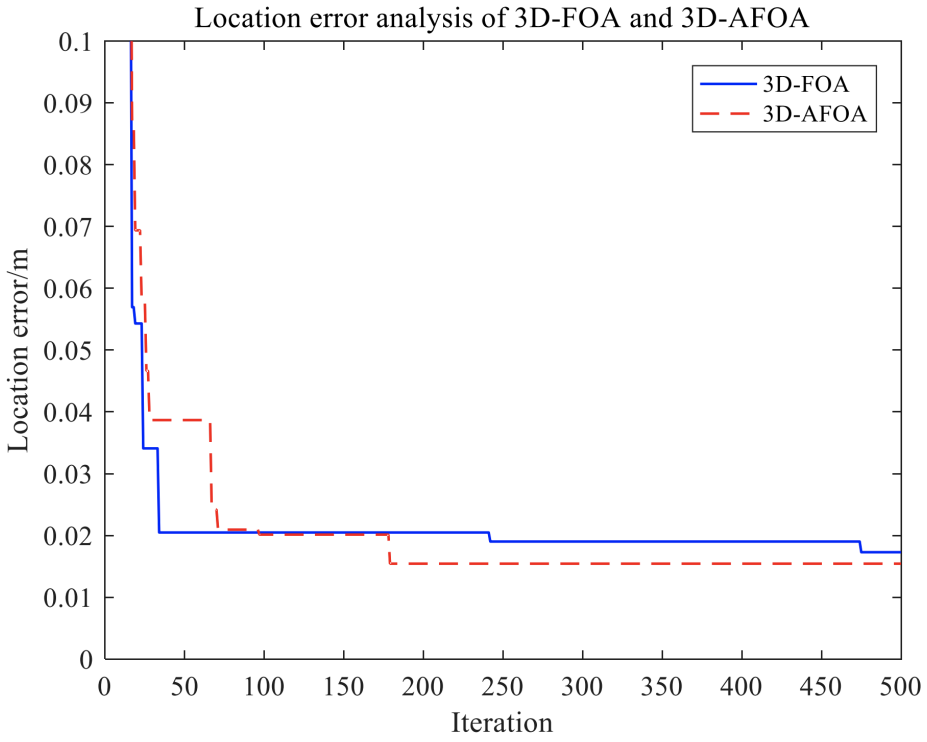

Figura 1.2: Resultados de error de detección de 3D-FOA y 3D-AFOA.

Como se puede ver en la figura 1.2, el algoritmo 3D-AFOA tiene un mejor desempeño, pues converge en una mejor solución en menos generaciones.

Cone l objetivo de verificar la efectividad del algoritmo, se probó con 3, 4 y 5 nodos fijos en la red para detectar nuevamente un recurso en el taller simulado. Los resultados se muestran en 1.3.

Analizando los resultados expuestos en 1.3, se puede apreciar que, conforme incrementa el número de nodos en el algoritmo, el tiempo de convergencia acelera. Sin embargo, el error de detección de los objetivos, aunque mejora con más nodos, no existe gran diferencia. Esto es debido a que solamente la ubicación de un solo nodo fue optimizada en los algoritmos.

Para finalizar, los investigadores concluyeron que 3D-AFOA es altamente aplicable al problema de optimización en la ubicación de nodos en una red WSN y no requiere de un alto presupuesto computacional para lograr buenas soluciones.

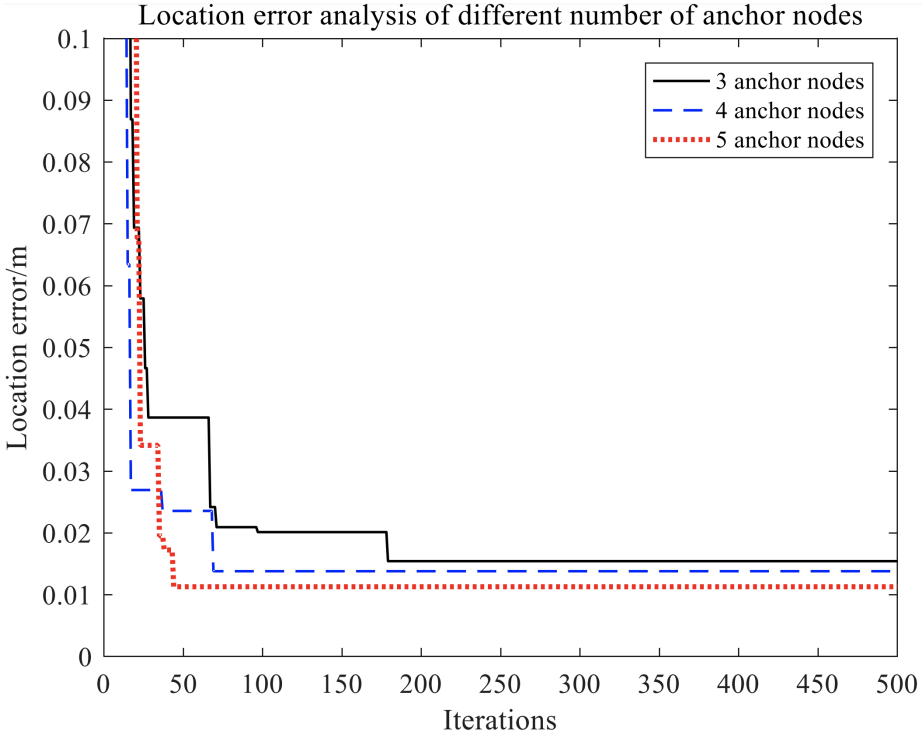

Figura 1.3: Resultados de error de detección con diferente cantidad de nodos.

1.2.2. Asignación óptima de unidades de generación distribuidas en sistemas de energía radiales

Hantash et al, en 2020 [12], publicaron un estudio en el que proponen una modificación al estándar PSO-iw para calcular las posiciones y tamaños de unidades generadoras distribuidas en una red radial de energía eléctrica.

Los autores partieron del estándar PSO-iw y, después de una investigación, encontraron que valores grandes en el peso inercial (w) facilitan la búsqueda global, mientras que valores pequeños de w favorecen la búsqueda local [9, 13]. Gracias a estos hallazgos, los autores proponen una estrategia no lineal para obtener valores de w variables con el tiempo. Esta estrategia está descrita en la Ecuación 1.4:

$$w = w_{max} e^{(((maxiter - iter)/maxiter) - 1)} - w_{min} \tag{1.4}$$

en donde w_{max} es el peso máximo, w_{min} es el peso mínimo, $iter$ es la iteración actual y $maxiter$ es el número máximo de iteraciones.

Finalmente, para validar sus resultados, el grupo de investigadores decidió comparar su estrategia propuesta contra un PSO convencional. Para comparar los resultados arrojados por ambos algoritmos, se usó la red de energía

estándar IEEE 34. Después de la ejecución de los experimentos, los investigadores obtuvieron pérdidas de energía 31.6 % menores con su propuesta de PSO a comparación del PSO convencional sin el valor de w variable. Además de esto, la propuesta consumió 62.2325 s para proveer una solución optima, cuando el PSO convencional consumió 62.2325 s. La comparación completa entre el PSO propuesto y el convencional está en la Figura 1.4.

	Proposed PSO	Convectional PSO
DG1_ size (MW)	1.6115	1.6722
DG1_ Vc (pu)	1.0155	1.0055
DG1_ location	24	10
Fitness losses	0.0401	0.0406
Average _elapsed time (s)	62.2325	78.6212

Figura 1.4: Comparación de resultados entre el PSO propuesto y el convencional.

1.2.3. Optimización de ubicación de trampas para mosquitos

El trabajo realizado por Sánchez y el Departamento de bioestadística y epidemiología de la universidad de Berkley [14] aún no ha sido publicado y sigue en progreso, sin embargo, es el estado del arte en el área de este trabajo. Sánchez et al están trabajando en un paquete llamado MGSurvE, el cual está diseñado para optimizar la ubicación de trampas de mosquito en entornos espacialmente heterogéneos.

En el estado actual del paquete cuenta con diversas funcionalidades, tales como: diferentes kernels de movimiento para mosquitos machos y hembras; kernels de atractivo para trampas personalizables; soporte para trampas inamovibles; rutinas de trazado de mapas integradas; integración de rutinas para optimización con GA; y mucho más, con más funciones esperadas en futuras actualizaciones.

Como se mencionó, las rutinas de optimización para la ubicación de las trampas para mosquito se implementa un algoritmo genético (GA), el cual es la variante estándar y es importado directamente del paquete DEAP [15].

Para el correcto funcionamiento del algoritmo de optimización, 3 grupos de características deben ser definidas: del entorno, de las trampas y del comportamiento de migración de mosquitos. En la figura 1.5 se describe cómo estos tres grupos de características forman el "Landscape", el cual es usado por el algoritmo genético.

En la fecha de realización de este capítulo, aunque no hay ningún experimento publicado oficialmente, la documentación del paquete sí cuenta con ejemplos y tutoriales de optimización en la siguiente URL `https://chipdelmal.github.`

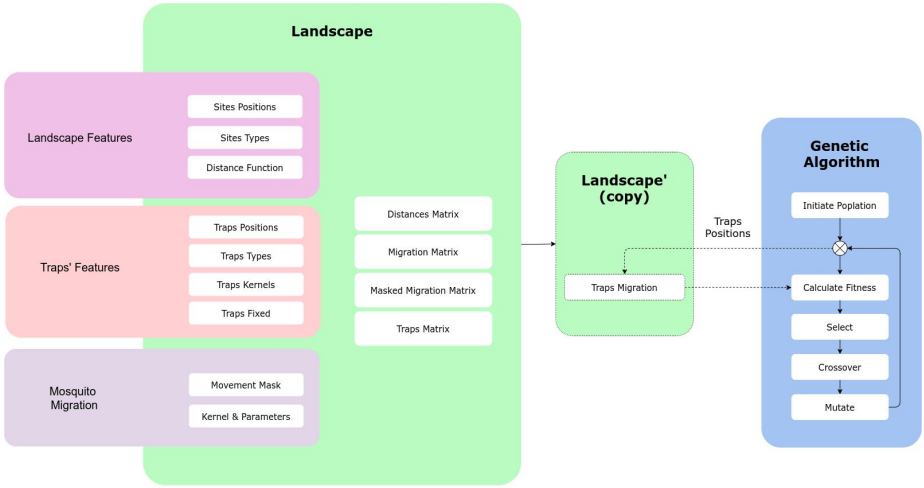

Figura 1.5: Interacción del Landscape con el algoritmo genético.

`io/MGSurvE/build/html/GA.html`. En los tutoriales, el número de población es de 10 por cada trampa a optimizar. Si se desean optimizar 4 trampas, se utiliza una población de entre 40 o 50.

Para finalizar, el paquete permite obtener el mejor cromosoma y trazar en un diagrama el mapa de las trampas y la migración de los mosquitos.

1.3. Comparación de algoritmos

Los trabajos mencionados en la sección anterior proponen diferentes algoritmos para solucionar problemas muy similares: encontrar la mejor ubicación de nodos en una red de nodos para optimizar algún resultado de una medición. Un aspecto en común de las soluciones propuestas por los diferentes investigadores es que todos hacen uso de algoritmos metaheurísticos iterativos para optimización. Otra similitud es que sus soluciones están basadas en inteligencia de enjambre y más específicamente en PSO.

Para resaltar las ventajas y desventajas de los diferentes tipos de algoritmos, en la tabla 1.6 se muestra una comparación detallada entre PSO y GA.

Además de los algoritmos propuestos por las investigaciones mencionadas en la sección anterior, se pueden encontrar una gran variedad usados en la actualidad para resolver problemas optimización. En un estudio realizado en 2021 por Piotrowski et al [16], encontraron que las principales familias de algoritmos usados para resolver problemas de optimización relacionados a la pandemia de COVID-19 fueron Swarm Intelligence (SI) y Evolutionary Algorithms (EA).

En 2017, un estudio similar realizado por Piotrowski et al [17], sometieron a 33 metaheurísticas propuestas entre 1960 y 2016 a 22 experimentos, cada una con diferentes presupuestos computacionales. Al finalizar los experimentos, en-

Algoritmo	Ventajas	Desventajas
Algoritmos Genéticos	• Resultados sobresalientes con presupuesto computacional alto. • Procesos más exploratorios.	• Require poblaciones grandes (al menos 50 individuos). • Malos resultados con presupuesto computacional bajo. • Lenta convergencia.
Particle Swarm Optimization	• Requiere poblaciones pequeñas (20-40 partículas). • Resultados sobresalientes con presupuesto computacional bajo. • Simple de implementar.	• Malos resultados con presupuesto computacional alto. • Riesgo de convergencia prematura.

Figura 1.6: Ventajas y desventajas de PSO y GA basado en [16, 17].

contraron que algoritmos basados en inteligencia de enjambre tienen un mejor desempeño cuando el presupuesto computacional es bajo. A diferencia de los algoritmos basados en inteligencia de enjambre, los basados en procesos evolutivos tienen un mejor desempeño cuando el presupuesto computacional es alto.

Aunque lo anterior es cierto, en un estudio de 2020 realizado por Piotrowski et al [18], se encontró que variantes modernas de PSO y SI sin peso inercial logran mejores resultados con una población de entre 70 y 500 partículas, contradiciendo las recomendaciones para lograr mejores resultados con variantes clásicas. Debido a esto, es de gran importancia conocer las condiciones adecuadas para cada variante de algoritmos basados en PSO. Si la cantidad adecuada de partículas para un problema en cuestión es desconocido, la recomendación es usar entre 70 y 100 partículas.

1.4. Conclusión

En la actualidad, existe una gran variedad de algoritmos de optimización los cuales tienen comportamientos diferentes para cada tipo de problema [17]. Por esto, es de suma importancia escoger correctamente el algoritmo que será empleado en alguna solución, para así obtener resultados realmente óptimos.

Como se ve en la tabla 1.6 y se discutió en la sección anterior, los algoritmos genéticos tienen un mejor comportamiento y desempeño cuando el presupuesto computacional y la población es grande (más de 50), por lo que el trabajo realizado en MGSurvE [14] no está optimizando correctamente las ubicaciones de las trampas para mosquitos, pues usan poblaciones de entre 40 y 50, además de un presupuesto computacional bajo. Esta situación da paso a una oportunidad de mejora, para implementar un algoritmo que se ajuste en mejor manera a la situación de MGSurvE, con la posibilidad de mejorar la optimización de la ubicación de las trampas para mosquitos.

Bibliografía

[1] S. Russell and P. Norving, "Artificial Intelligence: A modern approach", 3rd ed. Prentice Hall, New Jersey: Pearson Education, 2009.

[2] J. M. Spector and S. Ma, "Inquiry and critical thinking skills for the next generation: From Artificial Intelligence back to human intelligence," Smart Learning Environments, vol. 6, no. 1, Sep. 2019.

[3] E. K. Burke and G. Kendall, "Search methodologies: Introductory tutorials in optimization and decision support techniques". New York, New York: Springer, 2014.

[4] V. J. Rayward-Smith, "Modern Heuristic Search Methods". Chichester, New York: Wiley, 1996.

[5] F. Glover and M. Laguna, "Tabu Search". Boston, Massachusets: Kluwer Academic Publishers, 1997.

[6] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen, "Evolutionary algorithms for solving multi-objective problems," Genetic and Evolutionary Computation Series, 2007.

[7] A. S. Fraser, "Simulation of genetic systems by Automatic Digital Computers II. effects of linkage on rates of advance under selection," Australian Journal of Biological Sciences, vol. 10, no. 4, p. 492, 1957.

[8] J. Kennedy and R. Eberhart, "Particle swarm optimization," Proceedings of ICNN'95 - International Conference on Neural Networks, 1995.

[9] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360), 1998.

[10] S. Cheg, H. Lu, X. Lei, and Y. Shi, "A quarter century of particle swarm optimization,", Complex and Intelligent Systems, vol. 4, no. 3, pp. 227 239, 2018.

[11] S. Li, C. Zhang, and J. Qu, "Location optimization of wireless sensor network in Intelligent Workshop based on the three-dimensional adaptive fruit

fly optimization algorithm," International Journal of Online Engineering (iJOE), vol. 14, no. 11, p. 202, Nov. 2018.

[12] N. Hantash, T. Khatib, and M. Khammash, "An improved particle swarm optimization algorithm for optimal allocation of distributed generation units in Radial Power Systems," Applied Computational Intelligence and Soft Computing, vol. 2020, pp. 1-8.

[13] H. Liang and F. H. Kang, "Adaptive mutation particle swarm algorithm with dynamic nonlinear changed inertia weight," Optik-International Journal for Light and Electron Optics, vol. 127, no. 19, pp. 8036-8042, 2016.

[14] H. M. Sanchez, "MGSurvE's documentation," MGSurvE's documentation! - MGSurvE documentation, 2021. [Online]. Available: https://chipdelmal.github.io/MGSurvE/build/html/index.html. [Accessed: 31-Mar-2022].

[15] DEAP Project, "DEAP documentation," DEAP documentation - DEAP 1.3.1 documentation, 2022. [Online]. Available: https://deap.readthedocs.io/en/master/. [Accessed: 01-Apr-2022].

[16] A. P. Piotrowski and A. E. Piotrowska, "Differential Evolution and particle swarm optimization against covid-19," Artificial Intelligence Review, Jul. 2021.

[17] A. P. Piotrowski, M. J. Napiorkowski, J. J. Napiorkowski, and P. M. Rowinski, "Swarm intelligence and Evolutionary Algorithms: Performance versus speed," Information Sciences, vol. 384, pp. 34-85, Apr. 2017.

[18] A. P. Piotrowski, J. J. Napiorkowski, and A. E. Piotrowska, "Population size in particle swarm optimization," Swarm and Evolutionary Computation, vol. 58, May 2020.