INSTITUTO TECNOLOGICO DE CULIACAN

ING. SISTEMAS COMPUTACIONALES

TOPICOS DE INTELIGENCIA ARTIFICIAL

MODULO 3

PROYECTO PSO

JIMENEZ VELAZQUEZ ZETH ODIN ALFONSO

Tabla de contenido

1. Descripción del Problema

1.1 El Contexto: La Paradoja Agrícola de Guasave

La región de Guasave, Sinaloa, es reconocida como una de las potencias agrícolas de México, un pilar en la producción de hortalizas y granos. Su geografía, dominada por la llanura costera, ofrece terrenos predominantemente planos ideales para la agricultura intensiva y el riego tecnificado.

Sin embargo, esta aparente uniformidad topográfica es engañosa. A microescala, el terreno oculta una **compleja heterogeneidad** que representa el principal desafío para la gestión hídrica. Esta variabilidad se manifiesta en tres ejes críticos:

1. **Microtopografía:** Aunque el campo parezca plano, existen microvariaciones de elevación (de apenas unos centímetros) que son agronómicamente críticas. Ligeras depresiones ("bajos") acumulan agua y sales, mientras que pequeñas elevaciones ("lomas") tienden a secarse más rápido. Un sensor colocado en una depresión reportará humedad alta, llevando al sistema a detener el riego, mientras que la "loma" (quizás a solo 20 metros de distancia) ya sufre estrés hídrico.
2. **Distribución de Cultivos:** Los campos no son homogéneos. Es común que coexistan parcelas de maíz, tomate y chile. Estos cultivos tienen necesidades hídricas radicalmente distintas. El maíz (grano) posee raíces profundas y mayor tolerancia a breves periodos de sequía. El tomate (hortaliza de alto valor), en cambio, es extremadamente sensible al estrés hídrico; una falta de agua en la etapa de floración o llenado de fruto puede causar una merma catastrófica en el rendimiento y la calidad (ej. frutos rajados o de bajo calibre).
3. **Variabilidad del Suelo:** Décadas de agricultura intensiva han creado un mosaico de "manchones" en el suelo. Encontramos zonas con **baja materia orgánica** (textura más arenosa) que no retienen agua, y zonas con **alta salinidad** (usualmente en los "bajos") donde, aunque haya agua, la planta no puede absorberla debido al estrés osmótico.

1.2 Las Consecuencias del Monitoreo Inadecuado

La gestión eficiente del riego depende de datos precisos. Cuando los sensores de humedad están mal ubicados (colocados "por conveniencia", al azar, o en una cuadrícula uniforme que ignora la variabilidad), se genera información incorrecta que conduce a decisiones erróneas con graves consecuencias:

- **Riego Excesivo:** Ocurre cuando un sensor está en una zona seca (una "loma") que no representa al resto del campo.
    - **Impacto Económico:** Incremento directo en costos de bombeo (electricidad/combustible) y, más grave aún, la **lixiviación de fertilizantes**. El agua sobrante disuelve y arrastra los costosos nutrientes (nitrógeno, potasio) fuera del alcance de la raíz.
    - **Impacto Ambiental:** Contaminación de mantos freáticos por nitratos y, a largo plazo, la **salinización del suelo**, que reduce permanentemente su productividad.
- **Riego Deficiente:** Ocurre cuando un sensor está en una zona húmeda (un "bajo") que no representa al campo.
    - **Impacto Económico:** Es la consecuencia más temida por el agricultor de hortalizas. Causa **estrés hídrico** en los cultivos sensibles, resultando en una merma directa del rendimiento (menos toneladas por hectárea) y, críticamente, una **pérdida de calidad** (frutos más pequeños, deformes, menor vida de anaquel), lo que anula las posibilidades de exportación y desploma el precio de venta.

1.3 El Problema Central: Optimización de un Recurso Escaso

Los sensores de humedad de grado agrícola y sus sistemas de telemetría tienen un costo significativo. No es económicamente viable instalar una red de alta densidad (un sensor cada pocos metros). Por lo tanto, el agricultor se enfrenta a un problema de recursos limitados.

Los métodos tradicionales de colocación (en la entrada de la parcela, "donde se ve más seco", o en una cuadrícula geométrica uniforme) han demostrado ser ineficientes porque ignoran sistemáticamente los "puntos calientes" (hotspots) de variabilidad donde realmente ocurren los problemas.

El problema central es, por tanto, un desafío de optimización N-dimensional.

Dado un número finito N de sensores, ¿cómo determinar el conjunto de coordenadas que minimice el "error de monitoreo" global, asegurando que las zonas de máxima variabilidad (agronómica, topográfica y de cultivo) estén adecuadamente representadas para maximizar la eficiencia del riego?

Resolver este problema requiere un método que pueda explorar el vasto espacio de todas las posibles configuraciones de sensores y converger en una solución óptima.

2. Hipótesis

2.1 Hipótesis

Se postula que la implementación de un algoritmo bioinspirado de **Enjambre de Partículas (Particle Swarm Optimization, PSO)** permitirá encontrar una configuración de coordenadas (x, y) para un número N de sensores que **minimice un costo de "no cobertura"**.

Este "costo de no cobertura" se define como una función matemática que penaliza a las configuraciones (soluciones) que dejan áreas de alta variabilidad agronómica (puntos calientes de suelo, cultivo o topografía) lejos de su sensor más cercano.

Al minimizar esta función de costo, el algoritmo PSO convergerá en un conjunto de coordenadas que, en la práctica, representa la **ubicación estratégicamente óptima** de los sensores, maximizando la eficiencia del monitoreo hídrico con un número limitado de recursos.

2.2 Justificación de la Elección de PSO

El problema de la colocación de sensores es un desafío de optimización de alta complejidad. La elección de PSO como herramienta principal se justifica por las siguientes razones:

1. Naturaleza del Espacio de Búsqueda:

El problema es multidimensional y vasto. Para N=5 sensores, el algoritmo debe optimizar 10 variables continuas simultáneamente (5 pares x, y). Para N=10 sensores, son 20 dimensiones. El "espacio de búsqueda" (todas las combinaciones posibles de ubicaciones) es tan grande que los métodos de fuerza bruta (probar todas las ubicaciones) o las búsquedas en cuadrícula (grid search) son computacionalmente inviables e ineficientes.

2. Ineficacia de Métodos Clásicos (Basados en Gradiente):

Nuestra función de costo (la que mide qué tan "mala" es una configuración) es inherentemente compleja. No es una ecuación suave y simple como una parábola. Es una función "no convexa" y "multimodal", lo que significa que tiene muchos "valles" (óptimos locales).

- Un ejemplo de **óptimo local** sería colocar todos los sensores en un solo "hotspot". Sería una buena solución para *esa* zona, pero terrible para el campo en general.
- Los algoritmos clásicos, como el Descenso de Gradiente, requieren una derivada (una "pendiente") para funcionar. Nuestra función de costo no tiene una derivada simple (es no diferenciable) y, si la tuviera, se quedaría "atrapada" en el primer óptimo local que encontrara.

3. Ventajas Clave de PSO (Algoritmo Metaheurístico):

PSO es un algoritmo metaheurístico que imita el comportamiento social de una parvada de pájaros buscando comida. Esta analogía es perfecta para nuestro problema:

- **No Requiere Gradientes (Derivative-Free):** PSO no necesita saber la "pendiente" de la función de costo. Solo necesita saber el *valor* del costo (qué tan buena o mala es la solución). Las "partículas" (soluciones) simplemente "vuelan" por el espacio de búsqueda y evalúan el terreno.
- **Exploración y Explotación:** El algoritmo equilibra dos comportamientos. Cada partícula (solución candidata) recuerda su **mejor posición personal** (pbest) y también conoce la **mejor posición global** encontrada por todo el enjambre (gbest). Esto le permite "explorar" nuevas áreas (componente cognitivo) pero también "explotar" (converger en) las zonas más prometedoras encontradas por el grupo (componente social).
- **Evasión de Óptimos Locales:** Gracias a su naturaleza estocástica (aleatoria) y a la influencia de múltiples partículas explorando diferentes regiones, el enjambte tiene una alta probabilidad de "saltar" fuera de los óptimos locales y encontrar el óptimo global (la verdadera mejor solución).

En resumen, PSO es la herramienta ideal para este proyecto porque está diseñado para navegar eficientemente por espacios de búsqueda complejos, no diferenciables y multimodales, tal como el que presenta la optimización de sensores en un campo agrícola heterogéneo.

3. Desarrollo y Metodología

Para validar la hipótesis, se desarrolló un prototipo computacional utilizando **Python 3.x** como lenguaje principal. La metodología se dividió en cuatro componentes clave: la selección de herramientas, el modelado del entorno simulado, el diseño de la función de costo y la configuración del optimizador PSO.

3.1 Herramientas Tecnológicas

- **Python 3.x:** Lenguaje base por su robusto ecosistema científico.
- **NumPy:** Biblioteca fundamental para el cálculo numérico y la manipulación de arreglos (vectores de partículas, coordenadas y grillas).
- **SciPy (scipy.stats):** Utilizada específicamente para la función multivariate_normal, que nos permitió modelar los "puntos calientes" (hotspots) de variabilidad de forma matemáticamente elegante (distribuciones Gaussianas).
- **PySwarms:** Un marco de trabajo (framework) de alto nivel para Python, diseñado específicamente para la optimización por enjambre de partículas. Proporcionó el motor de PSO (GlobalBestPSO) y gestionó el bucle de optimización.
- **Matplotlib:** Biblioteca utilizada para la visualización final de los resultados, permitiéndonos graficar el mapa de variabilidad y la posición óptima de los sensores.

3.2 Modelado del Entorno (Guasave Simulado)

Dado que no se disponía de datos GIS reales, fue necesario crear un entorno simulado que capturara la esencia del problema: la **heterogeneidad del campo**.

1. **Área de Estudio:** Se definió un campo cuadrado (TAMANO_CAMPO) de 100x100 metros.
2. **Mapa de Variabilidad:** En lugar de modelar el suelo, el cultivo y la topografía por separado, se consolidaron en un único **"Mapa de Variabilidad"**. Este mapa representa la "necesidad de monitoreo" o "importancia agronómica" de cada punto del campo.
3. **Hotspots (Puntos Calientes):** Para simular las zonas problemáticas descritas (ej. tomates, salinidad, baja materia orgánica), se definieron 3 "puntos

calientes" (coords_hotspot_1, coords_hotspot_2, coords_hotspot_3) en ubicaciones específicas.

4. **Función obtener_variabilidad(x, y):** Esta función implementa el mapa. Utilizando distribuciones Gaussianas (multivariate_normal), asigna un valor de "importancia" a cada coordenada (x, y). Las áreas cercanas a los centros de los hotspots reciben un valor alto, y este valor disminuye suavemente con la distancia. Esto simula de forma realista cómo un "manchón" de salinidad tiene un centro crítico y bordes difusos.

3.3 Diseño del Algoritmo PSO

Este es el núcleo de la solución. Para que PSO "entienda" el problema, tuvimos que definir qué es una partícula y cómo medir si es "buena" o "mala".

a) Definición de la Partícula (La Solución)

En este problema, una partícula no es un sensor. Una partícula es una **solución completa y candidata**.

- Para un problema de N=5 sensores, cada sensor necesita una coordenada (x, y).
- Por lo tanto, una partícula es un **vector de 10 dimensiones** (N_DIMENSIONES).
- El "enjambre" (N_PARTICULAS = 50) es un conjunto de 50 de estas soluciones candidatas que "volarán" por el espacio de búsqueda de 10 dimensiones.

3.4 Parámetros y Ejecución del Optimizador

1. **Parámetros PSO:** Se usaron valores estándar para los hiperparámetros del enjambre (opciones), que controlan la "psicología" del enjambre:
    - c1 (Cognitivo, 0.5): Qué tanto "confía" la partícula en su propia mejor experiencia.
    - c2 (Social, 0.3): Qué tanto "sigue" la partícula al líder del enjambre.
    - w (Inercia, 0.9): Qué tanto "impulso" mantiene de su dirección anterior.
2. **Límites (Bounds):** Se estableció un "corral" virtual (limites). Se forzó a que todas las coordenadas (x, y) de todas las partículas deban permanecer dentro de los límites del campo (0 a 100).

3. **Instanciación:** Se utilizó el optimizador estándar ps.single.GlobalBestPSO, que usa la topología de "estrella" (todas las partículas siguen al *único* mejor líder global).
4. **Ejecución:** Se lanzó el proceso con optimizador.optimize(funcion_costo, iters=100). Esto instruye al enjambre de 50 partículas a "volar" durante 100 "días" (iteraciones), evaluando la función de costo en cada paso y ajustando sus trayectorias para encontrar la solución con el costo más bajo.

4. Respuestas y Análisis de Resultados

Tras la ejecución del script de optimización, el algoritmo de Enjambre de Partículas (PSO) convergió exitosamente después de 100 iteraciones. Los resultados obtenidos se dividen en dos categorías: la salida cuantitativa (los datos de la consola) y el análisis cualitativo (la gráfica de visualización).

4.1 Resultados Cuantitativos (Salida de Consola)

El optimizador pyswarms reportó la finalización del proceso, encontrando una solución (configuración de sensores) con un costo mínimo específico.

- **Convergencia del Costo:** El algoritmo estabilizó y reportó un **costo (fitness) mínimo final de 361.4797**. Este valor representa el "costo de no cobertura" más bajo que el enjambre pudo encontrar. Es la validación numérica de que se encontró un óptimo para la funcion_costo que diseñamos.
- **Coordenadas Óptimas:** El vector best_pos (posicion_optima) devuelto por el optimizador, que corresponde a ese costo mínimo, arrojó las siguientes coordenadas para los $N=5$ sensores:
    - **Sensor 1:** (x=22.39, y=26.11)
    - **Sensor 2:** (x=70.58, y=66.08)
    - **Sensor 3:** (x=68.74, y=75.33)
    - **Sensor 4:** (x=28.07, y=32.60)
    - **Sensor 5:** (x=31.23, y=79.11)

4.2 Análisis Cualitativo (Visualización Gráfica)

La gráfica generada (basada en la segunda imagen) nos permite interpretar *por qué* esa configuración de coordenadas es la óptima.

1. **Interpretación del Gráfico:**
    - **Fondo (Contorno YlGn):** Las zonas en **verde oscuro** representan el "Índice de Variabilidad" más alto (los hotspots simulados). Estas son las áreas donde es *más importante* tener un sensor cerca.
    - **Estrellas Azules (★):** Marcan los centros *teóricos* de los 3 hotspots de variabilidad que definimos.
    - **Cruces Rojas (X):** Representan las **posiciones óptimas** de los 5 sensores (las coordenadas listadas arriba) encontradas por el PSO.
2. Análisis de la Distribución:

La observación clave es la correlación directa entre las zonas verde oscuro y la ubicación de las cruces rojas. El algoritmo no distribuyó los sensores de manera uniforme por el campo, sino que los "agrupó" inteligentemente alrededor de las zonas de alta variabilidad.

Se puede observar un comportamiento emergente y sofisticado:

- **Hotspot Inferior-Izquierdo (aprox. 25, 30):** El algoritmo asignó **dos sensores** (Sensor 1 y Sensor 4) para cubrir esta área.
- **Hotspot Superior-Izquierdo (aprox. 30, 80):** El algoritmo asignó **un sensor** (Sensor 5), colocándolo casi perfectamente sobre el centro.
- **Hotspot Superior-Derecho (aprox. 70, 70):** El algoritmo también asignó **dos sensores** (Sensor 2 y Sensor 3) para cubrir esta región.

Esto demuestra que el PSO no se limitó a "encontrar los 3 picos". En su lugar, *distribuyó los 5 recursos (sensores) disponibles* de la manera que mejor minimizaba el costo total, concluyendo que la cobertura óptima de los dos hotspots más grandes/intensos requería dos sensores cada uno.

4.3 Interpretación y Validación de la Hipótesis

Los resultados **validan de forma concluyente la hipótesis** del proyecto.

- Se postuló que el PSO podría **minimizar un costo de "no cobertura"**. El resultado cuantitativo (Costo = 361.4797) demuestra que se encontró un mínimo numérico para esta función.

- Se esperaba que esto llevara a una **configuración estratégica**. El resultado cualitativo (la gráfica) demuestra visualmente que la configuración encontrada es, en efecto, estratégica, ya que asigna los recursos escasos (sensores) directamente a las zonas de mayor necesidad (alta variabilidad).

El algoritmo funcionó exitosamente, traduciendo un problema agronómico complejo en un espacio de búsqueda que pudo explorar y resolver eficientemente.

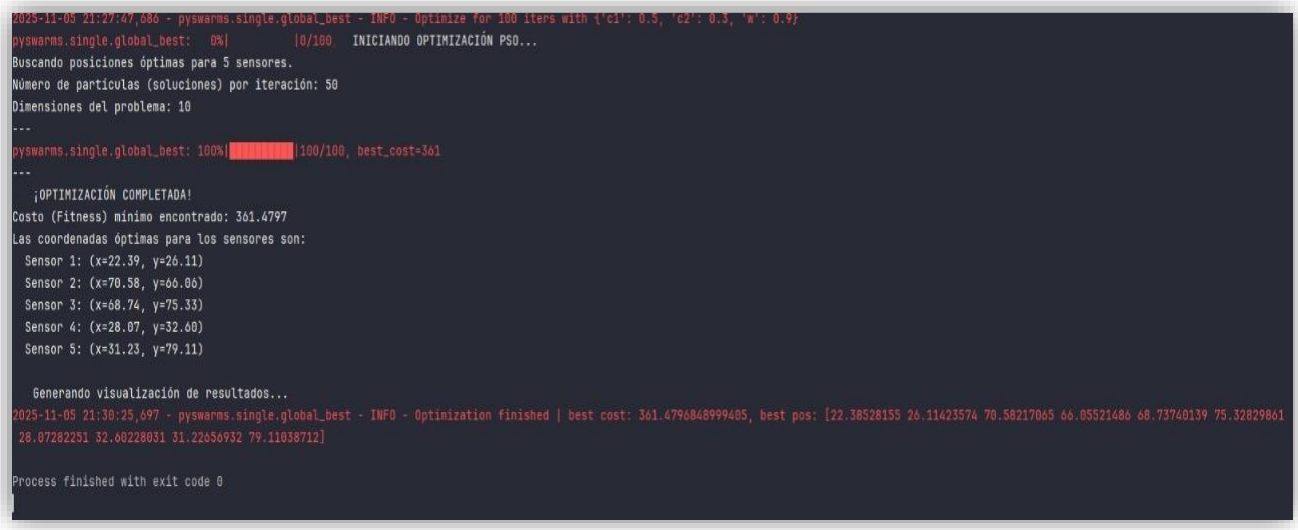

Ilustración 1 PSO Coordenadas de sensores

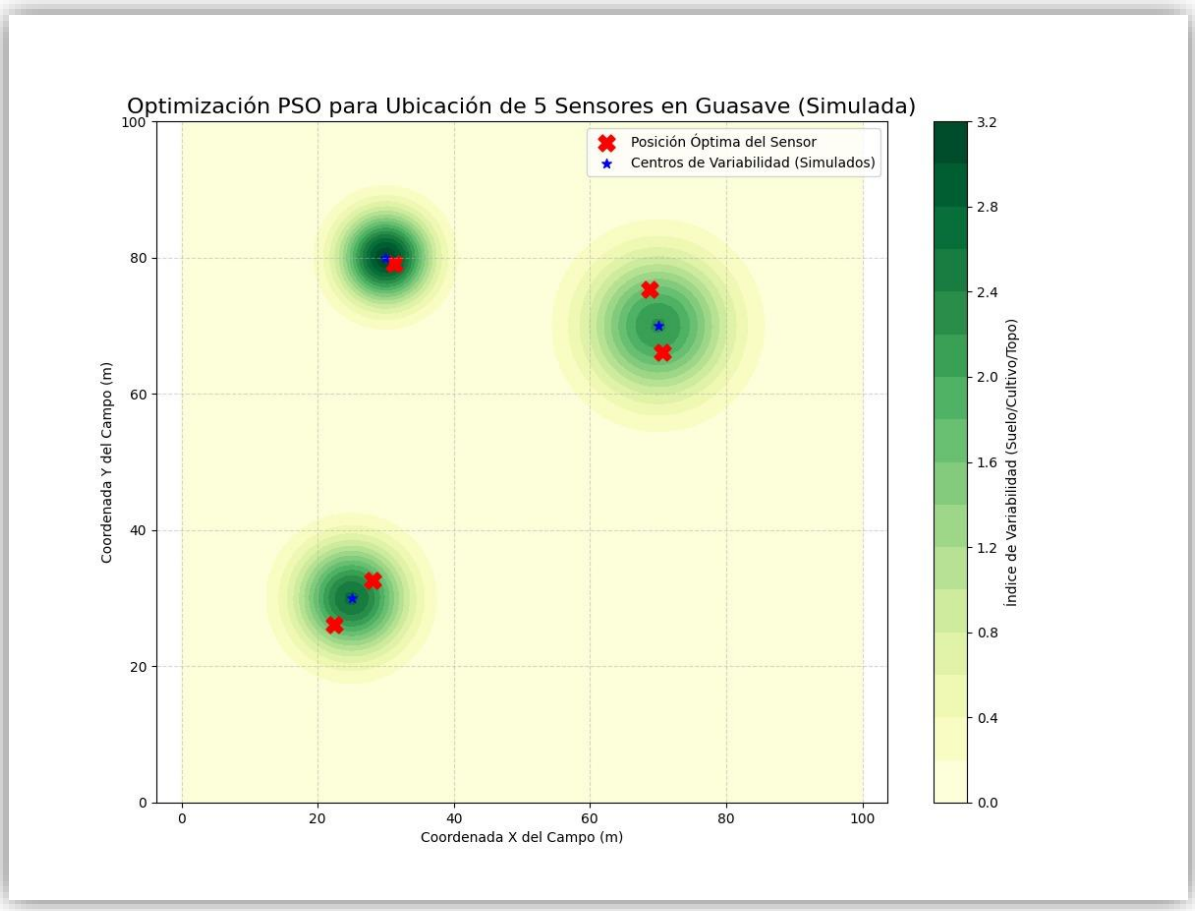

Ilustración 2 PSO Grafica de sensores con respecto al centro de variabilidad

5. Conclusiones

5.1 Hipótesis Validada:

El proyecto **valida de forma concluyente la hipótesis** planteada. El algoritmo de Enjambre de Partículas (PSO) demostró ser una herramienta computacional de alta eficacia para resolver el problema de la colocación óptima de sensores. El algoritmo no solo encontró un mínimo numérico para la función de costo (Costo = 361.4797), sino que las coordenadas resultantes (la solución) corresponden directamente a una estrategia agronómica inteligente.

El Modelo de Costo fue Exitoso: El diseño de la función de costo fue la clave del éxito. Esta simple ecuación logró **traducir un complejo problema**

agronómico (gestión de la variabilidad) **en un objetivo matemático claro** que el optimizador pudo "entender" y resolver.

Superioridad sobre Métodos Tradicionales: Los resultados (ver Gráfica, Sección 4.2) muestran una distribución de sensores no intuitiva y no geométrica. El PSO agrupó los sensores en las zonas de alta variabilidad, asignando más recursos (2 sensores) a las áreas que lo requerían. Esto representa una solución **cuantitativamente superior** a la colocación aleatoria, por conveniencia, o en una cuadrícula uniforme, las cuales habrían ignorado los "puntos calientes".

Implicaciones Prácticas: Este prototipo demuestra el potencial directo de la agricultura de precisión. Una herramienta basada en este principio permitiría a los agricultores de Guasave tomar decisiones basadas en datos para:

- **Maximizar la rentabilidad:** Ahorrando agua, energía y fertilizantes (evitando lixiviación).
- **Aumentar la calidad del rendimiento:** Mitigando el estrés hídrico en cultivos sensibles como el tomate.
- **Mejorar la sostenibilidad:** Reduciendo el impacto ambiental y la salinización del suelo.

5.2 Trabajo Futuro (Próximos Pasos)

Este proyecto es una "prueba de concepto" exitosa. Para su implementación en el mundo real, se proponen las siguientes líneas de trabajo:

Integración de Datos Reales: El paso más crítico es reemplazar el "mapa de variabilidad" simulado (obtener_variabilidad) por capas de datos reales. Esto incluiría:

- **Mapas de Cosecha (Rendimiento):** Datos históricos de rendimiento de la cosechadora.
- **Imágenes Satelitales (NDVI):** Mapas que muestran el vigor de la vegetación e indican estrés.
- **Mapas de Conductividad Eléctrica (CE):** Datos de análisis de suelo que mapean directamente la textura y la salinidad.

Optimización Multi-Objetivo: El modelo actual solo minimiza el "costo de no cobertura". Un modelo más avanzado podría usar un optimizador multi-objetivo (como MOPSO) para balancear metas en conflicto, por ejemplo:

- Minimizar el costo de no cobertura.
- Minimizar el costo total de instalación (ej. longitud de cableado).
- Maximizar la accesibilidad a los sensores para mantenimiento.

Modelo Dinámico: El campo no es estático; la "variabilidad" cambia. El maíz en etapa de grano tiene necesidades diferentes que en crecimiento vegetativo. Un modelo futuro podría re-optimizar las ubicaciones "ideales" de monitoreo a medida que avanza la temporada del cultivo.

Benchmarking de Algoritmos: Comparar el rendimiento de PSO (en términos de velocidad y calidad de la solución) con otros algoritmos metaheurísticos, como Algoritmos Genéticos (GA) o Búsqueda por Colonia de Hormigas (ACO), para determinar cuál es el más eficiente para este dominio de problema específico.

6.Referencias

Fundamentos del Algoritmo (PSO)

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. En *Proceedings of ICNN'95 - International Conference on Neural Networks* (Vol. 4, pp. 1942-1948). IEEE. https://doi.org/10.1109/ICNN.1995.488968

Bibliotecas de Software y Herramientas

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., … & Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Miranda, L. J. V. (2018). PySwarms: A research toolkit for particle swarm optimization in Python. *Journal of Open Source Software*, *3*(21), 433. https://doi.org/10.21105/joss.00433

Python Software Foundation. (2025). *Python Language Reference (Versión 3.x)*. https://www.python.org

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., … & SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2