Optimización de Riego
Con Enjambre de partículas

Instituto Tecnológico de México Campus Culiacán

Tópicos Avanzados de I.A.

Zuriel Dathan Mora Félix

Integrantes de equipo:

Sánchez Arévalo José Antonio – 21170474

Félix Avendaño Mateo – 21170314

05/11/2025

Descripción del Problema

En la agricultura de la región de Guasave, Sinaloa, la gestión eficiente del riego es un factor determinante para mantener la productividad de los cultivos y el aprovechamiento racional del agua. Sin embargo, la variabilidad del suelo, las diferencias en los requerimientos de humedad de los cultivos (como maíz, tomate y chile), y las micro variaciones topográficas del terreno dificultan una distribución homogénea del agua.

Por lo tanto, se requiere una estrategia que permita determinar la ubicación óptima de los sensores de humedad, considerando simultáneamente factores como la topografía, la variabilidad del suelo y la distribución de los cultivos, con el objetivo de maximizar la eficiencia del riego y minimizar las pérdidas de agua.

Justificación del Uso de PSO (Particle Swarm Optimization)

El algoritmo de Enjambre de Partículas (PSO) es una técnica de optimización inspirada en el comportamiento colectivo de los enjambres naturales, como bandadas de aves o cardúmenes de peces. Este método es particularmente adecuado para problemas complejos de optimización continua y multidimensional, como la ubicación de sensores en un campo agrícola.

El algoritmo de Enjambre de Partículas ofrece diversas ventajas para este tipo de aplicación:

Exploración eficiente del espacio de soluciones: permite encontrar configuraciones óptimas incluso en entornos con múltiples variables (humedad, tipo de cultivo, salinidad, etc.).

Rápida convergencia: alcanza soluciones de buena calidad en menos iteraciones que otros métodos heurísticos como los algoritmos genéticos.

Facilidad de implementación y adaptación: su estructura sencilla permite incorporar fácilmente restricciones o características del terreno agrícola.

Adecuación a sistemas no lineales: las condiciones del suelo y del riego no siguen relaciones lineales simples, y PSO puede modelar adecuadamente estas interacciones.

Diseño del Algoritmo

1. Entradas

Tabla de Datos de los cultivos con Coordenadas Geoespaciales (WGS 84) que contiene: La humedad, cultivo, elevación, salinidad, temperatura, latitud y longitud. (Todo esto en formato .CSV)

Cantidad de Sensores

2. Salidas

-Posiciones óptimas para la ubicación de cada sensor y el mejor costo del algoritmo PSO.

-Una gráfica con las ubicaciones de los sensores y la distribucion de los diferentes cultivos (Maiz, Tomate, Chile).

3. Proceso del Algoritmo

Inicio

- **Leer datos de la tabla de los cultivos**
- **Inicializar población de partículas (posibles ubicaciones de sensores)**
- **Para cada iteración:**
    - **Evaluar eficiencia de riego para cada partícula**
    - **Actualizar la mejor posición personal y global**
    - **Modificar posiciones y velocidades de partículas**
- **Fin iteraciones**
- **Seleccionar la mejor distribución encontrada para los sensores**

Fin

Resultados Obtenidos

Los resultados obtenidos se presentan de esta forma.

N. Sensores = 5

Particulas = 50

Iteraciones = 100

c1 = 0.5

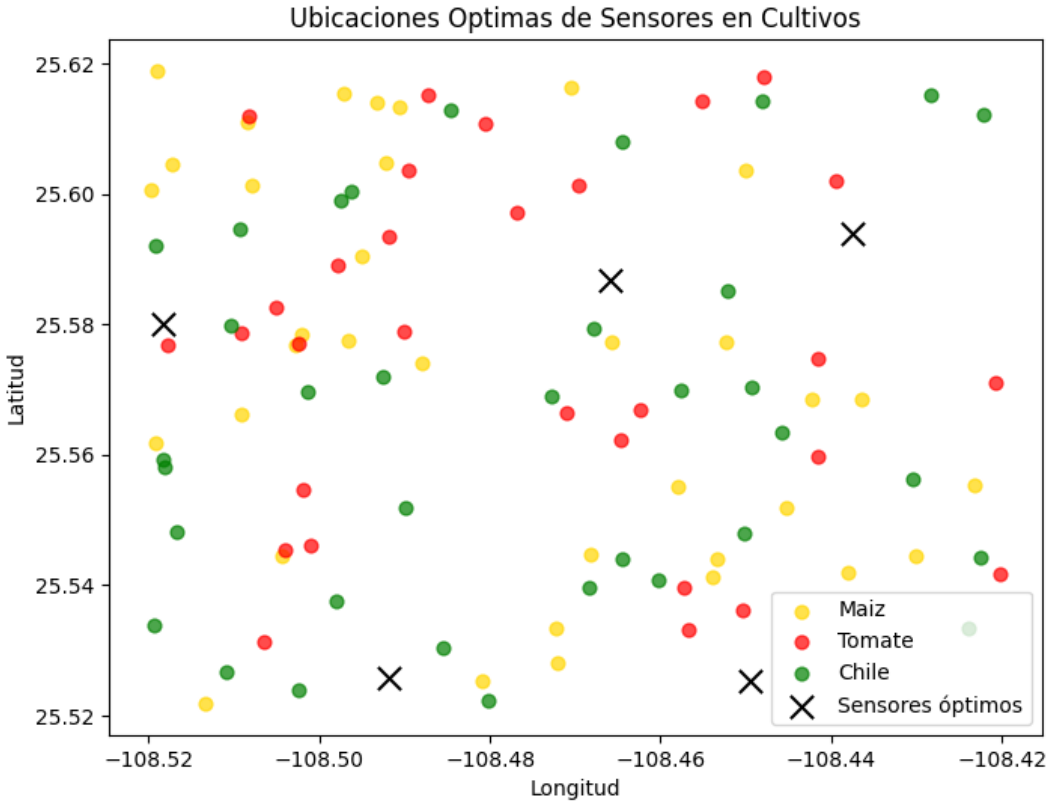
c2 = 0.3

w = 0.9

coverage_radius = 0.01 (El coverage radius significa que es el radio que esta alrededor de cada sensor dentro del cual los puntos del dataset se consideran vecinos.)

Mejor Costo = 50

```
PS C:\Users\jose.sanchez\Documents\TopicosIA\Unidad3_Enjambre> & C:\Users\jose.sanchez\AppData\Local\Programs\Python\Python313\p
ython.exe c:/Users/jose.sanchez/Documents/TopicosIA/Unidad3_Enjambre/main.py
2025-11-05 23:01:52,721 - pyswarms.single.global_best - INFO - Optimize for 100 iters with {'c1': 0.5, 'c2': 0.3, 'w': 0.9}
pyswarms.single.global_best: 100%|██████████|100/100, best_cost=50
2025-11-05 23:02:23,626 - pyswarms.single.global_best - INFO - Optimization finished | best cost: 50.0, best pos: [  25.59403404
 -108.43755268   25.58008203 -108.51833184   25.52585903
 -108.4917923    25.52541259 -108.44945404   25.58685298 -108.46598488]
------------------------------------------------------------
Mejor Costo: 50.00
Mejores posiciones para sensores (Latitud, Longitud):
Sensor 1: Latitud=25.594034, Longitud=-108.437553
Sensor 2: Latitud=25.580082, Longitud=-108.518332
Sensor 3: Latitud=25.525859, Longitud=-108.491792
Sensor 4: Latitud=25.525413, Longitud=-108.449454
Sensor 5: Latitud=25.586853, Longitud=-108.465985
```



Pruebas Funcionales

Se realizaron diferentes pruebas con diferentes valores para verificar la eficiencia del algoritmo PSO

Prueba 1:

N. Sensores = 5

Particulas = 50

Iteraciones = 50

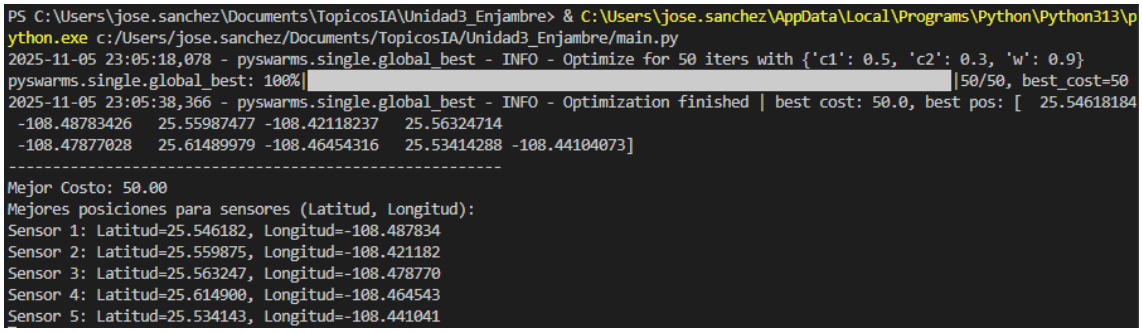
c1 = 0.5

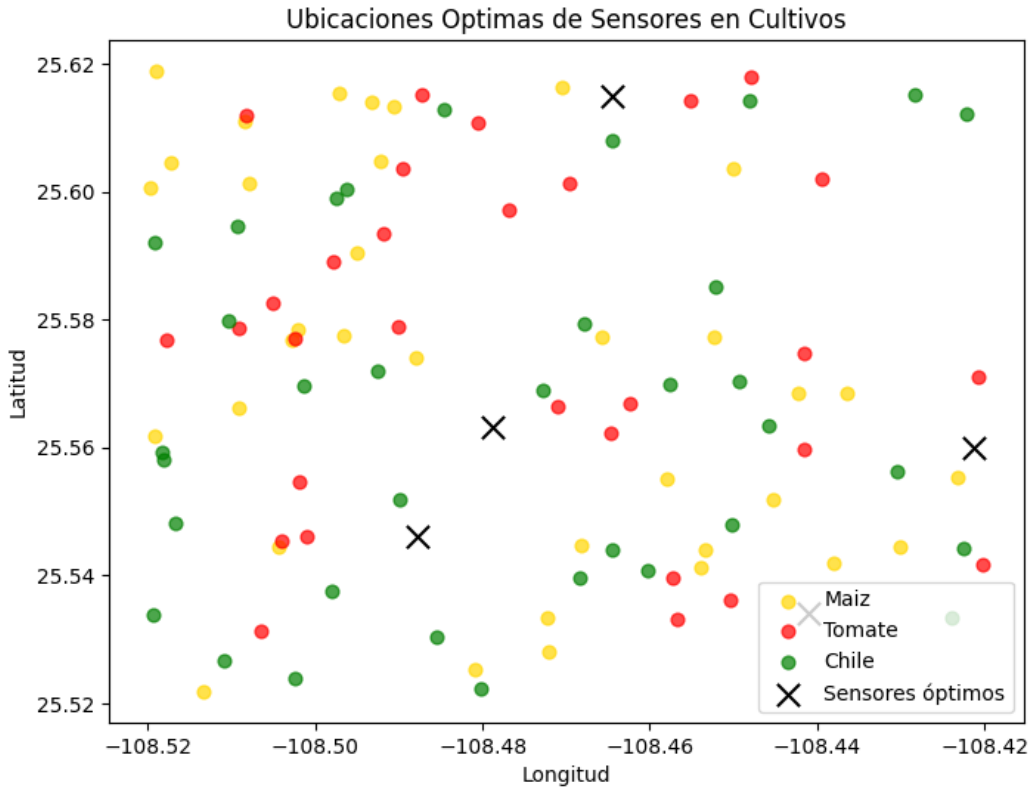
c2 = 0.3

w = 0.9

coverage_radius = 0.01

Mejor Costo = 50.00





Prueba 2:

N. Sensores = 5

Particulas = 30

Iteraciones = 100

c1 = 0.5

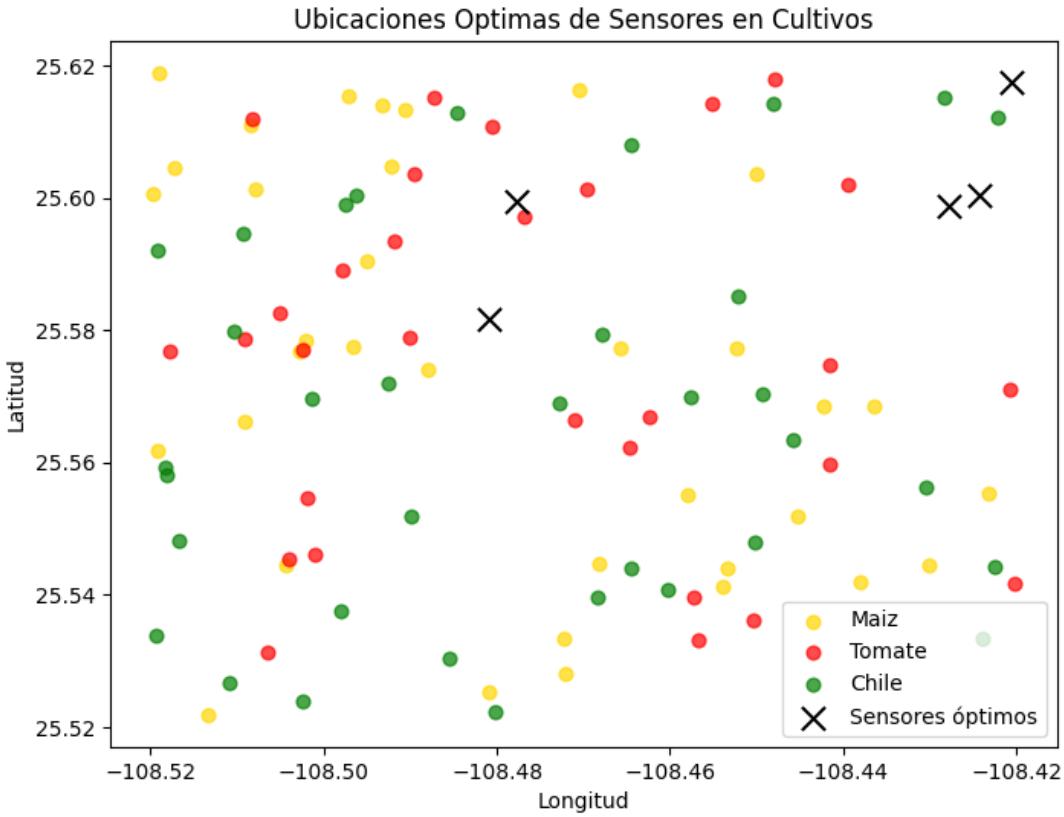
c2 = 0.3

w = 0.9

coverage_radius = 0.025

Mejor Costo = 209.16

```
PS C:\Users\jose.sanchez\Documents\TopicosIA\Unidad3_Enjambre> & C:\Users\jose.sanchez\AppData\Local\Programs\Python\Python313\p
ython.exe c:/Users/jose.sanchez/Documents/TopicosIA/Unidad3_Enjambre/main.py
2025-11-05 23:08:06,633 - pyswarms.single.global_best - INFO - Optimize for 100 iters with {'c1': 0.5, 'c2': 0.3, 'w': 0.9}
pyswarms.single.global_best: 100%|                                                                                |100/100, best_cost=209
2025-11-05 23:08:50,445 - pyswarms.single.global_best - INFO - Optimization finished | best cost: 209.1606404557992, best pos: [
  25.58173368 -108.48084335   25.59867912 -108.42769933   25.61740698
 -108.4204969    25.60043428 -108.42416107   25.59952116 -108.47764422]
------------------------------------------------------------
Mejor Costo: 209.16
Mejores posiciones para sensores (Latitud, Longitud):
Sensor 1: Latitud=25.581734, Longitud=-108.480843
Sensor 2: Latitud=25.598679, Longitud=-108.427699
Sensor 3: Latitud=25.617407, Longitud=-108.420497
Sensor 4: Latitud=25.600434, Longitud=-108.424161
Sensor 5: Latitud=25.599521, Longitud=-108.477644
```



Prueba 3:

N. Sensores = 7

Particulas = 50

Iteraciones = 100

c1 = 0.3

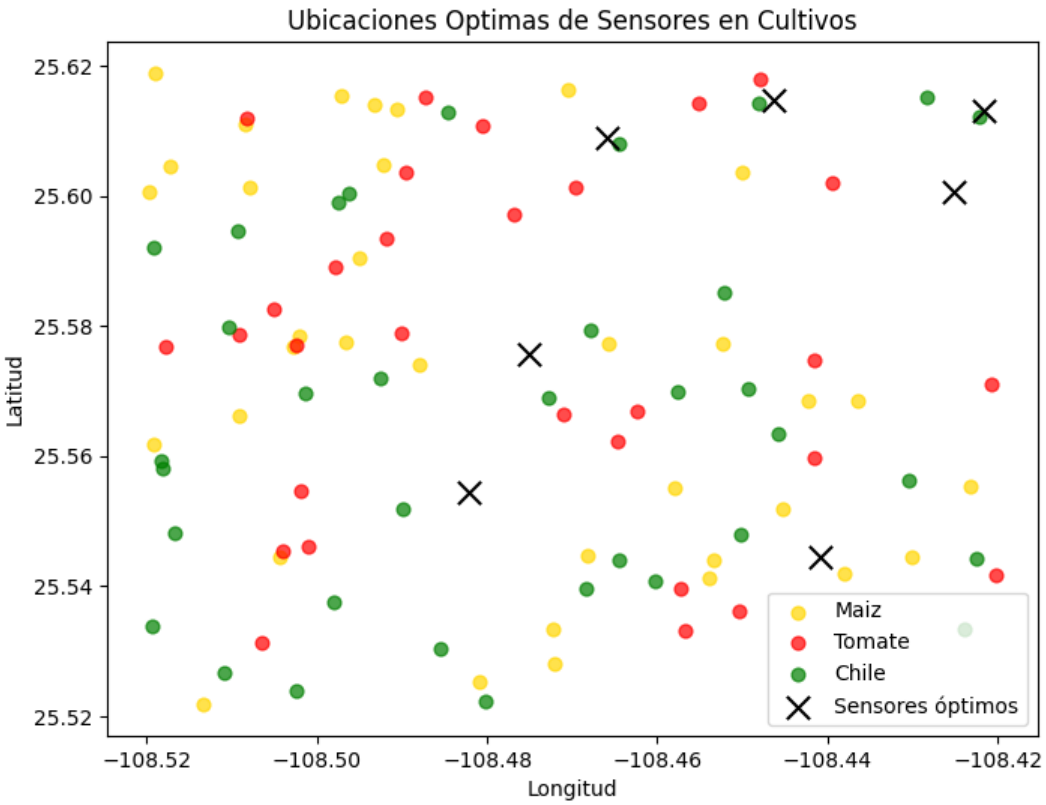
c2 = 0.6

w = 0.5

coverage_radius = 0.025

Mejor Costo = 329.14

```
PS C:\Users\jose.sanchez\Documents\TopicosIA\Unidad3_Enjambre> & C:\Users\jose.sanchez\AppData\Local\Programs\Python\Python313\p
ython.exe c:/Users/jose.sanchez/Documents/TopicosIA/Unidad3_Enjambre/main.py
2025-11-05 23:10:36,325 - pyswarms.single.global_best - INFO - Optimize for 100 iters with {'c1': 0.3, 'c2': 0.6, 'w': 0.5}
pyswarms.single.global_best: 100%|██████████|100/100, best_cost=329
2025-11-05 23:11:50,383 - pyswarms.single.global_best - INFO - Optimization finished | best cost: 329.14088987026855, best pos:
[  25.57559086 -108.4750289    25.6005384  -108.42516707   25.6130884
 -108.42162206   25.60901217 -108.46593428   25.5543562  -108.48210854
   25.54447236 -108.44088677   25.61476625 -108.44628653]
------------------------------------------------------------
Mejor Costo: 329.14
Mejores posiciones para sensores (Latitud, Longitud):
Sensor 1: Latitud=25.575591, Longitud=-108.475029
Sensor 2: Latitud=25.600538, Longitud=-108.425167
Sensor 3: Latitud=25.613088, Longitud=-108.421622
Sensor 4: Latitud=25.609012, Longitud=-108.465934
Sensor 5: Latitud=25.554356, Longitud=-108.482109
Sensor 6: Latitud=25.544472, Longitud=-108.440887
Sensor 7: Latitud=25.614766, Longitud=-108.446287
```



Análisis de eficiencia

Robustez

Se realizó la prueba de robustez probando 10 veces el algoritmo para verificar si se cambiaron los costos, al final estos son los resultados:

Datos Probados:

Particulas: 30

Iteraciones = 100

Repeticiones = 10

N.Sensores = 5

N.Dimensiones = N.Sensores * 2 = 10

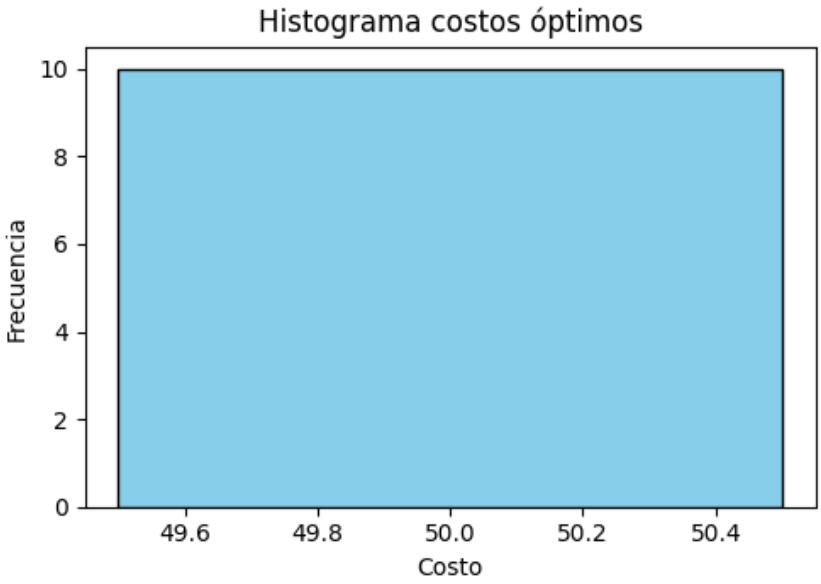
c1 =0.5

c2 =0.3

w =0.9

coverage_radius = 0.01

```
Robustez del algoritmo PSO:
  Repeticiones: 10
  Media: 50.0000
```



Se muestra un histograma en la cual indica que el costo se mantiene en 50 por lo cual no hay cambios en el costo (Para visualizar eso en el histograma, se necesita expander un poco el rango para poder graficar la barra).

Eficiencia por tiempo de ejecución

Evaluamos el tiempo de ejecución del algoritmo de Enjambre de Partículas (PSO)

Particulas: 50

Iteraciones = 30

Repeticiones = 5

N.Sensores = 5

c1 =0.3

c2 =0.6

w =0.5

coverage_radius = 0.025

```
Resumen:
Tiempos (s): [14.002, 14.9907, 18.618, 20.7054, 15.7664]
Min: 14.0020  Max: 20.7054  Promedio: 16.8165
```

Estos valores reflejan una consistencia temporal aceptable entre repeticiones, lo que indica que el algoritmo mantiene un desempeño estable.

Conclusión

La aplicación del algoritmo de enjambre de partículas en este problema de optimización de riego demostró un desempeño eficiente en la búsqueda de localizaciones óptimas para la colocación de sensores. Los resultados obtenidos demuestran que el método utilizado es efectivo para maximizar la eficiencia del riego y resolver la retención de agua.

Podemos concluir en que el enfoque basado en PSO es una alternativa sólida y adaptable para la automatización de la gestión de riego agrícola. La capacidad que este modelo ofrece para incorporar múltiples variables ambientales y de cultivo en un mismo proceso de optimización lo convierte en un candidato fuerte con respecto a otros modelos (como el algoritmo genético).

Link del repositorio github del codigo PSO: https://github.com/JSarevalo25/Unidad-3-Enjambre-de-Particulas