

MAESTRÍA EN INTELIGENCIA ARTIFICIAL

ALGORITMOS EVOLUTIVOS I

Desafío práctico: Optimización por Enjambre de Partículas (PSO)

Autor:
Maximiliano Torti

Docente:
Miguel Augusto Azar

Este trabajo fue realizado en noviembre de 2025.

Índice general

1. Desarrollo	1
1.1. Introducción	1
1.2. Datos	1
1.3. Mapa	1
1.4. Mapa de aptitud	2
1.5. Partícula	3
1.6. Simulación	4
1.6.1. Datos crudos	4
1.6.2. Enjambre con series de tiempo con ruido	4
1.6.3. PSO	4
1.7. Resultados	6
1.7.1. Datos crudos	6
1.7.2. Enjambre con series de tiempo con ruido	6
1.7.3. PSO	8
1.8. Conclusiones	10
1.9. Trabajo futuro	11
A. Repositorio	12

1 Desarrollo

1.1. Introducción

En el presente trabajo, se desarrolla la implementación del algoritmo de Optimización por Enjambre de Partículas (PSO) en la búsqueda de una solución óptima a un problema de posicionamiento y recorrido. El objetivo es evaluar el rendimiento del algoritmo y compararlo con otros métodos más simples.

En el caso planteado, se parte de una posición global conocida y luego se dispone solamente de mediciones de velocidad y ángulo de movimiento respecto al norte. La meta es poder continuar trazando una ubicación precisa e idónea del recorrido realizado sin contar con las mediciones de posicionamiento. Esta situación es común en sistemas embebidos cuando se pierde la señal del GPS (por ejemplo por un diseño incorrecto de la antena de recepción, por fallas en el software del dispositivo o por pérdida de línea de vista al cielo) o cuando intencionalmente se hace un apagado temporal del GPS para reducir el consumo eléctrico.

Inicialmente se realizó una prueba utilizando datos reales de sensores y se comprobó que el ruido inherente de los mismos genera un recorrido alejado del real. Por este motivo, se prosiguió con una simulación con optimización PSO que ajusta las posiciones a valores óptimos que mejoran la precisión del recorrido.

1.2. Datos

Los datos sobre los cuales se trabajó fueron obtenidos realizando un circuito en auto por la ciudad de Rosario y utilizando un celular para hacer el registro de los sensores disponibles y del GPS. En la figura 1.1 se muestran los primeros registros recolectados. De estos, solo se utilizó la latitud y longitud del GPS del punto de inicio y los valores de la velocidad (en metros/segundos) y ángulo de movimiento (en grados respecto al norte) de todo el recorrido.

Timestamp	Accel_x	Accel_y	Accel_z	Gyro_x	Gyro_y	Gyro_z	Light	Magnetic_x	Magnetic_y	Magnetic_z	Proximity	Latitude	Longitude	Elevation	Speed	Acceleration	Bearing
1.63E+15	-0.348	-1.021	9.815	-0.003	-0.013	0.004	4105	-47.831	18.263	-58.669	5.0	-3.295.812	-6.063.654	52.8	8.083.333	-1.021	86.323.690
1.63E+15	0.049	-0.755	10.256	-0.005	0.028	-0.001	4105	-48.469	15.900	-60.525	5.0	-3.295.813	-6.063.644	53.6	8.361.111	-0.755	61.604.633
1.63E+15	-0.499	-1.013	9.925	-0.023	-0.014	-0.005	4105	-48.975	18.281	-59.419	5.0	-3.295.815	-6.063.635	51.9	8.361.111	-0.013	30.706.606
1.63E+15	-0.447	-0.975	10.308	-0.008	-0.013	0.019	3725	-48.788	16.856	-58.800	5.0	-3.295.816	-6.063.627	51.2	8.111.111	-0.975	82.547.460
1.63E+15	-0.504	-0.999	9.621	-0.008	-0.003	0.012	3725	-50.456	19.406	-60.488	5.0	-3.295.818	-6.063.618	50.6	7.750.000	-0.999	88.920.540

FIGURA 1.1. Primeros registros de datos recolectados.

1.3. Mapa

Ejecutar las simulaciones requirió del desarrollo de utilidades que realizaran gráficos y cálculos geográficos. Con este fin se implementó en Python la clase *Map* con las siguientes características:

- Utiliza la librería OpenCV y Matplotlib para trazar el mapa de la zona de interés y marcar sobre el mismo puntos rojos que representan posiciones (latitud y longitud) y trayectos. En la figura 1.2 se observa el mapa utilizado.
- Método de calibración para transformar entre latitud/longitud \Leftrightarrow pixeles verticales/horizontales sobre la imagen del mapa, de manera de poder utilizar indistintamente cualquiera de los dos sistemas de referencia. Durante la calibración, se tomaron 4 puntos de referencia conocidos y se entrenaron dos modelos de regresión lineal.
- Algoritmo que permite obtener el punto final (latitud y longitud) luego de desplazarse una cierta cantidad de metros, con un cierto ángulo respecto al norte, desde una posición inicial conocida.
- Algoritmo Haversine para obtener la distancia en metros y el ángulo respecto al norte del segmento sobre la superficie terrestre que conecta dos puntos con posiciones conocidas.
- Soporte de mapa de aptitud con máscara de probabilidad.

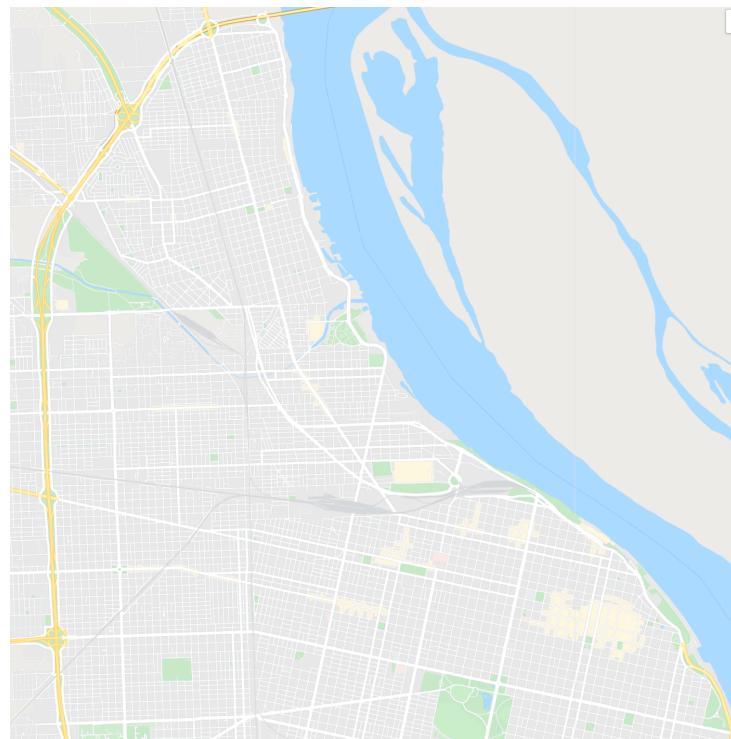


FIGURA 1.2. Mapa de zona de interés (ciudad de Rosario).

1.4. Mapa de aptitud

Cuantificar la idoneidad de una posición o trayecto requiere de una métrica medible. Esta se puede obtener utilizando un mapa de aptitud con una máscara de probabilidad que asigna un valor entre 0 y 1.0 a cada pixel del mapa original, según la posición sea más o menos probable. Las calles y avenidas tienen un valor de 1.0, mientras que los centros de manzana, el río y zonas donde no es posible transitar tienen valor 0. Entre ambos extremos, la máscara asigna valores decrecientes en forma suave. Así, para obtener la aptitud de una posición única se

utiliza la probabilidad de la máscara en ese pixel, y para la de un recorrido se acumulan las probabilidades individuales de cada una de las posiciones. En la figura 1.3 se muestra el mapa de aptitud utilizado en este trabajo.



FIGURA 1.3. Mapa de aptitud con máscara de probabilidad.

1.5. Partícula

Otra utilidad necesaria fue la clase que modela el objeto que se desplaza por el mapa. Este objeto se implementó en Python en la clase *Particle* con las siguientes características:

- Se inicializa con una posición inicial (latitud, longitud), las series de tiempo de velocidad y ángulo de movimiento del recorrido, y la instancia de la clase *Map* con la zona de interés.
- Posee métodos para moverse un paso de tiempo, utilizando las series de tiempo provistas.
- Posee métodos para acumular distancia desplazada. Esto resulta útil al optimizar cálculos ya que permite no ejecutar los algoritmos en cada paso de tiempo, donde el desplazamiento puede haber sido de 0 metros si el objeto se encontraba detenido.
- Posee métodos para evaluar una posición o un movimiento en base al mapa de aptitud de la clase *Map*.
- Posee métodos para ejecutar el algoritmo PSO.

1.6. Simulación

1.6.1. Datos crudos

Inicialmente, y a fin de tener una base de comparación, se creó una única partícula con posición inicial conocida y se utilizaron los datos originales de movimiento (velocidad y ángulo) sin procesar. Ejecutando el método *move* de la clase *Particle* repetidas veces hasta agotar todos los movimientos disponibles, se obtuvo el recorrido para la primer simulación.

Algorithm 1 Algoritmo simple con datos crudos

```

particle  $\leftarrow$  Particle( $x(0)$ ,  $v(t)$ ,  $\theta(t)$ )
for each  $v(j) \in v(t)$ ,  $\theta(j) \in \theta(t)$  do
     $x(t + 1) = move(x(t), v(j), \theta(j))$ 
end for
```

1.6.2. Enjambre con series de tiempo con ruido

En una segunda simulación se utilizó un enjambre de 500 partículas. Cada partícula se inicializó con posiciones iniciales ligeramente desplazadas (dentro de un radio de 50 metros) y con copias de las series de tiempo de velocidad y ángulo de movimiento con ruidos gaussianos superpuestos. Se ejecutó el trayecto completo para todas las partículas y se evaluó la aptitud acumulada de cada recorrido, seleccionando la mejor partícula como aquella con mayor valor de aptitud acumulada.

Algorithm 2 Algoritmo enjambre con datos con ruido

```

repeat for 500 particles
    particle $_i \leftarrow$  Particle( $x_{noise}(0)$ ,  $v_{noise}(t)$ ,  $\theta_{noise}(t)$ )
    Fitness $_i(0) = MapProbability(x_{noise}(0))$ 
    for each  $v(j) \in v_{noise}(t)$ ,  $\theta(j) \in \theta_{noise}(t)$  do
         $x_i(t + 1) = move(x_i(t), v_i(j), \theta_i(j))$ 
        if  $x_i(t + 1) - x_i(t) \geq MinDistance$  then
            Fitness $_i(t + 1) = MapProbability(x_i(t + 1))$ 
        end if
    end for
end
BestParticle  $\leftarrow Max(\sum Fitness(t))$ 
```

1.6.3. PSO

En una última simulación, se realizó una mejora sobre el caso anterior consistente en ejecutar cada cierta longitud de desplazamiento (configurada inicialmente en 50 metros) instancias de 50 épocas del algoritmo PSO, donde la partícula se mueve una porción (dada por w) en la dirección y velocidad de la inercia (paso de movimiento anterior), otra porción (dada por c_1) en la dirección de la mejor posición encontrada por dicha partícula hasta el momento y una porción (dada por c_2) en la dirección de la mejor partícula del enjambre. La forma en la que se determinó las mejores posiciones locales y globales fue mediante la función de aptitud ya mencionada. Los coeficientes w , c_1 y c_2 son hiperparámetros que fueron ajustados manualmente.

Algorithm 3 PSO

```

repeat for 500 particles
    particlei  $\leftarrow$  Particle( $x_{noise}(0)$ ,  $v_{noise}(t)$ ,  $\theta_{noise}(t)$ )
    Fitnessi(0) = MapProbability( $x_{noise}(0)$ )
end
for each  $v(j) \in v_{noise}(t)$ ,  $\theta(j) \in \theta_{noise}(t)$  do
    repeat for 500 particles
         $x'_i(t+1) = move(x_i(t), v_i(j), \theta_i(j))$ 
    end
    if  $\exists i / x'_i(t+1) - x_i(t) \geq MinDistance$  then
        repeat for 50 psoepochs
            repeat for 500 particles
                 $x_i(epoch+1) = move(w * \Delta x_i(epoch), c_1 * (pbest_i - x_i(epoch)),$ 
                 $c_2 * (gbest - x_i(epoch)))$ 
                pbesti  $\leftarrow$  MaxMapProbability(pbesti,  $x_i(epoch+1)$ )
            end
            gbest  $\leftarrow$  MaxMapProbability(pbesti  $\forall i$ )
        end
    end if
    repeat for 500 particles
         $x_i(t+1) = pbest_i$ 
        Fitnessi(t+1) = MapProbability(pbesti)
    end
end for
BestParticle  $\leftarrow$  Max( $\sum$  Fitness(t))

```

1.7. Resultados

1.7.1. Datos crudos

El trayecto obtenido para una única partícula y datos de movimiento sin procesar se muestra en la figura 1.4. Se observa que el recorrido se aleja de posiciones válidas y la mayor parte del mismo transcurre por encima del agua.

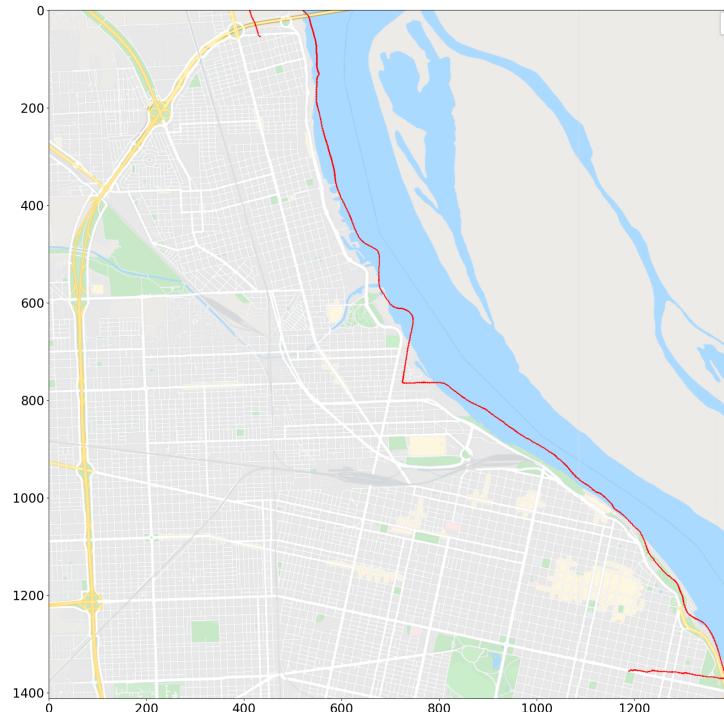


FIGURA 1.4. Trayecto partícula única con datos sin procesar.

1.7.2. Enjambre con series de tiempo con ruido

El trayecto obtenido para la mejor partícula de la segunda simulación se muestra en la figura 1.5. Si bien existe una mejora respecto al caso anterior, aún existen segmentos donde el recorrido se aleja de posiciones válidas, en especial en la mitad del mismo. En el siguiente [video](#) se puede validar el paso a paso de la evolución del trayecto a lo largo de la ejecución del algoritmo para todo el enjambre.

Para poder comparar cuantitativamente este y el algoritmo siguiente, se crearon gráficas de la evolución de la aptitud acumulada en función de los pasos de movimiento. Este valor se calculó tanto en forma totalizada para las 500 partículas del enjambre en la figura 1.6, como para la mejor partícula en la figura 1.7. En ambos casos se puede analizar con mayor detalle lo mencionado previamente: un crecimiento inicial sostenido, zonas medias de estancamiento y una zona final de crecimiento suave. Más aún, considerando solo los valores finales, se obtuvo un valor de aptitud acumulada para todo el enjambre de 23889, equivalente a un promedio de 0.164 por partícula por época (muy alejado del ideal de 1.0); mientras que para la mejor partícula se obtuvo un acumulado de 153, equivalente a 0.5 por época promedio, también alejado del ideal.

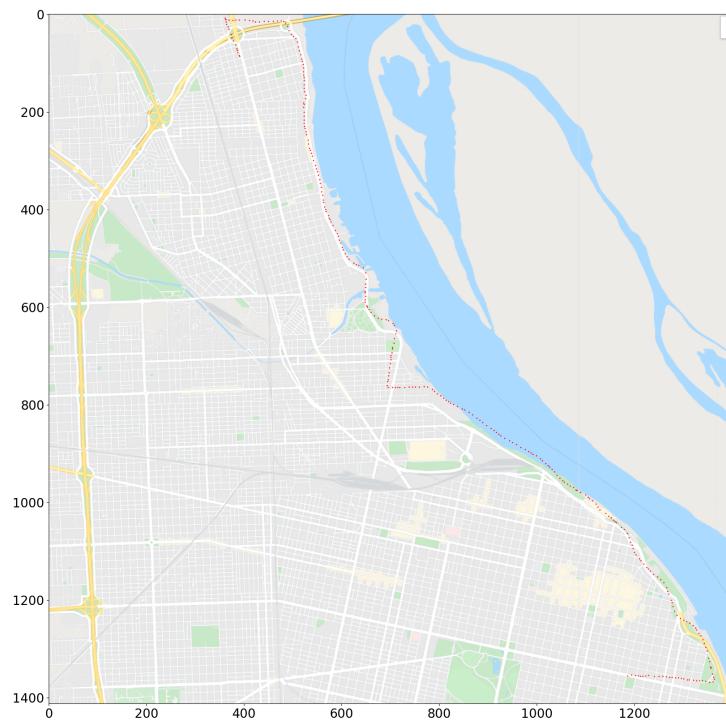


FIGURA 1.5. Trayecto mejor partícula en segunda simulación.

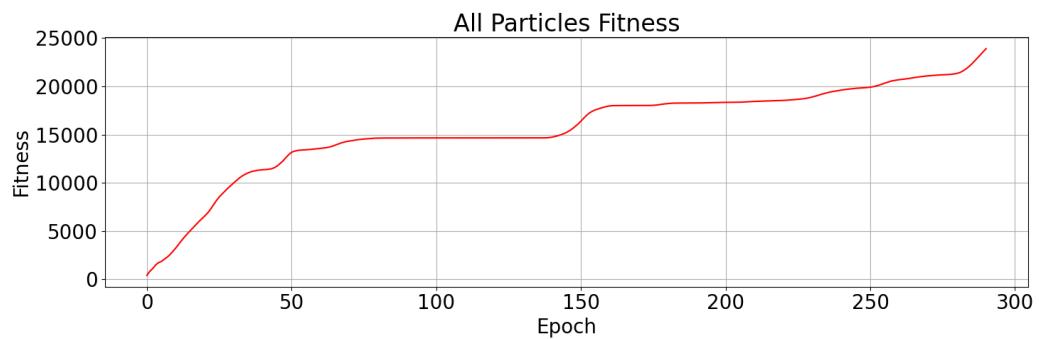


FIGURA 1.6. Aptitud acumulada del enjambre en segunda simulación.

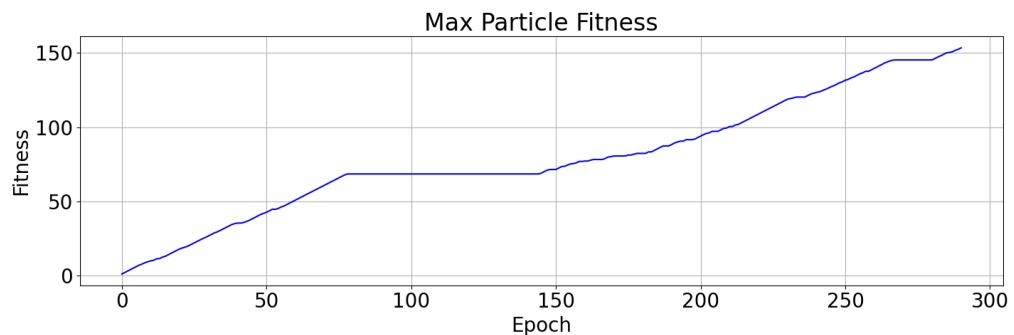


FIGURA 1.7. Aptitud acumulada de mejor partícula en segunda simulación.

1.7.3. PSO

El trayecto obtenido para la mejor partícula del enjambre de la tercera simulación (algoritmo PSO) se muestra en la figura 1.8. Se observa una mejora notoria respecto a los métodos anteriores ya que la partícula se mantiene en posiciones válidas en todo momento. En el siguiente [video](#) se puede validar el paso a paso de la evolución del recorrido a lo largo de la ejecución del algoritmo.

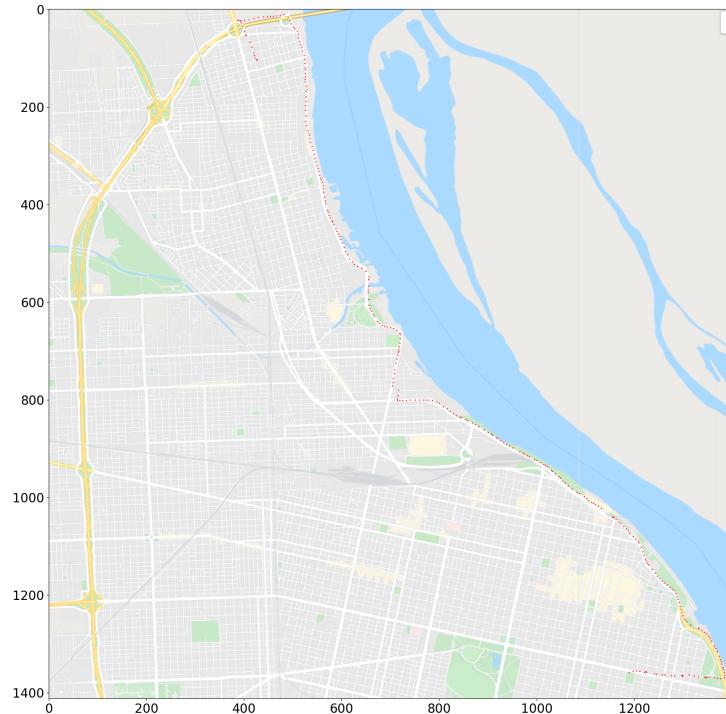


FIGURA 1.8. Trayecto mejor partícula en simulación PSO.

Para profundizar el análisis comparativo con el caso anterior, se muestra la evolución de la aptitud acumulada en función de los pasos de movimientos realizados en forma totalizada para las 500 partículas del enjambre en la figura 1.9 y para la mejor partícula en la figura 1.10. A diferencia del caso anterior, se observa un crecimiento sostenido constante en ambos gráficos. Al considerar los valores finales, se obtuvo un valor de aptitud acumulada para todo el enjambre de 144536, equivalente a un promedio de 0.98 por partícula por época (prácticamente ideal); mientras que para la mejor partícula se obtuvo un acumulado de 292, equivalente a 0.99 por época promedio, también prácticamente ideal.

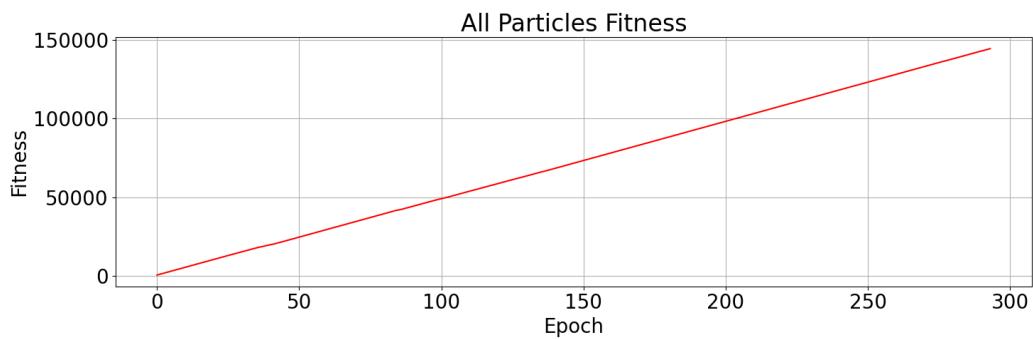


FIGURA 1.9. Aptitud acumulada del enjambre en simulación PSO.

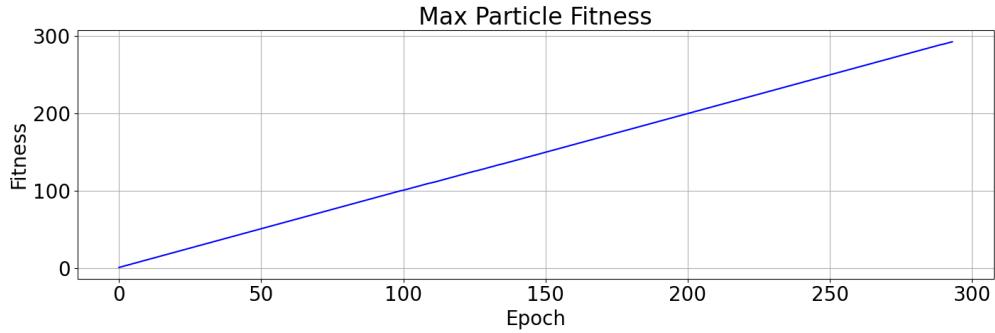


FIGURA 1.10. Aptitud acumulada de mejor partícula en simulación PSO.

A continuación se buscó evaluar el desempeño de las diferentes instancias de PSO en distintas partes del trayecto. Para esto se graficó la evolución de la función de aptitud (acumulada y promedio) para todas las partículas para 3 puntos del trayecto: inicial, medio y final. Los resultados obtenidos se muestran en las figuras 1.11 y 1.12. Se observa que, excepto por el punto medio donde las partículas aleatoriamente ya iniciaron en posiciones válidas, en los otros dos casos las partículas comenzaban relativamente dispersas (aptitud promedio menor a 0.85) y con las sucesivas ejecuciones de PSO todas las partículas convergen a posiciones válidas (aptitud promedio tiendente a 1.0).

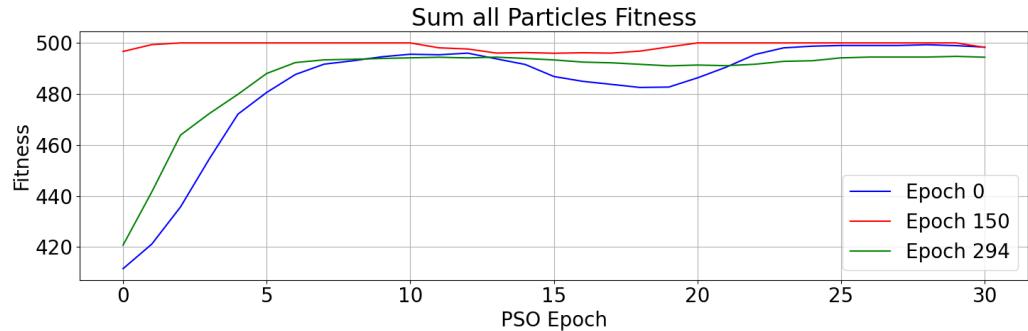


FIGURA 1.11. Aptitud del enjambre durante ejecución PSO en diferentes etapas de la trayectoria.

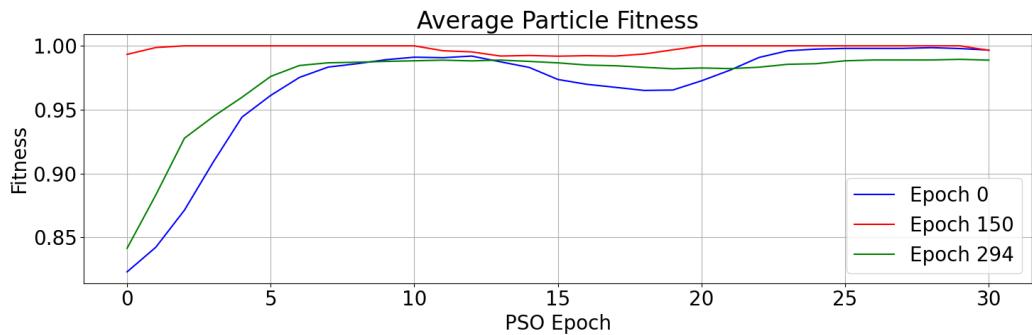


FIGURA 1.12. Aptitud promedio de las partículas del enjambre durante ejecución PSO en diferentes etapas de la trayectoria.

Para finalizar, se graficó un diagrama de cajas para las mismas instancias de PSO en la figura 1.13. Se observa que la mayor parte de los valores (exceptuando unos

pocos outliers) se distribuyen alrededor del puntaje ideal, lo que indica que el algoritmo logrado es estable, preciso y consistente.

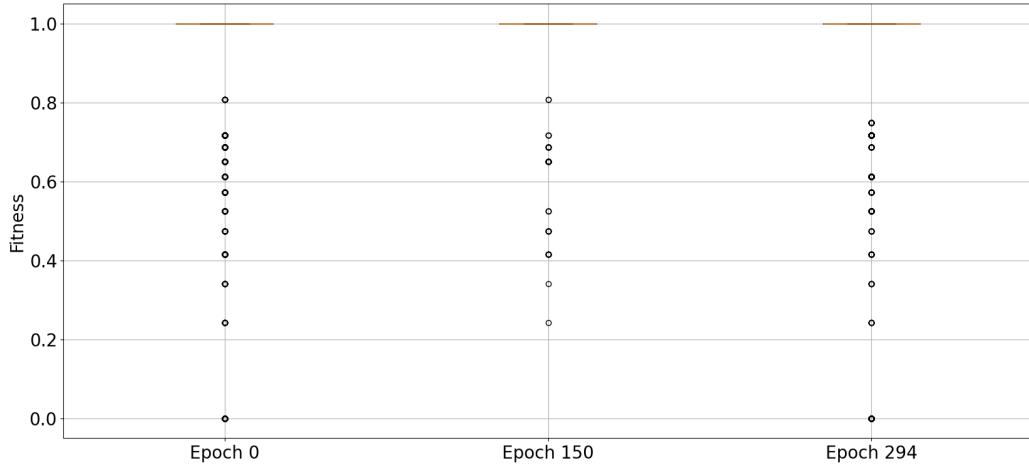


FIGURA 1.13. Diagrama de cajas de aptitud durante ejecución PSO en diferentes etapas de la trayectoria.

1.8. Conclusiones

Se comprobó la utilidad del algoritmo PSO para ajustar trayectorias obtenidas al mover un objeto desde un punto inicial conocido en la superficie terrestre y utilizando mediciones de velocidad y ángulo de movimiento respecto al norte obtenidas de sensores físicos.

Se evidenció que utilizar las mediciones sin procesar conduce a trayectorias mayormente inválidas y que realizar múltiples simulaciones con el artillugio de agregar ruido gaussiano tampoco converge a resultados aceptables. Bajo estas condiciones, se demostró que PSO logra resultados cualitativa y cuantitativamente aptos, ajustando las posiciones a zonas válidas y preservando la forma general de la trayectoria original.

Entre las dificultades encontradas se menciona principalmente el costo computacional requerido. Ejecutar el algoritmo de PSO para muchas partículas requiere un tiempo de cómputo considerable, por lo que fue necesario optimizar el número de partículas utilizadas en el algoritmo e implementar el mecanismo de mínima distancia recorrida para disminuir el número de instancias totales de PSO ejecutadas. Más aún, se intentó realizar una prueba de concepto partiendo de una posición inicial también desconocida, llenando el mapa de partículas y descartando aquellas que aún ejecutando el algoritmo anterior terminaban en posiciones poco probables, pero el costo computacional era tan elevado que hacía inviable lograr resultados en el tiempo disponible. Por otro lado ajustar los hiperparámetros de PSO w , c_1 , c_2 y pso_{epochs} también tuvo su complejidad, ya que se buscaba lograr una buena exploración del espacio mientras que se mantenían la estabilidad y velocidad de convergencia del algoritmo y también se respetaba la forma general de la trayectoria marcada por los datos de los sensores.

1.9. Trabajo futuro

Como trabajo futuro se plantea profundizar en el caso donde la posición inicial es desconocida, siendo necesario trabajar en diversas optimizaciones (posiblemente ejecutar los cálculos en forma paralela) para poder lograr tiempos de ejecución razonables.

A Repositorio

El repositorio con el código fuente y archivos adicionales de utilidad se encuentra disponible en el siguiente link:

https://github.com/maxit1992/MIA_AE1