Implementación de un algoritmo genético para optimizar trayectorias en un modelo del sistema solar. IIMAS

C. Iván Pineda S.[1]

[1] *Universidad Nacional Autónoma de México*
4 de septiembre de 2018

Resumen

El objetivo de este trabajo es modelar el sistema solar y su dinámica utilizando las ecuaciones de movimiento para un sistema de n cuerpos. Para darle solución a este problema de modelado se empleó un algoritmo genético con el fin de encontrar las trayectorias óptimas para el viaje desde la Tierra hasta otro cuerpo del sistema solar, la función de evaluación de dicho algoritmo recibe la velocidad inicial de la nave, un tiempo de inicio del viaje y utiliza el método de Euler para resolver las trayectorias de cada uno de los cuerpos. La evaluación es multiobjetivo y toma en cuenta la distancia más corta a la que estuvo la nave de su destino, la última distancia de la nave a su destino (en caso de que no llegase a la superficie del mismo), el tiempo que tomó realizar el viaje y el tiempo que la nave estuvo cerca del destino.

Introducción

Mecánica newtoniana

La interacción gravitatoria no relativista de dos cuerpos con masa (m_1 y m_2) se describe con la segunda ley de Newton y con la ley de gravitación universal [José and Saletan, 2000].

$$m_1 \frac{d^2(\mathbf{r_1} - \mathbf{r_2})}{dt^2} = -G \frac{m_1 m_2}{|\mathbf{r_1} - \mathbf{r_2}|^3} (\mathbf{r_1} - \mathbf{r_2}) \tag{1}$$

Donde:

$G = 39.4784176\ AU^3\ YR^{-2}\ SM^{-1}$, G es la constante de gravitación universal, AU=Unidad Astronómica, YR=Año, SM=Masas solares.

$\mathbf{r_n}$ es el vector de posición de cada cuerpo.

Simplificando la ecuación (1), obtenemos una función para la aceleración (2):

$$m_1 \frac{d^2(\mathbf{r_1} - \mathbf{r_2})}{dt^2} = -G \frac{m_1 m_2}{|\mathbf{r_1} - \mathbf{r_2}|^3} (\mathbf{r_1} - \mathbf{r_2}) \Rightarrow \frac{d^2(\mathbf{r_1} - \mathbf{r_2})}{dt^2} = -G \frac{m_2}{|\mathbf{r_1} - \mathbf{r_2}|^3} (\mathbf{r_1} - \mathbf{r_2}) \tag{2}$$

La ecuación (2) determina la dinámica de un cuerpo causada por la atracción gravitacional de otro cuerpo con masa, el problema a resolver es de n cuerpos. La fuerza que siente un cuerpo en un determinado momento es la suma de las fuerzas de cada uno de los cuerpos a su alrededor exceptuándose a si mismo. Utilizando (2), esto es:

$$\mathbf{a_k} = \sum_{i=1, i\neq k}^{n} \frac{d^2(\mathbf{r_k} - \mathbf{r_i})}{dt^2} \tag{3}$$

Utilizando (2) y (3) se obtiene:

$$\mathbf{a_k} = \sum_{i=1, i\neq k}^{n} [-G\frac{m_i}{|\mathbf{r_k} - \mathbf{r_i}|^3}(\mathbf{r_k} - \mathbf{r_i})] = -G \sum_{i=1, i\neq k}^{n} m_i \frac{\mathbf{r_k} - \mathbf{r_i}}{|\mathbf{r_k} - \mathbf{r_i}|^3} \tag{4}$$

La k representa al cuerpo del cual queremos obtener la aceleración y la i es cada uno de los n cuerpos que influyen gravitacionalmente sobre k.

Utilizando el método de Euler para integrar una ecuación diferencial y así obtener la velocidad y la posición de de un cuerpo k en un instante t, de (4) obtenemos:

$$\mathbf{v_{k_{t+1}}} = \mathbf{v_{k_t}} + \mathbf{a_k}dt = \mathbf{v_{k_t}} - G \sum_{i=1, i\neq k}^{n} m_i \frac{\mathbf{r_k} - \mathbf{r_i}}{|\mathbf{r_k} - \mathbf{r_i}|^3} dt \tag{5}$$

$$\mathbf{r_{k_{t+1}}} = \mathbf{r_{k_t}} + \mathbf{v_{k_{t+1}}}dt = \mathbf{r_{k_t}} + [\mathbf{v_{k_t}} + \mathbf{a_k}dt]dt = \mathbf{r_{k_t}} + \mathbf{v_{k_t}}dt - G \sum_{i=1, i\neq k}^{n} m_i \frac{\mathbf{r_k} - \mathbf{r_i}}{|\mathbf{r_k} - \mathbf{r_i}|^3} dt^2 \tag{6}$$

Que son funciones para la velocidad y la aceleración de un cuerpo k.

Implementación en pseudocódigo

```
//Una implementación para un módelo del sistemas solar basado en el método de Euler.
n = [Sol, Mercurio, Venus, Tierra, ..., nave]
for k in n:
    a_k = vector(0, 0, 0)
    for i in n:
        if k!=i:
            a_k += -G * m_i * (r_k - r_i) / |r_k - r_i|^3 * dt
        end if
    end for
    r_k += v_k * dt + a_k * dt^2
    v_k += a_k * dt
end for
```

Algoritmos genéticos

Los algoritmos genéticos forman parte de los métodos heurísticos de optimización y se utilizan para encontrar los máximos y/o mínimos de una función evitando las derivadas y cálculo de gradientes, se basan en la selección natural [Gestal et al., 2010].

Codificación binaria

En la naturaleza la información necesaria para codificar las proteínas que constituyen a los seres vivos está almacenada en una cadena de ADN, dicha cadena se compone de solo 4 bases; adenina, guanina, citosina y timina. Posteriormente se realiza una transcripción a ARN que se compone de las mismas bases, solo se cambia timina por uracilo y finalmente se realiza la traducción a proteínas.

Para este algoritmo se utilizó una codificación en números binarios, es decir, una cadena de solo 0 y 1 que posteriormente fue traducida a un número decimal que se encuentra dentro de un intervalo dado.

Se creó un número determinado de cadenas de binarios que fue nuestra población, cada una de estas cadenas tiene un número determinado de bits al que llamamos n y la distribución de ceros y unos es aleatoria.

Para convertir la cadena binaria a decimal se realizó un procedimiento idéntico al del siguiente ejemplo:

$$dec = 101011_2 = 1 * 2^5 + 0 * 2^4 + 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0 = 32 + 0 + 8 + 0 + 2 + 1 = 43$$

Para que dicho número esté en el intervalo en el que vamos a evaluar a nuestra función utilizamos:

$$decInt = a + \frac{dec}{2^n - 1}(b - a) \tag{7}$$

Donde a y b son los límites mínimo y máximo respectivamente, n es el número de bits de nuestra cadena y dec es nuestra primer conversión de binario a decimal, $decInt$ es el decimal que recibe la función de evaluación.

Selección

El objetivo es encontrar los mínimos. Por lo tanto, seleccionamos aquellas cadenas que al ser convertidas a decimal y al evaluarlas en la función $f(v_{x_0}, v_{y_0}, inicio)$ den los resultados más pequeños, es decir, las cadenas que produzcan las trayectorias con mejor $fitness$. La función evidentemente tiene un gran número de mínimos, obtenerlos con métodos numéricos o analíticos convencionales no es lo más efectivo.

El punto P es un mínimo local de f si existe un entorno reducido de centro x_0, que cumpla que $f(x) \geq f(x_0)\ \forall x \in E'(x_0)$. $P(x_0, f(x_0))$ es un mínimo absoluto de $f \iff \forall x \neq x_0, A, f(x_0) \leqslant f(x)$, donde A es un subconjunto del dominio de f.

La cantidad de cadenas a seleccionar puede variar. Para este trabajo se utilizó la mitad de la población, es decir solo el 50 % más adaptado puede dejar descendencia.

Reproducción

Una vez que se seleccionan los individuos más aptos, es necesario que dejen descendencia. Se seleccionan dos cadenas al azar, a partir de las cuales se crea una cadena hija. Para este algoritmo se genera una cantidad de cadenas hijas igual a la cantidad de cadenas padres, es decir, la mitad de la población inicial, así padres e hijos sumaran la misma cantidad de cadenas que al inicio.

Hay varias formas de combinar los genes (unidades de información, en este caso bits), aquí se utilizó el cruce uniforme que consiste en formar una cadena hija a partir de dos padres, donde cada

bit de las cadenas padres tiene un 50 % de probabilidad de pasar al hijo. Esto se logra seleccionando aleatoriamiente un bit en una posición específica de las cadenas padres (lo más fácil es empezar en la posición 0 y terminar en el último carácter de la cadena). Dicho bit se asigna en la misma posición para el hijo y se descarta esa posición.

Mutación

Las mutaciones son esenciales para mantener la variabilidad en nuestra población, sin estas, nuestros individuos tendrán el mismo valor en un menor número de generaciones, esto es ideal sí ese valor corresponde a un mínimo, pero casi nunca es así.

Para mutar a un individuo es necesario cambiar aleatoriamente el valor de algunos de sus genes o bits, para esto se genera un número aleatorio en un intervalo que va desde el 0 hasta la longitud de la cadena menos 1, se escoge la posición en la cadena correspondiente a ese número y se intercambia su valor por 0 si era 1 y viceversa, se realiza lo mismo el número de veces que se quiera mutar al individuo y esto se repite para una cantidad aleatoria de individuos.

Algoritmo principal

Los pasos que se llevan a cabo son los siguientes:

1. Generación de una población inicial de cadenas.
2. Evaluación.
3. Selección de las más aptas.
4. Reproducción.
5. Mutación solo en los hijos.
6. Generación de una nueva población que es la suma de los padres y los hijos.
7. Se regresa al paso 2 con la generación obtenida en el paso 6 hasta una condición de paro o cuando se alcance el número específico de iteraciones.
8. Se imprimen o guardan los valores de todas las generaciones.

Métodos

Este trabajo está basado en un artículo sobre el Diseño de un programa en python para la enseñanza de la transferencia de órbita de Hohmann [Méndez Hincapié and Monroy Cañón, 2016].

Se utilizó python para la elaboración de los algoritmos, la librería de math y numpy para el desarrollo matemático, cython y multiprocessing para optimizar el tiempo de ejecución, random para generar números aleatorios y Visual Python para el modelo visual, el código se puede revisar en `https://github.com/Ivan252512/AGSistemaSolar.git`. Visual python es la única biblioteca que no se instala con Anaconda, para hacerlo es necesario ejecutar desde la terminal de Linux **pip install vpython**, para más información visitar el sitio `http://vpython.org` y `https://www.anaconda.com/`.

La velocidad del cohete varía entre 40320 km/hr (velocidad de escape para la Tierra) y 60000 km/hr. Para la simulación se agregaron 15 cuerpos, entre estos están todos los planetas del sistema solar, el Sol, 4 satélites de Júpiter, la Luna, un satélite de Saturno y la nave. Para resolver las trayectorias se utilizaron las ecuaciones (5) y (6).

Resultados

Luna

Tabla 1. Valores para la Luna

Variable	Población	Generaciones	Tiempo total	Tiempo de inicio	dt
Valor	40	70	30.4 días	0-15.2 días	1 min

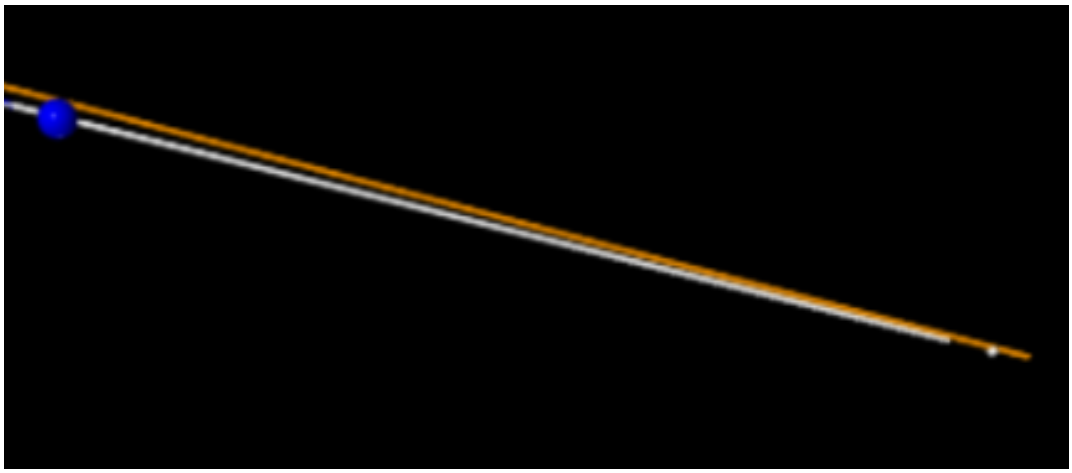

Figura 1: Trayectoria óptima de la nave (naranja) desde la Tierra (azul) a la Luna (blanco), se encontró una ruta que llega directo a la Luna.

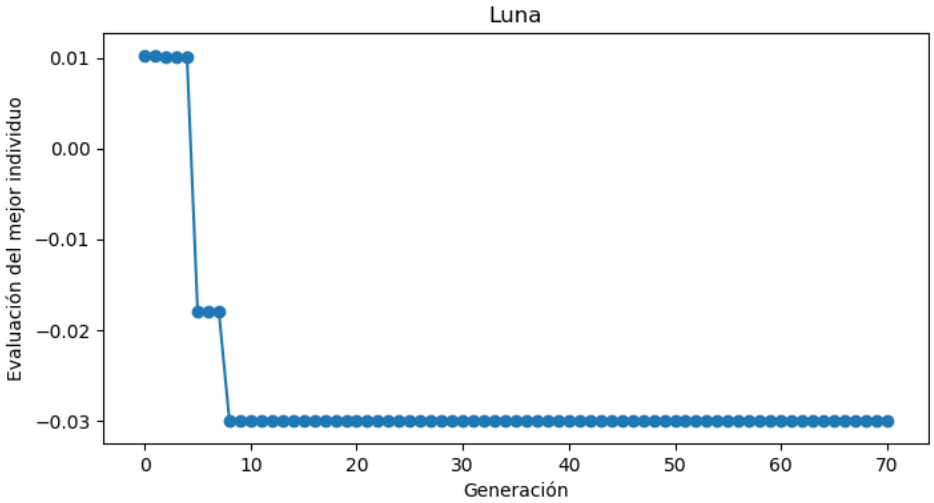

Figura 2: Gráfica que compara el resultado de la evaluación para el mejor individuo por generación contra su generación, desde la novena generación los resultados comienzan a ser constantes.

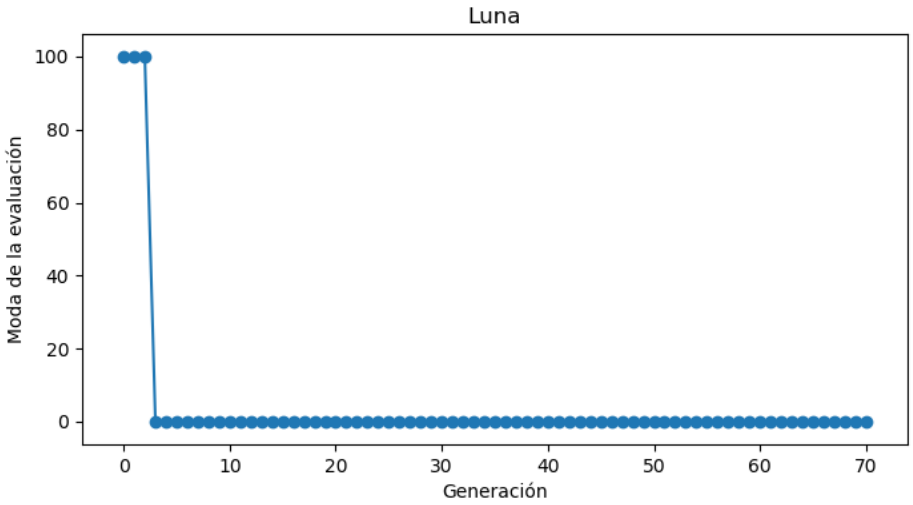

Figura 3: Gráfica que compara el resultado de la evaluación para la moda de cada generación, los resultados son similares a los de la gráfica del mejor individuo, esto indica que se cumple que los mejores individuos son los que dejan mayor descendencia, exceptuando las primeras generaciones.

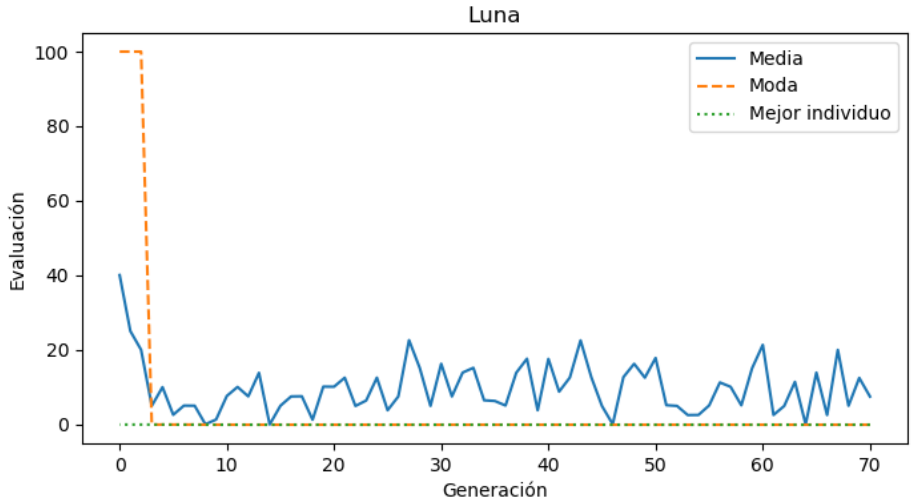

Figura 4: Comparación del mejor individuo, la moda y la media de cada generación.

Tierra

El objetivo era lograr que la nave orbitara a la Tierra, no se logró, pero la trayectoria sí es cercana a dicho planeta.

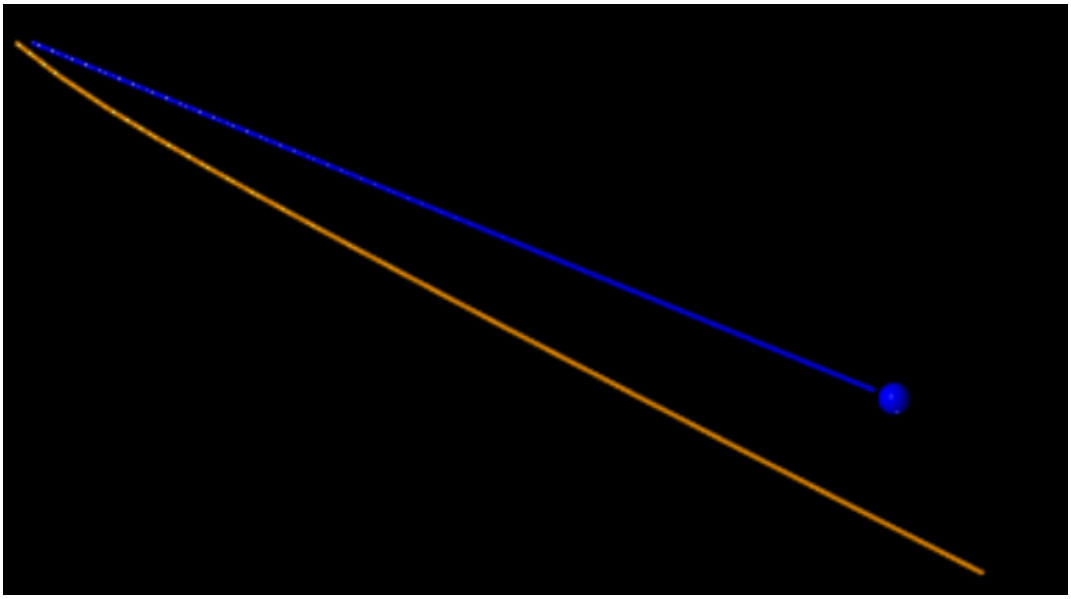

Figura 5: Trayectoria óptima de la nave (naranja) desde la Tierra (azul).

Tabla 1. Valores para la Tierra

Variable	Población	Generaciones	Tiempo total	Tiempo de inicio	dt
Valor	40	70	15.2 días	0-7.1 días	1 min

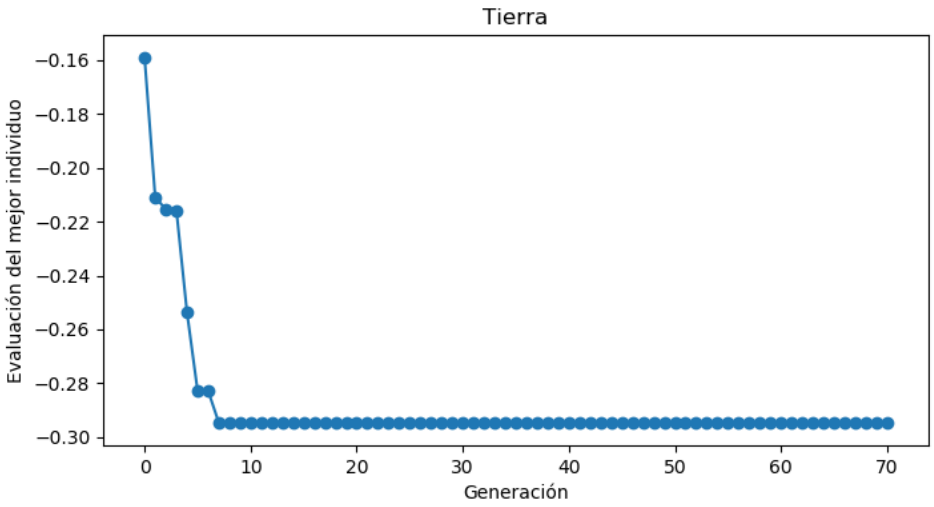

Figura 6: Gráfica que compara el resultado de la evaluación para el mejor individuo por generación contra su generación, desde la octava generación los resultados comienzan a ser constantes.

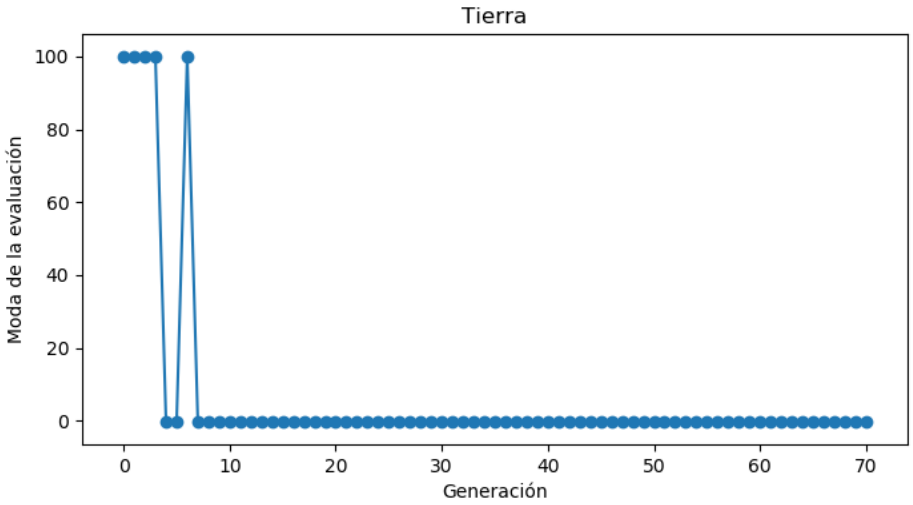

Figura 7: Gráfica que compara el resultado de la evaluación para la moda de la generación, los resultados son similares a los de la gráfica del mejor individuo, esto indica que se cumple que los mejores individuos son los que dejan mayor descendencia, exceptuando las primeras generaciones.

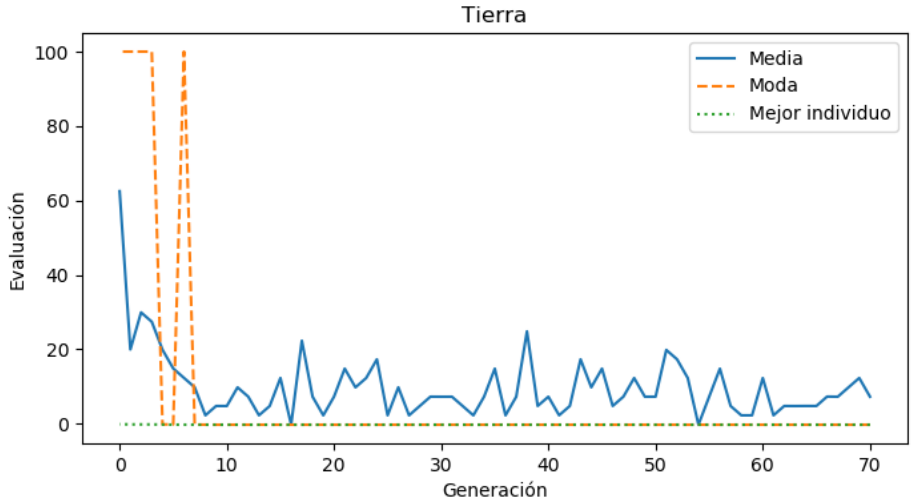

Figura 8: Comparación del mejor individuo, la moda y la media de cada generación.

Júpiter

Para lograr trayectorias largas en un tiempo factible es necesario usar dt grandes, por lo que la propagación de errores es también significativa.

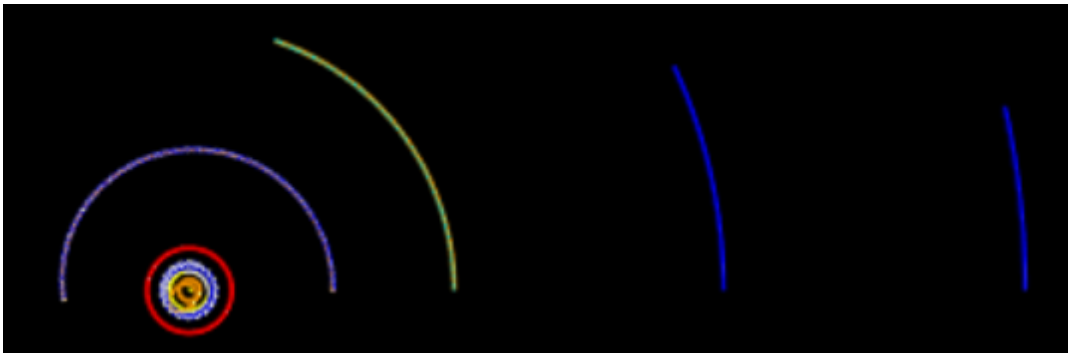

Figura 9: Trayectoria óptima de la nave (naranja) desde la Tierra (azul) a Júpiter(Verde), no se acerca la nave, las órbitas en general son muy erradas por el dt.

Tabla 1. Valores para Júpiter

Variable	Población	Generaciones	Tiempo total	Tiempo de inicio	dt
Valor	20	20	10 años	0-5 años	100 min

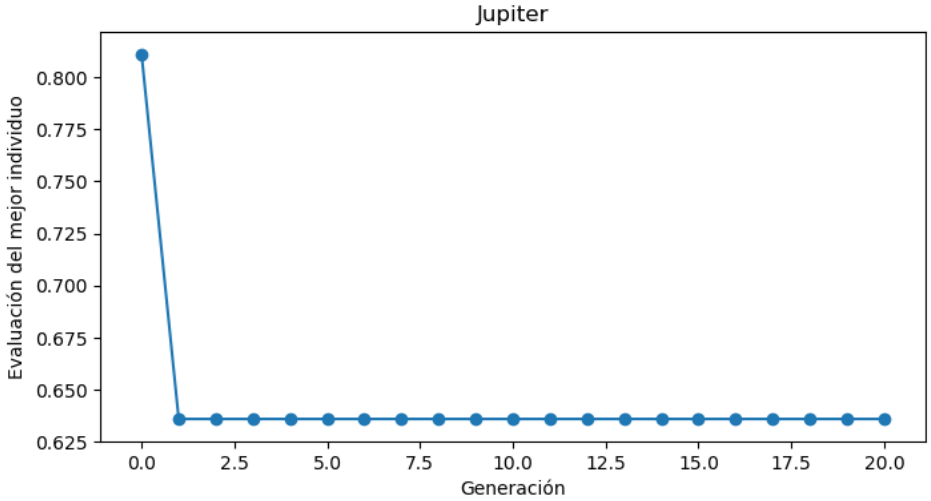

Figura 10: Gráfica que compara el resultado de la evaluación para el mejor individuo por generación contra su generación, desde la segunda generación los resultados comienzan a ser constantes.

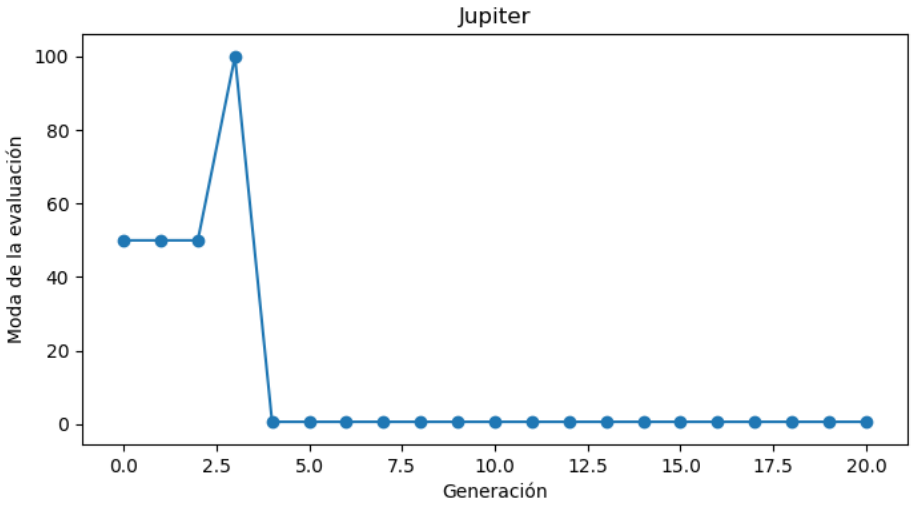

Figura 11: Gráfica que compara el resultado de la evaluación para la moda de la generación, los resultados son similares a los de la gráfica del mejor individuo, esto indica que se cumple que los mejores individuos son los que dejan mayor descendencia, exceptuando las primeras generaciones.

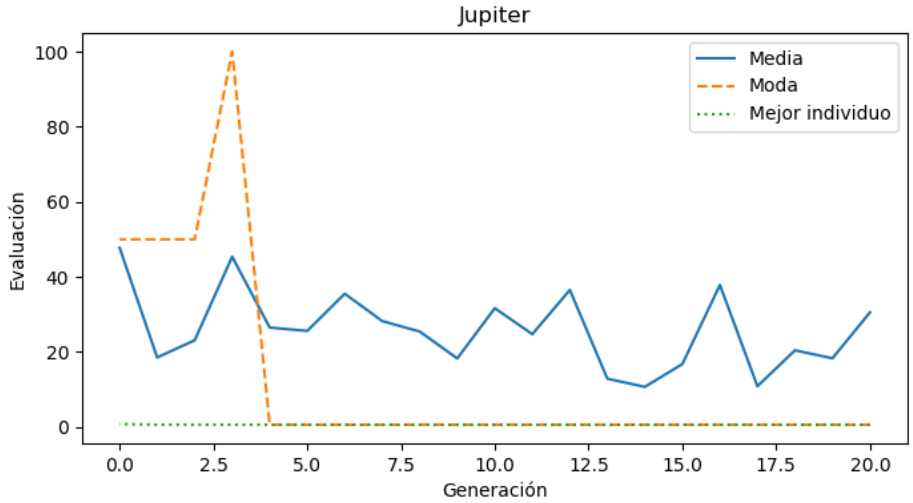

Figura 12: Comparación del mejor individuo, la moda y la media de cada generación.

Conclusiones

Los algoritmos genéticos son una herramienta potente para problemas de optimización. Sin embargo, dado que se necesita realizar repetidamente la evaluación para cada uno de los individuos y para varias generaciones, el tiempo de computo puede llegar a ser insostenible, todo depende de la complejidad en tiempo del algoritmo y principalmente de la función de evaluación. En el caso de la implementación de este problema, se almacenaron en memoria durante el tiempo de ejecución los datos conforme se iban evaluando, para el momento en el que se tenían cerca de 2000 individuos se utilizaban cerca de 12 Gb, una buena optimización a futuro sería hacer persistentes los datos conforme va terminando cada generación, así se evitaría utilizar toda la RAM del equipo.

Es aconsejable utilizar los algoritmos genéticos cuando los métodos «precisos» no se puedan implementar correctamente , aunque los AG pueden optimizar, no siempre encuentran los mínimos o máximos absolutos, esto es algo a tomar en cuenta a la hora de solucionar un problema, hay métodos que ayudan a no estancarse en puntos críticos relativos y de ser posible deben implementarse, pero si se puede hacer un mapeo o encontrar de forma analítica dichos puntos deberían contemplarse estas opciones sobre los AG.

Referencias

Gestal, M., Rivero, D., Rabuñal, J. R., Dorado, J., and Pazos, A. (2010). Introducción a los algoritmos genéticos y la programación genética. *A Coruña*, 2010:30–68.

José, J. and Saletan, E. (2000). Classical dynamics: a contemporary approach.

Méndez Hincapié, N. F. and Monroy Cañón, I. A. (2016). Diseño de un programa en python para la enseñanza de la transferencia de órbita de hohmann. *Tecné, Episteme y Didaxis: TED*, (39):81–102.