Esteban Quesada Quesada B96157

Explicaciones especiales sobre implementaciones de particular interés en el código fuente, con referencias a la manera en que se implementó el pseudo-código del paper

En primer lugar, se crea un *struct* llamado *Particula*, el cual guarda todos los atributos de una de ellas, es decir la velocidad en *x* y en *y*, las posiciones, tanto en el eje *x* como en el *y*, asimismo el *fitness* actual luego de evaluar la partícula. Y por último un registro de la mejor posición encontrada, así como el mejor *fitness* local de esa partícula.

Posteriormente en un método denominado *funcionObjetivo* se coloca la función para poder evaluar cada partícula y determinar el fitness de cada una de las 5 partículas. Seguidamente se encuentra la función *asignarParametrosIniciales,* mediante la cual se define una posición inicial para cada una de las partículas dentro del rango *[-1000,1000] x [-1000,1000],* asimismo, se inicializa la velocidad de cada una de las partículas dentro de un rango *[-10,10]*. La mejor posición local de cada partícula al igual que el mejor *fitness* se inicializan con las posiciones actuales de la primer partícula, así como el *fitness* actual de la primer partícula (la cual también se inicializa en esta función con ayuda de la *funcionObjetivo*).

Asimismo, se cuenta con una función denominada exceso de límites, la cual se encarga de localizar aquellas posiciones de cada partícula que exceden el dominio *[-1000,1000].* La técnica empleada es alusiva al juego de Pacman, en el cual cada vez que se llega al límite por un lado el personaje sale por el lado contrario.

El pseudocódigo construido con base en referencias del paper y adaptaciones es el siguiente:

Algoritmo PSO
Criterio de parada, mientras que no se cumple hacer (10000 iteraciones del ciclo más extern o, o cuando la función objetivo permanezca sin disminuir en 0.001 o más durante 50 iteraci ones consecutivas):
    Imprimir # de iteración, posición de la mejor solución global G y su valor de la funció n, coordenadas de cada una de las partículas Xi con 5 decimales y valor de la función objetivo para esa partícula con 7 decimales
    Evaluar cada partícula con la función objetivo
    Cada partícula:
        Actualizar posición y velocidad de cada partícula
            *Velocidad de cada particula x_i & y_i:
                Fórmula 2 del paper se emplea
            *Posicion de cada particula x_i & y_i (xi(t+1) = xi(t) + vi(t+1))
                Fórmula 1 del paper se emplea
            *Rebotar posiciones en caso de irrespetar domino

        Evaluar nueva posición y velocidad de cada partícula
        Actualizar su mejor personal si f(G local) > f(nueva local)
        Actualizar el global del sistema, si f(G) > f(particula i), entonces f(G) = f(particula)
        Se comprueba el registro de distancia entre soluciones para observar si es mayor a 0.001
Dar el mejor global obtenido (Al detenerse, se debe imrimir el mensaje "Solucion final: " y las coordenadas y valor de la mejor solución encontrada hasta ese momento.)

Resultados Encontrados

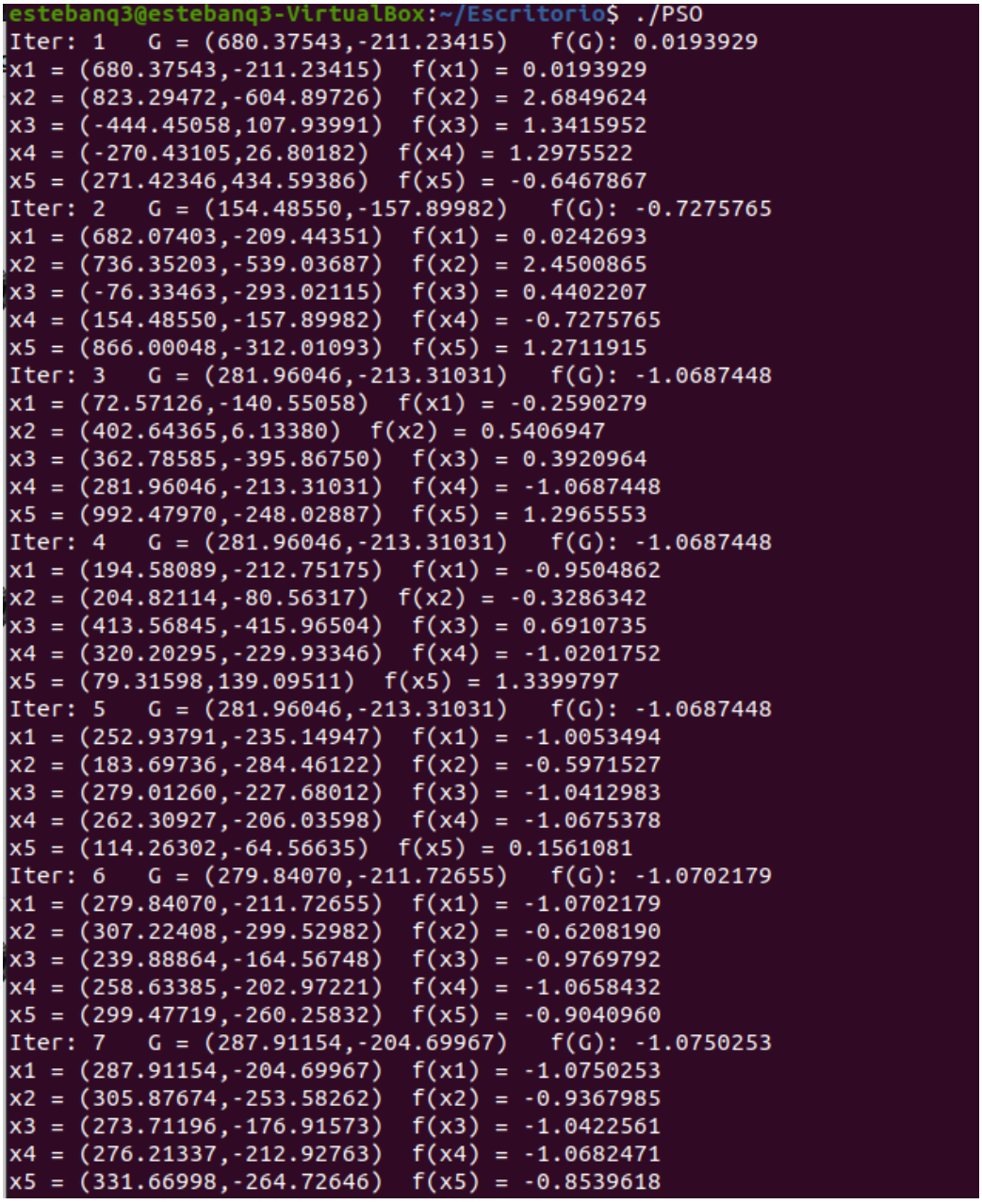

Figura 1: *Muestra de los resultados de las primeras 7 iteraciones.*

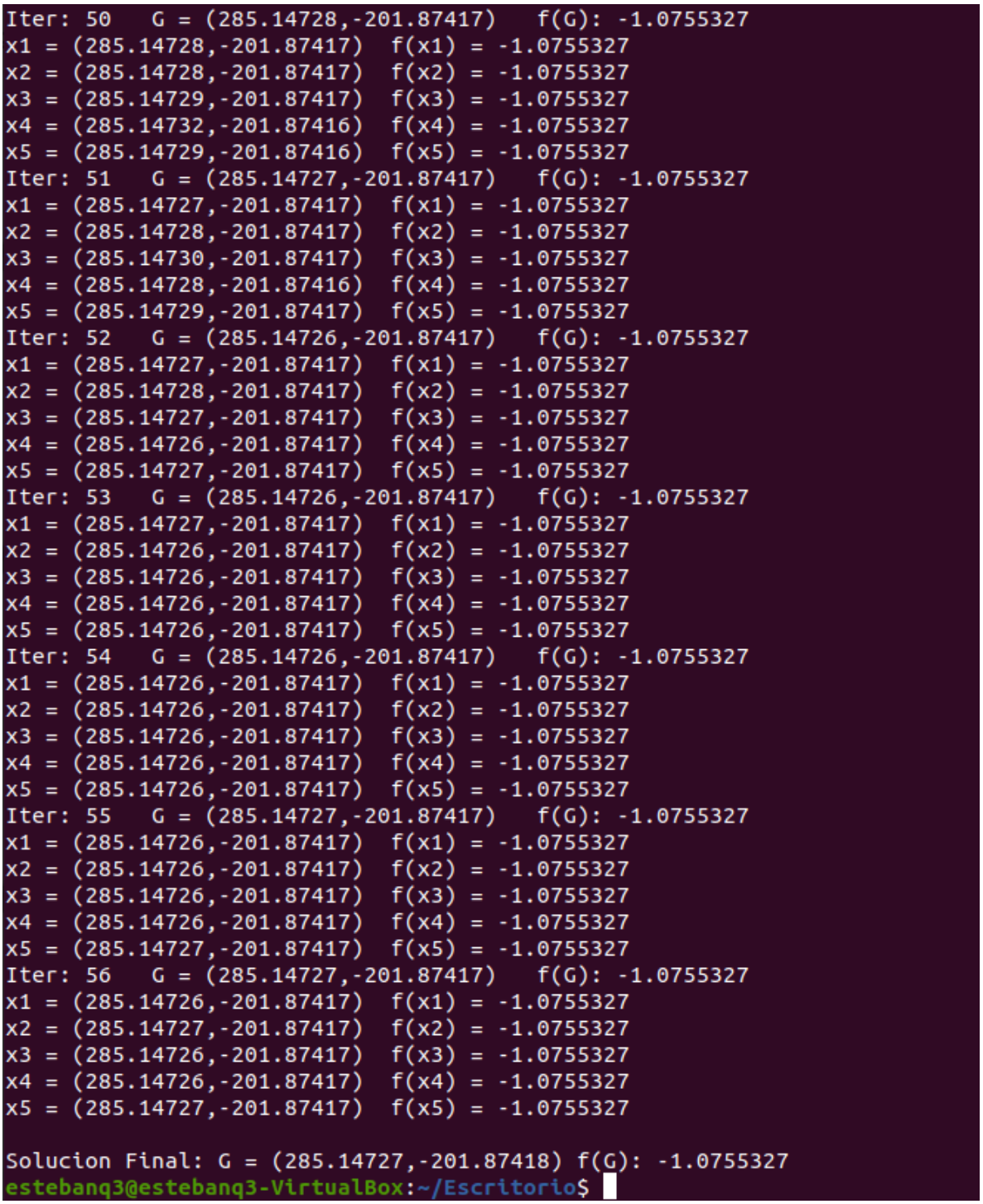

Figura 2: *Muestra de los resultados de las últimas 6 iteraciones y la solución final.*

Valoración personal sobre la calidad de la solución

En cuanto a probar la exactitud de la solución intenté buscar alguna calculadora de mínimos locales de la función otorgada, más ninguna pudo realizar el cálculo y mostrarme el mínimo correctamente, por lo tanto, a ciencia cierta no tengo la completa seguridad de que tal solución sea el mínimo local, asimismo, el algoritmo quizás se pudo haber estancado en un mínimo local y no uno global. Por otra parte, también afecta la condición de que la generación de números aleatorios siempre es la misma (*srand* con la misma semilla siempre) a la hora de querer generar distintos valores a los de siempre, mas, sin embargo, tomando en cuenta que el algoritmo funciona correctamente pensaría que la solución obtenida no se aleja mucho de la solución real debido a que se utilizaron ecuaciones bien planteadas y estudiadas, asimismo, el algoritmo tiene un buen planteamiento.

Línea de comando para compilar

gcc -o PSO PSO.c -lm

El *-lm* es utilizado debido a que se necesita hacer uso de la biblioteca math.h del lenguaje C.

Dificultades Encontradas durante la realización de esta parte

Quizás fue un poco difícil entender el algoritmo en primera instancia y construir un algoritmo mejor detallado al del paper, mas existía bastante información y tiempo para realizar las lecturas e investigaciones necesarias, así como la implementación del mismo, por lo tanto, no existieron muchas complicaciones en realidad y se aprendió bastante sobre el PSO.