Implementación de SFLA para la navegación autónoma de robots móviles

Sánchez Flores Saúl Isaac
Universidad de Guanajuato, Guanajuato, Gto. México
si.sanchezflores@ugto.mx

Resumen — Se utilizó una implementación del algoritmo bio-inspirado conocido como SFLA para realizar la planeación de ruta de un robot móvil a través de un mapa de obstáculos. En cada iteración, la rana obtenida por el algoritmo, que utiliza los sensores instalados en el robot como entrada, es utilizada como la siguiente posición intermedia a alcanzar.

Palabras clave— Robot móvil, Planeación de rutas, navegación reactiva.

I. INTRODUCCIÓN

La presencia de robot móviles, y otros tipos de agentes móviles se ha incrementado en los últimos años en distintos ámbitos ya sea industriales, de transporte, agricultura, e incluso en la vida diaria de las personas en hoteles y hospitales. Debido a su capacidad para navegar en entornos peligrosos, o a través de almacenes complejos, el desarrollo de soluciones para la generación de rutas de navegación en entornos dinámicos se ha convertido en un tema de estudio bastante popular.

El problema de la planeación de rutas consiste, en esencia, en generar una ruta libre de colisiones en un entorno, mientras se consideran distintos criterios de optimización. En este trabajo se propone el uso de un robot bio-inspirado para la generación de rutas a partir de información local recogida por el robot o agente móvil. Conocido como *Shuffled Frog Leaping Algorithm*, o por sus siglas SFLA, es un algoritmo de optimización que mimetiza el comportamiento de caza de las ranas. Este algoritmo se ha utilizado, por su versatilidad y eficiencia, en otros problemas de ingeniería, que van desde la optimización de dispositivos electromagnéticos [1], ingeniería civil, y el diseño de dispositivos inalámbricos[2].

II. TRABAJOS RELACIONADOS

Es posible encontrar distintos trabajos que trabajan de manera de manera similar en la generación de rutas de navegación. Específicamente hablando del uso de algoritmos bio-inspirados, existen propuestas como el uso de redes neuronales [3], algoritmos genéticos (GA) [4], y algoritmos de optimización de partículas (PSO) [5]. Si bien estos métodos pueden resultar en la obtención de rutas optimizadas libres de obstáculos, estos requieren de información precisa del entorno a resolver. Con SFLA, un modelo detallado del entorno es innecesario, pues es posible generar la ruta haciendo uso de la información recopilada de manera local. Una desventaja de esto es la posibilidad de no alcanzar la ruta más eficiente, la cual podría ser alcanzada con otros métodos.

Tanto Hassanzadeh [6] como Jianjun [7] han utilizado SFLA anteriormente para la elaboración de rutas, y se han recogido elementos de los trabajos de ambas investigaciones para la realización de este trabajo.

III. DESCRIPCIÓN DE SFLA

Este algoritmo, una modificación al algoritmo propuesto por Eusuff y Lansey [8], es un método de enjambre basado en la población. En esencia, combina los beneficios de los algoritmos genéticos, y los comportamientos sociales de los algoritmos de optimización de partículas o PSO [9].

El principio de funcionamiento básico del algoritmo es como sigue:

A. Inicialización

Se genera una población inicial de ranas en el espacio de posibles soluciones. Cada rana representa una posible solución en este espacio.

B. Partición

Una vez generada la población inicial, esta se ordena de manera descendente, según su valor de fitness, y dividida en grupos denominados memeplexes, de manera similar a la que se reparte una baraja, es decir cada grupo recibe una rana a la vez de manera cíclica.

C. Actualización

El objetivo es cambiar la posición de la peor rana de cada grupo, actualizándola al utilizar la información de la mejor rana en su grupo según la siguiente formula:

$$X_w^{new} = X_w + rand(\ \ ) * (X_b - X_w)$$

Donde X_w es la peor rana de cada grupo y X_b la mejor, mientras que rand() representa un numero aleatorio entre 0 y 1. En caso de que este proceso no genera una mejor rana, la mejor rana de toda la población, X_g, ocupará su lugar.

D. *Shuffling*

Una vez concluida la búsqueda local, se desea realizar una búsqueda global en la población de ranas. Todas las ranas son mezcladas, y el proceso de partición y actualización vuelven a ser repetidos hasta que la condición de convergencia se alcance.

IV. DESARROLLO

Se implementó el algoritmo siguiendo las siguientes consideraciones.

- El robot posee información sobre su posición y la de su objetivo, además de las posiciones recogidas a través de sus sensores durante la ejecución.
- Al igual que en el trabajo de Hassanzadeh [6], las coordenadas obtenidas se asumen como puntos, es decir, no se toma en cuenta la forma y tamaño de los obstáculos, lo cual puede afectar la seguridad del robot.

Se desarrolló un mapa basado en potholes de Maciej Kalisiak, en el cual distintos obstáculos se encuentran dispersos en una superficie, la cual se estableció de 20x20m.

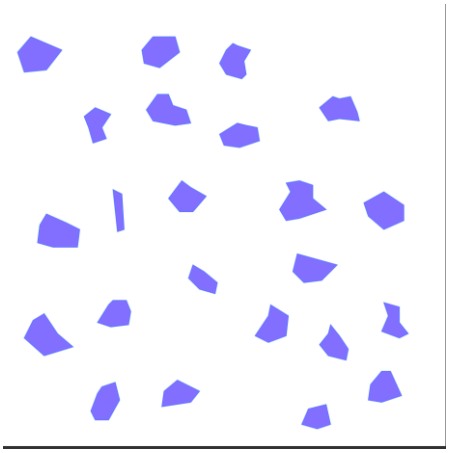

Figura 1 Mapa de obstáculos

Es posible considerar la generación de rutas como un problema de optimización, en el cual la función a reducir es la función fitness, la cual evalúa a las ranas generadas por el algoritmo. Se recoge la propuesta por Jianjun[7].

$$f(X_i) = w_1 * e^{-min\left|\left|X_i - O_j\right|\right|} + w_2 * ||Xi - T||$$

Donde X_i es la rana por evaluar, X_i-O_j es la distancia de la rana al obstáculo más cercano y X_i-T es la distancia de la rana al objetivo. Finalmente, w_1 y w_2 son los pesos que reciben ambos parámetros. Un valor elevado para w1 provocará que la ruta se aleje de los obstáculos, mientras un valor elevado de w_2 acelerará el acercamiento hacia el objetivo.

V. RESULTADOS OBTENIDOS

Para verificar el funcionamiento del algoritmo, este fue implementado en Python, de dos maneras.

En la primera, utilizando todas las posiciones de los obstáculos, es decir, el algoritmo conoce la ubicación inicial del robot, el objetivo y todos los obstáculos. Se obtuvieron los siguientes resultados.

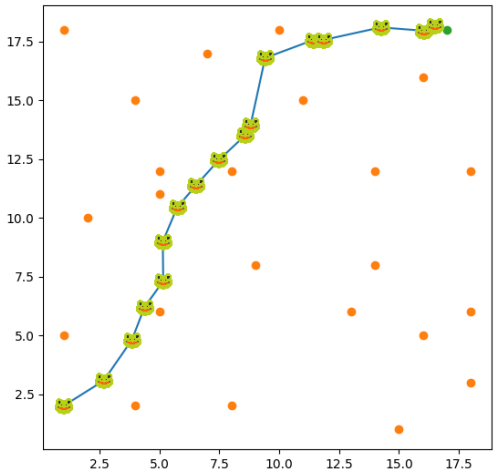

Figura 2 Simulación teórica 1

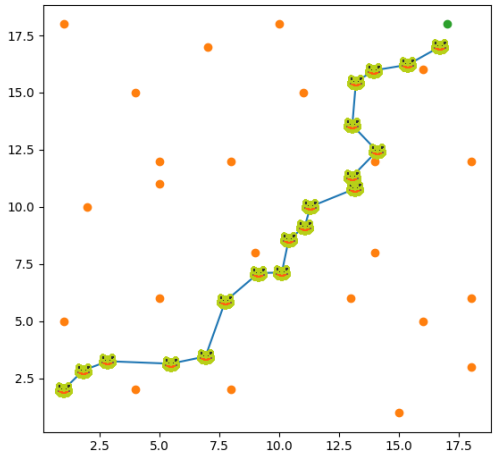

Figura 3 Simulación teórica 2

Tanto en la figura 2 y 3 los puntos representados por naranja representan las posiciones por los obstáculos. Por otro lado, las ranas representan los puntos intermedios de la ruta generada por el algoritmo.

Los parámetros utilizados se presentan en la siguiente tabla.

Parámetro	Valor	Descripción
F	30	No. de ranas
m	6	No. de grupos
n	5	No. de ranas en cada grupo
L	10	No. de generaciones para cada grupo
G	100	No. de iteraciones para mezclar
w_1	7	Peso de los obstáculos
w_2	12	Peso del objetivo

Estos mismos parámetros se utilizaron para la simulación con Coppelisim. En dicha simulación una vez más se definió el comportamiento mediante Python y la API remota de Coppeliasim.

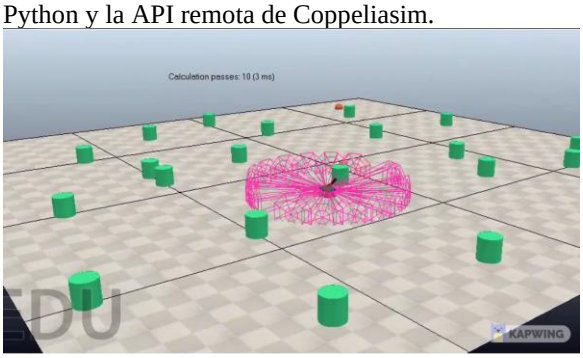

Figura 4 Simulación con el robot Pioneer

El robot simulado fue un Pioneer P3DX. Se utilizó el algoritmo de Braitenberg de manera auxiliar para corregir la cercanía de la ruta a los obstáculos, y se estableció la lectura de los sensores ultrasónicos como única entrada del algoritmo. La ruta obtenida por el robot fue la siguiente.

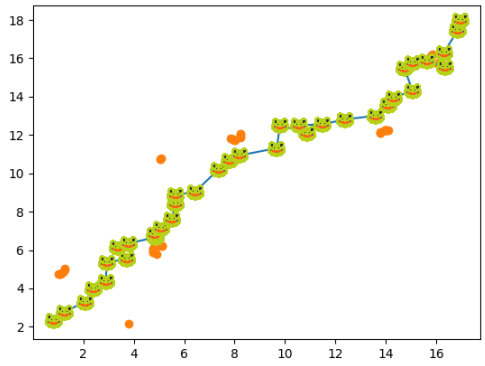

Figura 5 Ruta obtenida a través de los obstáculos localizados.

En la figura 3 es posible apreciar la diferencia de la ruta generada, a partir de los obstáculos localizados por los sensores a bordo del robot.

A. Tiempo de Ejecución

Se realizaron tres pruebas para medir el tiempo de ejecución promedio del algoritmo en el peor escenario, esto es, alimentando al algoritmo la lista de todos los obstáculos en el mapa.

Prueba	**Tiempo de Ejecución**	**Distancia Recorrida**
1	26.91s	26.09m
2	38.20s	25.62m
3	27.44s	23.13m
Promedio	30.85s	24.94m

Dichos resultados permiten apreciar el tiempo utilizado el algoritmo para calcular todas las posiciones que conforman la ruta en promedio. Adicionalmente la distancia promedio al ser relativamente corta confirma la efectividad del algoritmo al acercarse de manera heurística a una solución global.

VI. Conclusión

El algoritmo propuesto para la generación rutas es capaz de encontrar rutas viables de manera solo exitosa, sino también de manera rápida en tiempo real. A diferencia de otros algoritmos, SFLA no requiere de un modelo preciso del entorno a navegar, siendo capaz de generar soluciones bastante buenas a partir de la información local.

Los resultados obtenidos dan espacio a la mejora, la cual puede ser obtenida al afinar los valores de los parámetros utilizados.

VII. Bibliografía

[1] W. Yang, S. L. Ho, and W. Fu, “A modified shuffled frog leaping algorithm for the topology optimization of electromagnet devices,” *Appl. Sci.*, vol. 10, no. 18, 2020, doi: 10.3390/APP10186186.

[2] S. Swayamsiddha, S. Parija, S. S. Singh, and P. K. Sahu, “Bio-inspired algorithms for mobile location management—a new paradigm,” *Adv. Intell. Syst. Comput.*, vol. 516, no. Mlm, pp. 35–44, 2017, doi: 10.1007/978-981-10-3156-4_4.

[3] L. Wang, S. X. Yang, and M. Biglarbegian, “Bio-inspired navigation of mobile robots,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7326 LNAI, pp. 59–68, 2012, doi: 10.1007/978-3-642-31368-4_8.

[4] C. Lamini, S. Benhlima, and A. Elbekri, “Genetic algorithm based approach for autonomous mobile robot path planning,” *Procedia Comput. Sci.*, vol. 127, pp. 180–189,

2018, doi: 10.1016/j.procs.2018.01.113.

[5] J. M. Hereford, M. Siebold, and S. Nichols, “Using the particle swarm optimization algorithm for robotic search applications,” *Proc. 2007 IEEE Swarm Intell. Symp. SIS 2007*, no. Sis, pp. 53–59, 2007, doi: 10.1109/SIS.2007.368026.

[6] I. Hassanzadeh, K. Madani, and M. A. Badamchizadeh, “Mobile robot path planning based on shuffled frog leaping optimization algorithm,” *2010 IEEE Int. Conf. Autom. Sci. Eng. CASE 2010*, pp. 680–685, 2010, doi: 10.1109/COASE.2010.5584758.

[7] J. Ni, X. Yin, J. Chen, and X. Li, “An improved shuffled frog leaping algorithm for robot path planning,” *2014 10th Int. Conf. Nat. Comput. ICNC 2014*, no. 1, pp. 545–549, 2014, doi: 10.1109/ICNC.2014.6975893.

[8] P. Luo, Q. Lu, and Chenxi Wu, “Modified shuffled frog leaping algorithm based on new searching strategy,” *Proc. - 2011 7th Int. Conf. Nat. Comput. ICNC 2011*, vol. 3, no. 1, pp. 1346–1350, 2011, doi: 10.1109/ICNC.2011.6022273.

[9] E. Elbeltagi, T. Hegazy, and D. Grierson, “Comparison among five evolutionary-based optimization algorithms,” *Adv. Eng. Informatics*, vol. 19, no. 1, pp. 43–53, 2005, doi: 10.1016/j.aei.2005.01.004.