UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Aplicaciones prácticas para algoritmos de robótica de enjambre

Protocolo de trabajo de graduación presentado por Daniela María Baldizón García, estudiante de Ingeniería Mecatrónica

Guatemala,

2021

Resumen

Antecedentes

Implementación de Algoritmos de Inteligencia de Enjambre

En la fase anterior a este trabajo [1] se implementó el algoritmo *Ant System* (AS) (o *Ant Colony* (ACO)) como planificador de trayectorias. Dicho algoritmo se basa en el comportamiento de las hormigas para buscar y hallar alimento. Con esta implementación se obtuvo una alternativa de planificación de trayectorias a parte de la del *Modified Particle Swarm Optmimization* (MPSO), desarrollada en el proyecto Robotat [2]. Estos algoritmos de inteligencia de enjambre tienen diferentes enfoques, el de AS es en búsqueda de grafos y el del MPSO es en funciones de costo.

Se encontraron los parámetros para el correcto funcionamiento del AS y se validaron por medio de simulaciones computarizadas que permiten visualizar el comportamiento de las feromonas depositadas por la colonia. Se adaptaron los modelos de movimiento y de cinemática de los Bitbots de UVG al ACO, y así se logró realizar la implementación en Webots para comparar su desempeño con el del MPSO.

Se implementó distintos controladores para considerar que los motores que controlan al robot tienen un límite de velocidad y garantizar que este haga lo deseado. Entre los controladores que se implementaron están: Transformación de uniciclo (TUC), PID de velocidad lineal y angular, PID de acercamiento exponencial, de pose, de pose de Lyapunov, Closed-loop steering, Regulador cuadrático lienal (LQR) y Integrador cuadrático linal (LQI). Se concluyó que los modelos de movimiento de los Bitbots desarrollados en el proyecto Robotat [2] funcionan tanto en AS como en MPSO.

Se realizaron pruebas con computación paralela para agilizar la planeación de trayectorias de los robots con el Ant System. También se realizaron pruebas con algoritmos genéticos para explorar si estos pueden ser alternativa al AS y MPSO, pero los resultados no fueron satisfactorios.

Robotarium de Georgia Tech

El proyecto de Robotarium del Tecnológico de Georgia en Estados Unidos proporciona una plataforma de investigación de robótica de enjambre accesible de forma remota de libre acceso. El objetivo de este proyecto es que cualquier persona pueda cargar y probar sus ideas en un hardware robótico real, y así librar las inversiones significativas en mano de obra. Para utilizar el Robotarium solo se necesita descargar el simulador de MATLAB o Python y registrarse en la página para recibir la aprobación de la administración del Robotarium [3].

Justificación

Objetivos

Objetivo General

Implementar algoritmos de inteligencia de enjambre para la realización de tareas prácticas.

Objetivos Específicos

- Implementar un algoritmo de inteligencia de enjambre para exploración y planificación de trayectorias, tomando en cuenta obstáculos que puedan presentarse dentro de las trayectorias.
- Implementar un algoritmo de inteligencia de enjambre para la recolección de muestras especificadas.
- Validar las implementaciones de los algoritmos mediante simulaciones en webots.

Marco teórico

Particle Swarm Optimization (PSO)

Es un algoritmo de optimización meta-heurístico ya que utiliza analogías con otros procesos para resolver un problema. El algoritmo se inspira en la evolución del comportamiento colectivo, trata de imitar el comportamiento social de grupos de animales como manadas, cardúmenes, parvadas, etc. Los métodos meta-heurísticos no se enfocan en resolver un problema en particular, por lo que estos pueden emplearse en cualquier problema y obtener un resultado aceptable. El PSO no es determinista, lo que significa que los resultados obtenidos no siempre serán los mismos aunque se trate de una misma función. Los algoritmos de enjambre de partículas se caracterizan por ser eficientes y de bajo costo computacional [4].

A lo largo del tiempo se han propuesto variaciones para el PSO, esta sección se enfoca en la versión inercial o clásica. El objetivo de un problema de optimización es encontrar un vector $X = [x_1, x_2, x_3, \ldots, x_n]$ que minimice o maximice cierta función f(X). El vector variable X es el vector de posición que representa un modelo variable y es de dimensión n, donde n es la cantidad de variables que hay que encontrar en el problema. La función f(x) es la función objetivo, esta evalúa que tan bien o mal es la posición X [4].

Tomando un enjambre con P cantidad de partículas, hay un vector posición $X_i^t = (x_{i1}, x_{i2}, x_{i3}, \ldots, x_{in})^T$ y un vector velocidad $V_i^t = (v_{i1}, v_{i2}, v_{i3}, \ldots, v_{in})^T$ a una t-iteración para cada una de las partículas que compone al enjambre. Estos vectores son actualizados a través de una dimensión j de acuerdo a las siguientes ecuaciones [4]:

$$V_{ij}^{t+1} = wV_{ij}^t + c_1 r_1^t \left(pbest_{ij} - X_{ij}^t\right) + c_2 r_2^t \left(g \text{ best }_j - X_{ij}^t\right) \tag{1}$$

$$X_{ij}^{t+1} = X_{ij}^{t} + V_{ij}^{t+1} \tag{2}$$

Donde $i = 1, 2, \ldots, P$ y $j = 1, 2, \ldots, n$.

La ecuación (1) actualiza la velocidad y (2) actualiza la posición de las partículas. w es la constante de inercia peso, para búsquedas globales o exploraciones este parámetro es alto, y para búsquedas locales o explotación este parámetro es bajo. El primer término de la ecuación (1) es la influencia del movimiento anterior en el actual. Depende de w que tanto el movimiento anterior de la partícula influencia al movimiento actual. El segundo término representa la cognición individual y es calculado por medio de la diferencia entre la mejor posición de la propia partícula (*pbest*) y su posición actual. La idea detrás de este término es que mientras la partícula está más lejos de su mejor posición la diferencia aumenta; por consecuencia, el término aumenta atrayendo a la partícula a su mejor posición. El parámetro c_1 multiplica al término como una constante positiva y es un parámetro individual-cognición, y marca la importancia de las experiencias anteriores de cada partícula. Otro término que compone el producto del segundo término es r_1, y este es un parámetro de valor aleatorio con rango [0,1]. Este valor evita las convergencias prematuras [4].

El tercer término de (1) es el de aprendizaje social. Esto es así debido a que todas las partículas del enjambre pueden intercambiar información sobre el mejor punto alcanzado sin importar cual lo encontró (*gbest*). La diferencia que se muestra en este término actúa como atractor para las partículas al mejor punto global encontrado en una iteración t. Similarmente, c_2 es el parámetro de aprendizaje social, y su valor representa la importancia del aprendizaje global del enjambre. Y r_2 tiene el mismo rol que r_1 [4].

Ant Colony Optimization (ACO)

Uno de los comportamientos que han estudiado los entomólogos es la habilidad que tienen las hormigas de encontrar el camino más corto entre su hormiguero y las fuentes de alimento. A partir de estos estudios y observaciones, Marco Dorigo desarrolló el primer algoritmo que modela el comportamiento de las hormigas al buscar alimento, dicho algoritmo fue llamado como Ant Colony Optimization Meta-Heuristic (ACO-MH). Después de esto, se han desarrollado numerosas variantes para dicho algoritmo, como lo son el Ant System (AS), Ant Colony System (ACS), Max-Min Ant System (MMAS), Ant-Q, Fast Ant System, Antabu, AS-Rank y ANTS [5].

Las hormigas reales empiezan la búsqueda de alimento comportándose inicialmente de forma aleatoria o con un patrón de actividad caótica. Al encontrar una fuente de alimento, los patrones de actividad se vuelven más organizados con más hormigas siguiendo el mismo camino hacia la fuente de alimento. Lo que ocurre es que las hormigas que encuentran el alimento influencian a las demás hacia la comida, este comportamiento es resultado del mecanismo de reclutamiento de las hormigas. La mayoría de las especies de hormigas utilizan como forma de reclutamiento la comunicación indirecta, la cual hacen por medio de rastros de feromonas y se denomina *stigmergy*. Cuando una hormiga encuentra una fuente de alimento, esta regresa al hormiguero por el mismo camino llevando alimento, al hacer esto deja rastros de feromonas en el camino para indicar a las demás hormigas qué camino seguir. Las hormigas recolectoras deciden qué camino seguir basándose en las concentraciones de

feromonas de los diferentes caminos. Los caminos con mayor concentración de feromonas tienen una mayor probabilidad de ser elegidos. Mientras más hormigas sigan el mismo camino, aumenta la concentración de feromonas, por lo que este camino es el que más será utilizado por el conjunto de hormigas. El comportamiento colectivo resultante es una forma de comportamiento auto-catalítica, donde la retroalimentación positiva de un camino hacia el alimento causa que este camino sea seguido por las demás hormigas [5].

Deneubourg estudió el comportamiento de las hormigas por medio de un experimento, en el que el hormiguero está separado de la fuente de alimento por un puente con dos caminos de la misma longitud. Al pasar el tiempo, las hormigas eligieron de manera aleatoria uno de los dos caminos para que fuera la ruta hacia la fuente de alimento. Este experimento es conocido como "El puente binario". Goss extendió el experimento del puente binario haciendo que uno de los caminos fuera más largo que el otro. Las hormigas seleccionaron el camino más corto como su ruta hacia el alimento. Como las hormigas que van por el camino corto regresan antes al hormiguero, la cantidad de feromonas en este camino es reforzada antes que en el camino largo. La probabilidad de las hormigas de escoger el camino más corto aumenta al incrementar la diferencia del largo de los caminos, este efecto se conoce como "largo diferencial" [5].

Los caminos más cortos tendrán una mayor concentración de feromonas al transcurrir el tiempo, esto se debe a que las hormigas regresan antes por esos caminos. Las feromonas se evaporan con el tiempo, y como consecuencia de esto las concentraciones de feromonas decrecen más rápido en los caminos largos. La feromona artificial imita las características de la feromona real, indica la popularidad de una solución del problema de optimización que se está resolviendo. De hecho, la feromona artificial funciona como una memoria a largo plazo del proceso completo de la búsqueda [5].

Simple Ant Colony Optimization (SACO)

El SACO es una implementación algorítmica del experimento del puente binario de Deneubourg. Considerando el problema general de encontrar el camino más corto entre dos nodos en un grafo $G = (V, E)$, donde V es el conjunto de vértices o nodos y E es la matriz que representa la conexión entre los nodos. El largo L^k del camino construido por la hormiga k es calculado como la cantidad de saltos en el camino desde el nodo que representa al nido hasta el que representa el destino con la comida. En la Figura 1 se muestra un ejemplo de un grafo y el camino seleccionado. El camino elegido está marcado con flechas continuas y su longitud es 2. Cada arista (i, j) del grafo tiene una concentración $\tau_{i,j}$ asociada [5].

En el algoritmo, a cada arista se le asigna un valor aleatorio pequeño de feromona ($\tau_{ij}(0)$) (Aunque en la realidad esto debería ser cero). Luego, las hormigas deciden de forma aleatoria que camino/arista seguir. k hormigas se colocan en el nodo fuente. En cada iteración del algoritmo cada hormiga construye un camino (solución) hacia el nodo de destino. En cada nodo i, cada hormiga k selecciona el siguiente nodo j, basándose en la probabilidad de transición,

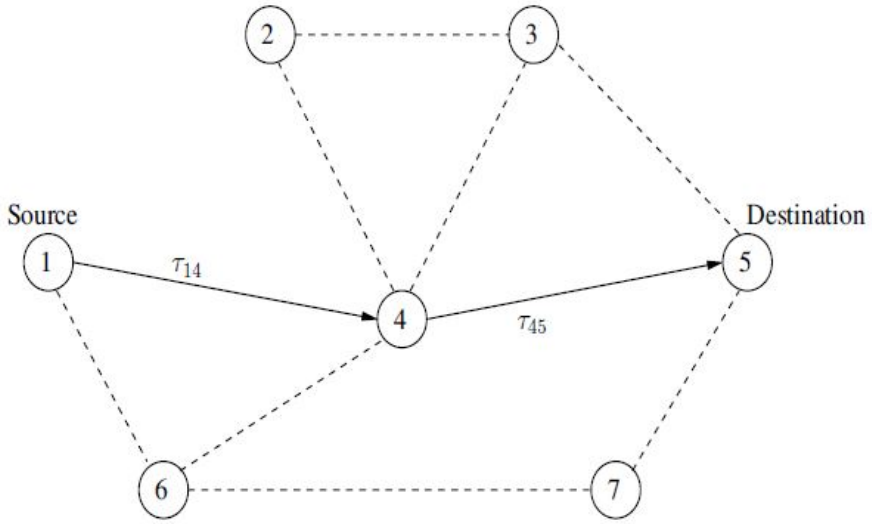

Figura 1: Ejemplo de grafo para el camino más corto [5]

$$p_{ij}^k(t) = \begin{cases} \frac{\tau_{ij}^\alpha(t)}{\sum_{j \in \mathcal{N}_i^k} \tau_{ij}^\alpha(t)} & \text{if } j \in \mathcal{N}_i^k \\ 0 & \text{if } j \notin \mathcal{N}_i^k \end{cases} \tag{3}$$

donde $\mathcal{N}_i^k$ es el conjunto de nodos viables conectados al nodo i, respecto a la hormiga k. Si se llega a dar el caso $\mathcal{N}_i^k = 0$, entonces el nodo anterior se incluye al conjunto. α es una constante positiva que amplifica la influencia de las concentraciones de feromona, es decir, para valores grandes de α la importancia de la feromona es grande, especialmente para los valores iniciales aleatorios de feromona, lo cual puede hacer que la convergencia sea rápida y resulte en caminos no óptimos [5].

Una vez todas las hormigas han construido un camino completo desde el nodo de origen al nodo de destino, cada hormiga vuelve a recorrer su camino hacia el nodo fuente, y deposita una cantidad de feromona,

$$\Delta\tau_{ij}^k(t) \propto \frac{1}{L^k(t)} \tag{4}$$

en cada arista (i, j) del correspondiente camino. $L^k(t)$ es el largo del camino construido por una hormiga k en el tiempo t. De esta forma, considerando que la feromona puede evaporarse y así evitar la convergencia temprana y no óptima, la cantidad de feromona en cada arista está dada por

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{k=1}^{n_k} \Delta\tau_{ij}^k(t) \tag{5}$$

donde n_k es la cantidad de hormigas. La constante $\rho \in [0,1]$ y especifica la tasa a la que las feromonas se evaporan, causando que las hormigas "olviden" las decisiones anteriores.

Para valores grandes de ρ la feromona se evapora más rápido. Mientras más feromonas se evaporen, la búsqueda se vuelve más aleatoria [5].

Se sabe que

$$\Delta\tau_{ij}^{k}(t) = \frac{Q}{L_k}(t) \tag{6}$$

donde Q es una constante y L es el costo del trayecto, como el largo de este. El resultado de esto representa el cambio de feromona entre el nodo i y j que la hormiga visitó en la iteración t [6].

Ant System (AS)

Este algoritmo es una mejora del SACO (A pesar que el AS se desarrolló antes). La probabilidad de transición cambia para incluir la información heurística y agregar capacidad de memoria con una lista tabú, la cual previene que una hormiga visite el mismo nodo dos veces [6]. La ecuación de probabilidad de transición es,

$$p_{(i,j)}^{k}(t) = \begin{cases} \frac{(\tau_{ij}(t))^{\alpha}\cdot(\eta_{ij}(t))^{\beta}}{\sum_{k\in J_k}(\tau_{ij}(t))^{\alpha}\cdot(\eta_{ij}(t))^{\beta}} & \text{si j} \in N_i^k \\ 0 & \text{si j} \notin N_i^k \end{cases} \tag{7}$$

donde η_{ij} representa la efectividad *a priori* del movimiento desde i a j, o el inverso del costo de la arista [5].

Por lo tanto,

$$\eta_{ij} = \frac{1}{d_{ij}} \tag{8}$$

donde d_{ij} es la distancia (o costo) entre los nodos i y j[5].

Aplicaciones del Ant Colony Optimization

Los algoritmos de ACO se utilizan normalmente para resolver problemas no determinísticos, los cuales son aquellos que no se pueden solucionar por algoritmos con estructura polinomial, por lo que requieren del tipo exponencial. Para la solución, se realizan iteraciones con las que se pueden encontrar respuestas globales y particulares [7].

Algunos campos de aplicación de los algoritmos por optimización por colonia de hormigas son: redes neuronales, inteligencia artificial, optimización de funciones numéricas, sistemas difusos, procesamiento de imágenes, control de sistemas, problemas del hombre viajero, enrutamiento de vehículos y líneas de producción de carros [7].

Metodología

Cronograma de actividades

Índice preliminar

Referencias

[1] G. Iriarte, "Aprendizaje Automático, Computación Evolutiva e Inteligencia de Enjambre para Aplicaciones de Robótica," en *Trabajo de Graduación, Modalidad Tésis*, Facultad de Ingeniería Universidad del Valle de Guatemala, 2020.

[2] A. Aguilar, "Algoritmo Modificado de Optimización de Enjambre de Partículas (MPSO)," en *Trabajo de Graduación, Modalidad Tésis*, Facultad de Ingeniería Universidad del Valle de Guatemala, 2019.

[3] *Robotarium | Institute for Robotics and Intelligent Machines*. dirección: `http://www.robotics.gatech.edu/robotarium` (visitado 29-03-2021).

[4] B. S. G. d. Almeida y V. C. Leite, "Particle Swarm Optimization: A Powerful Technique for Solving Engineering Problems," en, *Swarm Intelligence - Recent Advances, New Perspectives and Applications*, dic. de 2019, Publisher: IntechOpen. DOI: `10.5772/intechopen.89633`.

[5] A. P. Engelbrecht, *Computational Intelligence: An Introduction*, 2nd. Wiley Publishing, 2007, ISBN: 0470035617.

[6] M. N. Ab Wahab, S. Nefti-Meziani y A. Atyabi, "A Comprehensive Review of Swarm Optimization Algorithms," *PLOS ONE*, vol. 10, n.º 5, págs. 1-36, mayo de 2015. DOI: `10.1371/journal.pone.0122827`. dirección: `https://doi.org/10.1371/journal.pone.0122827`.

[7] C. A. R. Algarín, "Optimización por colonia de hormigas:" ES, *Ingeniería Solidaria*, vol. 6, n.º 10-11, págs. 83-89, 2010, Number: 10-11, ISSN: 2357-6014. dirección: `https://revistas.ucc.edu.co/index.php/in/article/view/454` (visitado 29-03-2021).