UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Algoritmo Modificado de Optimización de Enjambre de Partículas (MPSO)

Trabajo de graduación presentado por
Aldo Stefano Aguilar Nadalini
Para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala, Julio 2019

Protocolo de Trabajo de Graduación

Algoritmo Modificado de Optimización de Enjambre de Partículas (MPSO)

Anteproyecto de Trabajo de Graduación

Resumen

El algoritmo Modificado de Particle Swarm Optimization (MPSO) surge primeramente de dos ecuaciones vectoriales del PSO clásico para la búsqueda de mínimos y máximos dentro de un espacio determinado. Estas ecuaciones evaluan los puntos de posición de una partícula en una función de costo. La primera ecuación de velocidad se compone del factor de memoria o velocidad pasada, el factor cognitivo siendo la diferencia entre el costo de la posición actual y la mejor posición personal encontrada por la partícula, y el factor social siendo la diferencia entre el costo de la posición actual y la mejor posición global encontrada por el enjambre. La segunda ecuación de posición es la actualización de la posición actual a una posición nueva tomando la nueva velocidad de partícula por un delta de tiempo específico.

Ecuaciones de PSO clásico:

$$V_{i+1} = \varphi_o(\omega V_i + c_1\rho_1\delta_i^* + c_2\rho_2\delta_i^+)$$
$$\delta_i^* = \frac{p_{i+1}^* - p_i^*}{\Delta t}$$
$$\delta_i^+ = \frac{p_{i+1}^+ - p_i^+}{\Delta t}$$
$$P_{i+1} = P_i + V_{i+1}\Delta t$$

Las modificaciones realizadas al algoritmo clásico de PSO surgen a causa de la necesidad de implementar dicho algoritmo de búsqueda a robots que no se comportan como partículas, sino que se ven limitados por sus características físicas (dimensiones y limitaciones de movimiento). Tanto las ecuaciones de cinemática de los Bitbots diseñados en UVG, como las restricciones del espacio de búsqueda tienen efecto sobre los parámetros numéricos ideales que se buscan para la operación de estos. Se busca que estos se comporten como un enjambre capaz de rastrear y alcanzar una meta específica dentro del espacio de búsqueda de una manera óptima. Para la validación del diseño de dicho algoritmo, se debe realizar la simulación del comportamiento de los Bitbots utilizando softwares de MATLAB y WeBots que permiten emular el espacio de trabajo.

Justificación

Los Bitbots diseñados en la Universidad del Valle de Guatemala como parte de la Fase I del Megaproyecto Robotat [1] [2] son capaces de comportarse como un enjambre combinado de dimensiones reducidas. Sin embargo, estos aún no son capaces de llevar a cabo acciones como búsqueda de puntos objetivo (metas) dentro de un espacio de búsqueda utilizando las rutas óptimas. De aquí, nace la necesidad de utilizar el algoritmo de Particle Swarm Optimization [3] para darles a los Bitbots la capacidad de que, por medio del enjambre, puedan llegar hacia metas definidas en rutas óptimas.

Las modificaciones del MPSO se deben realizar para que sea posible la implementación del algoritmo de búsqueda, ya que los robots poseen dimensiones y características limitadas en el mundo físico. Es imposible utilizar directamente el PSO clásico de control de partículas [3], ya que los Bitbots deben cumplir con sus propias relaciones de cinemática [4]. Se debe lograr modelar dichos robots diferenciales de tal manera que puedan ser controlados teniendo únicamente las velocidades dadas por el algoritmo de partículas.

Se debe validar el algoritmo MPSO por medio de simulaciones por software para determinar si, efectivamente, con las modificaciones elegidas, se puede controlar a los Bitbots para que converjan hacia una meta definida. Estas simulaciones, deben realizarse en softwares que permitan la simulación del comportamiento dinámico de los robots físicos [5] para asemejarlos lo más posible a los agentes reales y permitir una futura fase de implementación física en los agentes de la UVG.

Objetivos

Generales

Implementar un algoritmo de búsqueda basado en la modificación del método de Particle Swarm Optimization (PSO) estableciendo los parámetros indicados para un mejor comportamiento del enjambre y las ecuaciones para adaptar dicho algoritmo a las dimensiones físicas y cinemática de los Bitbots de UVG.

Específicos

1. Encontrar el valor de los parámetros de la ecuación de PSO que permitan a los Bitbots comportarse como un enjambre que, en conjunto, busca una meta definida y converge en ella en un tiempo finito.
2. Validar las ecuaciones y parámetros PSO establecidos por medio de simulaciones computarizadas que permitan la visualización del comportamiento del enjambre de partículas.

3. Establecer las ecuaciones necesarias para modelar el movimiento y la cinemática de los Bitbots y hacer posible la adaptación del algoritmo PSO en ellos.
4. Validar las simulaciones del MPSO en Bitbots por medio de la implementación del algoritmo en robots físicos simulados en el software de WeBots.

Marco Teórico

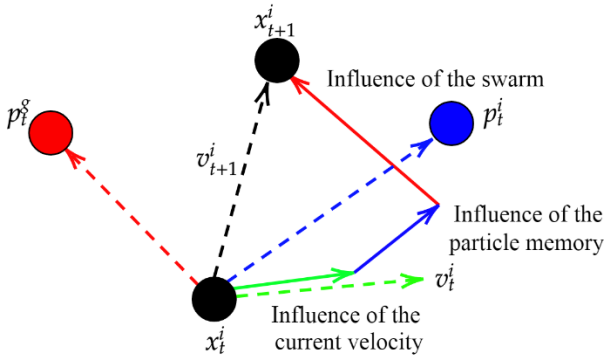
1. Particle Swarm Optimization (PSO)

Generalidades del algoritmo

El algoritmo clásico de optimización de enjambre de partículas (PSO por sus siglas en inglés) fue desarrollado por el Dr. Russell Eberhart y el Dr. James Kennedy en 1995 como un algoritmo de optimización basado en comportamiento de poblaciones [6]. Este algoritmo es similar a técnicas evolutivas de computación como los algoritmos genéticos (AG) explicados más a detalle en [7]. El algoritmo se inicializa con una población de soluciones aleatorias de una función de costo o fitness function; estas soluciones son denominadas partículas. Este algoritmo busca la solución óptima a la función de costo por medio de actualización de generaciones de partículas.

A diferencia de los algoritmos genéticos, el PSO no toma en cuenta operadores evolutivos como crossover o mutación de partículas al actualizar cada generación. Las partículas del PSO símplemente se mueven a través de espacio de trabajo siguiendo las partículas óptimas actuales de cada generación. El PSO tiene la ventaja de que es fácil de implementar a comparación de los algoritmos genéticos, ya que posee una menor cantidad de parámetros que se deben ajustar. Esto ha permitido que dicho algoritmo haya sido implementado exitosamente en campos de estudio como entrenamiento de redes neuronales, control de sistemas con lógica difusa, optimización de funciones computacionales e incluso en aplicaciones de optimización de construcción de estructuras como se expone en [8].

Figura 1. Representación vectorial de Ecuación de Velocidad PSO



2. Parámetros importantes del algoritmo

Parámetro de Constricción

Este parámetro denotado por la letra φ, es una ponderación que se le da a la ecuación completa de velocidad PSO. El resultado de la suma de todos los factores ponderados cognitivo, social y de memoria se multiplican por este parámetro. Este sirve para ajustar la longitud de los pasos que cada partícula puede dar en cada iteración [3].

Parámetro de Inercia

Este parámetro denotado por la letra ω, es una ponderación que se le da al factor de memoria de la ecuación de velocidad PSO. Y. Shi y R. Eberhart investigaron acerca de este nuevo parámetro en [3] y determinaron que su valor puede cambiar la velocidad de convergencia del enjambre en el mínimo/máximo de la función de costo. En la experimentación que realizaron, se descubrió que valores de $\omega > 0.8$ favorecían una convergencia más lenta del enjambre dándole así más capacidad de exploración. Con valores de $\omega < 0.8$, se obtenía una convergencia más rápida hacia mínimos locales dándole al enjambre mayor capacidad de explotación. En [3], se plantea un parámetro de inercia decreciente en el tiempo para que, al aumentar el número de iteraciones del algoritmo, este cambie el carácter exploratorio inicial del enjambre a un carácter más convergente y de explotación. También se puede aplicar inercia caótica, random y natural exponencial.

Parámetros de Uniformidad

Kennedy y Eberhart [6], también descubrieron que se le debe aplicar cierta dinamicidad estocástica a dichos factores para que el enjambre de partículas no se mueva únicamente en una dirección común, sino que explore adecuadamente el espacio de búsqueda. Para esto, se utilizan los parámetros de uniformidad para cada factor. Estos parámetros poseen un valor aleatorio real entre 0 y 1, y están denotados por la letra ρ.

Parámetros de escalamiento

Asímismo, Kennedy y Eberhart descubrieron que darle un factor de escalamiento a los factores cognitivo y social cambiaba el comportamiento del enjambre. Un mayor peso al factor cognitivo resulta en una mayor dispersión de las partículas aumentando la capacidad de exploración del enjambre. Un mayor peso al factor social resulta en una mayor convergencia del enjambre hacia mínimos locales aumentando la capacidad de explotación de un punto específico [9].

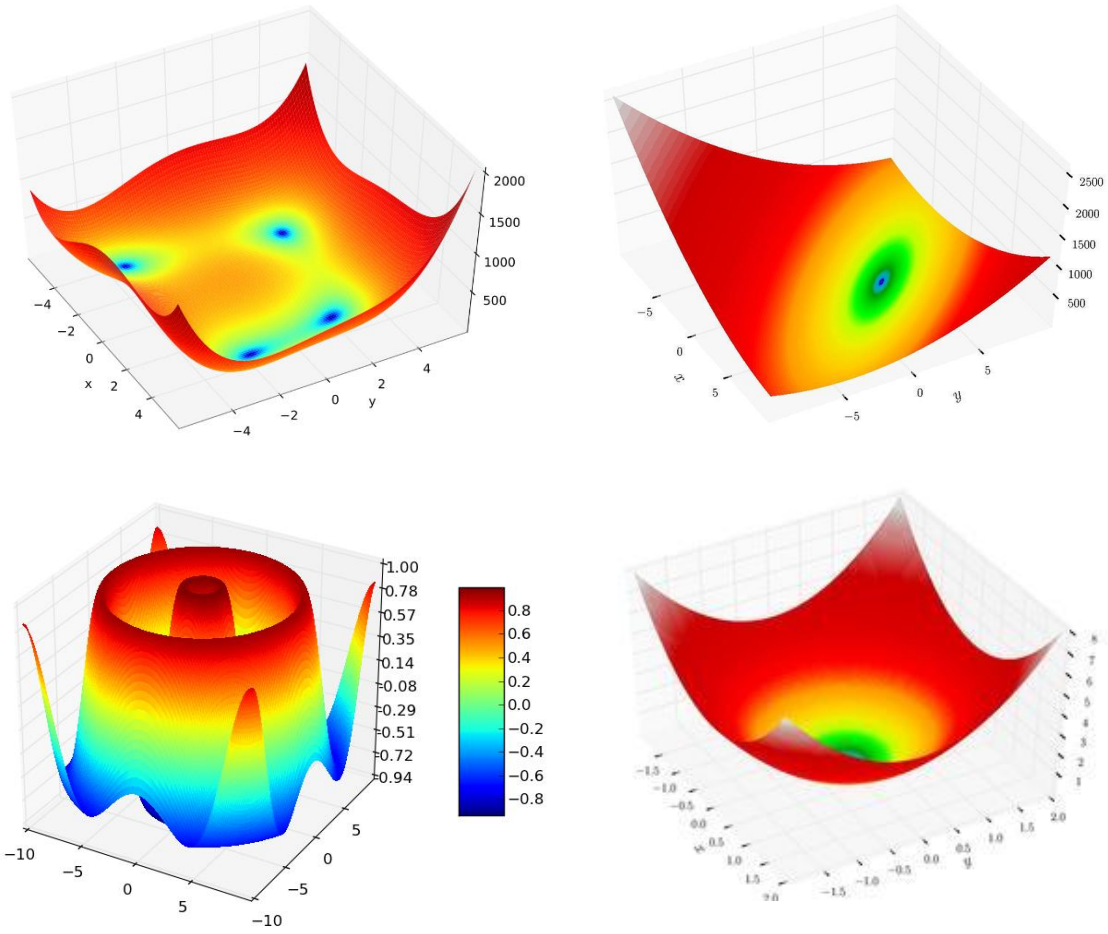
3. Funciones de costo

Para la evaluación de efectividad y eficiencia del algoritmo PSO se utilizan funciones de prueba o *benchmark functions* que presentan topologías con múltiples mínimos y máximos. Estas son útiles para determinar si los parámetros elegidos para el algoritmo son los adecuados. Se pueden elegir funciones símples con un solo mínimo para evaluar velocidades

de convergencia o se pueden elegir funciones con varios mínimos locales y un mínimo absoluto para determinar la capacidad de exploración del algoritmo para encontrar el máximo absoluto sin converger erróneamente a algún mínimo local. En [6], se utilizan las siguientes funciones de costo:

- Función Schaffer F6
- Función Booth
- Función Himmelblau
- Función Sphere
- Función Rosenbrock

Figuras 2-5. Funciones de costo en espacio tridimensional

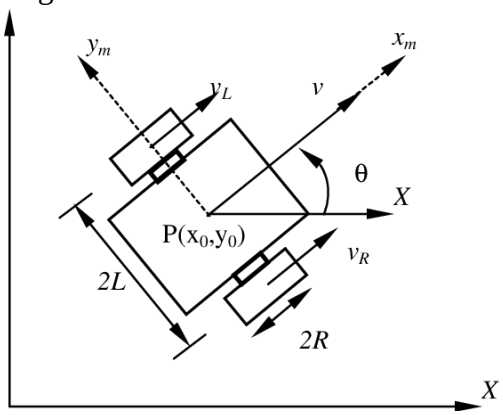


4. Robots diferenciales

Existen diferentes tipos de robots móviles que son capaces de trasladarse en un medio planar por medio de actuadores rotacionales. Dentro de esta categoría se encuentran los robots diferenciales. Los robots diferenciales se destacan por tener dos ruedas montadas sobre un

mísmo eje de rotación. Cada rueda es propulsada independientemente por su propio actuador. Esto permite controlar tanto la tracción como la orientación del robot con las mismas ruedas. Los robots diferenciales, al poder ser controlados con solamente dos motores y dos ruedas, son ideales para aplicaciones en donde se requiere que estos posean dimensiones físicas reducidas. Con este tipo de diseño de robot móvil, se pueden realizar movimientos lineales y movimientos rotacionales sin traslación. El control de este tipo de robots se reduce al envío adecuado de velocidades angulares a cada uno de los actuadores para poder describir las curvas de movimiento necesarias [10].

Figura 6. Modelo de robot diferencial



Metodología

1. Se debe investigar acerca de las diferentes aplicaciones que se le ha dado al PSO en simulaciones de partículas y se debe observar qué parámetros se utilizan en dichas simulaciones.
2. Se debe estructurar una tabla por cada parámetro investigado y se deben realizar anotaciones sobre los efectos del valor de dicho parámetro en cada una de las simulaciones observadas. Con esto se puede determinar la relación entre este parámetro y el comportamiento del enjambre. Los parámetros principales a investigar son:
    a. C_1 y C_2: tipo de parámetros utilizados para dar una ponderación a cada uno de los factores que componen la ecuación de PSO. Con este tipo de parámetro se elige si se le da más importancia al parámetro local o al parámetro social.
    b. ρ_1 y ρ_2: tipo de parámetros que poseen un valor aleatorio entre [0,1]. Se debe determinar si estos parámetros son necesarios para el algoritmo a diseñar para los Bitbots de UVG.
    c. ω: tipo de parámetro de inercia que le da ponderación a la velocidad actual del robot para calcular la velocidad futura. Se debe determinar el valor de este para el movimiento adecuado de los Bitbots de UVG.

    d. φ: tipo de parámetro para limitar los cambios entre una velocidad y otra del robot. Este parámetro depende de la cantidad de obstáculos que el enjambre tenga que superar en el espacio de trabajo.
3. Se deben determinar las funciones adecuadas para asignar un costo a cada punto en el mapa y que estos valores puedan ser utilizados por el algoritmo de MPSO para orientar a los robots hacia una meta definida en la posición con el costo más bajo. Estas funciones de costo o *fitness functions* deben tomar en cuenta la distancia actual entre una posición del mapa y la posición de la meta.
4. Se deben determinar las ecuaciones de cinemática que se acoplen a la forma de movimiento de los robots o agentes que poseen dimensiones físicas.
5. Ya teniendo las ecuaciones de cinemática y la fitness function, se debe estructurar el algoritmo en software para que pueda ser utilizado por robots simulados.
6. Teniendo el algoritmo de MPSO programado, se debe comenzar a realizar las pruebas de simulación, primeramente, utilizando agentes que posean la misma dimensión que los robots físicos. En la simulación, se debe probar con diferentes valores de los parámetros de la ecuación de PSO tomando en cuenta los comportamientos investigados en el paso 1 y 2.
7. Con la simulación, se deben validar las ecuaciones de cinemática y los parámetros elegidos experimentalmente. Se debe tabular datos como los valores de parámetros utilizados, la cantidad de agentes usados en las simulaciones, la variación en la velocidad angular de los motores de los robots y diferentes parámetros de ajuste de las ecuaciones cinemáticas utilizadas.

Cronograma de Actividades

1. Semana No.2 (08/07) – Avances prototipo 1: Se debe investigar aplicaciones del PSO clásico y tabular los efectos de cada uno en el comportamiento del enjambre. Se simularán las partículas PSO en MATLAB. En cuanto a los robots, se debe fijar el tamaño de llantas, distancia entre llantas y velocidad máxima de motores de robots para estructurar las ecuaciones de cinemática.

2. Semana No.3 (15/07) – Avances trabajo escrito 1: Se documentarán los resultados obtenidos en las simulaciones previas de partículas realizadas en MATLAB, así como se estructurará y revisará el marco teórico.

3. Semana No.4 (22/07) – Avances prototipo 2: Se seleccionarán las funciones de costo para evaluar el rendimiento del MPSO y se procederá a la programación del algoritmo en lenguaje C. Se realizarán simulaciones del movimiento del enjambre de robots en WeBots para observar el comportamiento de este de manera realista.

4. Semana No.6 (05/08) – Avances prototipo 3: Se realizarán nuevas simulaciones del movimiento del enjambre de robots en WeBots variando los parámetros PSO elegidos en MATLAB. También se probará implementar un controlador PID para la velocidad angular y un filtro de picos para evitar saturaciones en velocidades angulares de actuadores o motores de los robots simulados.

5. Semana No.7 (12/08) – Avances trabajo escrito 2: Se documentarán los resultados obtenidos en las simulaciones de robots en el software de WeBots.

6. Semana No.8 (19/08) – Avances prototipo 4: Se determinará si es necesario realizar más simulaciones variando algunas secciones del algoritmo o si se procede a dejarlo como prototipo final.

7. Semana No.10 (02/09) – Entrega final prototipo: Se entregará simulación final con el algoritmo completamente implementado con los parámetros PSO elegidos y las ecuaciones finales de cinemática.

8. Semana No.11 (09/09) – Entrega final trabajo escrito: Se entregará trabajo escrito final de la investigación para sus correcciones finales.

9. Semana No.12 (16/09) – Fase final: se comenzará a transcribir el trabajo escrito a una publicación científica para su consecuente publicación al finalizar la presentación del trabajo de graduación.

Índice Preliminar

Referencias

[1] A. Rodas, “Desarrollo e implementación de algoritmo de visión por computador en una mesa de pruebas para la experimentación con micro-robots móviles en robótica de enjambre”, UVG, 2019, págs. 11-51.

[2] M. Castillo, “Diseñar e implementar una red de comunicación inalámbrica para la experimentación en robótica de enjambre”, UVG, 2019, págs. 22-42.

[3] Y. Shi y R. Eberhart, “A modified particle swarm optimizer”, en 1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360), IEEE, 1998, págs. 69-73.

[4] M. Egerstedt, Control of Mobile Robots, Introduction to Controls. Georgia Institute of Technology, 2014.

[5] S. Konduri, E. O. C. Torres y P. R. Pagilla, “Dynamics and control of a differential drive robot with wheel slip: application to coordination of multiple robots”, Journal of Dynamic Systems, Measurement, and Control, vol. 139, n.o 1, pág. 014 505, 2017.

[6] J. Kennedy y R. Eberhart, “Particle Swarm Optimization”, IEEE, 1995, págs. 1-7.

[7] J. Carr, “An introduction to genetic algorithms”, Senior Project, vol. 1, No. 40, pág. 7, 2014.

[8] A. Kaveh, Advances in metaheuristic algorithms for optimal design of structures. Springer, 2014.

[9] T. Beielstein, K. E. Parsopoulos y M. N. Vrahatis, Tuning PSO parameters through sensitivity analysis. Universitätsbibliothek Dortmund, 2002.

[10] R. Siegwart, I. R. Nourbakhsh y D. Scaramuzza, Introduction to autonomous mobile robots. MIT press, 2011.