Análisis de Convergencia Temprana en Algoritmos PSO con Función Objetivo Lineal

Miguel A. Azar, FabiolaP. Paz, Analía Herrera Cognetta
Facultad de Ingeniería / Universidad Nacional de Jujuy
Av. Italia y Av. Martiarena / S. S. de Jujuy / Provincia de Jujuy/Tel. 0388-4221591
auazar@live.com, lic.faby@gmail.com, anihco@yahoo.com

Resumen

El presente trabajo está centrado en hallar el número de ciclos aceptable para la interrupción del proceso de iteraciones del algoritmo PSO, mediante la extrapolación de una función exponencial que debe ser hallada durante las primeras iteraciones.
Si bien, en el algoritmo básico PSO, las partículas siguen una variación del peso de inercia lineal en cada iteración para encontrar el valor óptimo de una función, en el presente trabajo, se plantea el uso de la regresión exponencial, como alternativa para lograr una convergencia temprana. Se estudia la convergencia de partículas, para luego proponer la regresión exponencial, como factor de aproximación. Para finalizar esta etapa de investigación, se utiliza la función lsqnonlin de MATLAB®, a fin de obtener los valores de los coeficientes que satisfacen un cúmulo de puntos de las posiciones de cada partícula.

Palabras clave: regresión exponencial, metaheurísticas, inteligencia de enjambre, particle swarm optimization, restricciones, convergencia.

Contexto

El presente proyecto se desarrolla a partir de un trabajo final de carrera de grado, en el que se plantea una solución alternativa al problema del mecanismo de manejo de restricciones en PSO.
En cuanto al contexto de investigación, el proyecto pretende formar parte de un trabajo vinculado al desarrollo de técnicas innovadoras para la implementación de sistemas autónomos y autoorganizados.
Las tareas investigativas son analizadas, coordinadas e implementadas por los integrantes de la cátedra Trabajo Final de Sistemas de la carrera Licenciatura en Sistemas de la UNJu.

Introducción

La técnica de optimización por enjambre de partículas PSO (Particle Swarm Optimization) es una metaheurística en el campo de los sistemas inteligentes que permite encontrar el valor óptimo de una función objetivo determinada. Para ello utiliza un conjunto de partículas que se desplazan y colaboran entre sí en un espacio n-dimensional. Este proceso demanda un costo computacional que depende de la función analizada y de la estrategia empleada. La reducción de dicho costo significa lógicamente una mayor eficiencia de los recursos y por ende un mejor aprovechamiento del hardware, esto es, el microprocesador y la memoria.
Los sistemas inteligentes basados en metaheurísticas se caracterizan por la cualidad de obtener resultados óptimos aproximados con un error que está en función del número de iteraciones.
La inteligencia de enjambres (SI) está basada en poblaciones de individuos (soluciones potenciales candidatas) que cooperan entre sí y estadísticamente son cada vez mejores a través de las iteraciones con las que, finalmente, encuentran una o varias soluciones [1].

Los algoritmos generados mediante la SI tienen una gran ventaja con respecto a otros métodos de optimización matemáticos. Estos algoritmos sólo requieren conocer el valor de la función objetivo para cada una de las soluciones candidatas, así de esta manera, pueden generar nuevas y mejores soluciones. Esto es, el empleo de algoritmos flexibiliza el amplio conjunto de problemas de optimización, sin la necesidad de tener un conocimiento profundo y detallado de las características implícitas de la función que se desea optimizar [2].

Entre las técnicas más estudiadas en este ámbito se encuentran los algoritmos PSO que permiten resolver diferentes problemas de optimización y son aplicados en otras ramas de la ciencia. Esta metaheurística imita el comportamiento social y comunicacional que se establece en grupos de individuos, tales como aves y peces, y utilizan la inteligencia colectiva como recurso para hallar alimento o refugio; esto significa que ajustan sus movimientos para evitar depredadores o buscar comida [3].

La estrategia consiste en seguir al ave que está más cerca del alimento. Cada ave se representa mediante una partícula que está siempre en continuo movimiento dentro del espacio de búsqueda; almacenando, y en algunos casos, comunicando a todo el enjambre la mejor solución que han encontrado hasta el momento [4].

El factor de inercia w

El modelo del PSO puede ser puramente cognitivo si la tendencia de la partícula depende de las mejores posiciones encontradas en su pasado personal; o bien, puede ser puramente social si esa tendencia es proporcional al pasado del cúmulo. Si el modelo tiene ambas componentes (cognitiva y social) la velocidad de cada partícula puede determinar la forma en que estas convergen al valor óptimo. Es por ello que Shi y Eberhart [5] propusieron una mejora del algoritmo básico, modificando la fórmula de actualización de la velocidad e introduciendo una nueva variable denominada factor de inercia *w*.

Tanto el proceso de exploración y explotación del espacio de búsqueda, como así también, el control que permite que la velocidad no exceda sus límites establecidos, son equilibrados mediante el factor de inercia. Dicho factor regula la velocidad, multiplicando su peso por la velocidad previa. Un gran peso de inercia facilita la búsqueda global, mientras que un pequeño peso de inercia facilita la búsqueda local. Generalmente el mejor valor para *w* es dependiente del problema.

Al incorporar el factor de inercia *w* la actualización de la velocidad queda determinada por:

$$v_{id} = w * v_{id} + r_1 * p_1 * (pb_{id} - pos_{id}) + r_2 * p_2 * (gb_d - pos_{id})$$

En donde la variable v_{id} es la velocidad de la iteración anterior de la partícula *i* en la dimensión *d* (siendo *d* el número de variables del problema), r_1 y r_2 son valores aleatorios en el rango [0,1], p_1 y p_2 son los coeficientes de aprendizaje personal y social respectivamente, pb_{id} (personal best) es la mejor posición personal encontrada por la partícula *i* en la dimensión *d*, pos_{id} es la posición actual de la partícula *i* en la dimensión *d* y gb_d (global best) es el vector de posición global, mejor encontrado por todas las partículas, obtenido en la función objetivo y se actualiza después de cada iteración del algoritmo.

El valor del factor de inercia *w* puede permanecer fijo o bien puede ser linealmente decrementado en cada ciclo de vuelo.

Las primeras versiones de PSO con factor de inercia utilizaban un valor constante durante toda la búsqueda, generalmente con *w*=7. Sin embargo, estudios posteriores demostraron que para algunos problemas, el decremento del factor *w* en el tiempo permite combinar la exploración

de una búsqueda global con la explotación de una búsqueda local, es decir un *w* dinámico.
Si este factor tiene características dinámicas, *w* es reducido linealmente utilizando 2 valores límites: $w_{inicial}$ y w_{final} que generalmente toman los valores: 0.9 y 0.4, respectivamente. Luego, mediante:

$$w = \frac{(ciclo_{max} - ciclo) * (w_{inicial} - w_{final})}{ciclo_{max}} + w_{final}$$

el factor de inercia es actualizado en cada ciclo del proceso, hasta el final de las iteraciones ($ciclo_{max}$).

Convergencia de las partículas

El factor de inercia *w*, como se indicó anteriormente, posee características lineales por lo que el movimiento de cada partícula debiera acompañar tal cualidad. Sin embargo, en general las partículas en PSO se desplazan con movimiento exponencial en función de los ciclos; esto es, durante las primeras iteraciones de exploración la velocidad de cada partícula toma valores extremos mientras que en los últimos ciclos se reduce hasta alcanzar una convergencia aproximada. En [6], basado en este concepto, se propone el cálculo de un peso de inercia adaptativa exponencial (Exponential Particle Swarm Optimization) que mejora el rendimiento de la búsqueda de las funciones de referencia de manera significativa. Por otro lado, mediante esta cualidad, en [7] se exponen diferentes estrategias novedosas para la dinámica del peso de inercia exponenciales.
La Figura 1 ilustra los puntos extremos que una partícula asume en función de los ciclos del algoritmo PSO hasta aproximarse al valor óptimo.
Ahora bien, dependiendo de las características del problema a optimizar, el número de ciclos necesarios para una convergencia segura es siempre indeterminado. En un problema de optimización en donde el número de variables de decisión es considerable, para obtener un resultado aceptable, la dificultad aumenta. Por lo general se adopta un número fijo de iteraciones de manera que el algoritmo obtenga una convergencia conveniente. Dicha indeterminación incrementa el costo computacional, dado que no es posible conocer con antelación, a partir de que iteración la convergencia es confiable.

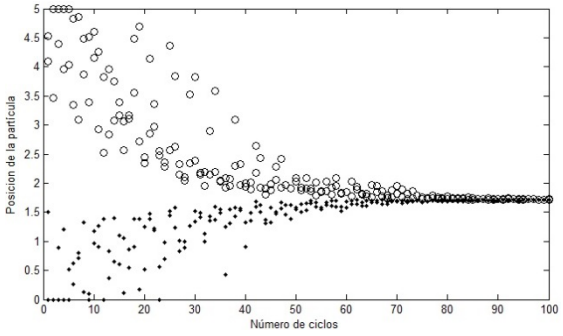

Figura 1

Frente a esta problemática, se propone la regresión exponencial como factor de aproximación, para obtener una convergencia temprana, extrapolando la función exponencial hallada.

Regresión exponencial en PSO

La regresión exponencial básica consiste en encontrar aquellos coeficientes pertenecientes a una función exponencial que satisfagan con la máxima aproximación a los puntos de interés.
La función exponencial simple de la forma:

$$a * exp^{b*x}$$

requiere para su resolución la determinación de los coeficientes *a* y *b* tal que los puntos conocidos y definidos por *x* sean representados en forma aproximada por la función.
Una función simple de estas características constituye la base que posibilita una regresión con un error que depende del nivel de dispersión de los puntos de

interés. Es por ello que resulta conveniente ampliar los términos de esta expresión a una función de la forma:

$$a * exp^{b*x} + c * exp^{d*x} + e * exp^{f*x}$$

Precisamente, a mayor cantidad de términos exponenciales representativos, mejor será el ajuste entre la curva exponencial hallada y el conjunto de puntos de interés.

Aplicación de la regresión exponencial

La función lsqnonlin de MATLAB® permite resolver problemas no lineales de mínimos cuadrados, incluidos los problemas de ajuste de datos no lineales. A través de esta función es posible la obtención de los coeficientes buscados logrando una aproximación aceptable.
A continuación en la Figura 2 puede observarse el código que permite encontrar los valores de los coeficientes que satisfacen un cúmulo de puntos de una partícula.

```
function resta = regresion_exp(vector_coef,vx,vy)
    a = vector_coef(1);
    b = vector_coef(2);
    c = vector_coef(3);
    d = vector_coef(4);
    e = vector_coef(5);
    f = vector_coef(6);
    resta = a.*exp(b.*X)+c*exp(d.*X)+e.*exp(f.*X) - Y;
```

Figura 2

Los parámetros utilizados para la definición del algoritmo PSO son los siguientes: Cantidad de partículas 20, Dimensión (cantidad de variables que utiliza la función objetivo) 2, Factor de inercia inicial *winicial* 0.9, Factor de inercia final *wfinal* 0.4, Constante de aprendizaje cognitivo y social 1.49445.
Para verificar la hipótesis de convergencia temprana se utilizó una función simple lineal con la siguiente estructura:

```
F.O.: Min Z = 180X1 + 160X2
Sujeta a: 6X1 + 1X2 >=12
          3X1 + 1X2 >=8
          4X1 + 6X2 >=24
          X1, X2<=5
```

Figura 3

La adquisición de los puntos de interés a los que se aplica la regresión exponencial son los valores que toma cada partícula en cada iteración. En este caso, tal como se muestra la Figura 1, se separan todos aquellos puntos que forman la curva cóncava hacia abajo antes de aplicar la regresión.

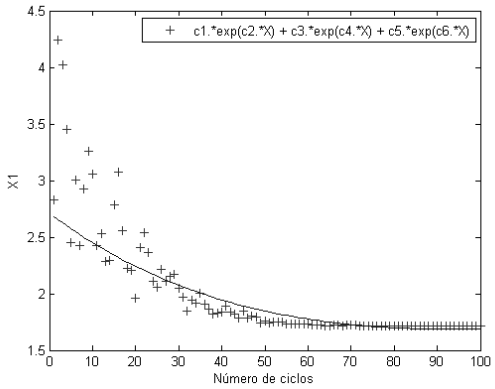

Figura 4

En la Figura 4 se observan los valores que toma la variable X1 en función del número de ciclos del algoritmo PSO. Los puntos de interés marcados con '+' son representados por la curva de línea continua en donde aproximadamente en el ciclo 70 se intersectan.
El análisis realizado en diferentes pruebas da cuenta de que es posible tomar dicho punto de intersección como condición de parada del algoritmo PSO obteniendo una convergencia temprana con un error promedio inferior al 0.3% evitando llegar al ciclo 200 que se sugiere habitualmente para asegurar una mejor aproximación.

Líneas de investigación y desarrollo

El uso de una función lineal se decidió a efectos de iniciar un primer acercamiento, a la reducción del costo computacional que implica el uso de algoritmos PSO, ante la incertidumbre que supone la condición de parada que, para asegurar una convergencia aproximada, debe estar en el orden de los 200 ciclos o más.
Se pretende en futuras pruebas emplear funciones benchmark, disponibles para

evaluar este tipo de algoritmos, y así obtener conclusiones más significativas en cuanto a su aplicación.

Resultados y objetivos

El trabajo es el resultado de una serie de análisis enfocados a mejorar y aplicar los algoritmos PSO en futuras investigaciones y que busca el siguiente objetivo:

Objetivo general

Desarrollar nuevas técnicas para la mejora de metaheurísticas en sistemas autónomos y autoorganizados.
En cuanto a este objetivo, se desarrolló una primera etapa de convergencia temprana con éxito, utilizando una función objetivo lineal, que será reemplazada por funciones de mayor complejidad en etapas sucesivas.

Formación de recursos humanos

El equipo de trabajo está formado por: Ing. Miguel Augusto Azar (Facultad de Ingeniería - UNJu), Lic. Fabiola Patricia Paz (Facultad de Ingeniería – UNJu) y Lic. Analía Herrera Cognetta (Facultad de Ingeniería - UNJu). El proyecto tiene su origen-idea, en el trabajo final de grado para la carrera de Licenciatura en Sistemas, presentado y aprobado por Fabiola P. Paz, en el que propuso optimizar los insumos, a través de la aplicación de PSO, para la producción de alimentos.

Referencias

[1] Panigrahi B.K., Shi Y., and Lim M., *Handbook of Swarm Intelligence: Concepts Principles and Applications*, Springer, 2011.

[2] Eiben A.E., Smith J.E., *Introduction to Evolutionary Computing*. Springer Natural Computing Series, 1st edition, ISBN: 3-540-40184-9, 2003.

[3] Kennedy J., Eberhart R.C., *Particle Swarm Optimization*, in Proceedings IEEE International Conference of Neural Network, IEEE Service Center, Piscataway, pp. 1942-1948,NJ, 1995.

[4] Paz F.P., Azar M.A., Herrera Cognetta A., Pérez Otero N.M., *Una alternativa para el mecanismo de manejo de restricciones en algoritmos PSO*, in Proceedings Second Argentine Conference on Human-Computer Interaction, Telecomunications, Informatics and Scientific Information HCITISI, Córdoba, Argentina, ISBN 978.88.96.471.25.8, Noviembre 21-22, 2013.

[5] Shi Y., Eberhart R.C., *Parameter Selection in Particle Swarm Optimization*, in Proceeding of the Seventh Annual Conference on Evolutionary Programming, San Diego, California, pp. 591–600, USA, March 25–27, 1998.

[6] Wu J., Liu W., Zhao W., Li Q., *Exponential Type Adaptive Inertia Weighted Particle Swarm Optimization Algorithm*. Second International Conference on Genetic and Evolutionary Computing, pp. 79-82. IEEE Press, New York, 2008.

[7] Chauhan P., Deep K., Pant M., *Novel inertia weight strategies for particles warm optimization*. Memetic computing, Vol. 5, pp. 229-25, ISSN: 1865-9284, 2013.