Universidad de Burgos

Computación Neuronal y Evolutiva

Optimización por Enjambre (*Particle Swarm Optimization*)

La optimización por enjambre de partículas (*Particle Swarm Optimization* - PSO) hace referencia a una metaheurística que evoca el comportamiento de algunos animales como enjambres de insectos en la naturaleza.

Estudiantes:

David Miguel Lozano
Javier Martínez Riberas

Profesores de la asignatura:

Álvaro Herrero Cosío
Bruno Baruque Zanón

1º semestre 2016

Índice

A. Introducción

La optimización por enjambre de partículas (*Particle Swarm Optimization* - PSO) es un algoritmo metaheurístico que hace una búsqueda iterativa en el espacio de soluciones posibles para intentar mejorar las soluciones actuales.

Fue desarrollado por James Kennedy y Russell Eberhart en 1995, partiendo de una simulación simplificada de un entorno social basado en una parvada.

La principal inspiración del PSO es la vida artificial (*A-life*) en general, con el estudio de bandadas de aves, bancos de peces o la inteligencia de enjambre en particular. Aunque, según mencionan los autores en su trabajo original [4], también se inspira en campos como los algoritmos genéticos y la computación evolutiva.

Sus aplicaciones principales son problemas no lineales y multidimensionales, aunque se puede adaptar a casi todo tipo de problemas.

B. Historia

En 1995, James Kennedy y Russell Eberhart publicaron el artículo "*Particle Swarm Optimization*"[4], en el que propusieron una prueba de concepto para optimizar funciones no lineales basándose en la metodología de enjambre de partículas.

El objetivo inicial de los autores era realizar una simulación gráfica de un modelo de conducta social basado en el movimiento imprevisible de una bandada de aves. Sin embargo, posteriormente descubrieron que su modelo podía ser utilizado como un optimizador.

En su primera aproximación, la velocidad de cada agente (en su caso pájaros) se ajustaba en relación con la velocidad de su vecino más próximo. Sin embargo, este método desembocaba en que la parvada establecía rápidamente una dirección fija.

Para solventar este problema, introdujeron una variable estocástica llamada "locura". De tal manera, que en cada iteración se añadía algún cambio aleatorio. Lo que proporcionaba un aspecto más real a la simulación, aunque la variación seguía siendo totalmente artificial.

Inspirados por la simulación de aves de Hepner, se propusieron añadir un objetivo a los agentes: encontrar comida. Para esto, modificaron los agentes para que cada uno supiese en todo momento la mejor posición global de cada uno y cambiaron el ajuste de las velocidades de acuerdo a estos valores.

Por lo tanto, el escenario planteado era el siguiente: una bandada de pájaros se encuentra buscando comida aleatoriamente en un área donde sólo hay una pieza de comida. Ninguno de los pájaros sabe donde está la comida, pero, debido a la función a optimizar elegida, sí que saben cómo de lejos están de ella.

Con las modificaciones introducidas, vieron como las aves eran capaces de encontrar la comida gracias a la forma en la que el conocimiento de cada individuo era compartida en el medio social. Los resultados eran muy buenos, estas se aproximaban a la comida de forma realística y sincronizada, hasta que finalmente se "posaban" en el objetivo. De esta forma, demostraron que el método era capaz de optimizar funciones lineales de dos dimensiones.

En una nueva aproximación, se intentó generalizar el método a un entorno no lineal y multidimensional. Para ello, se utilizó una matriz de vectores para guardar las velocidades de cada dimensión. Se probó la nueva versión con diferentes problemas, entre ellos, el entrenamiento de varias redes neuronales. Para todos ellos, el algoritmo arrojó unos buenos resultados.

El algoritmo resultante, tras algunas mejoras y optimizaciones posteriores, sentó las bases de la optimización por enjambre de partículas.

El término enjambre había sido definido por Millonas en 1994 en el artículo "*Swarms, phase transitions, and collective intelligence*"[5]. En él, se establecen los cinco principios que deben cumplir los enjambres inteligentes:

1. **Principio de proximidad:** la población tiene que ser capaz de realizar cálculos simples del espacio y tiempo.
2. **Principio de calidad:** la población tiene que ser capaz de responder a factores de calidad en el entorno.
3. **Respuestas diversas:** la población no debe llevar a cabo sus acciones por canales excesivamente estrechos.
4. **Estabilidad:** la población no debe de cambiar su modo de comportamiento cada vez que cambia el entorno.
5. **Adaptabilidad:** la población tiene que poder cambiar su modo de comportamiento cuando merezca la pena computacionalmente.

El término partículas fue escogido por compromiso. Al tratarse de agentes sin masa ni volumen podían haberse denominado puntos. Sin embargo, se quiso incidir en las propiedades de velocidad y aceleración que estos poseen. Por esto, se vio más adecuado el uso del término partículas. [4]

En una nueva publicación en 1998 [7], Shi y Eberhart discutieron la necesidad de utilizar la velocidad en la propia ecuación de actualización de la velocidad. Si se eliminaba, PSO dejaba de explorar y se limitaba a explotar los mínimos locales. Si

no se eliminaba, PSO gastaba demasiado tiempo en explorar, en vez de intentar buscar el mínimo local que podría ser el global.

Para poder generalizar un comportamiento intermedio entre búsqueda exploratoria y explotación local, propusieron añadir una variable nueva. Esta variable multiplica la velocidad, de manera que, cuando es mayor de 1 fomenta la exploración y cuando es menor fomenta la explotación.

En la solución propuesta, la variable se inicializa por encima de 1 y se va decrementando linealmente hasta llegar a 0 en el número máximo de iteraciones. De esta manera, el algoritmo prioriza la exploración en una etapa inicial, pero se centra en la explotación según avanza la ejecución.

En la actualidad se están desarrollando nuevas y más sofisticadas variantes del PSO con el fin de mejorar el rendimiento de este. En particular, se está trabajando en las siguientes vías de investigación: [8]

- Modificaciones del PSO (quantum-behaved PSO, bare-bones PSO, chaotic PSO, fuzzy PSO, PSOT-VAC, opposition-based PSO, etc.).
- Métodos híbridos que combinan PSO con otros métodos metaheurísticos, incluyendo algoritmos genéticos (GA), sistemas inmunológicos artificiales (AIS), búsqueda Tabú (TB), algoritmos de colonias de hormigas (ACO), algoritmos de recocido simulado (SA), algoritmos de colonias de abejas artificiales (ABC), evolución diferencial (DE), optimizaciones biogeográficas (BBO), búsquedas armónicas (HS), etc.
- Extensión del PSO a otros campos de optimización, como optimización multiobjetivo, con restricciones, discreta o binaria.
- Análisis teóricos del PSO. Sobretodo en la selección de parámetros y análisis de convergencia.
- Implementaciones paralelas de PSO: incluyendo programación multinúcleo, en GPU o en Cloud.

C. Funcionamiento

El funcionamiento es muy simple. Sin embargo, aunque hay varias teorías acerca de porqué funciona, los expertos no consiguen ponerse de acuerdo en una.

En PSO, las potenciales soluciones se llaman partículas y "vuelan" por el espacio del problema siguiendo a las partículas óptimas actuales.

Cada partícula posee unas coordenadas en el espacio del problema. Estas están asociadas con la mejor solución (*fitness*) encontrada hasta ahora. A su vez, cada

partícula almacena tres valores:

- velocidad: dirige el "vuelo" de la partícula.
- `pbest`: el valor de la mejor solución que ha encontrado hasta el momento (*fitness*).
- `gbest`: el valor de la mejor solución encontrada hasta el momento por cualquier partícula del enjambre.

El algoritmo se inicializa con un grupo de partículas aleatorias (soluciones) y posteriormente busca la solución óptima mediante la actualización de estas.

En cada iteración, se actualiza la velocidad de cada partícula siguiendo sus dos "mejores" valores: `pbest` y `gbest`. Y con su velocidad se calcula su nueva posición.

En concreto, se actualiza la velocidad de cada dimensión de acuerdo a la siguiente expresión:

```
vx[][]= vx[][] +
    2 * rand() * (pbestx[][] - presentx[][]) +
    2 * rand() * (pbestx[][gbest] - presentx[][])
```

Esta expresión actualiza la velocidad de una dimensión (`vx[][]`) de tal manera que la partícula se acerque al mejor punto conocido por esta. `presentx` hace referencia a la partícula actual (solución), `rand()` genera un número aleatorio entre (0, 1), `pbestx` y `pbestx` se corresponden con los valores descritos anteriormente.

La posición se actualiza de acuerdo a:

```
presentx[][]=presentx[][]+vx[][]
```

El pseudocódigo del algoritmo es el siguiente:

Algorithm 1 PSO básico.

```
1: for all partículas do
2:    Inicializar partícula con un vector aleatorio uniformemente distribuido.
3:    Inicializar pbest con su posición inicial.
4:    if fitness en pbest es mejor que en gbest then
5:       Actualizar gbest con pbest.
6:    end if
7:    Inicializar velocidad de la partícula.
8: end for
9: repeat
10:   for all partículas do
11:      Calcular el valor de fitness.
12:      if fitness actual es mejor que en gbest then
13:         Actualizar gbest con la posición actual.
14:      end if
15:   end for
16:   Obtener la partícula con el mejor fitness.
17:   for all partículas do
18:      Calcular su nuevo valor de velocidad.
19:      Actualizar su posición.
20:   end for
21: until no se cumpla el criterio de parada (iteraciones máx./error mín.)
22: Devolver gbest como solución encontrada.
```

En las siguientes figuras podemos observar dos instantes consecutivos del algoritmo. El sentido de la velocidad de cada partícula está indicada con una flecha. Vemos como el mejor resultado obtenido en el primer instante condiciona el movimiento de las partículas en el segundo.

Para visualizar gráficamente el funcionamiento del algoritmo se han grabado dos simulaciones realizadas en Matlab:

- Simulación 2D: [webm][flv]
- Simulación 3D: [webm][flv]

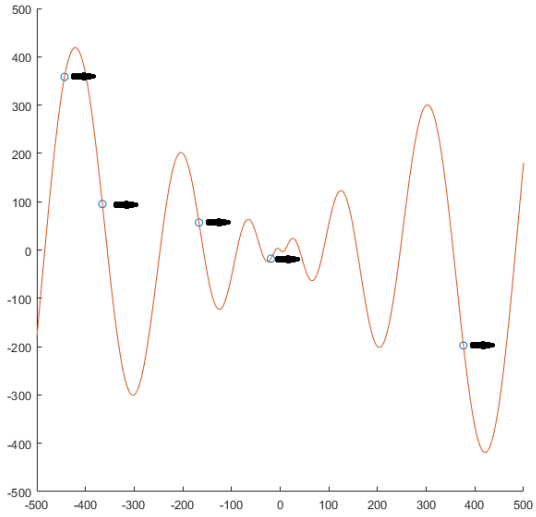


Figura 1: PSO t0

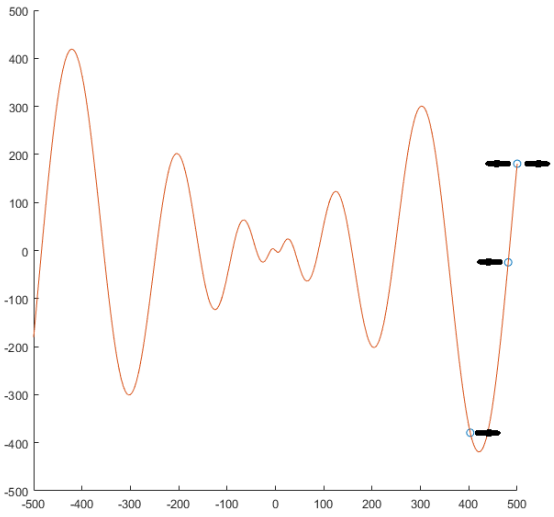


Figura 2: PSO t1

El número de parámetros a ajustar en PSO no es muy grande: [2]

- **Número de partículas:** el rango típico está entre 10 y 40 partículas. Para la mayoría de problemas 10 partículas es un número suficiente para obtener buenos resultados. Sin embargo, en problemas difíciles el número necesario se puede incrementar hasta las 100 o 200 partículas.
- **Dimensiones:** determinado por el problema a resolver.
- **Rangos:** los valores por lo que puede moverse cada partícula. También es particular del problema a resolver. Los rangos pueden ser diferentes en cada dimensión.
- **Velocidad máxima:** define el máximo cambio que puede sufrir una partícula en una iteración. Normalmente se establece con el valor del rango.
- **Condición de parada:** se suele utilizar un criterio doble. Por un lado se define el número máximo de iteraciones y por otro una cota inferior del error. Estos valores dependen del problema a resolver.

El por qué PSO funciona no está claro, como ya hemos comentado anteriormente. Existen dos teorías principales sobre su funcionamiento interno. [1]

La primera teoría y más extendida defiende que PSO realiza dos tipos de búsqueda. En un primer momento, realiza una búsqueda exploratoria amplia en el espacio de soluciones (comportamiento de exploración). Pero según aumenta el tiempo, cambia a una búsqueda local que se aproxima rápidamente hacia un óptimo, posiblemente local (comportamiento de explotación).

La otra teoría defiende que aún no se ha logrado comprender cómo afecta el com-

portamiento del enjambre a la calidad del proceso de optimización, especialmente en problemas de optimización con espacios de búsqueda multidimensionales, discontinuos o variables en el tiempo. Esta escuela de pensamiento no busca un equilibrio entre las fases de exploración y explotación, sino que busca encontrar los mejores parámetros posibles para el problema en concreto. Este planteamiento ha llevado a simplificar los algoritmos de PSO en variantes como *Bare Bones PSO* [3] o la optimización con enjambre de partículas aceleradas (*Acelerated PSO* - APSO). Ambos métodos prescinden de la velocidad.

D. Aplicaciones

Desde que se desarrolló el método de optimización por enjambre de partículas, se ha aplicado en numerosos campos.

Al tratarse de una metaheurística, tiene un carácter muy generalista para resolver cualquier problema que se pueda definir en términos de un problema de optimización. Aunque, como cualquier otra metaheurística, no asegura encontrar una solución óptima.

Una de las ventajas de PSO frente a otras metaheurísticas es que no usa descenso de gradiente. Por lo tanto, no es necesario que la función que define el problema sea derivable.

Otra de sus características positivas es el reducido número de parámetros que hay que ajustar. La misma versión del algoritmo con muy ligeras modificaciones puede resolver problemas muy diferentes.

Además, ha demostrado obtener buenos resultados de una manera más rápida y barata que muchos otros métodos de optimización.

Por todo lo anterior, PSO ha sido utilizado en infinidad de aplicaciones. A continuación exponemos las más relevantes clasificadas por categorías por orden de popularidad: [6, 8]

D.0.1. Telecomunicaciones

El campo más popular del PSO son las telecomunicaciones. Su uso por excelencia es el diseño de antenas de telecomunicaciones, de tal forma que optimicen la propagación o la recepción de la señal. Otros usos en este campo son el diseño y optimización de todo tipo de redes de comunicaciones. Por ejemplo, la elección

del emplazamiento óptimo de los equipos, la optimización del ancho de banda y la asignación de canales, redes inalámbricas, etc.

D.0.2. Medicina

La segunda categoría más popular es biomedicina. En donde se utiliza en aplicaciones muy diversas, desde herramientas para el diagnóstico de Parkinson, diseño de medicamentos, prótesis, clasificación de cánceres, agrupamiento de genes, selección de biomarcadores, etc.

D.0.3. Inteligencia artificial

El uso principal de PSO en cuanto a la inteligencia artificial son las redes neuronales, tanto su control y diseño como el entrenamiento. PSO se usa tanto en conjunto con las redes neuronales para conseguir modelos con más complejidad aunque también se usa PSO para entrenar las redes neuronales o incluso para diseño y control de las mismas. [se puede recalcar que fue la primera aplicación propuesta]

D.0.4. Data mining

Un buen número de aplicaciones están relacionadas con tareas de agrupamiento, clasificación, reducción de la dimensionalidad o selección de características. Que a su vez, pueden ser utilizadas para tareas de predicción como el pronóstico del tiempo, la carga de la red eléctrica, tráfico de vehículos, etc.

D.0.5. Procesamiento de imagen y sonido

Este es uno de los campos donde PSO tiene una gran repercusión. Existen infinidad de aplicaciones desde reconocimiento de caras, iris, peatones, frutas, detección de defectos, clasificación de objetos, reducción de ruido, tracking de objetos, filtros adaptativos, codificación de sonido, gráficos 3D, síntesis de texturas, etc.

D.0.6. Robótica

En el campo de la ingeniería eléctrica, electrónica y de automatismos también existe un importante número de artículos relacionados con PSO. Desde diseño

digital de circuitos electrónicos tolerantes a fallos, planificación de trayectos, hasta el control y planificación de motores.

D.0.7. Operaciones

Los PSO son ampliamente utilizados en tareas de control y planificación . Como planificaciones óptimas de procesos, calendarios laborales, horarios de trenes, etc.

E. Comparación

PSO comparte ciertas similitudes con técnicas de computación evolutiva como los algoritmos genéticos:

- El sistema es inicializado con una población de soluciones aleatorias.
- El valor de fitness de cada individuo se recalcula en cada iteración.
- El sistema busca la solución óptima mediante la actualización de sus generaciones.
- Ambos introducen valores aleatorios en su búsqueda del óptimo.
- Ninguno de los dos métodos garantiza encontrar la solución óptima.
- Ambos pueden resolver cualquier problema que se pueda formular como un problema de optimización.

Pero también poseen notables diferencias:

- En la actualización de generaciones, PSO no utiliza operadores evolutivos como la selección, el cruce o la mutación, sino que sus nuevos individuos se basan en los individuos óptimos actuales.
- Cada partícula posee memoria. Lo que supone una pieza clave del algoritmo.
- El mecanismo por el que se comparte información es diferente. En PSO, se utiliza un mecanismo de comunicación unidireccional mediante el cual sólo se comparte el valor `gbest` entre partículas.
- La velocidad de convergencia suele ser mayor que la de los algoritmos genéticos.
- Poseen menos parámetros configurables.

Uno de los usos que se les da a los algoritmos genéticos es para el entrenamiento de redes neuronales. Los pesos y/o la topología de la red se codifican como genotipos y se diseña una función de selección de acuerdo con los objetivos de la red. Por ejemplo, para un problema de clasificación, el valor de fitness puede ser el ratio de patrones clasificados mal.

Sin embargo, hay varios artículos que proponen el uso de PSO para reemplazar el algoritmo de entrenamiento *back-propagation*. Ya que es un método rápido y obtiene buenos resultados en la mayoría de los casos. Además, posee ciertas ventajas respecto a los algoritmos genéticos, en los que la selección de los operadores genéticos debe realizarse cuidadosamente.

F. Conclusiones

En este trabajo se ha introducido el método de optimización por enjambre de partículas. Como se ha visto, se trata de un algoritmo simple que permite optimizar un amplio abanico de funciones.

Como todos los métodos de optimización, PSO posee una serie de ventajas e inconvenientes. Es un algoritmo simple, con un número reducido de parámetros, puede optimizar funciones no derivables y es eficiente en un gran número de casos. Por otro lado, en su versión original, puede quedarse atrapado en mínimos locales, su complejidad computacional es algo elevada y su rendimiento es muy sensible a la parametrización escogida.

Desde su aparición, PSO ha sido utilizado en numerosas aplicaciones pertenecientes a disciplinas científicas muy diferentes. Desde su primera aplicación para en entrenamiento de redes neuronales, hasta la síntesis de medicamentos en medicina, optimización de las redes de comunicaciones y eléctricas, planificación de calendarios laborales o métodos de visión artificial.

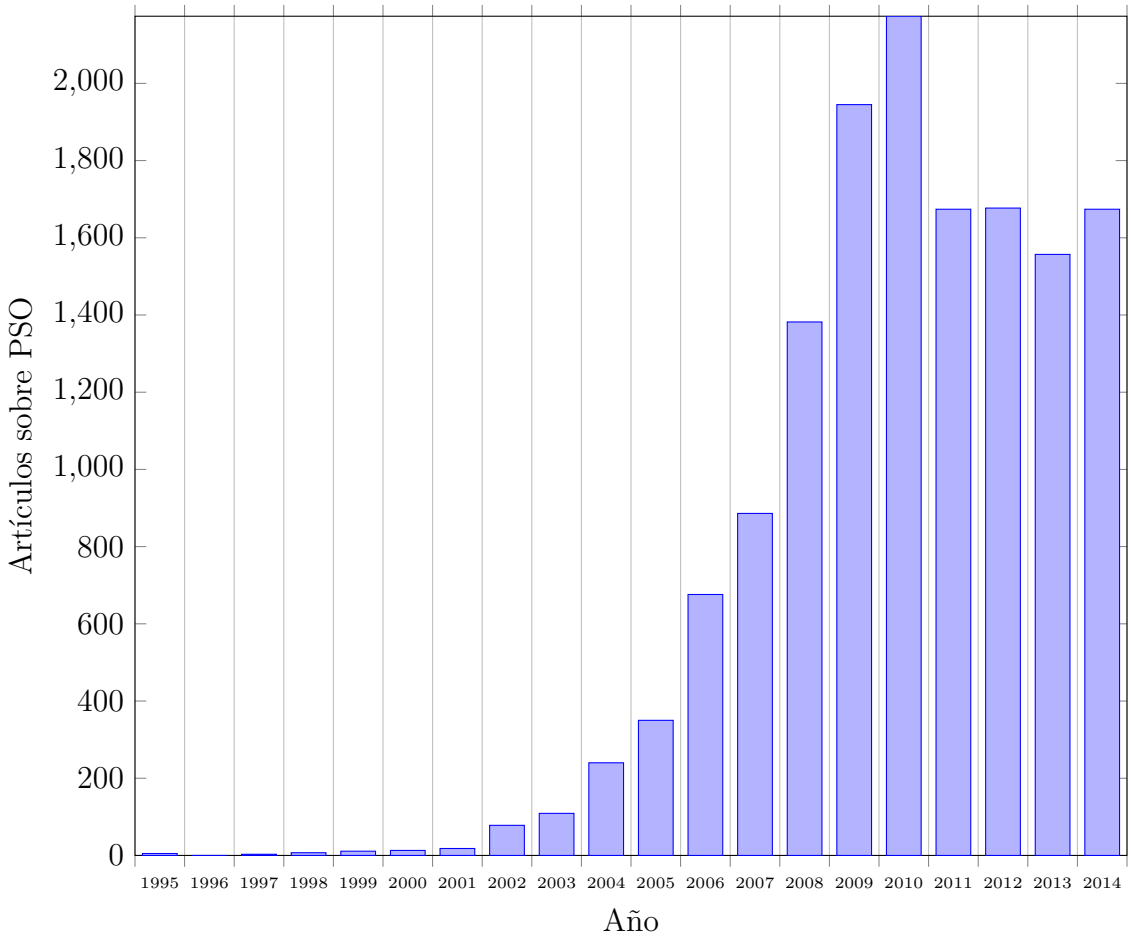
El futuro parece prometedor bajo nuestro punto de vista, ya que la cantidad de aplicaciones para las que se está usando PSO es muy grande y el conjunto de algoritmos de inteligencia de enjambre son todavía jóvenes (1993). Otros algoritmos parecidos como los genéticos (1971) son más antiguos y han tenido más tiempo para ser investigados.

Este futuro quizá se vea nublado por el hecho de que no sobresalgan en una 'familia' de problemas concretos y haya otros algoritmos más establecidos.

Por esto puede que necesiten un claro ejemplo en el que superen al resto de algoritmos existentes con creces para conseguir el reconocimiento y atención necesarios para ser ampliamente conocidos y estudiados en los ámbitos en los que más atención se les va a dar como puede ser el académico.

Cabe destacar que se está realizando gran cantidad de investigación en estos algoritmos como se puede ver en la gráfica.

Queríamos comentar que la investigación de PSO se ha estabilizado o al menos eso parece y esperamos que madure, para que acabe produciendo los resultados anteriormente mencionados para el progreso de la investigación.



Referencias

[1] Particle swarm optimization, September 2016. URL https://en.wikipedia.org/w/index.php?title=Particle_swarm_optimization&oldid=740776743. Page Version ID: 740776743.

[2] Xiaohui Hu. Particle Swarm Optimization: Tutorial. URL http://www.swarmintelligence.org/tutorials.php.

[3] J. Kennedy. Bare bones particle swarms. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No.03EX706)*, pages 80–87. IEEE, 2003. ISBN 0-7803-7914-4. doi: 10.1109/SIS.2003.1202251. URL http://ieeexplore.ieee.org/document/1202251/.

[4] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE, 1995. ISBN 0-7803-2768-3. doi: 10.1109/ICNN.1995.488968. URL http://ieeexplore.ieee.org/document/488968/.

[5] M. M. Millonas. Swarms, Phase Transitions, and Collective Intelligence. *Conference: 3. artificial life conference, Santa Fe, NM (United States), 15-19 Jun 1992*, 1992. URL https://arxiv.org/abs/adap-org/9306002.

[6] Riccardo Poli. Analysis of the Publications on the Applications of Particle Swarm Optimisation. *Journal of Artificial Evolution and Applications*, 2008: 1–10, 2008. ISSN 1687-6229. doi: 10.1155/2008/685175. URL http://www.hindawi.com/archive/2008/685175/.

[7] Y. Shi and R. Eberhart. A modified particle swarm optimizer. In *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, pages 69–73. IEEE, 1998. ISBN 0-7803-4869-9. doi: 10.1109/ICEC.1998.699146. URL http://ieeexplore.ieee.org/document/699146/.

[8] Yudong Zhang, Shuihua Wang, and Genlin Ji. A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications. *Mathematical Problems in Engineering*, 2015:1–38, 2015. ISSN 1024-123X. doi: 10.1155/2015/931256. URL http://www.hindawi.com/journals/mpe/2015/931256/.