3. Algoritmos

3.1. Introducción

El objetivo de las técnicas de identificación es obtener los parámetros de un modelo que mejor ajustan las medidas realizadas y las predicciones realizadas con el modelo.

El problema de la identificación óptima de parámetros podría formularse básicamente de la forma siguiente: "Dada una función $F(x_1, \dots, x_n)$, hallar las constantes $(x_1, \dots, x_n)$ tales que minimizan o maximizan F". Así pues, el problema de la identificación óptima de modelos se reduce a la resolución de un sistema de ecuaciones no lineales.

La identificación de las constates de los modelos de los motores a partir de la información de los catálogos presenta algunas dificultades importantes. La información reflejada en el catálogo deriva de medidas "reales". Al realizar observaciones del sistema físico real, es decir, al realizar las medidas, siempre se introducen errores asociados al proceso de medida (errores en la determinación experimental de los valores asociados a los transductores, los instrumentos y al propio método de medida). Según [4], en algunos casos es fácil comprobar que un error de tan sólo un 5% en una medida puede provocar un error en alguno de los parámetros estimados superior al 100%, o incluso provocar que sea negativo. En el caso de tener que realizar la identificación a partir de la información de catálogo, se presenta una complicación adicional derivada de las tolerancias admitidas por la normas para cada uno de los valores reflejados en el catálogo. Además, los datos que ofrecen los catálogos no corresponden a ninguna máquina en particular, sino que son valores medios de una serie de medidas de distintas máquinas, supuestamente idénticas pero que en la realidad nunca lo son. Por otro lado, los modelos de los sistemas físicos son descripciones matemáticas que describen "razonablemente" el comportamiento del sistema físico real. Es decir, que describen sólo la parte que interesa al observador y con cierto nivel de error, como se deduce de lo expuesto en el Capítulo 2: Modelos de circuito, que viene limitado de nuevo por las exigencias o necesidades del observador. La combinación de todos estos factores (errores de medida, tolerancias, datos medios y modelado) hace que lo que debería ser un sistema de ecuaciones (lineal o no lineal), de la que despejar los valores de los parámetros, se convierta realmente en un problema de optimización (maximización o minimización), en el que a lo más que se puede aspirar es a la obtención de un conjunto de constantes que minimicen las diferencias entre los valores derivados del modelo y los medidos o, en general, la información disponible sobre su respuesta.

En este capítulo se hace primero una descripción básica del ajuste por mínimos cuadrados, que es la técnica clásica más utilizada en los problemas de identificación de modelos. En una segunda parte se introducen los fundamentos de los algoritmos

genéticos. Finalmente, se exponen las bases teóricas de los algoritmos basados en los sistemas de partículas.

3.2. Método de los mínimos cuadrados para la resolución de ecuaciones no lineales

El método tradicionalmente utilizado para la búsqueda de la solución de un sistema de ecuaciones no lineales es el método de los mínimos cuadrados. Este método se basa en los algoritmos de Newton-Gauss, de Levenberg-Marquardt o de la Región de Confianza que se explicarán más adelante.

El problema de los mínimos cuadrados consiste en encontrar una solución, *x*, de forma que la suma de los cuadrados de los residuos sea mínima. Existen varios métodos para resolver este problema. Las condiciones ideales se tienen cuando f es continua, doblemente diferenciable y estrictamente convexa. Obviamente esto no se tiene o no se puede garantizar o verificar siempre. Hay métodos que utilizan segundas y primeras derivadas, es decir utilizan la matriz hessiana y el gradiente, hay métodos que utilizan únicamente derivadas primeras y hay métodos que no utilizan derivadas.

Generalmente, cuanta más información se utilice, es decir, cuanto mayor sea el orden de las derivadas utilizadas, más eficiente es el método en cuanto al número de iteraciones. Pero por otra parte, conseguir toda esa información es costoso y, dependiendo de la función, puede hacer lento el proceso. Por otro lado, a veces no se puede conseguir cierta información. Por ejemplo, en algunas ocasiones no es posible calcular derivadas primeras o segundas en funciones que no son continuas. En esos casos la única solución es utilizar métodos alternativos que no requieran esta información.

Matemáticamente, el problema de los mínimos cuadrados se escribe [9]:

$$\min_{x\in\mathbb{R}^n} f(x) = \frac{1}{2}R(x)^T R(x) = \frac{1}{2}\sum_{i=1}^{m} r_i(x)^2 \qquad (3.\ 1)$$

dónde $R(x)$ es la función residual y $r_i(x)$ es la componente *i*-ésima de la misma.

Para resolver un problema de mínimos cuadrados los distintos métodos existentes siguen una línea común. Dado un punto inicial x^0, se construye una sucesión de puntos $\{x^k\}$ con la esperanza de alcanzar el mínimo de la función.
En casi todos los métodos, dado un punto x^k se construye una dirección $d^k \neq 0$ y se obtiene λ_k^* minimizador de $\varphi(\lambda) = f(x^k + \lambda \cdot d^k)$ cuando λ varía en $\mathbb{R}$. Los métodos se diferencian en la construcción de la dirección d^k [10].

$$\lambda_k^* = \arg\min f(x^k + \lambda \cdot d^k) \quad (3.\ 2)$$

$$x^{k+1} = x^k + \lambda_k^* \cdot d^k \quad (3.\ 3)$$

Además, debe cumplirse que: $f(x^{k+1}) < f(x^k)$

3.2.1. Método de Newton

El método más básico para la resolución de problema de mínimos cuadrados es el método de Newton. La primera derivada de *R*(*x*) es el Jacobiano, $J(x)_{ij} = \frac{\partial r_i(x)}{\partial x_j}$. Una aproximación de *R*(*x*) en torno a un punto x_c es [9]:

$$M_c(x) = R(x_c) + J(x_c)(x - x_c) \quad (3.\ 4)$$

La primera derivada de $f(x) = \frac{1}{2} R(x)^T R(x)$ es:

$$\nabla \mathrm{f(x)} = \sum_{i=1}^{m} r_i(x) \cdot \nabla r_i(x) = J(x)^T R(x) \quad (3.\ 5)$$

De la misma forma, la segunda derivada es:

$$\nabla^2 f(x) = \sum_{i=1}^{m} \left(\nabla r_i(x) \cdot \nabla r_i(x)^T + r_i(x) \cdot \nabla^2 r_i(x) \right) = J(x)^T \cdot J(x) + S(x) \quad (3.\ 6)$$

dónde:

$$S(x) \triangleq \sum_{i=1}^{m} r_i(x) \cdot \nabla^2 r_i(x) \quad (3.\ 7)$$

que denota la información de segundo orden.

La aproximación cuadrática de $f(x)$ en torno a x_c es:

$$m_c = f(x_c) + \nabla f(x_c)^T(x - x_c) + \frac{1}{2}(x - x_c)^T \nabla^2 f(x_c)(x - x_c) = R(x_c)^T \cdot$$

$$\cdot R(x_c) + R(x_c)^T \cdot J(x_c)(x - x_c) + \frac{1}{2}(x - x_c)^T \cdot\cdot (J(x_c)^T \cdot J(x_c) + S(x_c)) \cdot \quad (3.\ 8)$$

$$\cdot (x - x_c)$$

Aplicando el método de Newton a (3.1) a partir de (3.8):

$$x_+ = x_c - \left(J(x_c)^T \cdot J(x_c) + S(x_c)\right)^{-1} \cdot J(x_c)^T R(x_c) \quad (3.\ 9)$$

Este es un método rápido para resolver el problema de los mínimos cuadrados ya que localmente converge cuadráticamente.

El problema de esta aproximación de Newton es que *S*(*x*) normalmente no está disponible o es difícil de conseguir y es muy costoso, computacionalmente, obtener una aproximación por diferencias finitas. Además, este método no converge si el punto inicial está muy lejos de la solución.

3.2.2. Método de Newton-Gauss

La primera clase de métodos que se consideran para resolver el problema de mínimos cuadrados usa la aproximación [9]:

$$M_c(x) = R(x_c) + J(x_c)(x - x_c) \quad (3.\ 10)$$

Por tanto el problema de mínimos cuadrados se transforma en:

$$\min_{x \epsilon \mathbb{R}^n} \frac{1}{2} \|M_c(x)\|^2 \triangleq \widehat{m}_c(x) \quad (3.\ 11)$$

La solución a este problema es:

$$x_+ = x_c - (J(x_c)^T \cdot J(x_c)^{-1}) \cdot J(x_c)^T \cdot R(x_c) \quad (3.\ 12)$$

El método iterativo consistente en usar (3.12) en cada iteración. Este método es conocido como *método de Newton-Gauss* y en él se omite el término *S*(*x*) del método de Newton (3.9). La convergencia del método de Newton-Gauss dependerá de la importancia del término $S(x_c)$ como parte de la segunda derivada, $\nabla^2 f(x)$.

La velocidad de convergencia del método de Newton-Gauss decrece con la no linealidad del problema. Además, si el valor del residuo es demasiado grande el método puede no converger. Aunque el método de Newton-Gauss padece de muchos problemas, es la base de algunos importantes y exitosos métodos para resolver problemas de mínimos cuadrados. Este método es ventajoso en problemas lineales o casi lineales y cuando los residuos son pequeños. Los problemas lineales se resuelven en una iteración. Si el problema es fuertemente no lineal o los residuos son grandes, la convergencia es lenta o incluso puede no converger. Además, la convergencia no es necesariamente global. El comportamiento de estos algoritmos depende en gran medida del punto de partida.

3.2.3. Método de Levenberg-Marquardt

Un método que evita el problema de la derivada segunda es el *método de Levenberg-Marquardt.* Levenberg (1944) y Marquardt (1963) sugirieron la siguiente fórmula [9]:

$$x_+ = x_c - (J(x_c)^T \cdot J(x_c) + \mu_c I)^{-1} \cdot J(x_c)^T R(x_c) \qquad (3.\ 13)$$

Existen muchas versiones de este método en función de la elección de μ_c. Las propiedades de convergencia del método de Levenberg-Marquardt son similares a las del método de Newton-Gauss. La convergencia local es lenta cuando el residuo es grande y cuando el problema es fuertemente no lineal. Sin embargo este método es superior en ciertos aspectos. Cuando $J(x_c)$ no es de rango completo, el método de Levenberg-Marquardt queda mejor definido y cuando el paso es grande, la dirección de descenso es mejor que en el método de Newton-Gauss. Por ello es preferible el uso del método de Levenberg-Marquardt para la mayoría de los problemas.

3.2.4. Método de la Región de Confianza

Otra modificación del algoritmo de Newton-Gauss se basa en las llamadas *regiones de confianza* [10]. En el método de la región de confianza se restringe la minimización de la aproximación cuadrática a una vecindad de x_c, o sea,

$$\min_{x_+ \in \mathbb{R}^n} \|R(x_c) + J(x_c)(x_+ - x_c)\|_2 \quad sujeto\ a\ \|x_+ - x_c\|_2 \leq \delta_c \tag{3. 14}$$

Suponiendo que a partir de un punto *x* en el espacio *n*-dimensional, se quiere mejorar moviéndose a un punto donde la función sea menor, la idea es aproximar *f* mediante una función *q*, que razonablemente refleja el comportamiento de la función *f* en las cercanías, *N*, del punto *x*. Esta zona es la *región de confianza*. Por tanto, para un algoritmo basado en la región de confianza se necesita elegir una aproximación *q* definida en el punto actual *x*, elegir y modificar la región de confianza *N* y la precisión del subproblema de la región de confianza.

En el método estándar de la región de confianza, la aproximación cuadrática *q* está definida por los primeros dos términos del desarrollo de Taylor de *f*; las cercanías de *x*, *N*, se toman generalmente como una esfera o un elipsoide.

El valor de δ_c puede variar en cada iteración de acuerdo a la mejoría en el valor de la función objetivo. Si la mejoría es grande, entonces para la iteración siguiente el valor de δ_c aumenta. Por el contrario, si la mejoría es muy pequeña (o si no hay mejoría), esto quiere decir que el radio utilizado es demasiado grande y debe ser disminuido para la iteración siguiente. En este caso, además, *x* no cambia, es decir, $x^{k+1} = x^k$.

3.3. Algoritmos genéticos

Los Algoritmos Genéticos son métodos adaptativos, generalmente empleados en problemas de búsqueda y optimización de parámetros, basados en la reproducción sexual, teniendo en cuenta incluso posibles mutaciones, y en el principio de supervivencia del más apto.

El algoritmo genético se basa en la teoría de la evolución de Darwin, que ha cobrado tremenda popularidad en todo el mundo durante los últimos años. Se presentarán aquí los conceptos básicos que se requieren para abordarla.

Durante millones de años las distintas especies que pueblan la Tierra se ha adaptado para poder sobrevivir. De esta forma, se podría tener una población de posibles soluciones que fuesen evolucionando para adaptarse a su medio, que sería el problema a resolver de forma que se llegase a una solución óptima.

Los orígenes de los métodos de optimización evolutivos datan de los años 60 del siglo XX cuando un grupo de biólogos usaron ordenadores para la simulación de sistemas biológicos. Uno de los pioneros en el desarrollo de los algoritmos genéticos fue John Holland de la Universidad de Michigan, quien en sus investigaciones tenía como objetivo imitar los procesos adaptativos de los sistemas naturales y diseñar sistemas artificiales (normalmente programas), que retuvieran los mecanismos importantes de los

sistemas naturales. A la técnica que inventó Holland se le llamó originalmente "planes reproductivos", pero se hizo popular bajo el nombre "algoritmo genético" tras la publicación de su libro en 1975 [11]. David Goldberg, ingeniero industrial, fue uno de los primeros que trató de aplicar los algoritmos genéticos a problemas industriales [12]. Entre los algoritmos evolutivos más conocidos se incluyen los algoritmos genéticos, programación evolucionaria, estrategias evolutivas y programación genética que se agrupan bajo el nombre de computación evolucionaria [13].

Los algoritmos genéticos son algoritmos de optimización que tratan de resolver problemas del tipo "Dada una función $F(x_1, \ldots x_n)$, hallar las $(x_1, \ldots x_n)$ tales que minimizan o maximizan F".

Para explicar cómo estos algoritmos logran su propósito, puede tomarse como punto de partida la definición de D. Golberg [12]: "Los Algoritmos Genéticos son algoritmos de búsqueda basados en los mecanismos de la selección natural y de la genética. Partiendo de estructuras tipo cadena, combinan la supervivencia de la más apta, con el intercambio estructurado, aunque aleatorizado, de información entre ellas, constituyendo así una búsqueda con algo en común a la forma en la que el hombre experimenta y progresa".

Los algoritmos genéticos tratan de imitar a la Naturaleza en tanto que operan en busca del óptimo, produciendo como resultado especies perfectamente adaptadas a su entorno. Para entender cómo lo hace, se revisan brevemente las dos ideas clave: selección natural y genética [14].

- Charles Darwin, en 1859, publicó "El origen de las especies" (*The Origin of Species*) [15]. En ese texto se describió, por primera vez, el mecanismo que explicaba el fenómeno y que Darwin llamó selección natural: sólo los individuos mejor dotados logran sobrevivir y reproducirse, heredando los hijos las buenas cualidades que permiten que la especie mejore y se perpetúe generación tras generación.

  El proceso de transformación que sufren las especies está de acuerdo con esta teoría; en un medio ambiente cambiante, ciertas características individuales empezarían a cobrar mayor relevancia, en detrimento de otras, en el sentido de que aquellos ejemplares que las presentaran en mayor grado contarían con ventaja en la lucha por alcanzar la edad adulta y reproducirse. De esta forma, tales cualidades individuales tenderían poco a poco a extenderse a la totalidad de la población. Tras siglos de evolución, los descendientes llegarían a diferenciarse tanto de sus primeros padres que darían lugar a una nueva especie.

- La Genética, posteriormente, explicó el por qué de la diversidad y cómo se produce la herencia: toda la información acerca de la constitución física de un organismo vivo se encuentra codificada a nivel celular en una molécula denominada ADN. O dicho de otro modo, el ADN, que se encuentra formando parte de los cromosomas (el genotipo), controla la producción de enzimas que

serán los responsables de la formación de los caracteres de cada individuo (el fenotipo). Así, se observaron dos hechos importantes:

- Los cromosomas de dos individuos de la misma especie no son idénticos (diversidad).
- Los cromosomas de las células sexuales de los padres son los encargados de transmitir las características propias de la especie, al mismo tiempo que sus manifestaciones peculiares (herencia).

En la naturaleza, los individuos de una población compiten constantemente con otros por recursos tales como comida, agua y refugio. Los individuos que tienen más éxito en la lucha por los recursos tienen mayores probabilidades de sobrevivir y, generalmente, dan lugar a una descendencia mayor. Esto implica que los genes de los individuos mejor adaptados se propagarán a un número cada vez mayor de individuos de las sucesivas generaciones.

Para ilustrar cómo se traduce todo esto en un algoritmo genético, considérese el siguiente problema de optimización:

$$\min \text{ ó } \max y = F(x_1, \ldots x_n)$$

sujeto a una serie de restricciones.

1. Se identifica '*y*' como la cualidad, o conjunto de ellas, que se pretende mejorar; las variables del problema, (x_1, ... x_n), como el fenotipo de cada individuo; y '*F*', como la función objetivo, de adaptación o bondad, que proporciona el grado de excelencia de cada individuo como solución potencial del problema.

2. Se codifica el fenotipo en lo que Golberg llama estructura cadena, esto es, un cromosoma artificial (genotipo).

3. Se parte de un conjunto de posibles soluciones (población de individuos), que empiezan a competir para ver cuál constituye la mejor solución (aunque no necesariamente la mejor de todas las posibles). Los pares individuo-función objetivo *F* (ambiente), ejercerán una presión selectiva sobre la población, de forma que los que dan un mejor valor de *F* (los más adaptados) sobrevivan.

4. Estos últimos se emparejan, intercambiando parte de su información y dejando su legado genético a las futuras generaciones. La mejora continua debe estar sustentada por factores que fomenten la diversidad.

El azar juega un papel importante ya que la selección y el emparejamiento se desarrollan con un cierto grado de aleatoriedad. No obstante, los algoritmos genéticos no son simples búsquedas azarosas (Goldberg habla de proceso "aleatorizado"). En lugar de ello, los algoritmos genéticos explotan eficientemente la información del pasado para especular sobre nuevos puntos hacia los que dirigir la búsqueda, esperando mejorar su actuación.

Entre las ventajas de los algoritmos genéticos cabe destacar las siguientes:

- El primer y más importante punto es que los algoritmos genéticos son intrínsecamente paralelos. La mayoría de los demás algoritmos funcionan en serie y sólo pueden explorar el espacio de soluciones hacia una solución en una dirección al mismo tiempo, y si la solución que descubren no es la óptima, deben comenzar de nuevo. Sin embargo, ya que los algoritmos genéticos tienen descendencia múltiple, pueden explorar el espacio de soluciones en múltiples direcciones a la vez. Si un camino resulta ser un callejón sin salida, pueden eliminarlo fácilmente y continuar el trabajo en avenidas más prometedoras, dándoles una mayor probabilidad en cada ejecución de encontrar la solución.
  Además, el algoritmo genético puede dirigirse hacia el espacio de búsqueda con los individuos más aptos y encontrar el mejor de ese grupo. En el contexto de los algoritmos evolutivos esto se conoce como *teorema del esquema*, y es la principal ventaja de los algoritmos genéticos sobre los otros métodos de resolución de problemas.
- Debido al paralelismo que les permite evaluar implícitamente muchos esquemas a la vez, los algoritmos genéticos funcionan particularmente bien resolviendo problemas cuyo espacio de soluciones potenciales es relativamente grande. La mayoría de los problemas que caen en esta categoría son los no lineales donde cambiar un componente puede tener efectos en cadena en todo el sistema. El paralelismo implícito de los algoritmos genéticos les permite encontrar con éxito resultados óptimos o muy buenos en un corto periodo de tiempo, tras muestrear directamente sólo regiones pequeñas del espacio de búsqueda.
- Otra ventaja notable de los algoritmos genéticos es que se desenvuelven bien en problemas con una función de aptitud discontinua, cambiante con el tiempo o con múltiples óptimos locales. La mayoría de los problemas prácticos tienen un espacio de soluciones enorme, imposible de explorar exhaustivamente. El reto es entonces cómo evitar los óptimos locales, es decir, soluciones que son mejores que todas las que son similares a ella, pero que no son mejores que otras soluciones situadas en algún otro lugar del espacio de soluciones. Muchos algoritmos de búsqueda pueden quedar atrapados en los óptimos locales: si llegan a una cima descubrirán que no existen soluciones mejores en las cercanías y concluirán que han alcanzado la mejor de todas, aunque existan picos más altos en algún otro lugar del mapa.
- Otra área donde destacan los algoritmos genéticos es en su habilidad para manejar muchos parámetros simultáneamente. Muchos problemas involucran múltiples objetivos, a menudo contradictorios: uno sólo puede mejorar a expensas de otro. Si una solución particular a un problema con múltiples objetivos optimiza un parámetro hasta el punto en el que ese parámetro no puede mejorarse más sin causar una correspondiente pérdida de calidad en algún otro parámetro, esa solución se denomina *óptimo paretiano* o *no dominada.*

- Finalmente, una de las cualidades de los algoritmos genéticos es que no es necesario que conozcan nada del problema que se desea resolver. Estos realizan cruces y mutaciones en sus soluciones candidatas y luego utilizan la función de aptitud para determinar si estos cambios producen una mejora. Como sus decisiones están basadas en la aleatoriedad, todos los caminos de búsqueda posibles están abiertos.

Aunque los algoritmos genéticos son bastante eficientes, tienen ciertas limitaciones, aunque casi todas pueden superarse.

- La primera consideración al crear un algoritmo genético es definir una representación del problema. El lenguaje utilizado para especificar las soluciones debe ser robusto, es decir, debe ser capaz de tolerar cambios aleatorios que no produzcan constantemente errores fatales o resultados sin sentido. Ciertos lenguajes creados por el hombre no son robustos debido a que el número total de palabras con significado es pequeño comparado con el número total de formas en las que se puede combinar las letras del alfabeto y por tanto un cambio aleatorio en una frase puede producir un sinsentido.
- La forma de escribir la función de aptitud debe considerarse cuidadosamente para que verdaderamente signifique una solución mejor para el problema dado. Si se elige mal una función de aptitud puede que el algoritmo sea incapaz de encontrar una solución
- Un problema muy conocido que puede surgir con un algoritmo genético se conoce como convergencia prematura. Si un individuo es más apto que la mayoría de sus competidores emerge muy pronto en el curso de la ejecución, se puede reproducir tan abundantemente que merme la diversidad de la población demasiado pronto, provocando que el algoritmo converja hacia el óptimo local que representa ese individuo en lugar de rastrear el resto del espacio para encontrar el óptimo global. La solución es usar métodos de selección que no den tanta ventaja a los individuos excesivamente aptos.

3.3.1. Codificación del problema

Dado un problema específico a resolver, la entrada del algoritmo genético es un conjunto de soluciones potenciales a ese problema, codificadas de alguna manera, y una métrica denominada función de aptitud (*fitness*) que permite evaluar cuantitativamente a cada solución candidata. Cualquier solución potencial a un problema puede ser presentada dando valores a una serie de parámetros. El conjunto de todos los parámetros (genes) se codifica en una cadena de valores denominada cromosoma (Figura 3.1). El

conjunto de los parámetros representados por un cromosoma particular recibe el nombre de genotipo. Cada uno de los elementos que componen un gen se denomina alelo.

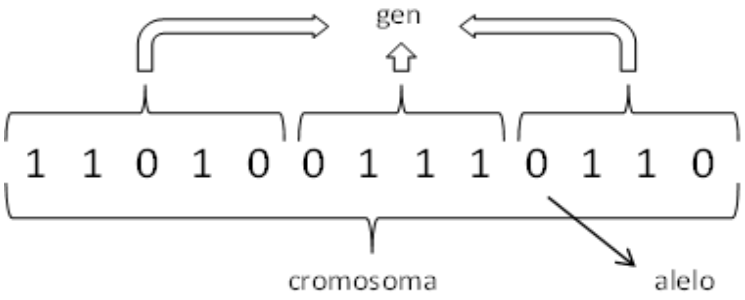

Figura 3.1. Composición de un cromosoma

La codificación suele hacerse mediante valores binarios. Se asigna un determinado número de bits a cada gen y se realiza una discretización de la variable representada por cada gen. El número de bits asignados depende del grado de ajuste que se desee alcanzar. Cada uno de los bits pertenecientes a un gen se suele llamar alelo. Sin embargo, también pueden existir representaciones que codifiquen directamente cada parámetro con un valor entero, real o en punto flotante.

Una vez que se ha definido la forma de codificar los elementos del problema, es necesario fijar un punto de partida. Los algoritmos genéticos trabajan sobre una población de individuos donde cada uno de ellos representa una posible solución al problema que se desea resolver. El algoritmo se encargará de favorecer en la población los individuos que están próximos a la solución del problema.

Los pasos en un algoritmo genético serían:

1. Crear una población inicial de individuos y calcular su grado de ajuste a la solución del problema (*fitness*).
2. Escoger dos miembros de esta población basándose en su aptitud, para convertirse en padres.
3. Usar un operador de reproducción (cruce o copia), sobre una gran parte de la población, para generar un nuevo individuo a partir de los padres.
4. Usar un operador de mutación, en una pequeña parte de la población, para conservar la diversidad genética de la población.
5. Calcular la aptitud del nuevo descendiente y reemplazar al peor adaptado de la población.
6. Volver al segundo paso hasta haber producido un número suficiente de descendientes.
7. El algoritmo termina cuando se engendra una solución lo suficientemente buena o por cualquier otro criterio de parada como puede ser un límite en el número de generaciones o repeticiones de la misma población.

Este algoritmo es capaz de encontrar buenas soluciones de un problema examinando solo un pequeño número de soluciones del total de soluciones posibles. Durante la reproducción, secciones del material genético de cada padre serán copiadas en los hijos. Como los padres han sido escogidos en función de su aptitud, hay muchas posibilidades de que la descendencia también tenga una buena aptitud lo que significa que no serán reemplazados por futuras generaciones demasiado rápido.

Una generación se obtiene a partir de la anterior por medio de los operadores de reproducción. Existen dos tipos:

- **Cruce:** Se trata de una reproducción de tipo sexual. Se genera una descendencia a partir del mismo número de individuos (generalmente dos) de la generación anterior
- **Copia:** Se trata de una reproducción de tipo asexual. Un determinado número de individuos pasa sin sufrir ninguna variación directamente a la siguiente generación

Una vez generados los nuevos individuos se realiza la mutación. Esencialmente es una modificación arbitraria que ayuda a prevenir una convergencia prematura de la población. La probabilidad de mutación suele hacerse muy baja, por lo general entre el 0.5% y el 2%.

Se sale de este proceso cuando se alcanza alguno de los criterios de parada fijados. Los más usuales suelen ser:

- Los mejores individuos de la población representan soluciones suficientemente buenas para el problema a resolver.
- La población ha convergido. Un gen ha convergido cuando el 95% de la población tiene el mismo valor. La media de la bondad de la población se aproxima a la bondad del mejor individuo.
- Se ha alcanzado el número de generaciones máximo especificado.

3.3.2. Operadores genéticos

En el paso de una generación a otra se aplican una serie de operadores genéticos. Los más empleados son los operadores de selección, cruce, copia, mutación y reemplazo.

3.3.2.1. Selección

Los algoritmos de selección son los encargados de escoger qué individuos van a disponer de oportunidades de reproducirse y cuáles no. Tal y como ocurre en la

naturaleza, se les debe otorgar mayores oportunidades de reproducción a los individuos más aptos. Por lo tanto la selección estará ligada al valor de ajuste de un individuo. Sin embargo, no se debe olvidar a otros individuos menos aptos puesto que en pocas generaciones se perdería diversidad y la población tendería a volverse homogénea. Lo común es seleccionar el primer individuo de los participantes en el cruce mediante algún método de selección y el segundo de manera aleatoria.

- Selección por ruleta

  También llamada *Selección de Montecarlo,* se trata de una forma de selección proporcional a la aptitud en la que la probabilidad de que un individuo sea seleccionado es proporcional a la diferencia entre su aptitud y la de sus competidores. Conceptualmente, esto puede representarse como un juego de ruleta, ocupando cada individuo un porcentaje de ésta de forma que la suma de todos los porcentajes sea la unidad. Los mejores individuos recibirán una porción de la ruleta mayor que la recibida por los peores. Generalmente la población está ordenada en base al ajuste por lo que las porciones más grandes se encuentran al inicio de la ruleta. Para seleccionar un individuo se genera un número aleatorio entre 0 y 1 y se escoge al individuo que ocupe esa posición en la ruleta.

  Es un método muy sencillo, pero ineficiente a medida que aumenta el tamaño de la población (su complejidad aumenta como n^2). Además presenta el inconveniente de que el peor individuo puede ser seleccionado más de una vez.

- Selección por torneo

  La idea principal es realizar la selección en base a comparaciones directas entre individuos. Se eligen subgrupos de individuos de la población, y los miembros de cada subgrupo compiten entre ellos. Resultarán ganadores aquellos que tengan valores de aptitud más altos.

- Selección por rango

  A cada individuo de la población se le asigna un rango numérico basado en su aptitud, y la selección se basa en este ranking, en lugar de en las diferencias absolutas en aptitud. La ventaja de este método es que puede evitar que individuos muy aptos ganen dominancia al principio a expensas de los menos aptos, lo que reduciría la diversidad genética de la población y podría obstaculizar la búsqueda de una solución aceptable.

Existen muchos otros algoritmos de selección. Unos buscan mejorar la eficiencia computacional, otros el número de veces que los mejores o peores individuos puedan ser seleccionados. Algunos de estos algoritmos son muestreo determinístico, escalamiento sigma, selección de jerarquías, estado uniforme, sobrante estocástico, brecha generacional, etc.

3.3.2.2. Cruce (*crossover*)

Se trata del operador genético principal. Una vez seleccionados los individuos, éstos son recombinados para que intercambien segmentos de su código, produciendo una descendencia artificial cuyos individuos son combinaciones de sus padres. La tasa de cruce suele tomarse elevada, rondando el 80%.

Los diferentes métodos de cruce podrán operar de dos formas diferentes. Si se opta por una estrategia destructiva los descendientes se insertarán en la población temporal aunque sus padres tengan mejor ajuste (trabajando con una única población esta comparación se realizará con los individuos a reemplazar). Por el contrario, utilizando una estrategia no destructiva, la descendencia pasará a la siguiente generación únicamente si supera la bondad del ajuste de los padres (o de los individuos a reemplazar). La idea principal del cruce se basa en que, si se toman dos individuos correctamente adaptados al medio y se obtiene una descendencia que comparta genes de ambos, existe la posibilidad de que los genes heredados sean precisamente los causantes de la bondad de los padres. Al compartir las características buenas de dos individuos, la descendencia, o al menos parte de ella, debería tener una bondad mayor que cada uno de los padres por separado. Si el cruce no agrupa las mejores características en uno de los hijos y la descendencia tiene un peor ajuste que los padres, no significa que se esté dando un paso atrás. Optando por una estrategia de cruce no destructiva se garantiza que pasen a la siguiente generación los mejores individuos. Si, aún con un ajuste peor, se opta por insertar a la descendencia, y puesto que los genes de los padres continuarán en la población (aunque dispersos y posiblemente levemente modificados por la mutación) en posteriores cruces se podrán volver a obtener estos padres, recuperando así la bondad previamente perdida.

Existen multitud de algoritmos de cruce. Sin embargo los más empleados son los tres siguientes:

- Cruce de un punto

  Es la técnica más sencilla. Conocida por las siglas SPX (*Single Point Crossover*), consiste en cortar los cromosomas de los padres por un punto seleccionado aleatoriamente para generar dos segmentos diferenciados en cada uno de ellos: la cabeza y la cola. Se intercambian las colas entre los dos individuos para generar los nuevos descendientes. De esta manera ambos descendientes heredan información genética de los padres, tal y como puede verse en la Figura 3.2.

- Cruce de dos puntos

  Se conoce como DPX (*Double Point Crossover*). Básicamente se trata del mismo procedimiento que el cruce de un punto pero los cromosomas de los

padres se cortan en dos puntos. Ninguno de los cortes debe corresponder a un extremo para garantizar que el cromosoma queda dividido en tres partes. Para generar la descendencia se toma el segmento central de uno de los padres y los segmentos laterales del otro padre (Figura 3.3).

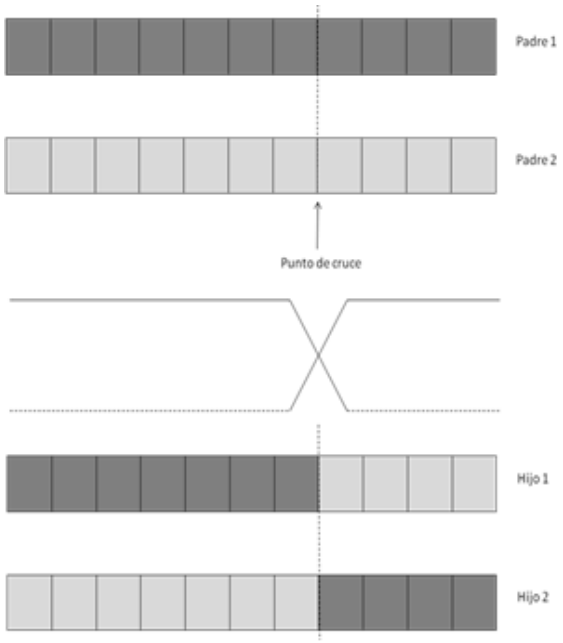

Figura 3.2. Cruce de un punto

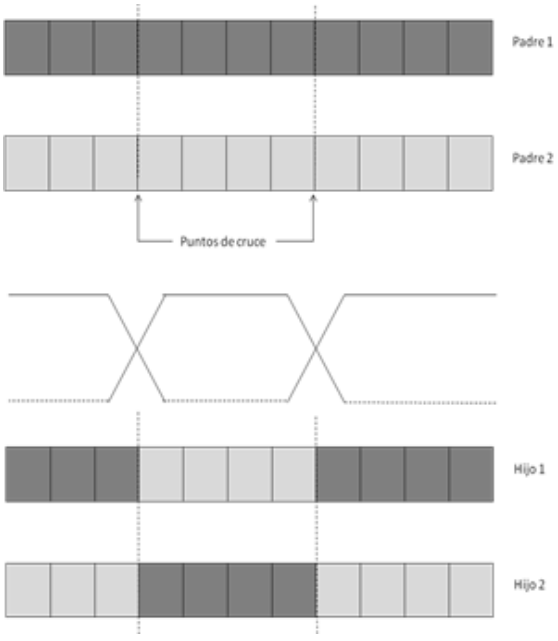

Figura 3.3. Cruce de dos puntos

Se pueden seguir añadiendo puntos de corte pero no siempre es aconsejable ya que puede dar lugar a un peor rendimiento del algoritmo. Al partir los cromosomas en demasiadas partes, y ser estas más pequeñas, es posible que por separado pierdan las características que poseían conjuntamente.

- Cruce uniforme

  Los puntos de corte se determinan por un patrón elegido aleatoriamente, por lo que cada gen de la descendencia tiene las mismas probabilidades de pertenecer a uno u otro padre. Si el patrón contiene un uno, el gen situado en esa posición se copia del primer padre, (Figura 3.4). Si hay un cero el gen se copia del segundo padre. El número efectivo de puntos de cruce es fijo pero suele hacerse, por término medio, L/2, siendo L la longitud del cromosoma.

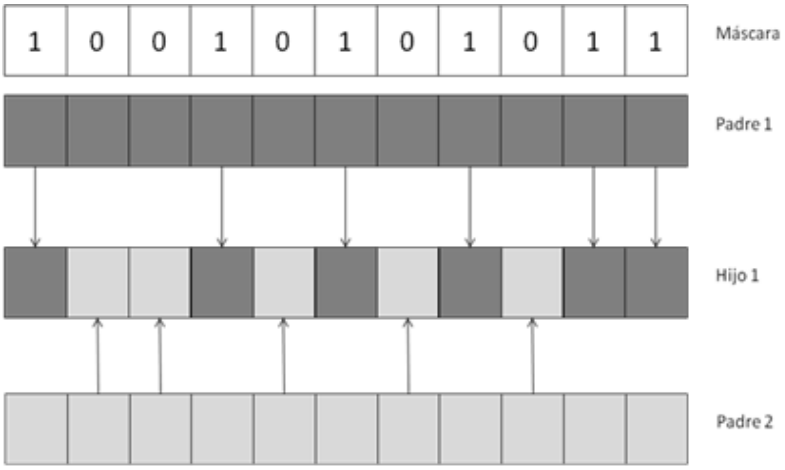

Figura 3.4. Cruce uniforme

Los tres tipos de cruce vistos son válidos para cualquier tipo de representación del genotipo. Si se emplean genotipos compuestos por valores enteros o reales pueden definirse otro tipo de operadores de cruce:

- **Media**: sólo se genera un descendiente que toma el valor medio de los genes de los padres.
- **Media geométrica**: cada gen de la descendencia toma como valor la raíz cuadrada del producto de los genes de los padres. Puede haber problemas con los signos de los padres.
- **Extensión**: se toma la diferencia existente entre los genes situados en las mismas posiciones de los padres y se suma al valor más alto o se resta del valor más bajo. Solventa el problema de generar un único descendiente.

3.3.2.3. Algoritmos de reemplazo

Cuando en vez de trabajar con una población temporal se hace con una única población, sobre la que se realizan las selecciones e inserciones, deberá tenerse en cuenta que para insertar un nuevo individuo deberá de eliminarse previamente otro de la población. Existen diferentes métodos de reemplazo:

- **Aleatorio**: El nuevo individuo se inserta en un lugar cualquiera de la población
- **Reemplazo de padres**: Como la descendencia suele conservar la bondad de los padres, el reemplazar a estos ayuda a conservar la diversidad y a evitar restringir el espacio de búsqueda
- **Reemplazo de similares**: una vez obtenido el ajuste de la descendencia se selecciona un grupo de individuos (entre seis y diez) de la población con un ajuste similar. Se reemplazan aleatoriamente los que sean necesarios
- **Reemplazo de los peores:** de entre un porcentaje de los peores individuos de la población se seleccionan aleatoriamente los necesarios para dejar sitio a la descendencia

3.3.2.4. Copia

La copia es la otra estrategia reproductiva para la obtención de una nueva generación a partir de la anterior. A diferencia del cruce, se trata de una estrategia de reproducción asexual. Consiste simplemente en la copia de un individuo en la nueva generación.

El porcentaje de copias de una generación a la siguiente debe ser relativamente reducido, pues en caso contrario se corre el riesgo de una convergencia prematura de la población hacia ese individuo. De esta manera el tamaño efectivo de la población se reduciría notablemente y la búsqueda en el espacio del problema se focalizaría en el entorno de ese individuo. Lo que generalmente se suele hacer es seleccionar dos individuos para el cruce, y si éste finalmente no tiene lugar, se insertan en la siguiente generación los individuos seleccionados.

3.3.2.5. Elitismo

El elitismo es un operador genético que selecciona los individuos más adaptados o con mejor *fitness* de una población y los transfiere automáticamente a la siguiente; de esta forma se garantiza que los mejores individuos no se van a perder producto de la manipulación genética de los otros operadores.

El número de individuos de élite se escoge de forma que se mantenga un equilibrio entre los individuos que intercambian material genético y los que se conservan para no perder diversidad, evitando caer en óptimos locales.

3.3.2.6. Mutación

La mutación tiene como objetivo cambiar alguna de las características de alguno de los individuos para introducir diversidad en la población. La mutación introduce cambios aleatorios en los genes de algunos individuos de forma aleatoria. Se suele realizar de manera conjunta con el operador de cruce. Primeramente se seleccionan dos individuos de la población para realizar el cruce. Si el cruce tiene éxito entonces uno de los descendientes, o ambos, se muta con cierta probabilidad P_m. Se imita de esta manera el comportamiento que se da en la naturaleza, pues cuando se genera la descendencia siempre se produce algún tipo de error, por lo general sin mayor trascendencia, en el paso de la carga genética de padres a hijos.

La probabilidad de mutación debe ser muy baja, generalmente menor al 1%. Esto se debe sobre todo a que los individuos suelen tener un ajuste menor después de mutados. Sin embargo, se realizan mutaciones para garantizar que ningún punto del espacio de búsqueda tenga una probabilidad nula de ser examinado. Así pues se puede decir que los algoritmos genéticos tratan de alcanzar un equilibrio entre dos objetivos aparentemente en conflicto: explotar las buenas cualidades de las potenciales soluciones y explorar el espacio de búsqueda.

Tal y como se ha comentado, la mutación más usual es el reemplazo aleatorio. Este consiste en variar aleatoriamente un gen de un cromosoma. Si se trabaja con codificaciones binarias consistirá simplemente en negar un bit. También es posible realizar la mutación intercambiando los valores de dos alelos del cromosoma.

3.4. Sistemas de Partículas

3.4.1. Introducción

Dentro de los algoritmos basados en la naturaleza, conjunto al que pertenecen los algoritmos genéticos, se engloba un grupo llamado Inteligencia de Enjambre, SI (Swarm Intelligence) por sus siglas en inglés. Los algoritmos del tipo SI se basan en el comportamiento que distintos grupos sociales tienen en la naturaleza, como por ejemplo, las colonias de hormigas, las manadas de vertebrados, los enjambres de abejas, los bancos de peces o las bandadas de pájaros.

Estos algoritmos se basan en los principios básicos que gobiernan el comportamiento dentro de sociedades complejas y la explotación de las formas de comunicación locales

y altamente distribuidas de control. De este modo, el estudio y adaptación de estos patrones naturalmente desarrollados y optimizados constituye un poderoso modelo que simplifica los procesos de resolución de problemas compuestos por un gran número de variables o que tienen un conjunto de soluciones altamente distribuido. Este tipo de aproximaciones ya han sido aplicadas con éxito en campos sobre todo de la Inteligencia Artificial, como son la robótica o la optimización de los análisis de flujos de carga, creando las llamadas *Smart Grids*, así como en multitud de otros problemas dentro del campo de la bilogía y la medicina.

Los algoritmos basados en la inteligencia del enjambre han sido empleados para resolver problemas de optimización en los que los métodos exactos y analíticos no son aplicables. Esto puede ser por la cantidad y tipo de las variables del problema o por las características de la propia función objetivo y/o sus restricciones. Siendo ésta una de las características que ha motivado el estudio de este tipo de algoritmos en la resolución del problema que ha motivado este trabajo.

Además, los procesos de resolución basados en la inteligencia colectiva, tienen una principal ventaja frente a otros métodos de optimización matemática. Sólo requieren conocer los valores que toma la función objetivo para cada una de las soluciones candidatas para poder proponer nuevas soluciones, mejores soluciones. Así pues, características de la función objetivo, requeridas por otros algoritmos, tales como diferenciabilidad y continuidad no son necesarias. De aquí la gran flexibilidad de este tipo de algoritmo para enfrentarse a una amplia variedad de problemas de optimización.

Dentro de los algoritmos basados en la SI se engloban los algoritmos basados en PSO (Particle Swarm Optimization) [17], [18].

3.4.2. Introducción al algoritmo PSO

Como se ha comentado, el algoritmo de optimización por cúmulo de partículas (Particle Swarm Optimization), es un subgrupo dentro de los algoritmos basados en la Inteligencia de Enjambre. El concepto básico del PSO está inspirado, tal y como lo describen sus creadores, en el comportamiento social de las bandadas de pájaros o los bancos de peces en su búsqueda de alimento y refugio, así como de los mejores lugares donde llevar a cabo el apareamiento y cría de sus retoños, teniendo estos lugares la especial característica de que no tiene porqué ser conocidos previamente por los individuos. Cada miembro de la red social es afectado, además de por su propia experiencia, por la experiencia de sus vecinos [19]. Así pues se desarrolla una técnica de computación evolutiva paralela que proporciona un modelo de búsqueda basado en la población [16], [17] y [18].

Este algoritmo fue originalmente desarrollado en 1995, en EE.UU., por el sociólogo James Kennedy y por el ingeniero Russ C. Eberhart.

Cada individuo, solución del problema, recibe el nombre de partícula. Ésta se va moviendo en un espacio multidimensional que representa su espacio social. Debido a su planteamiento, este tipo de algoritmo se adapta muy bien a problemas matemáticos, tanto de carácter continuo como discreto [19].

En palabras de los propios autores [18]:" Los individuos (partículas) que conviven en una sociedad tienen una 'opinión' que es parte del espacio de búsqueda, compartido por todos los individuos. Cada individuo puede modificar su opinión según tres factores:

- El conocimiento del entorno o ADAPTACIÓN.
- Experiencias anteriores del individuo o MEMORIA DEL INDIVIDUO.
- Experiencias anteriores de los individuos del vecindario o MEMORIA DEL VECINDARIO ".

Los individuos adaptan o modifican sus opiniones según aquella que le proporcionó más éxito, conocimiento propio, y aquellas que tienen más éxito dentro de su vecindario, conocimiento colectivo.

3.4.3. Funcionamiento

La población inicial que configura el enjambre (conjunto inicial de soluciones) se crea de forma aleatoria y evoluciona a lo largo del proceso iterativo de resolución. En esta evolución, el número de partículas permanece constante. No se tienen en cuenta los supuestos nacimientos (adiciones) y/o muertes (eliminaciones) de miembros del enjambre que sucederían en la naturaleza. Es decir, el proceso evolutivo se lleva a cabo con operadores de movimiento, pero no empleando operadores de cruce, copia y/o mutación como otros algoritmos inspirados en la naturaleza.

Es una técnica estocástica de optimización global, que se desarrolla en fases: inicialización y transformación.

Cada partícula tiene una pequeña memoria que le permite recordar su mejor posición encontrada hasta el momento. Las partículas son afectadas por su propia experiencia (mejor posición encontrada) y por la experiencia de sus vecinos (mejor posición encontrada por sus vecinos). El comportamiento de una partícula se describe según:

$$v_{id}(t+1) = \omega \cdot v_{id}(t) + lrn_1 \cdot rand_1 \cdot \left(p_{id}(t) - x_{id}(t)\right) + lrn_2 \cdot rand_2 \cdot \left(p_{gd}(t) - x_{id}(t)\right) \quad (3.15)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad (3.16)$$

En (3.15), v_{id} es la velocidad de la partícula i en la dimensión d. El primer término de la suma en (3.15) representa la fuerza de la inercia que empuja la partícula en su dirección más reciente, donde ω es el peso que controla esta fuerza de inercia. El segundo sumando en (3.15) corresponde a la experiencia cognitiva o personal. Este componente atrae a la partícula desde su posición actual, x_{id}, hacia la mejor posición encontrada por ella hasta el momento en esa dimensión, p_{id}. Para garantizar una búsqueda efectiva por todo el espacio de soluciones, se emplean unos coeficientes. En este caso esos coeficientes son el peso de aprendizaje propio, lrn_1, y una variable aleatoria uniformemente distribuida entre [0,1], $rand_1$. El tercer sumando representa la influencia social de los vecinos sobre la partícula. Este componente atrae a la partícula desde su posición actual, x_{id}, hacia la mejor posición encontrada por sus vecinos hasta el momento en esa dimensión, p_{gd}. Esta influencia es controlada, al igual que para el caso propio, por unos coeficientes, siendo esta vez lrn_2 y $rand_2$, respectivamente.

En cada iteración, como se describe en (3.16), cada partícula se mueve dando un paso de valor v_{id} en la dimensión d.

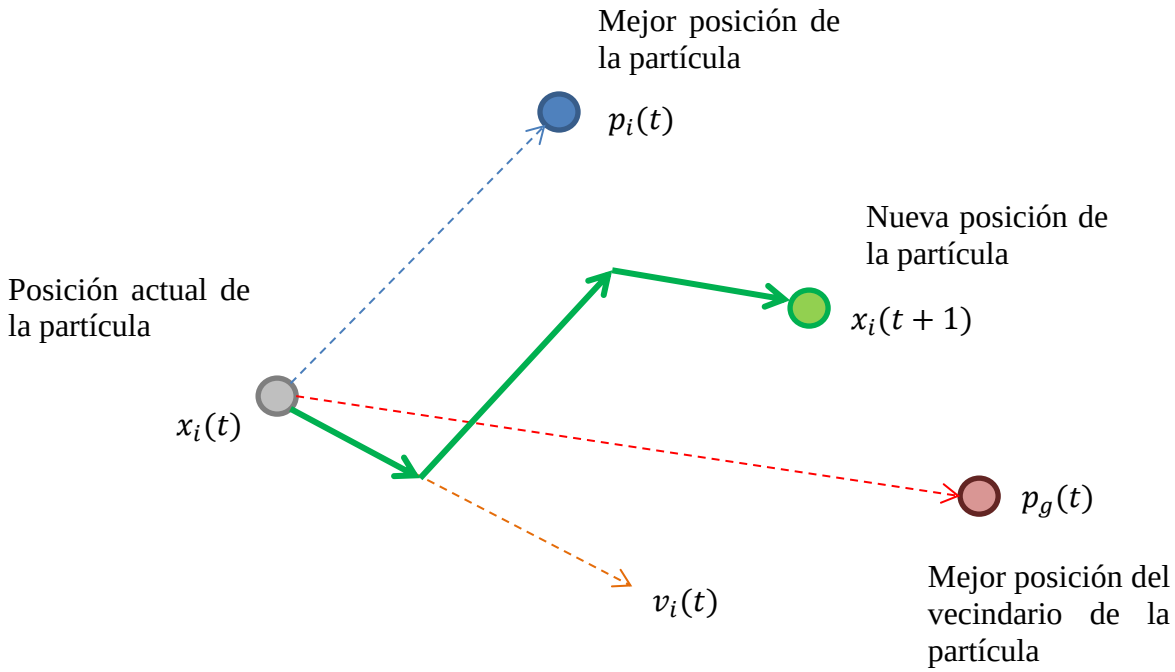

Figura 3.5. Representación gráfica del movimiento de una partícula

3.4.4. Tipos de PSO

Los algoritmos PSO han evolucionado desde que fuesen propuestos por Kennedy y Eberhart en 1995. El peso de la inercia no se incluía en la versión original. Fue añadido más tarde y se han publicado diversos estudios examinando los efectos de variar su valor [18].

La velocidad se ha limitado, para evitar la explosión de las velocidades de las partículas. Este límite será descrito en mayor profundidad más adelante en este trabajo, pues resulta de importancia primordial en las soluciones ofrecidas por el algoritmo, sobre todo cuando se trabaja con parámetros que pueden tomar valores muy diversos entre sí, incluso en orden de magnitud.

Los algoritmos PSO operan en tres espacios, el social, el paramétrico de las variables del problema y el de evaluación donde se determina la bondad de las soluciones según una función de evaluación o fitness.

Dentro del espacio social, originalmente, aparecieron dos versiones: una local y otra global.

En el caso global, PSO-g, todas las partículas están conectadas entre sí. Cada partícula sólo está influenciada por la mejor partícula de todo el grupo. Este tipo converge muy rápido, pero su debilidad estriba en la facilidad con la que cae en mínimos locales.

La otra variante original, el caso local, llamado PSO-l, organiza su estructura social como un anillo. En este anillo, cada partícula únicamente está conectada a dos partículas más. Este algoritmo converge más lentamente. Sin embargo, tiene una menor tendencia a caer en mínimos locales, además de favorecer una mayor diversidad de partículas. La influencia de cada partícula está limitada a sus dos más inmediatos vecinos, la partícula precedente y la sucesora en la estructura circular.

PSO local PSO global

Figura 3.6. Variantes originales del algoritmo PSO

Estas dos variedades de PSO están basadas en vecindarios estáticos. Al inicio de toda búsqueda se requiere una exploración de posibles soluciones, por lo que PSO-l es más adecuado al ampliar el espacio de búsqueda. En etapas posteriores, sin embargo, es preferible una explotación de las mejores soluciones encontradas en las primeras etapas de búsqueda, lo cual hace mejor el PSO-g. El carácter de la búsqueda también puede ser modificado variando el peso del término de inercia [20]. Valores pequeños promueven la exploración local, mientras que valores elevados favorecen una búsqueda más global.

Por este y otros motivos, algunos investigadores propusieron la implementación de vecindarios dinámicos.

En [21] se describe un vecindario que comienza con cada partícula conectada a sí misma, y que poco a poco se expande hasta conectar todas las partículas entre sí. El comportamiento de este algoritmo evoluciona desde un carácter claramente local hacia uno global.

Otros trabajos sugieren el empleo de un peso de inercia dinámico, que comience con un valor elevado y termine con un valor más pequeño, [18].

Cuando se trabaja con problemas con muchas variables, PSO puede quedar atrapado en un mínimo local con bastante frecuencia. Para solucionar esto, se han propuesto diversas alternativas como por ejemplo: añadir una partícula reina [24], la variación de la topología del vecindario [16], la introducción de una subpoblación y dotar a las partículas de una extensión física [25], etc. Dentro de estos últimos algoritmos se encuadra el Clubs-based PSO. Esta variante de PSO se basa en los clubes sociales para modelar su vecindario.

En este trabajo se han empleado dos versiones de PSO para comparar sus resultados. Una de ellas basada en los vecindarios estáticos, PSO-l con inercia dinámica, y otra en los vecindarios dinámicos, Club-based PSO. Este último se describe con mayor detalle en el próximo apartado.

3.4.4.1. Club-based PSO

La visión del enjambre de partículas como un grupo animal ha limitado la imaginación de los investigadores durante un tiempo. Recientemente, se le considera como un grupo de partículas, ya sean humanas, animales, u otra estructura cualquiera que tenga un comportamiento social [19].

Así en el algoritmo Club-based PSO (C-PSO), se crean clubes para partículas, análogos a los clubes sociales donde los humanos quedan e interaccionan. En este modelo una partícula cualquiera puede unirse a más de un club, y los clubes no tienen límites de aforo. También se permiten los clubes vacíos.

Después de inicializar aleatoriamente las posiciones y velocidades de las partículas, cada partícula se une a un número predefinido de clubes, llamado *número de afiliaciones por defecto*. La elección de estos clubes es aleatoria.

Cuando se evalúa el fitness de cada partícula, hay que tener en cuenta que ahora el vecindario de esa partícula lo componen aquellas partículas que pertenecen a los mismos clubes que la partícula en cuestión.

El proceso de actualización de posición y velocidad es idéntico al descrito más arriba para los algoritmos PSO.

Mientras se busca la solución óptima del problema, si una partícula muestra una adaptación superior a todas las partículas de su vecindario, se reduce la expansión de su influencia a través de la disminución de su número de afiliaciones. Es decir, se saca dicha partícula de uno de los clubes, elegido al azar, a los que pertenece. Con esta medida, se busca evitar una convergencia prematura. Por otro lado, si la aptitud de una partícula es la peor de todo su vecindario, se aumenta el número de membresía de dicha partícula. En otras palabras, se añade dicha partícula a otro club, elegido aleatoriamente, al que no perteneciera. Así se incrementa su red social, y por lo tanto, las oportunidades de aprender de mejores partículas.

Este proceso de adhesión a clubes y de salida de clubes se repite en cada iteración del algoritmo. De forma que si una partícula se mantiene como la peor de su vecindario, ésta irá aumentando su número de afiliaciones hasta que alcance el *número máximo de afiliaciones*. Del mismo modo, si una partícula se mantiene como la mejor de su

vecindario, ésta irá reduciendo su número de afiliaciones hasta alcanzar el *número mínimo de afiliaciones*.

Al mismo tiempo, aquellas partículas que no sean ni las mejores ni las peores de su vecindario, irán aumentando o disminuyendo su número de afiliaciones hasta alcanzar el *número de afiliaciones por defecto*. La velocidad a la que las partículas retornan al número de afiliaciones por defecto, es menor que la velocidad a la que las mejores y peores partículas disminuyen y aumentan, respectivamente, su número de afiliaciones. Es decir, mientras esto último se realiza en cada iteración, aquellas partículas que no muestran comportamientos extremos, no son las mejores ni las peores, verán modificado su número de afiliaciones cada "tr" iteraciones, siendo "tr" la tasa de retención, fijada por el usuario. El hecho de que esta velocidad sea menor, permite a la partícula quedarse durante más tiempo en los clubes, de forma añade estabilidad y suavidad al desarrollo del algoritmo.

A continuación, se presenta un ejemplo que, acompañado una figura, facilita la comprensión del algoritmo expuesto.

En este ejemplo la nube está compuesta por 8 partículas, y hay 6 clubes disponibles a los que unirse. El número de afiliaciones mínimo es 2, por defecto es 3 y el máximo 5. Se producen los siguientes cambios, que serán efectivos en la siguiente iteración, teniendo en cuenta que la iteración actual es un múltiplo de "tr":

1. La partícula 3 dejará el club 1, 2 ó 3 ya que es la mejor partícula en su vecindario (los clubes 1, 2 y 3).
2. La partícula 5 se unirá al club 1, 2 ó 4 porque es la peor partícula en su vecindario (los clubes 1, 2 y 4).
3. La partícula 2 dejará el club 1, 2, 3 ó 4 mientras que la partícula 4 se unirá al club 2, 3, 4 ó 6 para acercarse hacia el número de clubes por defecto pues ambas partículas no tiene una actuación extrema. No son ni las mejores ni las peores de sus respectivos vecindarios.

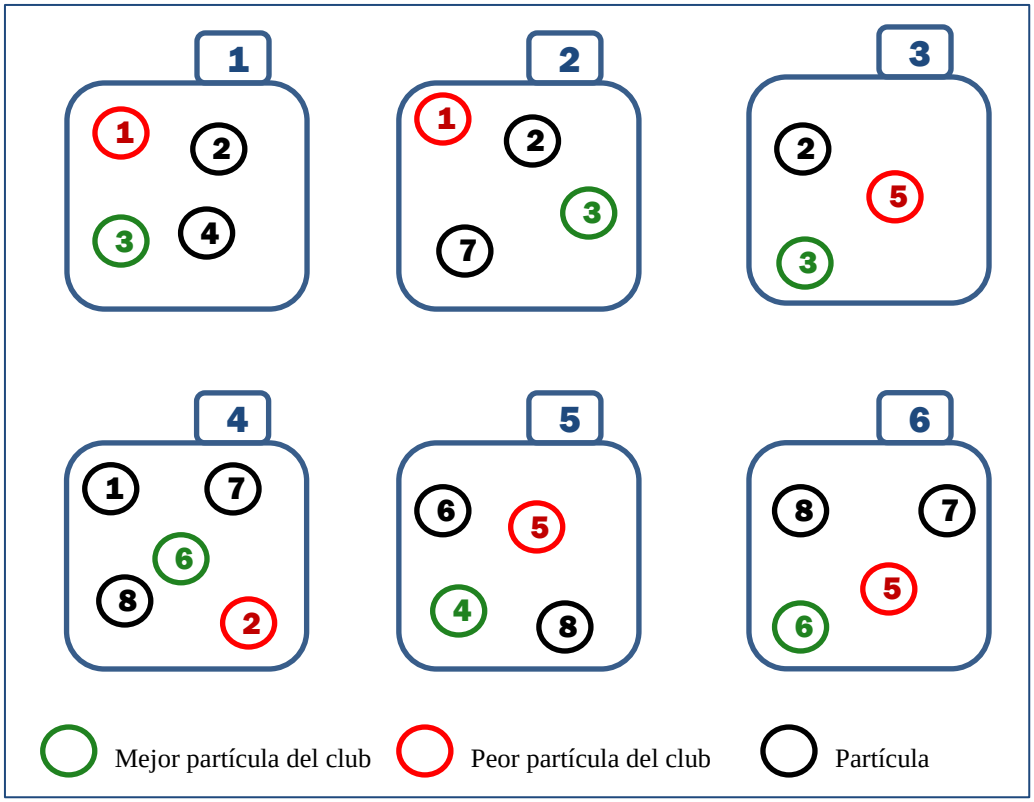

Figura 3.7. Situación de los clubes durante la simulación de C-PSO

Se supondrá que, elegidos al azar, la partícula 3 dejará el club 2; la partícula 5 se unirá al club 1; la partícula 2 dejará el club 2; la partícula 4 se unirá al club 6.

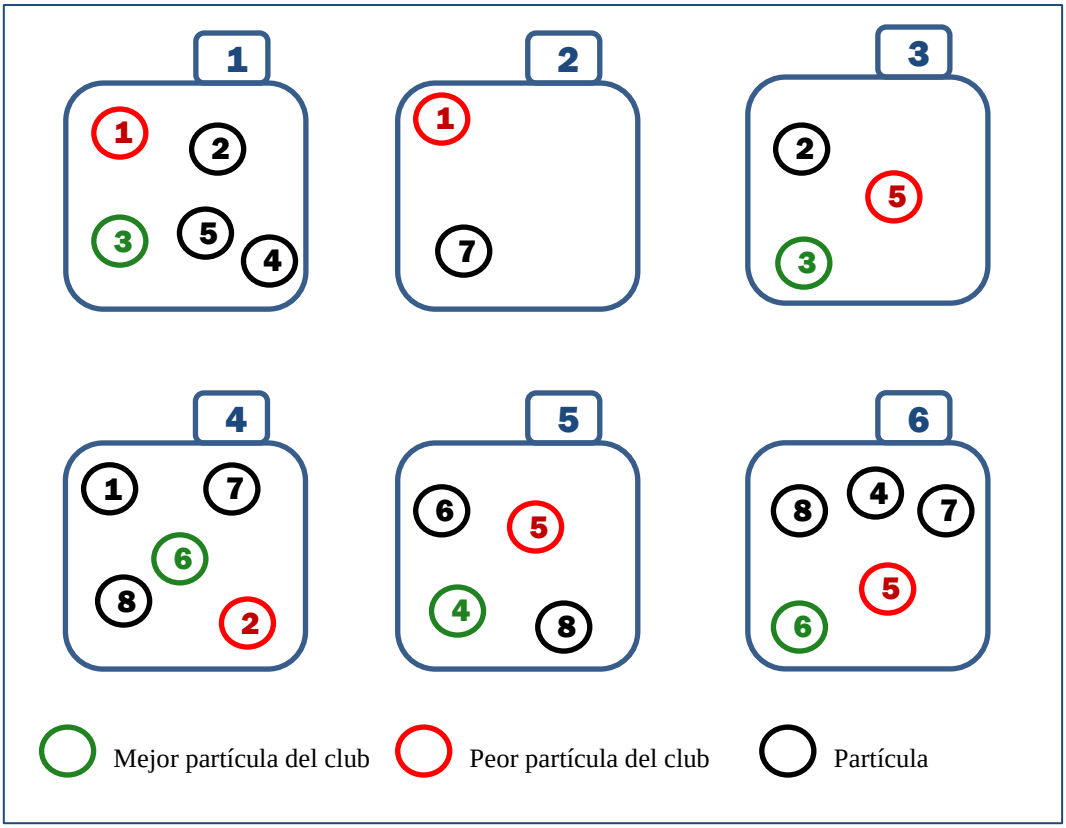

Figura 3.8. Situación de los clubes tras realizar las actuaciones descritas

Tras llevar a cabo estas actuaciones, para determinar quiénes son la mejor y la peor partícula de cada club, habría que volver a evaluar la aptitud de cada partícula.