Universidad Politécnica
de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Máster Universitario en Inteligencia Artificial

Trabajo Fin de Máster

Sobre el funcionamiento interno del PSO y construcción de nuevo método

Autor(a): David Erroz Arroyo
Tutor(a): Nik Swoboda

Madrid, Julio - 2022

Este Trabajo Fin de Máster se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Máster
Máster Universitario en Inteligencia Artificial

Título: Sobre el funcionamiento interno del PSO y construcción de nuevo método

Julio - 2022

Autor(a): David Erroz Arroyo
Tutor(a): Nik Swoboda
Inteligencia Artificial
ETSI Informáticos
Universidad Politécnica de Madrid

Resumen

Al ser el Particle Swarm Optimization (PSO) una metaheurística, sus fundamentos teóricos no son del todo rigurosos por no tener sus bases en la optimización matemática. Sabemos la metáfora que lo inspira y por ende la idea que hay detrás de su funcionamiento, pero desconocemos el funcionamiento exacto del método. No sabemos a priori por qué es tan efectivo para resolver problemas de optimización ni lo que hace que lo sea. Tampoco sabemos con precisión cuál es el rol de sus hiperparámetros en la optimización, cómo influyen éstos en el comportamiento del algoritmo y de qué manera afectan su rendimiento. Esto por supuesto es un condicionante bastante grave ya que nos limita mucho a la hora de tener que realizar la configuración óptima de hiperparámetros para nuestro problema o a la hora de investigar nuevas variantes o adaptaciones del PSO. En este trabajo se revisarán diferentes análisis publicados a lo largo de estos años sobre el método, algunos de los cuales derivan en intentos de mejora del estándar, todo con el fin de solucionar estos problemas. Además, también se presentará un análisis propio que da origen a un nuevo método PSO, que será a su vez comparado con otras versiones mejoradas del mismo. El nuevo método propuesto no solamente tiene por objetivo alcanzar un mejor rendimiento que el original, sino también facilitar el control del usuario sobre el proceso de optimización.

Abstract

As the Particle Swarm Optimization (PSO) is a metaheuristic, its theoretical foundations are not entirely rigorous because they are not rooted in mathematical optimization. We know the metaphor that inspires it and thus the idea behind its working, but we do not know the exact working of the method. We do not know a priori why it is so effective in solving optimization problems or what makes it so. We do not know with precision which is the role of its hyperparameters in the optimization task, how do they influence the algorithm's behaviour and in which way they affect its performance. This of course is a quite serious drawback since it limits us a lot when it comes to doing the optimal configuration of the hyperparameters for our problem or when investigating new variants or adaptations of the PSO. In this work different analyses about the method published along these years will be reviewed, some of which lead to improvement attempts of the standard, all in order to solve these problems. Moreover, an own analysis that gives rise to a new PSO method will also be presented, being this new PSO in turn compared to other improved versions of the original. The proposed new method is not only aimed at achieving better performance than the original one, but also to facilitate user control over the optimization process.

Tabla de contenidos

Capítulo 1

Introducción

El método de Optimización por Enjambre de Partículas (PSO por sus siglas en inglés) [6] es una de las metaheurísticas más utilizadas para la resolución de problemas de optimización continua. Sus bases tan intuitivas, su buen rendimiento y lo sencillo y simple de implementar que resulta ser, es lo que ha hecho que haya acaparado tanta atención desde que fuera propuesto (1995) hasta ahora. Durante todo este tiempo, el PSO ha sido aplicado a una gran variedad de problemas de optimización matemática de la vida real, que incluyen muchos campos diferentes [25, 26, 27, 28]. Además, las investigaciones sobre el método no solo se centran en la aplicación y evaluación de éste en diferentes problemas, sino que también se centran en su teoría y en estudiar posibles versiones distintas del método, mejoras de algún tipo, etc. Es por eso que desde su publicación han surgido diversas variantes [14, 16, 17] y adaptaciones del mismo [19, 21, 22].

Sin embargo, a pesar de ser un método muy eficiente en todos los sentidos y con un rango de aplicación elevadísimo, también tiene sus puntos negativos. Quizá el más evidente sea el de su gran número de hiperparámetros. El PSO tiene tres hiperparámetros principales que rigen su comportamiento y la adecuada elección de sus valores es crucial para el buen rendimiento del algoritmo. Esto se debe a que el método es sensible a cambios de sus hiperparámetros y diferentes configuraciones de éstos llevan a diferentes comportamientos del método y por tanto, a resultados distintos [141, 142]. Así, mientras que con una cierta configuración demostrará un muy buen rendimiento para algún problema de optimización, para otros no y viceversa con otra configuración. En definitiva, la manera de configurarlos es de gran importancia y por eso muchos focos de investigación se han puesto aquí. Al proceso de selección del valor de los hiperparámetros se le conoce como *hyperparameter tuning, hyperparameter optimization* o *hyperparameter selection*.

Al problema de selección de hiperparámetros hay que sumarle otro punto negativo del PSO: su comprensibilidad. El PSO, como cualquier metaheurística, tiene una inspiración intuitiva pero carece de fundamento teórico pleno. Es decir, se sabe la idea y las bases que hay detrás del método pero no cómo ni por qué funciona realmente para optimizar o qué es lo que hace que funcione tan bien [43, 120]. Como consecuencia de esta falta de comprensión, tampoco se conoce muy bien el efecto o la influencia que tienen los hiperparámetros en el comportamiento del método. Esto guarda relación directa con el punto anterior, ya que complica bastante el *hyperparameter tuning*.

De todos modos, como decíamos, hay muchos trabajos de investigación sobre el *hyperparameter selection*. Estos trabajos tratan o bien de aportar pautas generales para la selección o de proporcionar técnicas que automaticen este trabajo y mantengan al margen de todo el proceso de optimización al usuario. Estas técnicas pueden basarse en la experimentación o en el análisis. Las primeras realizan algún tipo de optimización matemática sobre los hiperparámetros, véase búsqueda aleatoria, exhaustiva en conjuntos finitos de configuraciones (grid search) o incluso búsquedas guiadas por metaheurísticas como el propio PSO [29]. Las segundas obedecen a algún tipo de análisis previo sobre el método, generalmente a análisis teóricos sobre la función de cada hiperparámetro [31], aunque también las hay basadas en análisis empíricos [30]. El segundo tipo de técnicas es mucho más eficiente porque no requiere de múltiples ejecuciones del algoritmo, motivo por el cual el grueso de la investigación se centra en ellas. De hecho, dentro de este grupo de técnicas no solo se estudian las que establecen configuraciones estáticas, sino que también las hay que permiten configuraciones dinámicas que cambian en el tiempo. Estas últimas cambian el valor de los hiperparámetros durante la ejecución ya sea según la iteración (técnicas dinámicas) [32] o según el estado de la simulación (técnicas adaptativas) [33]. Esto permite un mayor control aún sobre el comportamiento del PSO y puede traducirse como una mejora en la personalización y rendimiento del método para nuestro problema específico.

En resumidas cuentas, parece claro que para diseñar una técnica genérica de selección de hiperparámetros eficiente se necesita entender bien el rol de los hiperparámetros y de qué manera afectan al comportamiento del método. Así, de una forma u otra, el *hyperparameter tuning* requiere de un cierto nivel de comprensión del PSO. También se necesita este nivel de comprensión para diseñar nuevas variantes del método, mejores adaptaciones, etc. Es por eso que en este trabajo se pretende alcanzar un buen nivel de comprensión del PSO revisando los análisis teóricos de mayor relevancia propuestos hasta la fecha, así como las propuestas de mejora que muchos de ellos incluyen. También en este trabajo se presenta un análisis propio que deriva en un nuevo método PSO en el que los hiperparámetros tienen una función clara y bien definida, que incluso nos permite reducir el número de hiperparámetros del método facilitando así la labor de selección de hiperparámetros, el diseño de nuevas técnicas adaptativas o de nuevas variantes y adaptaciones. El nuevo método se vale también del análisis para introducir modificaciones del original que se han considerado beneficiosas, por lo que también será comparado con el PSO estándar y con otras variantes surgidas asimismo de análisis teóricos.

El trabajo está estructurado de la siguiente forma: en el Capítulo 2 se explican todos los conceptos preliminares necesarios para abordar el trabajo. En el Capítulo 3 se realiza un Estado del Arte sobre los análisis teóricos del PSO. En el Capítulo 4 se presenta el análisis propio y se propone el nuevo método derivado del análisis. En el Capítulo 5 se experimenta con el método propuesto para constatar sus beneficios y se discuten los resultados obtenidos. Finalmente, en el Capítulo 6 se extraen conclusiones sobre el trabajo y se indican las futuras líneas de investigación que se abren gracias a él.

1.1. Motivación

Lo que ha motivado este trabajo ha sido la dificultad que presenta el método (principalmente por su naturaleza estocástica) para comprender cómo funciona en términos de optimización y por ende, qué impacto tienen sus hiperparámetros en términos de exploración-explotación. Esto se ve reflejado en los múltiples trabajos que existen sobre técnicas de selección de hiperparámteros o sobre nuevas variantes y complica muchísimo la labor. La mayoría de artículos se centran en analizar el rol de cada hiperparámetro de manera independiente, lo cual no es ideal si se quiere alcanzar un conocimiento muy elevado sobre los mismos para poder realizar configuraciones paramétricas precisas que requieren el ajuste simultáneo de todos ellos. Además, pocos son los que profundizan en la utilidad de cada elemento del método aportando detalles concretos de su impacto en el proceso de optimización (por ejemplo de la aleatoriedad). Digamos que la información en torno al funcionamiento metafórico del método es abundante pero no lo es tanto la relativa al funcionamiento práctico de la optimización (cuándo las partículas pierden diversidad, en qué situaciones existen sesgos, si son realmente necesarios todos los elementos del método, cuáles son los más esenciales y por qué, qué función concreta desempeñan que resulta clave para el buen devenir del proceso optimizativo, etc). Todas estas cuestiones y más surgen debido a que el método no tiene sus raíces teóricas en la optimización matemática (es bioinspirado) y algunas tratan de ser respondidas por la comunidad investigadora. Apoyándonos en esta información y tratando de aportar nueva pretendemos alcanzar los objetivos del trabajo.

1.2. Objetivos

- Esclarecer la función de cada hiperparámetro del PSO y sus mecanismos de funcionamiento
- Tratar de dar con un método sin tantos hiperparámetros y sobre todo cuya función no sea tan confusa
- Intentar detectar defectos en el PSO, corrigiéndolos en el nuevo método propuesto
- Comparar rendimientos de distintos métodos PSO surgidos de análisis con el estándar y el propuesto

Capítulo 2

Preliminares

2.1. Optimización matemática

La optimización matemática es una rama de las matemáticas aplicadas que se encarga de resolver problemas de optimización. Un problema de optimización consiste en encontrar la solución de entre un conjunto de soluciones candidatas que sea mejor de acuerdo con algún criterio preestablecido para un problema dado. Así pues, se necesita cuantificar de alguna manera cómo de buena es cada posible solución para así poder compararlas y decidir cuál es la mejor. Para ello, se expresa el criterio en forma de función. Esta función nos permite obtener el grado de satisfacción del criterio que tiene cada solución y por tanto, poder resolver el problema de optimización matemáticamente encontrando la solución que maximiza (o minimiza) la función.

En otras palabras, se ve un problema de optimización como un problema de minimización (o maximización) de funciones. Formalmente, un problema general de optimización se define como el problema de hallar, dado un conjunto S de posibles soluciones y una función $f : S \longrightarrow \mathbb{R}$, el valor $\mathbf{x}^* \in S$ tal que

$$f(\mathbf{x}^*) \leqslant f(\mathbf{x}), \quad \forall \mathbf{x} \in S.$$

Se denota

$$\min_{\mathbf{x} \in S} f(\mathbf{x}).^{1}$$

Por tanto, cualquier problema de optimización consta de:

- *Una función a minimizar:* También denominada *función objetivo* o *función fitness*, que evalúa la calidad de las soluciones devolviendo un valor que indica su ajuste al criterio establecido. A dicho valor se le conoce como valor *fitness* de la solución.
- *Un conjunto de variables o incógnitas:* Son las variables de las que depende el problema. Cada posible solución viene constituida por el valor de estas variables. Es decir, $\mathbf{x} = (x_1, ..., x_i, ..., x_d), \quad \mathbf{x} \in S$. Lógicamente, la función objetivo es función de estas variables y el problema de optimización consiste en encontrar

[1] Claramente se puede utilizar la definición recíproca de maximización, pero, por convenio, se suele hablar de minimización. Es indistinto ya que maximizar f es lo mismo que minimizar $-f$.

los valores de cada x_i que hacen que el *fitness* sea mínimo.[2] El número de variables marca la dimensionalidad del problema.

- *Una serie de restricciones:* No cualquier combinación de valores para las variables x_i constituye una solución válida para el problema. Únicamente son soluciones aquellas que satisfacen todas las condiciones requeridas.[3] En otras palabras, para ser x solución ha de pertenecer al conjunto S, o sea, al dominio de la función objetivo. Al conjunto S se le conoce como *espacio de soluciones*, en el que cada posible solución viene representada por un punto en dicho espacio. Ahora bien, la forma de expresar S consiste en partir de algún conjunto numérico básico A^d del cual S sea subconjunto ($S \subseteq A^d$) como puede ser $\Re^d$ e imponerle algunas *restricciones* que cumplan exclusivamente los elementos de S. Estas restricciones pueden expresarse como un sistema de ecuaciones e inecuaciones. En definitiva, S se define como $S = \{\mathbf{x} \in A^d \mid g_j(\mathbf{x}) \leqslant 0 \wedge h_k(\mathbf{x}) = 0, \forall j \in \{1, ..., J\}, \forall k \in \{1, ..., K\}\}$, con g, h funciones de x utilizadas para expresar las J inecuaciones y K ecuaciones. Y por tanto, la función objetivo $f : S \longrightarrow \Re$ puede verse como otra función $f_1 : D = A^d \longrightarrow \Re$ cuyo dominio D es sujeto a algún tipo de restricción de cara al problema de optimización, es decir, no todos los puntos de D serán válidos o tenidos en cuenta para el problema a pesar de que sí tengan imagen asociada. Claramente, si $S = A^d$ entonces decimos que el problema no tiene restricciones, por el contrario, si $S \subset A^d$ diremos que sí las tiene.

Se pueden realizar diversas clasificaciones para un problema de optimización. La principal es según el tipo de variables que posee. En este sentido, podemos dividir los problemas en dos categorías: los que disponen de variables discretas (optimización discreta), en los que la solución debe encontrarse en conjuntos contables[4] y por otro lado, los que disponen de variables continuas (optimización continua), que son los que trabajan con variables reales ($x_i \in \Re$) y funciones continuas. También los problemas pueden ser clasificados según su número de variables (dimensiones) o según si el dominio de la función objetivo ha sido o no restringido para el problema.[5] El tipo y forma de la función fitness marca asimismo el tipo de problema, si es lineal o no lineal, si es unimodal (solo tiene un mínimo local) o multimodal (más de un mínimo relativo) cuando trabajamos en optimización continua, etc.

La optimización matemática se emplea en multitud de campos distintos: en transportes, redes, ingeniería, economía, marketing, finanzas, etc. Es muy importante ya que permite dar con la decisión óptima para un problema modelándolo matemáticamente y resolviéndolo computacionalmente, por lo que resulta muchísimo más efectivo y eficiente que hacerlo simplemente por intuición. Además, hay veces en las que la decisión a tomar comporta un gran gasto de recursos tanto económicos como temporales, hecho que hace inasumible el poder ir probando con diferentes decisiones para medir su rendimiento a posteriori. En estos y en la mayoría de los casos que se dan hoy en día, probablemente la optimización matemática resulte ser la única opción considerable debido al increíble ahorro que supone.

Existen varias técnicas para resolver un problema de optimización matemática. La

[2] Por supuesto satisfaciendo la condición de formar una solución válida.

[3] No necesariamente dichas soluciones han de ser óptimas.

[4] En optimización discreta S es un conjunto contable, lo cual quiere decir que es finito o numerable.

[5] En optimización continua las restricciones se aplican sobre el conjunto $\Re^d$, siendo d el número de dimensiones del problema.

elección de unas u otras dependerá del tipo de problema que queramos afrontar. Principalmente existen dos tipos: las técnicas exactas y las técnicas aproximadas. Las primeras encuentran la solución óptima al problema empleando algoritmos de optimización exactos que terminan en un número finito de pasos, mientras que las segundas, lo hacen de manera aproximada a través de métodos de búsqueda de soluciones óptimas que nos guían hacia ellas pero cuyo número de pasos no está acotado teóricamente.[6] Lógicamente, los métodos exactos son más precisos a costa de sacrificar más tiempo y recursos computacionales que los inexactos. A la hora de evaluar qué técnica hemos de emplear para nuestro problema, tenemos principalmente en cuenta estos dos aspectos (precisión y tiempo) y habitualmente solemos estar dispuestos a aceptar una solución muy buena aunque no óptima con tal de obtenerla en un tiempo razonable. Para la resolución de problemas altamente complejos no nos suele quedar más remedio que recurrir a técnicas aproximadas.

Por ejemplo, en optimización continua, podríamos resolver el problema de forma exacta empleando métodos analíticos que nos permitieran de manera directa obtener la solución que buscáramos. El caso más simple sería resolviendo el sistema de ecuaciones $[\frac{\partial f}{\partial x_i}(x) = 0;\ \forall i \in \{1, ..., d\}]$, que nos permitiría hallar los extremos relativos de la función (puntos en los que el gradiente de la función es el vector nulo) para luego hallar el absoluto. Sin embargo, esto no resulta nada sencillo cuando hablamos de funciones no lineales, multimodales y multivariables. Para los casos más extremos, podría llevarnos años resolverlos de esta manera, por lo que no queda otra que resolverlos de forma aproximada. Para resolverlos aproximadamente se contemplan principalmente dos técnicas: los métodos numéricos o iterativos y las técnicas de búsqueda aleatoria guiada. Los métodos iterativos utilizan técnicas de cálculo y se caracterizan por converger iterativamente a una solución pudiéndose valer para ello de información como el gradiente de la función, teoremas para funciones continuas, etc.[7] Mientras, las otras técnicas son heurísticas que ni siquiera han de converger necesariamente pero pueden proveer soluciones aproximadas a algunos problemas. Además en este último conjunto de técnicas, existen unas de propósito general que se sitúan por encima de las heurísticas, denominadas metaheurísticas porque suponen una guía genérica de cómo resolver conjuntos amplios de problemas mediante técnicas heurísticas. Estas técnicas hacen suposiciones mínimas sobre los problemas a tratar y no requieren de apenas información sobre la función objetivo ni sobre el contexto. Allá donde los métodos numéricos no llegan, sí lo hacen las metaheurísticas. Por ejemplo, cuando no podemos disponer de información relativa a la función objetivo (ya sea su expresión, sus derivadas, etc) sino que ésta permanece como una *caja negra* para el problema o cuando esta función tiene muchos óptimos locales o el espacio de soluciones es gigante, entonces estas técnicas se vuelven muy importantes. Las metaheurísticas se emplean mucho también en problemas complejos de optimización discreta (problema del viajante, n-reinas, etc).

[6] Es el usuario en estos métodos el que tiene que decidir cuándo poner fin a la ejecución y definir un criterio para ello.

[7] Son ejemplos de métodos numéricos el método de Newton-Raphson, el método de la bisección o el método de descenso por gradiente.

2.2. Metaheurísticas

Como comentábamos en la sección anterior, cuando las técnicas exactas o de aproximación convergentes no pueden o no convienen ser aplicadas para la resolución de un problema de optimización, es cuando recurrimos a otro conjunto de técnicas, las cuales utilizan búsquedas guiadas para encontrar la solución óptima. Estas técnicas las conocemos como *heurísticas*.

El término *heurística* proviene del griego y se asocia con el concepto de *encontrar*, aunque la idea genérica del término se relaciona con la tarea de resolver inteligentemente problemas reales usando conocimiento. En Inteligencia Artificial se habla de *heurística* para referirse a los procedimientos que, empleando conocimiento acerca un problema, tratan de aportar soluciones (o acercarse a ellas) utilizando una cantidad de recursos razonables (generalmente tiempo). Así, en optimización matemática, una *heurística* es una técnica que, basándose en el conocimiento sobre el problema que está tratando, es capaz de desarrollar una búsqueda inteligente por el espacio de soluciones que le permita encontrar con un grado alto de confianza una solución más que aceptable (con un alto grado de optimalidad) en un tiempo más que razonable. Cabe apuntar que al *espacio de soluciones* también se le conoce como *espacio de búsqueda*, pues es en este espacio donde estas técnicas buscan la solución óptima. Cada punto de este espacio representa una solución y estas técnicas lo recorren tratando de encontrar el punto óptimo, guiándose para ello de la información que van recaudando durante la búsqueda ya que a priori no conocen todos los puntos del espacio sino que los van generando poco a poco.

Es importante resaltar que las heurísticas suelen ser algoritmos especializados, es decir, están centrados en un tipo de problema concreto del cual aprovechan su conocimiento para resolverlo. Sin embargo, existen multitud de tipos de problemas diferentes para los cuales nos gustaría disponer de métodos más generales que pudieran ser adaptados a los distintos problemas sin necesidad de desarrollar un método para cada tipo de problema. Es aquí donde aparece el término *metaheurística*. Desde su introducción por primera vez en [1] como una heurística superpuesta a otra[8], se han utilizado diversas definiciones para este término. Quizá la definición más convencional es la dada en [2]: "Son marcos algorítmicos de propósito general que proveen una serie de pautas o estrategias para desarrollar algoritmos heurísticos de optimización". La idea básica es que son estrategias generales de diseño de procedimientos heurísticos que guían el proceso de búsqueda, independientes del problema. A los algoritmos de optimización heurísticos construidos con las metaheurísticas, también se les denomina *metaheurísticos* y son procedimientos aproximados de propósito general que guían una heurística subordinada de búsqueda por el espacio de soluciones. Tienen la ventaja de ser algoritmos fácilmente implementables y con gran éxito en la práctica.

Las bases en las que se sustentan las metaheurísticas varían significativamente y muchas no tienen siquiera una base teórica establecida. La mayoría de ellas se inspira en comportamientos o fenómenos observados en la naturaleza, utilizando metáforas que aparentemente nada tienen que ver con optimización para precisamente modelar este proceso. Algunos ejemplos de esto son los algoritmos genéticos (GA por sus siglas en inglés) [3], basados en la evolución natural, o el algoritmo de recocido simulado (SA) [4], basado en el proceso de recocido del acero y las cerámicas para

[8]El prefijo *meta* proviene también del griego y significa "más allá", por lo que el concepto *metaheurística* hace referencia a algo que va más allá de una heurística.

obtener material cristalizado.

A parte de por la base que las inspira, las metaheurísticas pueden ser clasificadas atendiendo a muchos otros criterios. Uno de los más empleados es el del número de puntos del espacio de búsqueda sobre los que trabajan simultáneamente. Más concretamente, si la técnica procesa una única solución por iteración se dice que es basada en trayectoria y si por el contrario trabaja sobre un conjunto o población de soluciones se dice basada en población.[9] También se pueden clasificar según si disponen o no de memoria, esto es, a la hora de buscar nuevas soluciones, si tienen o no en cuenta información de iteraciones pasadas y no solo de la actual.[10] Otra posible clasificación sería por el uso o no de aleatoriedad (estocásticas vs deterministas) aunque la inmensa mayoría la usa.

Por último, cuando hablamos de técnicas de búsqueda en general y de metaheurísticas en particular, hay dos conceptos clave que determinan su rendimiento y resultan imprescindibles en su diseño: la *exploración* y la *explotación*. La *exploración* se refiere a la capacidad de buscar en diferentes partes del espacio con el fin de encontrar regiones prometedoras, esto es, zonas del espacio de búsqueda donde las soluciones candidatas tienen un fitness destacado (bajo en comparación con el resto). Por su parte, la *explotación* se refiere a la capacidad de buscar en regiones prometedoras con el objetivo de afinar la búsqueda y encontrar soluciones aún mejores dentro de la región. Nótese que no existe una frontera clara entre estos dos conceptos, ya que la diferencia depende del tamaño del área que se explora, lo cual es relativo. Si se explora en un área más grande se estará siendo más explorativo y si se explora en un área más pequeña, más explotativo. Un algoritmo muy explotativo será aquel que concentre mucho sus esfuerzos en buscar en regiones pequeñas del espacio considerado y por tanto, tiene mayor riesgo de converger a óptimos locales o de hacerlo prematuramente[11] que uno muy explorativo, el cual se concentrará más en amplias regiones y no será capaz de precisar demasiado la solución óptima, pudiendo incluso no converger nunca. No obstante hay quien adopta una interpretación más conceptual (más abstracta) de estos términos y no distingue entre grados de exploración, sino que asocia la exploración a la idea de realizar una búsqueda global (no centrada en ningún punto) y la explotación a la de una búsqueda local por la vecindad de algún punto (centrada en un punto). Pero de nuevo, la diferencia está en el enfoque que quiera darle el usuario y no tanto en el proceso de búsqueda en sí, pues por ejemplo aunque conceptualmente entendamos que estamos explotando porque estamos focalizando la búsqueda en torno a un punto, en función de la vecindad (su tamaño) que contemplemos la capacidad de buscar en diferentes partes del espacio variará y con ello la capacidad de encontrar nuevas soluciones prometedoras. Nosotros optaremos por la primera de las nociones al considerarla menos ambigua y hablaremos de niveles o grados de exploración y explotación. Un balance adecuado entre estas dos características contrapuestas del proceso de búsqueda resulta completamente esencial para cualquier metaheurística, pues permite identificar rápidamente regiones del espacio con soluciones de buena calidad para así centrar la búsqueda en

[9] Hay metaheurísticas que no procesan soluciones desde el principio, sino que las van construyendo progresivamente. Son las denominadas constructivas.

[10] Si viéramos el proceso de búsqueda como un sistema dinámico discreto, podríamos distinguir entre procesos de Markov o no de Markov.

[11] Con convergencia prematura nos referimos a converger a un punto que no es siquiera extremo relativo de la función. Este comportamiento es totalmente indeseado y se da cuando el proceso de convergencia es tan rápido que ni siquiera es posible escapar de él.

ellas y evitar consumir tiempo en regiones no prometedoras, cumpliendo así el objetivo de obtener una solución con un grado alto de optimalidad en el menor tiempo posible. Generalmente, los algoritmos metaheurísticos suelen constar de parámetros que han de ser configurados por el usuario para regular el equilibrio adecuado explotación/exploración para el problema en cuestión que se quiera resolver. También conviene introducir otro término relacionado con la exploración como es la *diversidad*, que se emplea para referirse a lo diferentes que son las soluciones muestreadas durante la búsqueda. Un algoritmo explorativo que concentra sus esfuerzos en regiones amplias del espacio va asociado a una diversidad alta aunque ante un mismo nivel de exploración, pueden existir distintos niveles de diversidad.

2.3. Swarm Intelligence

Dentro de las metaheurísticas bioinspiradas, podemos encontrar un conjunto muy importante de ellas como son las basadas en la Inteligencia de Enjambre o *Swarm Intelligence* en inglés.

La Inteligencia de Enjambre (*Swarm Intelligence*) es definida como "la inteligencia colectiva emergente de un grupo de agentes simples" y supone una rama de la Inteligencia Artificial encargada de estudiar el comportamiento colectivo de sistemas descentralizados y autoorganizados. Estos sistemas inspirados en sistemas biológicos, están típicamente formados por una población de agentes simples que interactúan entre ellos (son sistemas multiagente) y con su medio ambiente. Aunque los agentes siguen reglas simples y no existe una estructura de control centralizado que dictamine su comportamiento, es la interacción entre ellos lo que conduce a la emergencia de un comportamiento global complejo, inteligente. A menudo hablamos de algoritmos de *Swarm Intelligence* para referirnos a los algoritmos o mecanismos distribuidos de resolución de problemas inspirados en el comportamiento colectivo de sociedades animales, generalmente insectos.

Las metaheurísticas basadas en *Swarm Intelligence* lo que hacen es modelar sistemas de *Swarm Intelligence* para llevar a cabo la optimización matemática. Constan de una población de agentes que rastrean el espacio de búsqueda interactuando entre ellos, siendo clave el comportamiento colectivo de enjambre para alcanzar el objetivo de acercarse a la solución óptima. Existen cada vez más metaheurísticas basadas en *Swarm Intelligence*. Entre las más destacadas están la Optimización por Colonia de Hormigas (ACO) [5], inspirada en el comportamiento de las colonias de hormigas a la hora de crear rutas en busca de comida o la Optimización por Enjambre de Partículas (PSO) [6], inspirada en el comportamiento social del vuelo de las bandadas de aves y el movimiento de los bancos de peces.

2.4. Particle Swarm Optimization

El método de Optimización por Enjambre de Partículas o *Particle Swarm Optimization* (PSO) [6] en inglés, es una metaheurística poblacional de *Swarm Intelligence* inspirada en el comportamiento social observado en las bandadas de aves y bancos de peces diseñada para resolver problemas de optimización continua.

PSO trata de simular la manera en la que este tipo de grupos animales logran acometer búsquedas de objetivos en el entorno natural en que viven, que de forma indivi-

dual no les sería posible. El ejemplo más clásico es el de sus búsquedas de alimento. Lo que les permite realizar búsquedas eficientes de fuentes de alimento a dichos animales es la interacción existente entre ellos y de eso se vale el método PSO para llevar a cabo la optimización matemática. PSO, al igual que los sistemas naturales, dispone de un conjunto de individuos o agentes pero que en lugar de buscar en un entorno físico natural lo hacen en el espacio de búsqueda abstracto definido para el problema de optimización en cuestión y que en lugar de buscar comida o algún otro tipo de material físico, lo que buscan es la solución óptima al problema.

En el PSO, a los agentes se les considera partículas, ya que sus características físicas (masa, etc) y biológicas no influyen para nada en la concepción del método.[12] De esta forma, lo único que se tiene en cuenta a la hora de recibir información del ambiente externo al enjambre es la posición de cada partícula. Recordemos que las coordenadas posicionales en este caso representan soluciones candidatas y tienen asociado un valor fitness que servirá de 'feedback' del ambiente a la partícula para que ésta conozca la calidad de su localización y pueda tomar una decisión al respecto. Las decisiones son en términos de movimiento y dependen no solo de la localización de la propia partícula sino de la localización de cada una de las partículas del enjambre. Y aquí es donde radica la interacción provocando que las partículas se muevan hacia lugares cada vez más prometedores y favoreciendo así una búsqueda eficiente. Lógicamente las partículas están comunicadas entre sí y la información detectada del ambiente por cada una de ellas es compartida instantáneamente al resto. Esta información es utilizada por cada partícula para actualizar su posición en el espacio basándose para ello de unas reglas simples de movimiento expresadas mediante fórmulas matemáticas sencillas. Estas reglas de movimiento/actualización consideran al espacio euclídeo, motivo por el cual solo sirven para espacios continuos (optimización continua). El tiempo en cambio sí ha de ser discretizado para la simulación. Cada avance en el tiempo supone una iteración del algoritmo, en la cual se actualizan las posiciones de las partículas y se gestiona la nueva información recibida. El hecho de contemplar varias soluciones (tantas como partículas) en cada iteración es lo que hace que el PSO sea una metaheurística poblacional.

Profundizando en las reglas de actualización de posiciones del método, hay que decir que se dividen en dos partes: regla de actualización de velocidades de las partículas y regla de actualización de posiciones de las partículas. En el PSO se utiliza el término 'velocidad' de una partícula para referirse al vector de desplazamiento de la misma en cada iteración. De este modo, la regla de actualización de posiciones depende exclusivamente de la velocidad y la posición actual de la partícula. La regla de actualización de velocidades es la que rige el movimiento y depende de tres elementos: la velocidad actual de la partícula, la mejor localización encontrada hasta la fecha por la propia partícula y la mejor localización encontrada hasta la fecha por cualquiera de las partículas del enjambre. Es pues a través de la mejor localización global que realmente se produce la interacción entre partículas. Gracias a la regla, las partículas son guiadas hacia las mejores posiciones encontradas tanto individual como colectivamente sin olvidar la dirección del último desplazamiento que sufrieron, respetando así la ley de inercia. Claro que, para ello, las partículas deben estar dotadas de memoria con el fin de almacenar su mejor posición a lo largo de la ejecución.

En resumen, una partícula viene determinada por:

[12] De hecho las partículas en el PSO no mueren nunca. Lo único importante de las mismas es que estén en constante movimiento.

- Su posición actual $X = (x_1, ..., x_j, ..., x_d)$ en el espacio de búsqueda d-dimensional
- Su velocidad actual $V = (v_1, ..., v_j, ..., v_d)$ indicando su último desplazamiento
- Su mejor posición histórica $pBest = (pbest_1, ..., pbest_j, ..., pbest_d)$

En cada iteración, la mejor posición individual se actualiza de acuerdo a la siguiente expresión:

$$pBest = \begin{cases} X, & si \quad f(X) < f(pBest) \\ pBest, & en\ caso\ contrario \end{cases} \tag{2.1}$$

Generalmente el enjambre está compuesto por N partículas ($N > 1$) y cada una de ellas es identificada por un índice i dentro del grupo ($1 \leqslant i \leqslant N$). A N también se le conoce como 'tamaño del enjambre'.

La mejor posición colectiva $gBest = (gbest_1, ..., gbest_j, ..., gbest_d)$ es la mejor de entre las mejores posiciones individuales. Es decir, $gBest = \underset{pBest_i}{\operatorname{argmin}} f(pBest_i)$ o, equivalentemente, $gBest = pBest_g$, con $g = \underset{i}{\operatorname{argmin}} f(pBest_i)$.

Con esta información ya podemos finalmente presentar la fórmula matemática de la regla de actualización de velocidades:

$$v_{i,j} = w * v_{i,j} + c_1 * rand_1 * (pbest_{i,j} - x_{i,j}) + c_2 * rand_2 * (gbest_j - x_{i,j}) \tag{2.2}$$

donde $rand_1$ y $rand_2$ son dos números aleatorios uniformemente comprendidos entre 0 y 1 ($rand_1, rand_2 \sim \mathrm{U}(0,1)$) generados para cada i, j [13] y tanto w como c_1 y c_2 son parámetros propios del método. Al primer sumando de la expresión se le conoce como 'componente inercial' y a w como 'factor de inercia'; al segundo, se le conoce como 'componente cognitivo' y al tercero, como 'componente social'. Así, c_1 y c_2 son los pesos que controlan los componentes cognitivo y social y se suele referir a ellos como factores de aprendizaje o aceleración. El componente cognitivo es el responsable de que parte del movimiento se deba a la propia experiencia de la partícula, mientras que el social lo es de que otra parte se deba a la influencia en los demás miembros del enjambre, tal y como se daría en las bandadas de pájaros por ejemplo.

Por su parte, la regla de actualización de posiciones se formula así:

$$x_{i,j} = x_{i,j} + v_{i,j} \tag{2.3}$$

siendo la regla completa de actualización conformada por la expresión (2.2) seguida por la (2.3). Obsérvese que dichas fórmulas contemplan cada componente dimensional j-ésimo por separado, abordándose así la actualización de manera independiente para cada dimensión. Aunque esta es la versión clásica, las fórmulas también pueden ser presentadas en formato vectorial, el cual quizá favorezca una mejor interpretación y comprensión del método (pues no hemos de olvidar que trabajamos con movimiento y posiciones en el espacio, lo que implica que las dimensiones no pueden ser entendidas de forma aislada).[14]

[13] La notación correcta para $rand_1$ y $rand_2$ sería en realidad $\{rand_1\}_{i,j}{}^{(t)}$ y $\{rand_2\}_{i,j}{}^{(t)}$ respectivamente, siendo (t) el índice de la iteración. Sería la notación adecuada ya que permite distinguir entre las diferentes variables aleatorias independientes entre sí que son (una por cada $(i, j, (t))$). Sin embargo, por motivos de simplicidad se abusa de la notación. Lo mismo sucede con el resto de variables al omitir el índice (t) correspondiente al número de iteración.

[14] A pesar de que este sea un aspecto puramente formal, sí tendrá cierta relevancia en este trabajo ya que en él se estudian diferentes interpretaciones y análisis del método.

El PSO, al igual que cualquier otra metaheurística, es efectivo solamente cuando busca en espacios bien definidos con límites o fronteras. Por eso se emplea para resolver problemas de optimización continua con restricciones de cota. Esto quiere decir que el espacio de búsqueda está acotado. Además la manera en que se impone la cota es definiendo unos límites superior e inferior $[l, u]$ para cada variable del problema (dimensión del espacio de búsqueda). Generalmente los límites se establecen iguales para todas las variables y por tanto el espacio de búsqueda es un hipercubo (si no, en cualquier caso es un hiperrectángulo). Así, el método tiene que ser capaz de gestionar posibles posiciones inválidas (salidas del espacio de búsqueda) ya que las reglas de actualización no pueden asegurar el mantenimiento de las partículas dentro del espacio. Para ello existen diversas estrategias [7, 8]. Entre las más comunes están reintroducir la partícula en alguna posición interior aleatoria, reintroducirla simulando un rebote con la 'pared' por la que salió, atraerla al borde más cercano o simplemente ignorarla y esperar a la siguiente iteración para volver a actualizar su posición.

La única parte que queda por mencionar del método es la de la inicialización de las velocidades y posiciones de las partículas. De nuevo aquí también existen múltiples propuestas [9, 10, 11, 12], si bien lo usual es hacerlo aleatoriamente empleando una distribución de probabilidad uniforme para cada variable en su correspondiente rango: $x_{i,j} \sim \mathrm{U}(l_j, u_j)$, $v_{i,j} \sim \mathrm{U}(-(u_j - l_j), u_j - l_j)$. Las velocidades también suelen inicializarse a 0.

Reordenando todos los pasos mencionados, tenemos finalmente el algoritmo PSO básico o estándar. Su descripción en pseudocódigo se muestra en Algoritmo 1. En él se deja a elección del usuario el criterio de terminación del bucle principal. Normalmente se querrá detener la ejecución cuando se haya encontrado una solución satisfactoria con un $f(gBest)$ lo suficientemente bajo. El problema es que en la mayoría de situaciones desconocemos cuál es la calidad de la solución óptima ($f(X^*)$), por lo que carecemos de referencia alguna para poder comparar la bondad del $gBest$. En estos casos solemos optar por detener el algoritmo tras un número prefijado de iteraciones o cuando veamos que se ha convergido lo suficiente a alguna solución comparando resultados de iteraciones anteriores [13]. También una combinación de ambas se utiliza. De hecho casi siempre se establece un número máximo de iteraciones por si los demás criterios no se llegan a cumplir en un tiempo razonable.

Como vemos, el método tiene varios parámetros que han de ser previamente configurados por el usuario. Cada problema requerirá de un ajuste diferente de estos parámetros para obtener el máximo rendimiento. Son lo que se conoce como *parámetros de control* o *hiperparámetros* del método porque sirven para proporcionar un control al usuario sobre el proceso de optimización que lleva a cabo el PSO. Dentro de los hiperparámetros podemos hacer una distinción entre los que son más propios o específicos del método y los demás, que son más generales. En los hiperparámetros de propósito general incluiríamos al número máximo de iteraciones $maxIter$ y demás parámetros que controlen el bucle principal del algoritmo, así como al tamaño de la población N principalmente. En los hiperparámetros característicos incluiríamos a los que nos permiten describir el funcionamiento de las partículas, véase w, c_1 y c_2. Son estos últimos los que determinan el comportamiento del método, decidiendo la atención que se le presta a la exploración y la explotación y por tanto, los que realmente influyen en su rendimiento. Aparte de los hiperparámetros, están los parámetros dependientes del problema de optimización como la función objetivo f, las restricciones de cota $[l_j, u_j]\ \forall j = 1, ..., d$ o el número de dimensiones d del espacio de

Algoritmo 1: Algoritmo PSO estándar

para *cada partícula* $i = 1, ..., N$ **hacer**
  Inicializar la posición X_i de la partícula dentro de los límites: $x_{i,j} \in [l_j, u_j]$
  Inicializar la mejor posición individual $pBest_i$ a la posición inicial: $pBest_i = X_i$
  si $f(pBest_i) < f(gBest)$ **entonces**
    Actualizar la mejor posición colectiva: $gBest = pBest_i$
  fin
  Inicializar la velocidad V_i de la partícula
fin
mientras *No se cumpla el criterio de terminación* **hacer**
  para *cada partícula* $i = 1, ..., N$ **hacer**
    Actualizar la velocidad V_i de la partícula según la ecuación (2.2) $\forall\, j = 1, ..., d$
    Actualizar la posición X_i de la partícula según la ecuación (2.3) $\forall\, j = 1, ..., d$
    Comprobar y gestionar si es necesario la factibilidad de las posiciones X_i
  fin
  para *cada partícula* $i = 1, ..., N$ **hacer**
    si $f(X_i) < f(pBest_i)$ **entonces**
      Actualizar la mejor posición individual: $pBest_i = X_i$
      si $f(pBest_i) < f(gBest)$ **entonces**
        Actualizar la mejor posición colectiva: $gBest = pBest_i$
      fin
    fin
  fin
fin

soluciones.

Desde su introducción han surgido múltiples variantes del PSO básico. Dichas variantes agregan distintas funcionalidades al método y/o modifican otras. Por ejemplo, algunas se centran en modificar la topología del enjambre[15] [14], otras la sincronía de las actualizaciones [15], otras las propias reglas de actualización [16], etc. Por supuesto también existen combinaciones del PSO con otros métodos de optimización, generalmente hibridaciones del PSO con otros metaheurísticos como los algoritmos genéticos [17] o el algoritmo de recocido simulado [18]. Todas estas variantes tienen por objetivo mejorar el rendimiento del PSO estándar y compiten entre sí para lograr el mejor rendimiento posible.

Por otra parte, también han surgido adaptaciones del método para poder abordar problemas de optimización discreta (ya sea optimización binaria [19], entera [20] o combinatoria [21]), problemas de optimización multiobjetivo [22] o de optimización con restricciones no solo de cota [23], incluso diversas versiones de PSOs paralelizables [24] centradas en disminuir el tiempo de ejecución del algoritmo.

[15]La topología del enjambre define el subconjunto de partículas con las cuales cada una puede compartir información. En el PSO estándar la topología es global ya que todas las partículas están comunicadas entre sí, el enjambre tiene estructura única.

Capítulo 3

Análisis del PSO en la literatura

Al tratarse el PSO de una simulación estocástica, no es nada fácil entender y comprender bien cómo se desarrolla este proceso para acabar encontrando la solución óptima. Sin embargo, afortunadamente existen bastantes trabajos de investigación en este sentido, que tratan de abordar el problema mencionado. En este capítulo haremos un repaso a la literatura para estudiar aquellos trabajos que de una manera u otra realizan un análisis del PSO estándar. Eso sí, solo consideraremos aquellos que basan su análisis en observaciones teóricas y no empíricas o experimentales. En base a la revisión de la literatura, hemos decidido clasificar los trabajos en tres grupos diferentes: los que realizan un análisis de convergencia del método, los que analizan los parámetros directamente (el rol teórico de cada uno) y los que analizan el método desde una perspectiva más general estudiando las reglas de actualización de posiciones en su conjunto. Todos ellos buscan el mismo objetivo pero lo hacen desde aproximaciones distintas. Muchos de ellos como resultado de sus análisis presentan además nuevas variantes que también mencionaremos.[1] Ahora bien, por motivos obvios se hace imposible presentar y explicar todos y cada uno de los trabajos existentes, por lo que hemos seleccionado los que, a nuestro juicio, resultan ser los más relevantes y/o adecuados al presente proyecto. El mayor énfasis lo hemos puesto en la tercera categoría de análisis puesto que es la más completa y de alguna manera abarca las dos anteriores.

Pero antes de comenzar con la revisión, puede ser de gran utilidad repasar brevemente la historia del método, su origen y evolución hasta la confección del PSO estándar. Esto sin duda nos ayudará a comprender el por qué de muchos de los componentes del método y forma parte de su análisis, además de ser clave para lo subsiguiente. Así pues, el capítulo está organizado en cuatro secciones: primero se comenzará hablando de la historia del método, después se presentará la review sobre los análisis de convergencia, posteriormente sobre los análisis paramétricos y por último, sobre los generales de método.

[1] Únicamente serán estudiadas aquellas variantes del PSO que tengan una justificación basada en un análisis teórico del PSO estándar, es decir, que surjan fruto de un análisis previamente compartido. Solo nos interesan aquellas que procuran mejorar o corregir alguna funcionalidad existente en el método original y no las que proponen nuevas funcionalidades

3.1. Historia del PSO

La historia del PSO se remonta al año 1995, fecha en la que fue introducido por Kennedy y Eberhart [6]. Tal y como cuentan ellos, el método fue descubierto a través de la simulación de un modelo social simplificado. Los autores andaban interesados en simular gráficamente la elegante y a la vez impredecible coreografía de una bandada de aves. Este trabajo se inspiraba en otros que también habían tratado de simular bandadas de aves basándose en la manipulación de las distancias entre individuos [34, 35] y tenía la aspiración de extraer conocimientos para aplicar a una eventual simulación del comportamiento social humano. Probaron con distintas propuestas de simulación hasta dar con la definitiva, la cual requería de un proceso de optimización matemática. Los autores consideraron que el movimiento de las 'aves' dependiera de una posición fija e independiente a todas ellas que actuara atrayéndolas, introduciendo así una fuerza artificial que resultara en una simulación realista. Pero decidieron que dicha posición fija no podía ser independiente de los agentes ya que en la vida real han de descubrirla, poniendo como ejemplo sus búsquedas de alimento. Así, propusieron una función matemática que permitiese a cada agente evaluar su posición y moverse en torno a las mejores posiciones encontradas a lo largo de la simulación. Este comportamiento social en el que cada miembro se beneficia de la experiencia previa y descubrimientos de todo el resto ya había sido teorizado para el caso de los bancos de peces. Sin darse apenas cuenta, lo que tenían entre manos era un potente optimizador.

La primera versión del método ajustaba las velocidades de las partículas teniendo solamente en cuenta el signo de la diferencia de posición por dimensión entre la partícula y su objetivo. Es decir, la velocidad disminuía si el objetivo estaba a la izquierda y aumentaba si estaba a la derecha:

$$v_{i,j} = \begin{cases} v_{i,j} - rand_1 * c_1, & si \quad x_{i,j} > pbest_{i,j} \\ v_{i,j} + rand_1 * c_1, & si \quad x_{i,j} < pbest_{i,j} \\ v_{i,j}, & si \quad x_{i,j} = pbest_{i,j} \end{cases} \qquad v_{i,j} = \begin{cases} v_{i,j} - rand_2 * c_2, & si \quad x_{i,j} > gbest_j \\ v_{i,j} + rand_2 * c_2, & si \quad x_{i,j} < gbest_j \\ v_{i,j}, & si \quad x_{i,j} = gbest_j \end{cases} \tag{3.1}$$

Las dos fórmulas de la ecuación (3.1) se aplicaban secuencialmente para cada dimensión $j = 1, ..., d$ y cada partícula $i = 1, ..., N$. Desde este momento, el papel de c_1 y c_2 ya comenzó a ser discutido por los autores. c_1 y c_2 regulaban el incremento de la velocidad cuando la partícula se veía afectada por el $pBest$ y el $gBest$ respectivamente. Según los autores, cuando éstos tomaban valores relativamente altos, las partículas parecían ser fuerte y rápidamente atraídas al objetivo, mientras que cuando tomaban valores bajos, lo hacían de manera más lenta y armoniosa. También observaron que un valor de c_1 alto en relación a c_2, resultaba en un excesivo aislamiento e independencia entre las partículas y lo contrario, resultaba en una convergencia prematura a un mínimo local.

Con el fin de hacer más fácil de entender el algoritmo y mejorar su rendimiento, los autores presentaron una segunda versión que no estuviera basada en el testeo de inecuaciones. Esta versión en lugar de tener en cuenta el signo de la diferencia de posición, tiene directamente en cuenta la diferencia de posición por dimensión entre las partículas y las mejores localizaciones:

$$v_{i,j} = v_{i,j} + rand_1 * c_1 * (pbest_{i,j} - x_{i,j}) + rand_2 * c_2 * (gbest_j - x_{i,j}) \tag{3.2}$$

Sin embargo, los autores seguían sin saber muy bien cómo establecer los valores de c_1 y c_2. Según ellos no había una buena manera de adivinar cuál de ellos debía ser mayor y por eso decidieron establecer ambos parámetros al valor constante de 2, pues así, al multiplicarse por el factor estocástico, la media sería de 1 ($\mathbb{E}(\mathrm{U}(0,1)*2) = 1$) y los agentes sobrevolarían el objetivo aproximadamente la mitad del tiempo. Esta versión fue su definitiva porque vieron que su rendimiento era superior a las anteriores, aunque dejaron pendiente de investigación cómo debería ser el proceso de selección del valor de estos hiperparámetros. También probaron más variaciones del método sin éxito como considerar solo el punto medio entre el $gBest$ y $pBest$ de cada partícula, al cual debieran sentirse atraídas, considerar dos tipos de agentes: exploradores y explotadores que usaran la primera y segunda versión propuesta respectivamente o eliminar el término inercial de la fórmula de actualización de velocidades.

Poco más tarde de esta primera publicación, los autores sacaron otra más centrada en la implementación y rendimiento del método para la resolución de problemas de optimización matemática [36]. En este artículo trataron por primera vez el tema del rango de las variables de optimización y la frontera del espacio de búsqueda. También como consecuencia de esto surgió el problema de las partículas volando fuera del espacio permitido, hecho que solventaron imponiéndole un valor absoluto máximo a las velocidades ($v_{i,j} = sgn(v_{i,j}) * min(|v_{i,j}|, v_{max})$) para prevenir que éstas fueran demasiado altas como para que las partículas se salieran de los límites.[2]

Tres años más tarde Shi y Eberhart expusieron en [37] que sin las partes social y cognitiva (es decir, quedándose solo con la inercial) las partículas seguirían moviéndose con la misma velocidad todo el rato hasta alcanzar la frontera y así el PSO no encontraría una solución aceptable a menos que se la cruzase por el camino. Por otro lado, sin la componente inercial las partículas irían convergiendo y el espacio de búsqueda se iría encogiendo poco a poco con el paso de las iteraciones, motivo por el cual serían incapaces de expandirse por el espacio como sí lo harían con inercia y el PSO solo encontraría la solución óptima en el caso de que ésta estuviera dentro del rango inicial del espacio ocupado por las partículas, lo cual dependería enormemente de la inicialización de las mismas. En otras palabras, la componente inercial aportaría exploración mientras que las otras dos, explotación. Como para cada problema a resolver se requiere un balance adecuado entre estas dos propiedades de búsqueda, los autores propusieron añadir un factor w que controlase la componente inercial así como c_1 y c_2 controlaban las componentes social y cognitiva. El resultado de esta modificación nos lleva al considerado PSO básico o estándar cuya regla de actualización de velocidades viene dada por la ecuación (2.2). Y es el estándar entre otras cosas porque es la generalización del propuesto en la publicación original, el cual utilizaba la expresión (3.2), que es igual a la expresión (2.2) cuando $w = 1$. En [37] también analizaron el uso de un peso inercial w dinámico que cambiara durante la ejecución y probaron uno linealmente decreciente con el número de iteraciones con el fin de explorar más al principio y explotar más al final. Hay que decir que todo ello, desde el análisis inicial hasta las pruebas finales, fue asumiendo un $c_1 = c_2 = 2$ y un límite para la velocidad $v_{max} = 2$. Sus conclusiones fueron que lo que mejor rendimiento daba era un $w \in [0.9, 1.2]$ o uno decreciente con el tiempo. El estudio del peso inercial linealmente decreciente se prolongó en [38], donde lo investigaron más extensivamente y llegaron a la conclusión de que podía faltar capacidad explorativa

[2] A este estado en el que las partículas se salen descontroladamente de los límites impuestos del espacio de búsqueda, se le conoce como *explosión*.

al final de la ejecución. En el estudio también observaron que el PSO no era sensible al tamaño de la población N y que escalaba bien al número de dimensiones d.

Desde la introducción del factor inercial w al método, éste se convirtió en el principal hiperparámetro a controlar, pues era el que tenía el rol más claro. Un w alto favorecía la exploración y uno bajo, la explotación. Sin embargo, aún se usaba conjuntamente con un límite de velocidad v_{max} cuya función ya no estaba bien justificada.[3] Fue en [39] donde por primera vez analizaron el impacto conjunto de estos dos elementos. Comenzaron razonando que la máxima velocidad permitida servía como restricción para controlar la máxima capacidad de exploración que podía tener el PSO. Si ésta era baja sí o sí el PSO sería más explotativo, mientras que si era alta, tendría más capacidad de control sobre la exploración que recaería en w. Dicho de otro de modo, la máxima velocidad permitida v_{max} afectaba de forma indirecta a la exploración y el factor inercial w lo hacía de forma directa, por lo que sería mejor poder controlar la exploración directamente solo a través de w. Es decir, proponen acabar con el límite de velocidad. Tras varios experimentos, concluyen que se puede obtener un rendimiento muy similar bajando w al subir v_{max} y que cuando v_{max} es pequeño (≤ 2) w debe estar cercano a 1 y cuando es grande (≥ 3), $w = 0.8$ es una buena decisión. Sobre todo concluyen que la velocidad límite es prescindible. A raíz de estos trabajos, la gran mayoría de futuras publicaciones sobre el PSO no contemplan límite de velocidad, considerando así a w como el hiperparámetro principal para regular la capacidad explorativa del método. Como es natural hay excepciones como [40], que trata de regularla mediante un límite de velocidad v_{max} dinámico que decrece con el paso de las iteraciones de forma similar a como se hacía con w en sustitución de este último, aunque con resultados no demasiado favorables.

3.2. Análisis de convergencia

A raíz de la incertidumbre mostrada en los primeros estudios acerca del proceso de convergencia y su control del PSO, comenzaron a aparecer análisis matemáticos del método que trataban de arrojar luz sobre su comportamiento convergente y que fundamentalmente buscaban proporcionar algún tipo de guía al usuario sobre qué valores de los hiperparámetros permitían un rendimiento deseado. Este tipo de análisis no está tan centrado en la funcionalidad que pueda tener cada componente sino en el resultado al que se llega.

3.2.1. Estabilidad determinista y/o de orden-1

El primer análisis en este sentido fue el de Ozcan y Mohan [41], que pretendía obtener formalmente las trayectorias de las partículas del PSO. Sin embargo, para hacerlo consideraron una versión bastante más simple a la original con $w = 1$, determinista (sin los componentes de aleatoriedad) y donde $gBest = pBest_i = p$, con p constante (tanto $gBest$ como $pBest_i$ eran estáticos). Además solo contemplaron una simple partícula ($N = 1$) en un sistema unidimensional ($d = 1$). Con todas estas simplificaciones los autores obtuvieron la ecuación recursiva de las posiciones de la partícula durante la ejecución y a partir de ésta, la forma cerrada de la recursión, es decir, la ecuación directa de la posición en función de la iteración que ahorra el cálculo de sus posicio-

[3] Recordemos que v_{max} se introdujo para paliar la sobreexploración, hecho con el que puede ahora lidiar w.

nes anteriores. Gracias a esta forma cerrada, pudieron analizar la trayectoria de la partícula en función del valor de c_1+c_2 y su velocidad inicial $V^{(0)}$. El análisis consistió en dividir el comportamiento de la partícula en 4 casos principales dependientes de $c = c_1 + c_2$. Únicamente cuando $0 < c < 4$ no se divergía, pues en el resto de casos, la partícula seguía una trayectoria divergente cuya posición cada vez se alejaba más del $gBest$. Y para el caso no divergente $0 < c < 4$, la ecuación de trayectoria era una onda sinusoidal cuya amplitud y frecuencia venía determinada por la elección de los parámetros. Quizá esa fuera la mayor conclusión del trabajo, que las partículas en lugar de 'volar' lo que hacían era 'surfear' el espacio de búsqueda 'saltando' así de unas olas a otras con el fin de moverse hacia la localización óptima. El mismo análisis con conclusiones análogas pero generalizado a sistemas multidimensionales ($d > 1$), con múltiples partículas ($N > 1$) y en los cuales no necesariamente $gBest = pBest_i$ lo realizaron en [42] los mismos autores.

Tras estos primeros trabajos, el primer gran análisis sobre la convergencia y estabilidad del PSO fue llevado a cabo por Clerc y Kennedy [43], donde además introdujeron una de las modificaciones más usadas del PSO hasta hoy. Partieron de la misma simplificación que [41] para primero hallar la matriz del sistema dinámico y sus valores propios e_1 y e_2, los cuales analizaron para averiguar la existencia de ciclos y la prevención de la explosión del sistema. Sobre todo andaban preocupados por esto último. Es por eso que tras definir el criterio de convergencia del sistema ($max(|e_1|, |e_2|) < 1$), lo modificaron añadiendo coeficientes a las expresiones del movimiento denominados 'factores de constricción' para así poder asegurar la convergencia del sistema de forma directa mediante estos coeficientes que fuerzan al sistema a comportarse de manera deseada. Centraron su estudio en la clase de casos más simple: la que solo trabajaba con un factor de constricción χ. Analizando cómo debía ser χ para que se cumpliera el criterio de convergencia para algunos de estos casos, se proporcionaron varios métodos de constricción. Finalmente todos estos resultados se generalizaron al método original, aunque eso sí con $w = 1$ y en ausencia de aleatoriedad. En este sistema el punto de convergencia se descubre es $P = \frac{c_1 * pBest + c_2 * gBest}{c_1 + c_2}$ y su ecuación de velocidad es equivalente a:

$$v_{i,j} = v_{i,j} + c * (p_{i,j} - x_{i,j}) \tag{3.3}$$

En resumen, aunque la nueva versión propuesta denominada PSO constrictivo admite varias expresiones de actualización diferentes, la más común tras reintroducir la aleatoriedad es:

$$\begin{aligned} v_{i,j} &= \chi(v_{i,j} + c_1 * rand_1 * (pbest_{i,j} - x_{i,j}) + c_2 * rand_2 * (gbest_j - x_{i,j})) \\ x_{i,j} &= x_{i,j} + v_{i,j} \end{aligned} \tag{3.4}$$

donde $\chi = \frac{2\kappa}{|2-c-\sqrt{c^2-4c}|}$ y $c = c_1 + c_2, \quad c > 4, \quad \kappa \in [0, 1]$. Típicamente κ se establece a 1 y $c_1 = c_2 = 2.05$, por lo que $\chi = 0.7298$. Más tarde en el tiempo se descubrió que esta versión era equivalente a la versión estándar con $w = \chi = 0.73$ y $c_1 = c_2 = 1.49$ aproximadamente [44].

En [31] fue donde se analizó por primera vez la convergencia del PSO con factor de inercia $w \neq 1$. En este artículo Trelea considera también un PSO simplificado sin componentes estocásticos y al igual que [43], se basa en la teoría de los sistemas dinámicos discretos para estudiar el comportamiento de una partícula. Así, descubre la región del espacio paramétrico (del plano (w, c)) que permite asegurar la convergencia

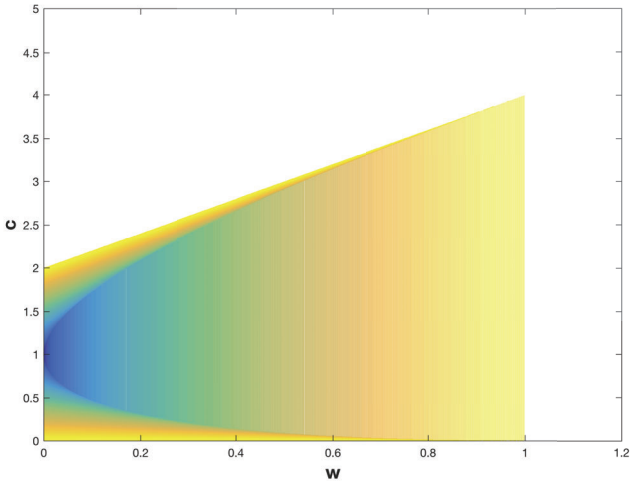


Figura 3.1: Región paramétrica de convergencia del PSO determinista y su tasa de convergencia para las diferentes configuraciones de la región. Para cada combinación de valores de los parámetros (c, w), se muestra el nivel de exploración o rapidez con la que converge el sistema asociado mediante un código de colores. Cuanto más claro es el color de la gráfica, mayor es la exploración (convergencia más lenta) y viceversa. Las regiones en blanco indican divergencia.

de la partícula al punto P. Dicha región se muestra en la figura 3.1. También descubre las regiones donde la partícula exhibe oscilaciones armónicas o comportamientos zigzagueantes alrededor de P antes de converger, cumplimentando esto con ejemplos para diferentes configuraciones. En general, se apunta que las configuraciones que caen cerca del centro de la región de convergencia inducen una convergencia rápida, mientras que las que caen en los bordes, requieren de más iteraciones para converger. Paralelamente en [45] se hizo el mismo tipo de análisis en el que se especificaban los valores de w, c_1 y c_2 que permitían alcanzar la estabilidad del sistema. También en [47] Zheng et al. analizaron la región de convergencia y fueron un paso más allá afirmando que, al contrario de lo expuesto en [37], un w bajo favorecería la exploración y uno alto, la explotación. Se basaban en que al reintroducir la aleatoriedad en el algoritmo, $\phi = c_1 * rand_1 + c_2 * rand_2$ que haría las veces de c en el determinista, sería un valor aleatorio uniformemente comprendido entre 0 y 4 y bajo esta nueva situación, examinando la región de convergencia determinista, un w alto otorgaría al sistema más posibilidades de converger y estabilizarse que uno bajo (esto se puede apreciar observando la figura 3.1). Su análisis derivó en la propuesta de un factor inercial w linealmente creciente [48].

Poco más tarde en [46], Van den Bergh y Engelbrecht presentaron la primera prueba formal de que P era el punto de atracción medio sobre el que se mueve la partícula en el PSO estándar. Primero, como se había hecho en [31], demostraron que la secuencia de posiciones $\{X^{(t)}\}_{t=0}^{+\infty}$ de una partícula en el PSO determinista de converger lo hacía a P:

$$\lim_{t \to +\infty} X^{(t)} = \frac{c_1 * pBest + c_2 * gBest}{c_1 + c_2}$$

Con la introducción de la aleatoriedad, el punto de atracción pasa a ser dinámico y dependiente de los componentes aleatorios. Ahora ya no es P sino P' con $p'_j =$

$\frac{c_1*rand_1*pbest_j+c_2*rand_2*gbest_j}{c_1*rand_1+c_2*rand_2}$.[4] De esta forma, la secuencia $\{X^{(t)}\}_{t=0}^{+\infty}$ se vuelve aleatoria y el límite de la misma no existe pero sí se puede predecir el comportamiento medio o estimado del sistema. Esto último es lo que derivan los autores:

$$\lim_{t\to+\infty} \mathbb{E}(X^{(t)}) = \mathbb{E}(P') = \frac{c_1 * \frac{1}{2} * pBest + c_2 * \frac{1}{2} * gBest}{c_1 * \frac{1}{2} + c_2 * \frac{1}{2}} = P \tag{3.5}$$

Aparte de esto, en [46] también estudian la región de convergencia determinista y tratan de extrapolar los resultados al PSO estándar con aleatoriedad. Lo hacen desde un enfoque muy similar al de [47], considerando al PSO estocástico un sistema que alterna aleatoriamente entre PSOs deterministas. Es decir, en cada iteración del método tratan a $c_1 * rand_1$ y $c_2 * rand_2$ como constantes que fueran a mantenerse para el resto de la ejecución para así poder valerse del análisis de convergencia determinista. De este modo tratan de analizar la convergencia del método estándar observando la probabilidad de que dados w, c_1 y c_2, la configuración de valores (w, ϕ) caiga dentro de la región de convergencia determinista. Así una iteración en la que dicha configuración esté en la región convergente se ve como un paso convergente del algoritmo, mientras que si cae fuera de ella, como uno divergente. Según dicen, cuanto mayor es la probabilidad de caer en la región convergente, mayor es la convergencia alcanzada por el algoritmo. Los autores tratan de contrastar estas ideas teóricas con ejemplos concretos y pruebas experimentales.

Como consecuencia del hallazgo de P como punto de convergencia del sistema, en [49] se propone una nueva regla de actualización de posiciones para el PSO que directamente consista en el muestreo de las coordenadas posicionales de una distribución de probabilidad gaussiana centrada en P, ahorrándose así toda la parte de velocidad de las partículas. A esta variante se le conoce como *Bare Bones PSO* y originalmente surgió a raíz de la observación empírica de que las posiciones de las partículas alrededor de sus $pBest$ y $gBest$ (asumidos constantes, es decir, sistema en estado de estancamiento) tenían una distribución con forma de campana de gauss centrada en la media aritmética entre $pBest$ y $gBest$ bajo la condición $c_1 = c_2$. Resumidamente su propuesta fue hacer que

$$x_{i,j} = \mathcal{N}(\frac{gbest_j + pbest_{i,j}}{2}, |gbest_j - pbest_{i,j}|) \tag{3.6}$$

si bien dicha expresión puede ser generalizada para cualquier configuración (c_1, c_2) simplemente haciendo que la media de la distribución normal sea P.

3.2.2. Estabilidad orden-2

Hasta ese momento todos los análisis de convergencia que se habían realizado y hemos revisado aquí o bien se olvidaban por completo del carácter estocástico del método o bien lo estudiaban desde un enfoque promedio, esto es, analizando el comportamiento esperado del sistema a través del estudio de convergencia de la secuencia $\{\mathbb{E}\,(X^{(t)})\}_{t=0}^{+\infty}$ en lugar de $\{X^{(t)}\}_{t=0}^{+\infty}$. Así, se concluía que si $\exists \lim_{t\to+\infty} \mathbb{E}(X^{(t)})$ entonces éste era igual a P según la expresión (3.5). A este tipo de análisis se le denomina análisis de estabilidad de orden-1. Sin embargo, este tipo de análisis no revela gran información acerca de la convergencia real de la secuencia original $\{X^{(t)}\}_{t=0}^{+\infty}$, ya que

[4] En realidad la expresión correcta para p'_j sería $p'_j = \frac{c_1*\{rand_1\}_j{}^{(t)}*pbest_j+c_2*\{rand_2\}_j{}^{(t)}*gbest_j}{c_1*\{rand_1\}_j{}^{(t)}+c_2*\{rand_2\}_j{}^{(t)}}$ aunque se obvia por simplicidad.

solo tiene en cuenta el valor esperado de los elementos de la secuencia, olvidándose de su varianza, la cual indica la dispersión que hay alrededor del teórico valor medio considerado. Lógicamente si dicha varianza es grande, la partícula del sistema no puede converger por mucho que sí lo haya hecho su posición esperada e incluso puede divergir si así lo hace su varianza. Por ello, es importante estudiar también la convergencia de la varianza del sistema, es decir, estudiar el $\lim_{t\to+\infty}\sigma^2(X^{(t)})$. Una secuencia para la cual existe este límite se dice que es estable de orden-2. Tan importante es la estabilidad orden-1 como la de orden-2, pues solo cuando se dan ambas es cuando se puede conseguir la convergencia. En concreto,

$$\lim_{t\to+\infty} X^{(t)} = P \iff \lim_{t\to+\infty} \mathbb{E}(X^{(t)}) = P \ \wedge \ \lim_{t\to+\infty} \sigma^2(X^{(t)}) = 0 \tag{3.7}$$

Uno de los primeros trabajos en realizar este análisis de estabilidad orden-2 fue el de Poli [50]. En este artículo se detallan las ecuaciones recursivas necesarias para describir la varianza $\sigma^2 = \mathbb{E}\left((X^{(t)})^2\right) - \left(\mathbb{E}\left(X^{(t)}\right)\right)^2$ y con ellas se define un sistema dinámico descrito en notación matricial al igual que hemos visto en la mayoría de trabajos, con la salvedad de que en esta ocasión el estado del sistema ya no solo depende de $\mathbb{E}\left(X^{(t)}\right)$ y sus variables auxiliares (las que permiten que se pueda calcular el estado actual de $\mathbb{E}\left(X^{(t)}\right)$), sino que depende también de $\mathbb{E}\left((X^{(t)})^2\right)$. De esta forma el sistema guarda implícitamente la información sobre σ^2 y el punto de equilibrio o de estabilidad del sistema (también denominado punto fijo), que es el estado al que converge, dependerá de la convergencia tanto de $\mathbb{E}\left(X^{(t)}\right)$ como de $\sigma^2(X^{(t)})$. El autor procede con el análisis del nuevo sistema, primero tratando de hallar la región paramétrica de convergencia mediante el estudio de los valores propios de su matriz y después, hallando los puntos fijos para cada variable que compone el estado del sistema.[5] Llega a la conclusión de que para alcanzar una convergencia real como la expuesta en (3.7), se requiere que $pBest = gBest$, hecho que solamente cumple una partícula dentro del enjambre. Sin embargo, el resto de partículas pueden alcanzar una estabilidad de órdenes- 1 y 2, donde el punto de convergencia de la varianza coincide en esencia con el observado en [49], pues es proporcional a $|gBest - pBest|$. Todo el análisis supone condiciones de estancamiento (no hay mejora y por tanto $pBest$ y $gBest$ estáticos) y $c_1 = c_2$. Bajo estas condiciones finalmente se proporciona el criterio de convergencia del modelo analizado, es decir, la región paramétrica que garantiza la estabilidad de órdenes- 1 y 2 del método:

$$c_1 = c_2 < \frac{12 * (w^2 - 1)}{5w - 7} \tag{3.8}$$

El trabajo de Poli [50] es uno de los más reconocidos pero, a decir verdad, no es ni el primero ni el único que trata de estudiar la estabilidad del PSO estándar teniendo en cuenta su carácter estocástico y tratando las posiciones de las partículas $X^{(t)}$ como variables aleatorias que son. Es más, un análisis muy similar ya había sido presentado por Jiang et al. [51] en el cual también se estudia la convergencia de las secuencias $\{\mathbb{E}\left(X^{(t)}\right)\}_{t=0}^{+\infty}$ y $\{\sigma(X^{(t)})\}_{t=0}^{+\infty}$ arrojando resultados equivalentes y además presentando una condición suficiente de convergencia de $pBest$ a $gBest$ y con ello, por tanto, de convergencia absoluta en el sentido de la expresión (3.7). Hay que decir

[5] Nótese que aunque el autor habla en todo momento de análisis de estabilidad de orden-2, lo que está haciendo en realidad es un análisis de estabilidad de órdenes- 1 y 2, ya que como hemos dicho, el equilibrio de este sistema se alcanza solo cuando $\exists \lim_{t\to+\infty} \mathbb{E}(X^{(t)})$ y $\exists \lim_{t\to+\infty} \sigma^2(X^{(t)})$.

que de nuevo se supone estancamiento de $gBest$.[6] El trabajo de Jiang [51] muestra como en el PSO $\exists \lim_{t\to+\infty} \sigma^2(X^{(t)}) \implies \exists \lim_{t\to+\infty} \mathbb{E}(X^{(t)})$, es decir, la estabilidad de orden-2 implica estabilidad de orden-1.

Otros trabajos, en lugar de analizar los sistemas dinámicos lineales deterministas correspondientes a $\{\mathbb{E}(X^{(t)})\}_{t=0}^{+\infty}$ y $\{\sigma^2(X^{(t)})\}_{t=0}^{+\infty}$, lo que hacen es estudiar el método desde la perspectiva de un sistema con realimentación no lineal que permite representar la influencia de los componentes estocásticos. El análisis de estabilidad absoluta se logra a través del método directo de Lyapunov[7] aplicado a este tipo de sistemas [53]. Algunos ejemplos de estos trabajos son [54] o [55]. El problema que tienen es que el uso de este tipo de análisis proporciona condiciones de estabilidad suficientes pero no necesarias, por lo que la región paramétrica de convergencia que deducen estos trabajos es limitada y conservadora. Además, su análisis solo puede centrarse en la partícula con la mejor posición histórica (la que cumple $pBest = gBest$), porque como ya hemos dicho, es la única que puede converger de forma absoluta satisfaciendo (3.7).

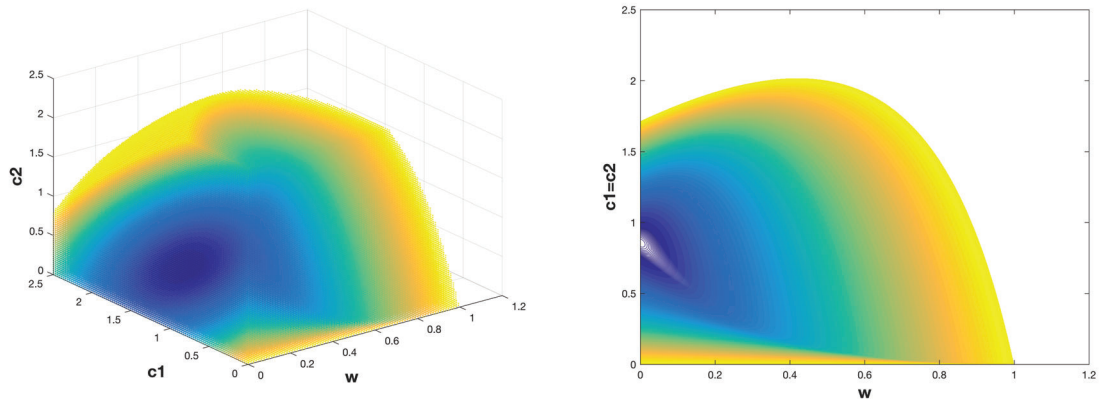
Más recientemente se han publicado trabajos en la línea de [50] y [51] pero que tratan de relajar las condiciones bajo las que se realiza el análisis. Por ejemplo en [56] se expone que la asunción de estancamiento total no es necesaria, sino que bajo la condición de estancamiento débil se pueden deducir los mismos resultados de estabilidad y lo hace demostrando que es suficiente con estudiar la convergencia de la varianza de la partícula dominante (la cual cumple $pBest = gBest$) para derivar la región de estabilidad de órdenes- 1 y 2. Así, $\lim_{t\to+\infty} \sigma^2({X_\tau}^{(t)}) = 0 \iff \exists \lim_{t\to+\infty} \sigma^2({X_j}^{(t)})\ \forall j = 1, ..., d$ con τ el índice de la partícula dominante, lo cual permite relajar las restricciones de [50, 51]. En [57] van un paso más allá considerando a $pBest$ y $gBest$ variables aleatorias también, eso sí, que siguen cada una una distribución de probabilidad fija e invariante con unas varianzas y valores esperados bien definidos.[8] En estas circunstancias el análisis de estabilidad muestra que la región de convergencia de órdenes-1 y 2 es independiente de dichas variables, por lo que sus resultados son los mismos que asumiendo estancamiento. Finalmente, uno de los últimos trabajos sobre convergencia del PSO como es [58], relaja aún más las restricciones de estancamiento al tratar a $pBest$ y $gBest$ como variables aleatorias que siguen distribuciones de probabilidad cambiantes con el tiempo con varianzas y valores esperados bien definidos que terminan convergiendo. Es decir, incorporan al análisis las secuencias convergentes $\{\mathbb{E}(pBest^{(t)})\}_{t=0}^{+\infty}$, $\{\sigma^2(pBest^{(t)})\}_{t=0}^{+\infty}$, $\{\mathbb{E}(gBest^{(t)})\}_{t=0}^{+\infty}$ y $\{\sigma^2(gBest^{(t)})\}_{t=0}^{+\infty}$, llegando de hecho a la conclusión de que la convergencia de las mismas es una condición necesaria para la estabilidad de órdenes- 1 y 2. Los resultados del análisis coinciden nuevamente con los de los demás trabajos, recogidos bajo la expresión (3.8) que indica el criterio de convergencia del PSO estándar cuando $c_1 = c_2$.

Todos los resultados presentados se resumen en la figura 3.2, la cual muestra el criterio de estabilidad orden-2 generalizado para los casos en que $c_1 \neq c_2$, además de la tasa de consecución de dicha estabilidad para cada configuración paramétrica

[6] Cuando solamente se asume el estancamiento de $gBest$, en la literatura se habla de condición débil de estancamiento.

[7] Este método se basa en la noción de estabilidad energética de un sistema. Un sistema es estable si su energía decrece con el tiempo hasta alcanzar un estado de equilibrio en el cual no resta energía. Lyapunov generalizó este concepto para la determinación de la estabilidad de un sistema cualquiera mediante el uso de funciones que simularan el rol de la función energía. Más información en [52].

[8] Aquí no se asume estancamiento en el sentido de los otros trabajos, pero sí se asume un estancamiento de distribución.



(a) (b)

Figura 3.2: Región paramétrica de convergencia del PSO estándar (estabilidad orden-2). Se muestra para cada configuración paramétrica la tasa de convergencia de $\{\sigma^2(X^{(t)})\}_{t=0}^{+\infty}$ mediante un mapa de colores que asocia linealmente la calidez del color con las tasas. A color más cálido, menor rapidez de convergencia. El color blanco indica divergencia. (a) representa la gráfica 4-dimensional de la función que hace corresponder a cada 3-tupla de valores paramétricos (w, c_1, c_2), la tasa de convergencia de $\{\sigma^2(X^{(t)})\}_{t=0}^{+\infty}$. (b) es una representación de (a) cuando $c_1 = c_2$.

estable. Ya que estabilidad orden-2 implica estabilidad orden-1 y un grado más alto de estabilidad (la estabilidad absoluta de (3.7)) ya no depende solo de la configuración paramétrica sino de la distribución de $pBest$ (o sea de la función objetivo, etc), podríamos decir que la figura 3.2 es quizá la mayor descripción de la convergencia alcanzada según la configuración en el PSO estándar que se pueda realizar desde un punto de vista teórico. De todos modos, hay que tener en cuenta que tampoco es una descripción del todo exacta de la estabilidad en la práctica, ya que proviene de análisis teóricos que requieren de simplificaciones del método. Concretamente, son análisis que realizan asunciones sobre $pBest$ y $gBest$, ya que lógicamente no es posible modelar por completo estos elementos al depender enteramente de una función objetivo y de los resultados que se vayan dando durante la ejecución (es donde reside el poder de la heurística). Por ejemplo, todos ellos asumen independencia entre $pBest$, $gBest$ y X, cuando realmente sabemos que esto no es así (ver ecuación (2.1)) además de alguna forma de estancamiento de $pBest$ y $gBest$, siendo el más débil el de convergencia en distribución de [58]. El hecho de tener que obviar de alguna manera las distribuciones de $pBest$ y $gBest$ es lo que hace que los resultados no sean del todo precisos, pues los parámetros también influyen en éstas. Por poner un caso, cuando $c_2 > c_1$ intuitivamente la partícula se ve más atraída por $gBest$ que por $pBest$, lo que brinda más opciones de que el $pBest$ resida más cerca del $gBest$ y también de que se convierta en nuevo $gBest$. Estas situaciones que pueden afectar la convergencia no se contemplan en estos análisis. Aún así, en [59] dan validez empírica a la descripción de convergencia de 3.2, pues deducen que dista muy poco de la realidad, aunque también señalan que fuera de dicha región también se alcanza la convergencia para según qué funciones y según la cercanía a la frontera de convergencia.

3.2.3. Convergencia local

Hasta ahora hemos visto varios estudios sobre la estabilidad del PSO estándar y hay otros tantos que hacen lo propio con otras variantes o generalizaciones del método [62, 63, 64], pero todos ellos analizan la convergencia a un punto P cualquiera sin importar las características de este punto. Es decir, se centran en ver cuándo $\exists \lim_{t\to+\infty} X^{(t)}$ y no si dicho límite es aceptable o deseable. Por supuesto lo ideal sería que $\lim_{t\to+\infty} X^{(t)} = X^*$, con X^* el óptimo global de la función objetivo, aunque eso no se pueda garantizar. Sin embargo, sí existen algunos trabajos en este sentido cuyo objetivo es analizar las garantías que ofrece el método de converger en óptimos aunque sea locales si no globales. Fue en [60] donde por primera vez se puso de manifiesto la falta de garantía del PSO para converger si quiera a un óptimo local. Sus autores observaron que las partículas podían dejar de moverse y por tanto parar la búsqueda cuando sus $pBest$ alcanzaban el $gBest$. Esto es lo que nosotros hemos llamado convergencia absoluta (3.7). Pero esta propiedad es indeseable ya que significa que las partículas convergen a la mejor posición descubierta por el enjambre hasta el momento ($gBest$), la cual no tiene por qué ser siquiera óptimo local pues no hay nada que garantice la optimalidad del $gBest$. Se puede producir pues una convergencia antes de tiempo (convergencia prematura) porque se para la búsqueda cuando todavía había margen de mejora para $gBest$ incluso en su vecindad. En [61] los mismos autores proporcionan una prueba formal de que el PSO estándar no tiene convergencia garantizada a un óptimo local. La solución que proponen a este problema es mutar la posición de la mejor partícula del enjambre (la partícula cuyo $pBest = gBest$) para que así se mantenga siempre en movimiento y otorgue una probabilidad no nula de mejorar el $gBest$, evitando con ello la convergencia prematura. Lo hacen sugiriendo una modificación de la regla de actualización de posición de la mejor partícula τ exclusivamente (el resto sigue con la regla original), que es la siguiente:

$$x_{\tau,j} = gbest_j + w * v_{\tau,j} + \rho * (1 - 2 * rand) \tag{3.9}$$

donde como siempre V_τ hace referencia al último desplazamiento sufrido por τ y como novedad el último término de la expresión, $\rho*(1-2*rand)$, es el encargado de generar una muestra aleatoria entorno al $gBest$, definida por el parámetro ρ que establece la amplitud del espacio de muestreo centrado en $gBest$ aunque luego desplazado por $w*V_\tau$. El valor de ρ es adaptado en cada iteración de acuerdo a la siguiente expresión:

$$\rho = \begin{cases} 2\rho, & si \quad \#successes > s_c \\ 0.5\rho, & si \quad \#failures > f_c \\ \rho, & en\ caso\ contrario \end{cases} \tag{3.10}$$

donde $\#successes$ y $\#failures$ denotan el número de aciertos y fallos consecutivos respectivamente, definiendo fallo como $f(gBest^{(t+1)}) = f(gBest^{(t)})$ y acierto lo opuesto. f_c y s_c son umbrales definidos por el usuario. La prueba de que este nuevo método bautizado como *Guaranteed Convergence PSO (GCPSO)* es localmente convergente también se encuentra en [61].

Como conclusión a este repaso a los análisis de convergencia, cabe decir que son muy útiles a la hora de describir las dinámicas de las partículas y determinar las configuraciones paramétricas que favorecen el buen rendimiento del método. Sin embargo, como ya hemos mencionado, se basan en modelos teóricos simplificados del PSO para hacer posible dichos análisis. Aún así logran resultados muy representativos de

los que se dan en la práctica con las condiciones reales. Con todo, ayudan pues a comprender mejor el comportamiento del método, lo que se traduce en una mayor facilidad para el usuario a la hora de realizar la selección de hiperparámetros, etc. No obstante, este tipo de análisis no contribuye demasiado a una mayor comprensión del funcionamiento en sí del método, pues aunque sí describe su comportamiento, no llega a cuestionar el por qué de los resultados ni qué es lo que hace que se produzcan éstos y no otros, etc. Por esto mismo, tampoco permiten detectar las claves de que el método funcione bien ni aquellos elementos indiferentes o que entorpecen el buen funcionamiento. El rol que juegan los parámetros y demás componentes del PSO no se desvela con estos análisis. Es por eso que no sirven de mucho para crear nuevas variantes o adaptaciones del método, es más, cada variante nueva requiere de un nuevo análisis de estabilidad realizado desde cero. Para extraer un conocimiento superior se requiere de un nivel mayor de abstracción, que trataremos de alcanzar en las siguientes secciones.

3.3. Análisis paramétrico

Este apartado está en realidad bastante relacionado con el anterior, ya que tiene por objetivo analizar el impacto que tienen los hiperparámetros del PSO en su comportamiento. Sin embargo, el enfoque es diferente. Mientras en la anterior sección estudiábamos los parámetros según su rendimiento y el resultado que conseguían obtener, en esta, nos centraremos más en su utilidad y función dentro del método. Para ello, visitaremos algunos trabajos que dan su interpretación sobre ello, generalmente derivando en propuestas de PSO adaptativos. Estas propuestas resultan interesantes ya que luego nos pueden permitir comparar rendimiento de los métodos y así vislumbrar qué idea era la más acertada acerca del rol de cada parámetro.

3.3.1. Factor de inercia

Comenzando por el factor de inercia w, sabemos que fue introducido en [36] para regular el balance exploración/explotación. Los autores veían que los componentes social y cognitivo servían para ayudar a la partícula a visitar regiones prometedoras del espacio de búsqueda y hacerlo de manera inteligente, pero no facilitaban el descubrimiento de nuevas regiones que pudieran ser prometedoras. Para esta parte explorativa se encontraba la componente inercial, pero ésta no era regulada por ningún parámetro como sí lo eran las otras dos. En definitiva, generalmente se cree que un w alto favorece la búsqueda global mientras que uno bajo, la local [39]. Aunque como vimos en [47] hay trabajos que opinan lo contrario. Sea como fuere, muchos estudios consideran a este parámetro como el más influyente en el rendimiento del algoritmo y por consiguiente, acapara gran atención investigativa.

Basados en este concepto de w hay varias estrategias que tratan de aprovechar este conocimiento para optimizar su rendimiento. Ya vimos como en [37] sugerían un factor inercial dinámico que decreciese linealmente con el paso de las iteraciones dentro de un rango dado para así disfrutar de una mayor exploración al principio de la ejecución y terminar con una mayor explotación al final:

$$w^{(t)} = w_{max} - (w_{max} - w_{min}) * \frac{t}{maxIter}$$

Otros trabajos proponían un decrecimiento no lineal que permitiera una asignación

temporal desigual para las diferentes tasas de exploración asociadas a w y por tanto un mayor dominio de la exploración o la explotación. Por ejemplo en [65] se propuso que

$$w^{(t)} = w_{min} + (w_{max} - w_{min}) * \left(\frac{maxIter - t}{maxIter}\right)^n$$

siendo n el elemento no lineal a escoger por el usuario, de manera que con $n > 1$ la tasa de decrecimiento es mayor al principio (más explotación) y con $n < 1$, es mayor al final (más exploración). En [68] pasa parecido con k como parámetro regulador usando en este caso la función tangente:

$$w^{(t)} = w_{min} + (w_{max} - w_{min}) * tan\left(\frac{7}{8} * \left(1 - \left(\frac{t}{maxIter}\right)^k\right)\right)$$

y en [69] con los parámetros a y b de la regla de actualización

$$w^{(t)} = w_{max} * e^{-a\left(\frac{t}{maxIter}\right)^b}$$

En [66] se propusieron dos estrategias de variación exponencial de w:

$$w^{(t)} = w_{min} + (w_{max} - w_{min}) * e^{-\left(\frac{10t}{maxIter}\right)}$$
$$w^{(t)} = w_{min} + (w_{max} - w_{min}) * e^{-\left(\frac{4t}{maxIter}\right)^2}$$

las dos siendo considerablemente más explotativas que la lineal. Algo parecido intentan en [67]. En [70, 79, 90] también diseñan una curva decreciente con forma exponencial para w y en [71] optan por emplear una función logarítmica decreciente, que aunque sigue siendo más explotativa que la lineal, lo es algo menos que las exponenciales. Más ejemplos de pesos inerciales no linealmente decrecientes son el uso de una función de Sugeno [77] o una función sigmoide [78].

Otros en cambio consideran un w aleatorio basándose en la incertidumbre de la necesidad de exploración o explotación en cada momento de la ejecución. Ejemplos de esto son [72] que emplea

$$w = 0.5 + \frac{rand}{2}$$

o [73] que simplemente considera w como un valor aleatorio uniformemente comprendido entre 0 y 1. En [74] para cubrir la incertidumbre, en lugar de utilizar aleatoriedad directamente, introducen el caos en la estrategia de actualización de w por su apariencia aleatoria aunque determinista que permite disfrutar de ambas ventajas. Lo hacen imponiendo

$$w = 4 * w * (1 - w)$$

siempre con $w \in (0, 1)$. También hacen lo propio en [75] combinando las dos estrategias anteriores:

$$w = 0.5 * rand + 0.5 * z; \quad z = 4 * z * (1 - z)$$

Además en ese mismo trabajo proponen otra estrategia que combina el caos con el decrecimiento de w:

$$w^{(t)} = (w_{max} - w_{min}) * \frac{maxIter - t}{maxIter} + w_{min} * z; \quad z = 4 * z * (1 - z)$$

Una alternativa al control aleatorio o caótico del peso inercial que asimismo lograra alternar entre exploración y explotación durante la ejecución, fue presentada por

[76] y constaba de un w oscilatorio que eso sí, reservara las iteraciones finales a un w constante para el refinamiento de la búsqueda en las regiones ya descubiertas (explotación al final):

$$w^{(t)} = \begin{cases} \frac{w_{min}+w_{max}}{2} + \frac{w_{max}-w_{min}}{2} \cos\left(\frac{2\pi t(4k+6)}{3*maxIter}\right) & si \quad t < \frac{3*maxIter}{4} \\ w_{min} & en\ caso\ contrario \end{cases}$$

También experimentaron con variantes de la propuesta como el reemplazo de w_{max} por $\left(1 - \frac{t}{maxIter}\right) w_{max}$ en la expresión anterior para que así la onda sinusoidal tuviera una amplitud linealmente decreciente y por tanto hubiera a su vez una transición paulatina de exploración a explotación. O incluso directamente probaron con un modelo que discretizase estas dos tendencias según el signo de la componente coseno de la ecuación.

Pero sin duda las estrategias que más centran la atención de los investigadores son las que adaptan el valor del factor de inercia en función no solo de la iteración sino sobre todo de la información que va recaudando el algoritmo durante la búsqueda (estrategias adaptativas) ya que son las que aportan el mayor control sobre lo que se precisa en cada momento teniendo en cuenta el impacto del parámetro. Una de las más conocidas y antiguas es la basada en sistemas difusos. La idea es que estos sistemas reciban como entrada variables que midan el rendimiento del PSO y devuelvan un nuevo valor para w, generalmente en términos de cambio con respecto al valor actual. Esta estrategia fue por primera vez planteada en [80], donde diseñaron un sistema que recibía el fitness normalizado de la actual mejor solución y el factor inercial actual como variables de entrada y devolvía el cambio del factor inercial. Las tres variables difusas están definidas para tener tres conjuntos difusos con funciones de pertenencia triangulares diferentes y el sistema difuso está compuesto por nueve reglas. Y a partir de este trabajo surgieron nuevas propuestas como la de [81] que aplicaba este sistema de manera individualizada para cada partícula según su estado, es decir, en lugar de tener en cuenta el valor fitness del $gBest$, se tenía en cuenta el del $pBest$ y así cada partícula gozaba de un balance diferente de exploración/explotación. Y otras muchas que directamente optaban por probar con otras variables indicadoras del estado del algoritmo que no fuesen solamente los mejores valores fitness sino, por ejemplo, la desviación de los fitness de la actual iteración, cambios en las mejores posiciones, velocidad relativa media, número de iteración, etc [82, 83].

Más allá de los sistemas difusos existen multitud de técnicas adaptativas para el control de w que utilizan información sobre el estado de la simulación equivalente a la expuesta en el párrafo anterior. Es el caso de [84] que utiliza la bondad relativa de la mejor solución encontrada por la partícula para asignarle a ésta un peso inercial individualizado, asignando a mayor bondad mayor explotación y viceversa:

$$w_i = \left(1.1 - \frac{f(gBest)}{f(pBest_i)}\right)$$

En [85] sin embargo proponen una adaptación general (no individualizada para cada partícula) de w en función de la distancia media normalizada entre la mejor partícula del enjambre τ y el resto, siendo ésta grande en fases exploratives y pequeña cuando se está convergiendo. El factor inercial a su juicio debe apoyar la fase en la que se encuentre el algoritmo, por lo que si está en fase de exploración se ha de apoyar la

exploración y al revés, quedando w proporcional a esta distancia de la cual depende:

$$w = \frac{1}{1 + 1.5e^{-2.6(s_\tau)_n}}; \quad (s_\tau)_n = \frac{s_\tau - \text{mín}(s_i)}{\text{máx}(s_i) - \text{mín}(s_i)} \quad s_i = \frac{1}{N-1} \sum_{k=1,k\neq i}^{N} \sqrt{\sum_{j=1}^{d} (x_{i,j} - x_{k,j})^2}$$

Otro trabajo que se aprovecha de información relativa a la distancia entre posiciones del espacio de búsqueda es [92]. En este caso tienen en cuenta la distancia entre el $gBest$ y el $pBest$ de cada partícula para diseñar una estrategia individualizada ya que este dato es tremendamente importante en el movimiento de la partícula. También sugieren que cuando esta distancia es grande se explore y cuando es pequeña se explote. Aportan dos estrategias en este sentido:

$$w_i = w_i - \left((w_i - 0.4) * e^{-\left(\frac{t}{maxIter}\sqrt{\sum_{j=1}^{d}(gbest_j - pbest_{i,j})^2}\right)} \right)$$

$$w_i = e^{-e^{-\frac{maxIter-t}{maxIter}\sqrt{\sum_{j=1}^{d}(gbest_j - pbest_{i,j})^2}}}$$

En [86] usan el porcentaje de éxito como 'feedback' para la adaptación, definido éxito como una mejora del $pBest$ de cada partícula. Según sus autores, un porcentaje alto indica que las partículas han convergido a un punto lejano al óptimo y se mueven muy lentamente hacia el óptimo mientras que uno bajo indica que las partículas oscilan alrededor del óptimo pero sin mucha mejora. Así, entienden que w ha de ser directamente proporcional al porcentaje de éxito P_s y en concreto deciden usar una función lineal:

$$w = (w_{max} - w_{min})P_s + w_{min}$$

Basándose en ese mismo concepto de porcentaje de éxito P_s, en [89] introducen dos estrategias adaptativas que también apuestan por una relación de proporción directa entre P_s y w. La primera es una modificación de la técnica dinámica de decrecimiento lineal de [37] y la segunda, una versión adaptativa de la técnica aleatoria de [72]:

$$w^{(t)} = (w_{max} - w_{min}) \left(\frac{maxIter - t}{maxIter} \right) + w_{min} * P_s$$

$$w = 0.5 * rand + 0.5 * P_s$$

Y lo propio hacen en [90] fusionando las estrategias caóticas de [75] con el elemento adaptativo P_s, exactamente por el mismo motivo, que es el de valerse de información relativa al estado de la optimización para guiar mejor el proceso manteniendo la esencia de las estrategias originales:

$$w^{(t)} = ((w_{max} - w_{min})(\frac{maxIter - t}{maxIter}) + w_{min}) * z; \quad z = 4P_s(1 - P_s)$$

$$w = (0.5P_s + 0.5) * z; \quad z = 4P_s(1 - P_s)$$

Como indicador de progreso en [87] emplean el valor máximo de la desviación estándar por dimensiones del conjunto de los $pBest$ y además tienen en cuenta no solo el de la iteración actual sino los de las últimas K mediante la declaración del array $s : s(k) = \text{máx}_{1\leq j\leq d}(\sigma((pbest_{i,j})_{1\leq i\leq N}))$, $k = (t \ mod \ K)$. Su idea es que cuando las mejores posiciones sean cada vez más parecidas, se incremente w para revertir

la tendencia y seguir explorando aunque siempre favoreciendo la vuelta a una fase explotativa. Para ello, siguen la siguiente regla:

$$w = 0.9 - 0.4\frac{s(k)}{\text{máx}_{1\leq k\leq K}\, s(k)}$$

En [88] apuestan por guiarse del error cuadrático medio de los valores fitness de la iteración en curso como medida de diversidad de la población para diseñar una estrategia que aumente la exploración (y por tanto w) cuando la diversidad es pequeña y la disminuya cuando sea grande. A su vez añaden una componente extra que actúe restando importancia a esta parte adaptativa a medida que se avanza en la ejecución:

$$w^{(t)} = w_{min} + (w_{max} - w_{min}) * \left(1 - \frac{2}{\pi}\arctan\left(\frac{\sum_{i=1}^{N}\left(f(X_i) - \frac{\sum_{i=1}^{N} f(X_i)}{N}\right)^2}{N}\right)\right) * e^{\left(\frac{-t^2}{2\left(\frac{maxIter}{3}\right)^2}\right)}$$

Un enfoque diferente se da en [91], donde son conocedores de que una velocidad alta supone exploración y una baja, explotación y por ello tratan de adaptar w buscando que la velocidad de las partículas se ajuste a una velocidad ideal predefinida. Dicha velocidad ideal acuerdan que sea decreciente y no lineal con el objetivo de disfrutar de más tiempo de exploración en las primeras fases y más de explotación en las finales comparado con una lineal, siendo la transición entre ambas relativamente rápida. Con lo cual, la adaptación se lleva a cabo comparando la velocidad media de las partículas con la velocidad ideal y en base a ello aumentando o disminuyendo w una cierta cantidad prefijada Δw de manera que:

$$w = \begin{cases} \text{máx}(w - \Delta w, w_{min}) & si \quad \frac{1}{N*d}\sum_{i=1}^{N}\sum_{j=1}^{d}|v_{i,j}| \geq \frac{u-l}{2} * \frac{1+\cos\left(\frac{\pi(t+1)}{0.95*maxIter}\right)}{2} \\ \text{mín}(w + \Delta w, w_{max}) & en\ caso\ contrario \end{cases}$$

Por otra parte en [93] se calcula el peso inercial para cada partícula según la posición que ocupa en el ranking por fitness de todas las partículas del enjambre. El puesto que ocupa cada partícula en ese ranking (R_i) determina lo bien o no que está situada, siendo el orden de mejor a peor. Así, la asociación entre w_i y R_i será lineal y dada por:

$$w_i = w_{min} + (w_{max} - w_{min})\frac{R_i}{N}$$

Podríamos sin duda mencionar muchos más trabajos sobre el ajuste dinámico del factor inercial y casi todos parten de la concepción de w como elemento meramente regulador de la exploración, que sirve para favorecer o no la importancia de V en el movimiento de la partícula, siendo V visto exclusivamente como un elemento disuasorio para la atracción directa hacia las mejores posiciones. Por ello, los trabajos asocian directamente exploración con w (proporcionalidad directa) y se centran en construir técnicas bajo esta asunción. No obstante, algún trabajo hay que considere un rol que va algo más allá del comentado. Es el caso de [94], que juega con la idea de que la información relativa a la dirección de V en un momento dado no es desdeñable. Ellos la utilizan para establecer el peso inercial de la mejor partícula del enjambre τ aduciendo que ésta ha de confiar en que su dirección de movimiento es la correcta (por eso es la mejor) y ha de acelerar en esa misma dirección que tenía para seguir buscando allí el óptimo global. Es por eso que su w ha de incrementarse para

darle más peso a su inercia. El resto de partículas siguen el procedimiento estándar de decrecimiento del factor de inercia. La estrategia queda definida por:

$$w_i = \begin{cases} w_i + \eta\frac{w_{max}-w_{min}}{maxIter} & si \quad i = \tau \\ w_i - \frac{w_{max}-w_{min}}{maxIter} & en\ caso\ contrario \end{cases}$$

con η una constante definida por el usuario que controla el ritmo de la aceleración.

En la literatura se pueden encontrar varias publicaciones que realizan comparativas de técnicas de asignación dinámica del peso inercial, entre ellas muchas de las aquí presentadas. Haciendo un repaso a algunas de estas publicaciones puede uno llegar fácilmente a la conclusión de que al final este tipo de técnicas no ofrecen unos resultados tan buenos como los que cabría esperar por su nivel de refinamiento y sofisticación. Por ejemplo en [95] destacan que a pesar del éxito reportado en las distintas publicaciones de las técnicas analizadas, los resultados empíricos muestran que la única estrategia de control inercial con un rendimiento semejante a la asignación constante de w es la del peso aleatorio. Algo similar deducen en [96], donde tras comparar treinta estrategias diferentes encuentran que la mejor en términos de precisión es la caótica-aleatoria, aunque en términos de eficiencia la asignación constante está por delante de todas. Estos resultados están en consonancia con los de [97], que determina también que los mejores métodos son los aleatorios y/o caóticos. En resumen, las técnicas más básicas y teóricamente con menor capacidad para guiar el proceso de búsqueda son las que mejor funcionan, lo cual quiere decir que por lo general la lógica que hay detrás de las estrategias de adaptación no es del todo acertada.

3.3.2. Factores social y cognitivo

Los factores social y cognitivo son los que controlan la lógica del método puesto que regulan la atracción de las partículas hacia las regiones prometedoras y por tanto en ellos reside la heurística en sí, lo que distingue al PSO de una simple búsqueda aleatoria. El factor social c_2 controla la atracción a la mejor posición encontrada por el colectivo ($gBest$) y el cognitivo c_1, la atracción a la mejor posición encontrada por la propia partícula ($pBest$). Ya vimos como inicialmente en [6] ambos valores se establecieron iguales a 2 aunque sin una justificación teórica contundente. Sin embargo, se puso encima de la mesa que tanto las magnitudes como la relación de proporcionalidad entre c_1 y c_2 influenciaban el comportamiento de las partículas en el método. Por un lado, si los valores de estas constantes eran altos, las partículas se veían rápida y fuertemente atraídas al objetivo mientras que si eran bajos, lo hacían lentamente. Por otro lado, si c_1 era incrementado, la atracción de la partícula hacia el $pBest$ aumentaba y decrecía la de la partícula hacia el $gBest$, lo que resultaba en un excesivo aislamiento e independencia entre las partículas mientras que lo contrario suponía una mayor atracción conjunta de todas las partículas hacia un mismo punto: el $gbest$. Este conocimiento sobre la utilidad y el impacto de los factores social y cognitivo en el funcionamiento del PSO ayudan, al igual que en el caso del factor inercial, a la construcción de técnicas inteligentes de ajuste dinámico de dichos parámetros que permitan obtener un mejor rendimiento del algoritmo.

El primer grupo de técnicas son las dinámicas no adaptativas, las que solamente dependen del tiempo. Dentro de este grupo se encuentra la técnica de decrecimiento lineal de los factores de aprendizaje, análoga a la del factor inercial de [37]. Primero

fue [98] quien testeó el decrecimiento lineal de ambos coeficientes con el paso de las iteraciones seguramente con la idea de disminuir la velocidad conforme avanzaba la ejecución y con ello la exploración:

$$c_1^{(t)} = \{c_1\}_{max} - (\{c_1\}_{max} - \{c_1\}_{min})\frac{t}{maxIter}$$

$$c_2^{(t)} = \{c_2\}_{max} - (\{c_2\}_{max} - \{c_2\}_{min})\frac{t}{maxIter}$$

En cambio el autor se dio cuenta de que fijar los valores de c_1 y c_2 daba mejores resultados, aunque la asignación óptima dependiera del problema. Más tarde en [99], con el mismo objetivo de explorar más al principio y explotar más al final, consideran el incremento lineal del componente social a la par del decrecimiento lineal del cognitivo, precisamente atendiendo al impacto causado por la relación de proporcionalidad entre c_1 y c_2 que explicamos antes:

$$c_1^{(t)} = \{c_1\}_{max} - (\{c_1\}_{max} - \{c_1\}_{min})\frac{t}{maxIter}$$

$$c_2^{(t)} = \{c_2\}_{min} + (\{c_2\}_{max} - \{c_2\}_{min})\frac{t}{maxIter}$$

De forma similar a como pasaba con w, también aparecen técnicas de decrecimiento/crecimiento no lineal de los coeficientes cognitivo y social respectivamente. Como muestra de ello tenemos que [100] propone tres estrategias distintas, aunque una de ellas es de prueba para ver cómo se comporta el método ante un ajuste incremental del factor cognitivo. Efectivamente comprueban que no funciona bien. Las otras dos juegan con distintos ritmos de transición entre exploración y explotación pero mantienen la esencia de [99]:

$$c_1^{(t)} = \{c_1\}_{max} + (\{c_1\}_{max} - \{c_1\}_{min})\left(\left(\frac{t}{maxIter}\right)^2 - \frac{2t}{maxIter}\right); \quad c_2^{(t)} = 3 - c_1^{(t)}$$

$$c_1^{(t)} = \{c_1\}_{min}\left(\frac{\{c_1\}_{max}}{\{c_1\}_{min}}\right)^{\frac{1}{1+10\frac{t}{maxIter}}}; \quad c_2^{(t)} = 3 - c_1^{(t)}$$

En [101] usan funciones trigonométricas para describir las curvas de evolución de los coeficientes de aceleración acotadas por unos valores máximo y mínimo iguales para los dos (c_{max} y c_{min}):

$$c_1^{(t)} = (c_{max} - c_{min}) * \sin\left(\frac{\pi}{2}\left(1 - \frac{t}{maxIter}\right)\right) + c_{min}$$

$$c_2^{(t)} = (c_{max} - c_{min}) * \cos\left(\frac{\pi}{2}\left(1 - \frac{t}{maxIter}\right)\right) + c_{min}$$

En [102] utilizan en cambio funciones exponenciales:

$$c_1^{(t)} = c_{min} + (c_{max} - c_{min}) * e^{-\left(\frac{4t}{maxIter}\right)^2}$$

$$c_2^{(t)} = c_{max} - (c_{max} - c_{min}) * e^{-\left(\frac{4t}{maxIter}\right)^2}$$

Y hay otros muchos trabajos parecidos que sugieren distintas funciones pero cuyo cometido es prácticamente el mismo, véase [103, 104, 105, 106].

Por otro lado están las técnicas adaptativas, que sí emplean información relativa al estado de la optimización para ajustar los factores de aprendizaje. Nuevamente aquí la metodología y la información empleada es la que se usaba para la adaptación de la inercia, con la importante salvedad de que ahora en lugar de tener que ajustar un único parámetro directamente relacionado con la exploración, tenemos que ajustar dos y la relación no es tan directa, pues depende también del valor relativo entre ellos. Por ejemplo, en [107] se basan en el fitness relativo de las posiciones más recientes de las partículas para diseñar una estrategia individualizada de ajuste dinámico que permita a las partículas mejor situadas explotar más y a las peor situadas, explorar más. Este control de la exploración lo llevan a cabo igual que los métodos anteriores, asociando un incremento del factor social con un decremento del cognitivo y una mayor exploración con un mayor valor relativo del cognitivo con respecto al social:

$$\{c_1\}_i = 1.2 - \frac{\text{mín}_{1\leq k\leq N}\, f(X_k)}{f(X_i)}$$

$$\{c_2\}_i = 0.5 + \frac{\text{mín}_{1\leq k\leq N}\, f(X_k)}{f(X_i)}$$

En [108] deciden ajustar únicamente el factor cognitivo para regular el ratio entre la influencia cognitiva y la social y con ello la exploración. Lo hacen considerando la distancia relativa referente al fitness (la llaman ξ) entre la partícula y el $gBest$, de modo que cuando la partícula esté cercana al $gBest$, su c_2 disminuya para atraerla más hacia el $pBest$ y aumentar exploración, mientras que si está lejana, su c_2 aumente para explotar más:

$$\{c_2\}_i = (c_{max} - c_{min}) * (1 - \cos(\frac{\pi}{2}\xi_i)) + c_{min}; \quad \xi_i = \begin{cases} \frac{f(X_i)-f(gBest)}{f(X_i)} & si \quad f(X_i) \neq 0 \\ 0 & en\ caso\ contrario \end{cases}$$

En [85] y [109] confeccionan una serie de reglas en base al estado en que se encuentra el enjambre que dictan cómo se debe actuar en cada situación en términos de incremento/decremento de los factores de aprendizaje. Concretamente, en [85] contemplan cuatro posibles estados determinados por la distancia media normalizada entre la mejor partícula del enjambre y el resto. Los estados con sus correspondientes acciones asociadas son los siguientes:

(1) Estado de exploración$\longrightarrow$ subir c_1 y bajar c_2. Se quiere explorar más y esta acción ayuda a incrementar la influencia del $pBest$ en el movimiento de la partícula.

(2) Estado de explotación$\longrightarrow$ subir ligeramente c_1 y bajar ligeramente c_2. Se quiere enfatizar la búsqueda alrededor del $pBest$. Este estado sería de transición entre el de exploración y el de convergencia, por lo que sus modificaciones son ligeras para que la transición sea gradual.

(3) Estado de convergencia$\longrightarrow$ subir ligeramente c_1 y subir ligeramente c_2. En este estado se quiere atraer a todas las partículas hacia el $gBest$ y alcanzar la convergencia a ese punto. Sin embargo, tampoco se quiere una convergencia prematura fruto de una atracción demasiado fuerte, hecho que según sus autores se daría si en lugar de subir ligeramente ambos coeficientes, se subiera c_2 y bajara c_1.

(4) Estado de salto$\longrightarrow$ bajar c_1 y subir c_2. Este estado representa la situación en la que se descubre una nueva mejor región lejos de la actual (se produce un salto hacia un mejor óptimo) y ahora sí, las partículas han de seguirla y llegar a ella lo antes posible.

Por otra parte en [109] también cuentan con cuatro reglas pero esta vez determinadas por un proceso de clustering de las partículas. Lo que hacen es dividir la población en tres clusters aplicando el algoritmo K-means y posteriormente analizar en qué clusters se encuentran la mejor y peor partícula. Nombrando al tamaño del cluster de la mejor y peor partícula como G_B y G_W respectivamente y al tamaño del mayor y menor cluster como G_{max} y G_{min}, las cuatro reglas son:

(1) $G_B = G_{min}, G_W = G_{max} \longrightarrow$ subir c_1 y bajar c_2. En este punto se quiere explorar lo máximo posible y se busca que las partículas se muevan hacia sus $pBest$.

(2) $G_B = G_W = G_{min} \longrightarrow$ bajar c_1 y bajar c_2. Aquí la información aún es escasa sobre la forma que va a tomar el enjambre, por lo que deciden que las partículas han de moverse libremente sin verse demasiado atraídas por $pBest$ y $gBest$.

(3) $G_B = G_W = G_{max} \longrightarrow$ subir c_1 y subir c_2. En este estado el efecto del $gBest$ ha de elevarse para guiar la convergencia del enjambre, pero al mismo tiempo la probabilidad de convergencia prematura ha de evitarse y por eso c_1 también se incrementa.

(4) $G_B = G_{max}, G_W = G_{min} \longrightarrow$ bajar c_1 y subir c_2. Este estado es el final, en el cual se quiere definitivamente explotar la región del $gBest$ para lo cual se quiere un c_2 relativamente alto y un c_1 relativamente bajo.

Como vemos, estos dos últimos trabajos ya no solo tienen en cuenta los valores relativos de los factores de aprendizaje sino también sus magnitudes. Asocian una mayor magnitud a una mayor atracción y viceversa. No son los únicos trabajos que se basan en este concepto, hay técnicas adaptativas que utilizan esta idea para ajustar los parámetros. Es el caso de [110], que realiza la adaptación de acuerdo a la distancia de las partículas a sus $pBest$ y $gBest$. Si la partícula está lejos de estos puntos de atracción, se emplea un coeficiente de aceleración grande para así atraerla rápidamente hacia ellos y al revés. Además trata por separado ambos componentes (social y cognitivo) analizando la magnitud de los dos factores y con ello la atracción a cada punto de manera independiente. Utiliza para dicho fin la función sigmoide acompañada de una serie de parámetros p, q, r, s a definir por el usuario, de tal modo que

$$\{c_1\}_i = \frac{q}{1 + e^{-p((pBest_i - X_i) - r)}} + s$$

$$\{c_2\}_i = \frac{q}{1 + e^{-p((gBest - X_i) - r)}} + s$$

La filosofía es la misma en [111], donde en lugar de la distancia de las partículas a sus atractores, utilizan la bondad de los mismos con relación a las posiciones actuales de las partículas:

$$\{c_1\}_i = \frac{f(pBest_i) - \text{máx}_{1 \leq k \leq N} f(X_k)}{\text{mín}_{1 \leq k \leq N} f(X_k) - \text{máx}_{1 \leq k \leq N} f(X_k)}$$

$$c_2 = \frac{f(gBest) - \text{máx}_{1 \leq k \leq N} f(X_k)}{\text{mín}_{1 \leq k \leq N} f(X_k) - \text{máx}_{1 \leq k \leq N} f(X_k)}$$

En [112] también se centran exclusivamente en las magnitudes de los coeficientes, sin embargo, la interpretación que hacen es diferente. Mientras que en los anteriores trabajos se asociaba una mayor magnitud a una mayor y más rápida y fuerte atracción (más explotación), en este se argumenta que una mayor magnitud expande el área de búsqueda y una menor, lo contrae. [112] actualiza los factores de aceleración multiplicándolos o dividiéndolos por 2 en función de la bondad relativa de las

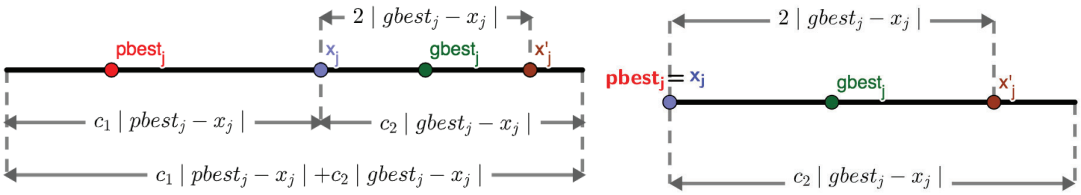
partículas dentro del enjambre. La actualización la lleva a cabo solo si el $gBest$ no ha cambiado en k sucesivas iteraciones y asume un $c_1 = c_2 = c$. Definido I como el conjunto de índices de las mejores $\frac{N}{2}$ partículas (las de menor $f(X_i)$), la estrategia adaptativa viene dada por la siguiente fórmula:

$$c_i = \begin{cases} 0.5 * c_i & i \in I \\ 2 * c_i & i \in \{1, ..., N\} - I \end{cases}$$

Similar conclusión acerca del rol de estos parámetros extraen en [113]. En este artículo se fijan en la probabilidad que tiene la partícula de acercarse o alejarse del $gBest$. Esta probabilidad p de acercarse al $gBest$ viene definida por los parámetros c_1 y c_2, pues son los que guían el movimiento. Estos parámetros determinan las posibles posiciones futuras de las partículas [9], dentro de las cuales se encuentran las que la acercan al $gBest$. Este hecho se ve perfectamente reflejado en la figura 3.3a. En ella se ve como para el caso de una dimensión concreta, las magnitudes de c_1 y c_2 determinan la amplitud del área de búsqueda con una mayor magnitud de la suma $c_1 + c_2$ resultando en mayor amplitud y menor probabilidad p de caer en una posición más cercana al $gBest$ que la actual. Es decir, cuanto mayores sean c_1 y c_2 (y por tanto $c_1 + c_2$), mayor será la exploración y viceversa. Para la situación en la que $pBest = X$, que se da siempre que hay una mejora individual histórica, el valor de c_1 es irrelevante y el mismo análisis se aplica pero solo teniendo en cuenta la magnitud de c_2. La figura 3.3b refleja esta circunstancia. Partiendo de esta lógica los autores proponen un ajuste adaptativo basado sobre todo en el ritmo de mejora de los $gBest$ y $pBest$ a lo largo de la ejecución, imponiendo una gran exploración cuando se observan mejoras pobres y pocos progresos durante el proceso. Los valores de los coeficientes se establecen en base al p que se quiera tener en cada momento, pues p es visto como un indicador de explotación (indica el grado entre 0 y 1 de explotación que hay en el algoritmo). Hay que decir que no tienen en cuenta la componente inercial, la eliminan del método.

En resumen, en lo concerniente a los factores de aceleración se conviene que tanto sus valores relativos (ratio entre ellos) como sus magnitudes afectan al comportamiento del método y a sus capacidades exploratívas. No obstante, la mayoría de trabajos en lo que se fijan es en lo primero para desarrollar sus estrategias de selección dinámica de estos hiperparámetros. Hay unanimidad en pensar que un incremento de c_1 en relación a c_2 aumenta la influencia del $pBest$ sobre el $gBest$ en el movimiento de las partículas y esto provoca una mayor independencia de las partículas a la hora de rastrear el espacio, lo que se traduce en una mayor diversidad y exploración. Estos trabajos dejan de lado la parte de la magnitud y controlan solo la proporción entre los dos factores. Otros trabajos en cambio se olvidan de la relación entre c_1 y c_2 y analizan los factores exclusivamente desde el punto de vista de sus magnitudes. Algunos lo hacen desde una perspectiva individual para cada factor y otros desde una conjunta (analizando los dos factores en conjunto). Y en estos trabajos centrados en magnitudes que no en relaciones, sí existen diversidad de interpretaciones. Hemos visto como algunos vinculaban una mayor magnitud con una mayor explotación, mientras que otros hacían justo lo opuesto. En definitiva, este apartado nos ha servido para descubrir que dependiendo de cómo se viera o desde qué punto de vista se analizara el impacto de estos parámetros, se daba lugar a diferentes interpretaciones y se

[9] Al conjunto de los puntos del espacio susceptibles de ser nuevas posiciones de la partícula en la siguiente iteración se le conoce como *dominio instantáneo de búsqueda*.



(a) $pbest_j \neq x_j$ (b) $pbest_j = x_j$

Figura 3.3: Representación gráfica del rango de movimiento de una partícula del PSO estándar en una dimensión concreta j obviando la componente inercial. Los puntos del segmento son puntos alcanzables por la partícula en la siguiente iteración. La longitud del segmento está determinada por las distancias entre la posición actual de la partícula (x_j) y las mejores posiciones históricas individual y global ($pbest_j$ y $gbest_j$) y por los valores de los factores de aprendizaje (c_1 y c_2). Una porción del segmento (señalada en la parte superior de la figura) representa los puntos que acercan x_j a $gbest_j$, que son todos los que caen entre x_j y su simétrico con respecto de $gbest_j$ (x'_j). La longitud de este subsegmento es el doble de la distancia entre x_j y $gbest_j$ y la proporción que representa del segmento total puede ser importante a la hora de valorar la exploración del método.

llegaba a diferentes conclusiones. Esto ha podido ayudar a constatar que aún a día de hoy el conocimiento acerca de estos parámetros no es el mejor puesto que no hay consenso claro sobre cómo ajustar sus valores, lo cual se ve reflejado también en el menor número de publicaciones sobre métodos adaptativos de estos coeficientes con respecto a las de adaptación del factor inercial, que son mucho más abundantes. Por supuesto hay una cifra importante también de publicaciones que realizan un ajuste dinámico de los tres hiperparámetros a la vez.

3.3.3. Notas finales

Aunque no interfieren en la lógica del método, los parámetros no específicos o generales sí tienen un gran impacto en su rendimiento. Sobre todo el parámetro N (tamaño de la población) es decisivo a la hora de explorar bien el espacio de búsqueda. Un N pequeño conlleva una peor búsqueda que uno grande, ya que se destinan menos recursos a la misma (menos agentes buscando). Esto se puede traducir en una búsqueda poco precisa y/o lenta. En cambio, cuanto mayor sea N, mejor será la búsqueda en cuanto a precisión e iteraciones requeridas (rapidez) pero consumirá cada vez más recursos computacionales, lo cual también aumenta el tiempo de ejecución. Un buen equilibrio se encuentra con un N entre 20 y 50 aunque en algunas publicaciones se usan valores más pequeños o más grandes en función del problema. En teoría, cuanto mayor es la dimensión del problema, mayor debería ser N, si bien es cierto que una observación común es lo poco sensible que es a los diferentes problemas [114]. Por tanto, no hay una preocupación notable por el ajuste de este hiperparámetro como sí la hay por los específicos (w, c_1 y c_2) y en cualquier caso su rol es evidente. A pesar de eso existen algunos métodos adaptativos de N en la literatura (véase [115]).

Como conclusión a esta sección podríamos decir que la dificultad principal a la hora de llevar a cabo la configuración paramétrica radica en los factores de aceleración c_1 y c_2, los cuales tienen un rol algo confuso que ha quedado de manifiesto cuando

repasábamos los distintos puntos de vista sobre ellos que existen en la literatura. El resto tienen una función más o menos clara y hay consenso acerca del efecto que suscitan en el comportamiento del método. Sin embargo, lo que hemos hecho en esta sección ha sido estudiar análisis por separado del papel de cada parámetro específico del PSO estándar, sin realmente tratar el método en su conjunto, lo cual aporta un grado de conocimiento aún más avanzado sobre la lógica y los mecanismos del mismo. Como el objetivo final del presente trabajo es llegar a comprender mejor el PSO, lo que haremos en adelante será precisamente eso: tratar de alcanzar un nivel de abstracción aún mayor analizando la regla de movimiento de las partículas en general.

3.4. Análisis del movimiento

Tras estudiar el PSO a través de sus parámetros como hemos hecho en las dos últimas secciones, es el momento de dar un paso más allá y hacerlo desde una perspectiva global, analizando el movimiento total de las partículas. Ya no estamos interesados en analizar el rol de los hiperparámetros o el rendimiento en sí del método sino en analizar el método en sí mismo. Esto fundamentalmente se consigue analizando cómo se mueven las partículas y cuáles son sus mecanismos para rastrear exitosamente el espacio de búsqueda, lo cual implica directamente analizar la regla de actualización de posiciones en su conjunto. Claramente esta regla está fuertemente influenciada por los componentes de aleatoriedad y por los hiperparámetros, así que su análisis pasa principalmente por examinar la función de la aleatoriedad en el método y su cohesión con los parámetros. En esta sección nos encargaremos de repasar en la literatura aquellas publicaciones que traten este tema.

3.4.1. Rol e impacto de la aleatoriedad

Podemos comenzar echando la vista atrás y recordando que, como vimos en el apartado de análisis de convergencia, el punto de atracción de las partículas, esto es, el punto alrededor del cual exploran, es P' con $p'_j = \frac{c_1 * \{rand_1\}_j * pbest_j + c_2 * \{rand_2\}_j * gbest_j}{c_1 * \{rand_1\}_j + c_2 * \{rand_2\}_j}$. Expresado de otra manera: $p'_j = a * pbest_j + (1-a) * gbest_j$ con $a = \frac{c_1 * \{rand_1\}_j}{c_1 * \{rand_1\}_j + c_2 * \{rand_2\}_j}$. Se ve claro como este punto es una media ponderada de los elementos $pbest_j$ y $gbest_j$, cuyos pesos vienen dados por las variables aleatorias y los coeficientes de aprendizaje. Tanto los coeficientes como las variables aleatorias determinan pues la influencia que tienen $pbest_j$ y $gbest_j$ en el movimiento de las partículas. Además, debido a la presencia de las variables aleatorias dicha influencia es variable y aleatoria, distinta en cada iteración excepto cuando $pbest_j = gbest_j$ (convergencia). Igualmente la exploración alrededor de P' depende de las variables aleatorias y los coeficientes de aceleración, siendo igual a $c_1 * \{rand_1\}_j + c_2 * \{rand_2\}_j$ y por tanto también variable y estocástica. Con esto podemos apreciar que el rol de las variables aleatorias está muy relacionado con el de los factores de aprendizaje ya que ambos van de la mano (aparecen multiplicados). Sin embargo, el impacto que tiene la presencia de la aleatoriedad no ha sido aún discutido.

En [31] se dice que la presencia de los números aleatorios en el algoritmo favorece una tendencia zigzagueante y ralentiza la convergencia de las partículas, con ello mejorando la exploración y previniendo convergencias prematuras a puntos no óptimos. En la misma línea apunta [117] destacando que la importancia de los coeficientes

aleatorios se debe a que con éstos los movimientos de las partículas son más complejos que sin ellos y les permiten explorar un área mayor del espacio de búsqueda. Gracias a este mayor área de búsqueda alrededor del $gBest$, las probabilidades de encontrar una mejor solución aumentan significativamente. En general, se asume que la presencia de los elementos de aleatoriedad contribuye a una mayor diversidad y una mayor exploración en el proceso de búsqueda. En varios artículos se compara el rendimiento del PSO con y sin la presencia de aleatoriedad en la regla de actualización de posiciones (PSO estándar vs PSO determinista) y en todos ellos se concluye que el determinista se comporta mucho peor. En concreto uno de ellos va más allá cuestionándose por qué la distribución de probabilidad de estas variables aleatorias ($\{rand_1\}_j$ y $\{rand_2\}_j$) es la uniforme en $[0, 1]$ ($\mathrm{U}(0, 1)$). El artículo en cuestión [116] decide comparar el método estándar con otras dos versiones del mismo en las que cambia la distribución de probabilidad de $\{rand_1\}_j$ y $\{rand_2\}_j$ por una uniforme en $[-1, 1]$ ($\mathrm{U}(-1, 1)$) y una normal de media 0 y varianza 1 ($\mathcal{N}(0, 1)$) respectivamente. Los resultados arrojados permiten a sus autores comprobar que el efecto positivo que provoca esta introducción de las variables aleatorias en la fórmula de actualización no es debida a que su distribución de probabilidad sea $\mathrm{U}(0, 1)$, es más, observan que con las otras distribuciones los resultados son incluso mejores. La explicación que dan es que con unos números aleatorios de mayor escala se expande el rango de velocidad de las partículas y esto hace que sean capaces de escapar de los óptimos locales y de encontrar el óptimo global.

3.4.2. *'Two steps forward, one step back'*

Aparte de lo comentado, existen varios trabajos que destacan propiedades de la aleatoriedad en el PSO y que podrían ser claves en su buen funcionamiento. Un ejemplo es el de [117], que expone que la aleatoriedad ayuda al PSO a superar el problema del '*two steps forward, one step back*'. Este problema aparece cuando una mejora en algo conlleva algún pequeño sacrificio o empeoramiento de alguna de sus partes. En el PSO se refiere a la potencial pérdida de un buen valor para alguna componente dimensional de una solución candidata (posición de la partícula) aunque la nueva solución sea mejor. Es decir, como consecuencia de que la actualización de las posiciones se hace de forma síncrona para las dimensiones (todas las componentes dimensionales se actualizan a la vez), puede suceder que alguna componente dimensional se aleje de la óptima tras la actualización a pesar de que la posición en sí se acerque a la óptima (y que lo hagan la mayoría de componentes). Lo ideal sería actualizar únicamente aquellas componentes que la acerquen pero esto no es posible ya que lógicamente desconocemos el óptimo. Pero lo que sí consiguen los coeficientes de aleatoriedad es que la atracción de la partícula hacia los atractores $pBest$ y $gBest$ sea desigual entre las diferentes dimensiones ya que cada dimensión j tiene asociadas sus propias variables aleatorias $\{rand_1\}_j$ y $\{rand_2\}_j$. Esto se traduce en una actualización que permite modificar más algunos componentes dimensionales que otros y por tanto conseguir que algunos componentes que pudieran ser buenos fueran preservados en la nueva posición, reduciendo los efectos del '*two steps forward, one step back*' aunque no solucionándolo completamente ya que todo en realidad depende de la aleatoriedad. Es por eso que, convencidos de la importancia de acabar con el problema, los autores proponen nuevas variantes del PSO estándar que pasan por actualizar solo algunas dimensiones. La primera (*PSORDS*) consiste en seleccionar aleatoriamente para cada partícula las dimensiones que vayan a ser actualizadas,

dejando el resto sin actualizar. Además las dimensiones seleccionadas se actualizan de acuerdo a la regla determinista sin componentes aleatorios:

$$v_{i,j} = w * v_{i,j} + c_1 * (pbest_{i,j} - x_{i,j}) + c_2 * (gbest_j - x_{i,j}) \tag{3.11}$$

No obstante esta primera variante sigue dependiendo de la aleatoriedad para evitar el '*two steps forward, one step back*' y es que no garantiza que las dimensiones seleccionadas sean las adecuadas. Por ello la segunda variante presentada ((*PSOHDS*)) utiliza una estrategia heurística de selección de dimensiones. En esta estrategia se seleccionan solo aquellas dimensiones que hacen mejor al $gBest$. La manera de hacerlo es comprobando para cada dimensión j si al reemplazar la componente de la peor posición global encontrada hasta la fecha ($gworst_j$) por la componente correspondiente de la mejor ($gbest_j$), la posición resultante $gWorst'$ mejora a $gWorst$. Si la mejora, entonces la dimensión j se selecciona para todas las partículas y si no, no. El proceso de selección se realiza únicamente cuando cambia el $gBest$ y su éxito recae en la suposición de que una componente dimensional del $gBest$ es un valor óptimo real si se puede obtener una mejor solución reemplazando la componente correspondiente del $gWorst$ por él porque implica que solo se seleccionen las componentes que verdaderamente hacen que el $gBest$ sea la mejor solución. Lo malo de esta variante es que selecciona las mismas dimensiones para todas las partículas cuando quizá lo más apropiado sería hacerlo de forma personalizada eligiendo cada partícula las dimensiones de las que aprender del $gBest$ en base a sus propias características. Esto es lo que procura la última de las variantes presentadas (*PSODDS*), que realiza la selección de dimensiones basándose en la diferencia entre las componentes del $gBest$ y las de X_i del siguiente modo:

$$s_{i,j} = \begin{cases} 1 & si \quad |gbest_j - x_{i,j}| > \frac{1}{d}\sum_{k=1}^{d}|gbest_k - x_{i,k}| \\ 0 & en\ caso\ contrario \end{cases} \tag{3.12}$$

con $s_{i,j}$ como la variable booleana que indica si la componente dimensional j de la partícula i ($x_{i,j}$) es seleccionada (1) o no (0) a ser actualizada. En este método se piensa que si la componente dimensional de la partícula está lejos del $gBest$, entonces aprender de él es urgente mientras que las dimensiones más similares no cambian. En conclusión, las tres variantes buscan que no todas las dimensiones sean actualizadas. Esto a criterio de los autores favorece la exploración, pues ayuda a mantener la diversidad del enjambre y prevenir la convergencia prematura. Además acaba con la existencia del problema '*two steps forward, one step back*'. Y según ellos, esas son las razones por las que la aleatoriedad funciona tan bien en el PSO estándar, ya que ésta contribuye a la actualización no uniforme de las dimensiones. Es por eso que se animan incluso a eliminarla por completo en sus dos últimas variantes potenciando de otra forma este efecto (la no uniformidad en la actualización). De las tres variantes, demuestran que la que ofrece un rendimiento superior es la tercera.

[117] no es el primer trabajo en abordar el problema del '*two steps forward, one step back*'. Previamente algún trabajo como [118] ya había presentado un método PSO para corregirlo. En este método (*CPSO-S*), en lugar de haber un único enjambre, hay d (uno por cada dimensión) y para formar una solución, las partículas unidimensionales de cada enjambre se combinan. En concreto la mejor partícula de cada enjambre es seleccionada para combinarse con las demás y formar la mejor solución histórica. A partir de esta solución, se va iterando por las distintas partículas de los distintos

enjambres actualizándola con el nuevo valor de la componente dimensional j correspondiente a la posición de la partícula i del enjambre j. La nueva solución es evaluada y su fitness se asocia con la partícula en cuestión, de manera que si supera el mejor fitness se convierta en la nueva mejor partícula de su enjambre. Por supuesto si la solución mejora también se actualiza el fitness asociado con las mejores partículas del resto de enjambres que componían la mejor solución. Tras haber iterado por todas las partículas de todos los enjambre, éstas actualizan sus posiciones de acuerdo a la regla original del PSO y se repite de nuevo el proceso. El algoritmo en pseudocódigo se muestra en 2. En [119] se propone un método híbrido del PSO estándar con el método recién mencionado porque se dice que este último tiene facilidad para quedarse atrapado en óptimos locales y al hibridizarlo con el PSO estándar, se conseguiría un método sin este inconveniente pero que a la vez conservara el buen rendimiento del nuevo método.

Algoritmo 2: Método PSO por dimensiones separadas de [118]

```
Crear e inicializar d enjambres PSO S_1 − S_d de N partículas 1-dimensionales
mientras No se cumpla el criterio de terminación hacer
    Seleccionar la mejor partícula b_1 − b_d de cada enjambre S_1 − S_d respectivamente
    para k = 1, ..., d, i = 1, ..., N hacer
        Seleccionar la partícula i−ésima del enjambre S_k (X_{i,k})
        Construir el vector solución sol_{i,k} = (b_1, b_2, ..., X_{i,k}, ..., b_d)
        Evaluar la solución y establecer fitness de la partícula X_{i,k} a f(sol_{i,k})
        Actualizar el mejor fitness de b_1 − b_d si es necesario
    fin
    Realizar actualizaciones PSO normales de todas las partículas en S_1 − S_d
fin
```

3.4.3. Diversidad en la búsqueda

Pero siguiendo con el análisis de la aleatoriedad, nos encontramos varios trabajos que adoptan una postura novedosa a la hora de analizar el PSO y es que lo hacen de forma vectorial, es decir, teniendo en cuenta todas las dimensiones a la vez y no tratándolas de manera independiente y aislada como hemos visto hasta ahora, por ejemplo en los anteriores trabajos. Esta perspectiva abre la puerta a nuevas observaciones. Así pues, comencemos definiendo el PSO de forma vectorial. Bajo esta nueva formulación, sus reglas de actualización de posiciones vendrían dadas por las siguientes expresiones:

$$V_i = w * V_i + c_1 * Rand_1 \circ (pBest_i - X_i) + c_2 * Rand_2 \circ (gBest - X_i) \tag{3.13}$$

$$X_i = X_i + V_i \tag{3.14}$$

donde los elementos $Rand_1$ y $Rand_2$ son vectores de números aleatorios uniformemente comprendidos entre 0 y 1 tales que $Rand_1 = (\{rand_1\}_1, ..., \{rand_1\}_j, ..., \{rand_1\}_d)$ y $Rand_2 = (\{rand_2\}_1, ..., \{rand_2\}_j, ..., \{rand_2\}_d)$, con $\{rand_1\}_j, \{rand_2\}_j \sim \mathrm{U}(0, 1)$. La operación $\circ$ representa el producto elemento a elemento (o de Hadamard) entre dos vectores. Hay quien, como alternativa a (3.13), utiliza la siguiente expresión:

$$V_i = w * V_i + c_1 * \Phi_1 \cdot (pBest_i - X_i) + c_2 * \Phi_2 \cdot (gBest - X_i) \tag{3.15}$$

donde Φ_1 y Φ_2 son matrices diagonales de números aleatorios uniformemente distribuidos en $[0, 1]$. Es decir, $\{\Phi_1\}_{j,j} = \{rand_1\}_j$ y $\{\Phi_2\}_{j,j} = \{rand_2\}_j$. En esta ocasión, la operación $(\cdot)$ es el producto matricial.

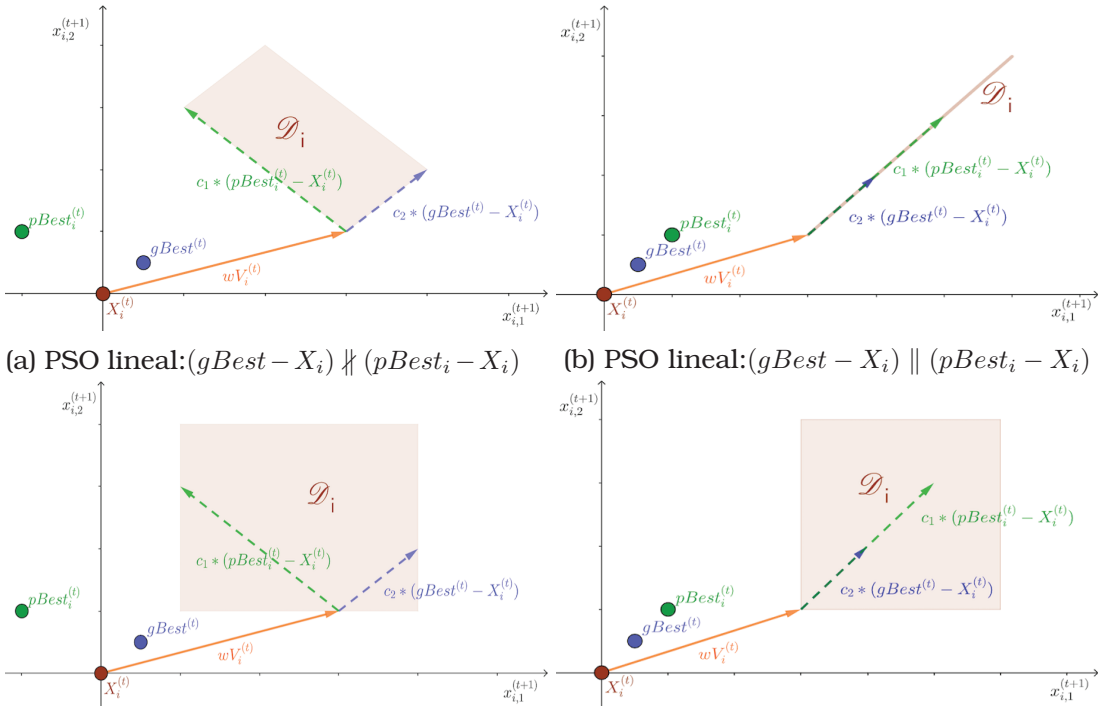
Uno de los precursores de este tipo de análisis vectoriales del PSO es el artículo de Wilke et al. [120]. En él se investiga el impacto que tiene la aleatoriedad en la diversidad en la búsqueda, comparando el comportamiento de las partículas en el PSO estándar con el de las de otra variante que difiere en la forma de introducir la aleatoriedad en la regla de actualización de velocidades. Esta variante denominada *PSO lineal* [121] lo que hace es escalar aleatoriamente los vectores social $(gBest - X_i)$ y cognitivo $(pBest_i - X_i)$ en vez de escalar cada uno de sus componentes por separado:

$$V_i = w * V_i + c_1 * rand_1 * (pBest_i - X_i) + c_2 * rand_2 * (gBest - X_i) \tag{3.16}$$

Es decir, en el PSO lineal todos los componentes de un mismo vector (el social o el cognitivo) son multiplicados por un mismo número aleatorio ($rand_1$ o $rand_2$). Lo que significa esto es que lo que están escalando estos números aleatorios en el PSO lineal es la magnitud de los vectores, cuando en el PSO clásico se escala cada componente de los vectores de forma independiente. Pues bien, analizando la ecuación (3.16) en [120] observan que puede interpretarse como la ecuación vectorial de un plano acotado en un espacio $d-$dimensional, aunque si los vectores cognitivo y social fueran paralelos, entonces la ecuación (3.16) sería la de un segmento (recta acotada). Esta observación nos desvela la forma que tiene el dominio instantáneo de búsqueda en el PSO lineal y queda reflejada para el caso bidimensional en las figuras 3.4a-3.4b. Las figuras 3.4c-3.4d muestran en cambio el dominio instantáneo de búsqueda en el PSO estándar. Comparando ambos dominios se ve claro como el del PSO estándar es capaz de abarcar más espacio (mayor dominio instantáneo) y por tanto ofrece mayor diversidad. Profundizando en este análisis, los autores investigan sobre las trayectorias de las partículas en uno y otro método (lineal vs clásico) y se dan cuenta de que en el lineal las partículas colapsan a la línea recta que une el $pBest$ con el $gBest$. Esto supone una ausencia total de diversidad direccional, o sea, direcciones de búsqueda fijas que no cambian con el tiempo. Cuanto más bajo es el factor inercial w, antes se alcanza el colapso a búsquedas en líneas. Este colapso se da siempre que el vector social sea paralelo al cognitivo como sabemos por la figura 3.4b, pero en realidad no hace falta que sea exactamente paralelo para notar los efectos comentados, pues con que el ángulo entre dichos vectores sea lo suficientemente pequeño, el dominio instantáneo de búsqueda será un estrecho y largo plano que no cambiará hasta encontrarse un nuevo $gBest$. Si lo que cambia es el $pBest$, la partícula seguirá buscando en una línea. Así, mientras el $gBest$ no sea actualizado, el enjambre conducirá N búsquedas en líneas por un número grande de iteraciones hasta que lo sea y las partículas vuelvan a buscar por un breve periodo de tiempo en planos. Este efecto de escasa o nula diversidad direccional no se observa en cambio en el PSO clásico, lo cual según los autores es clave para explicar su buen rendimiento y la utilidad de los componentes aleatorios en el método.

3.4.4. Invarianza rotacional

Tras demostrar que el escalamiento aleatorio de solo la magnitud de los vectores social y cognitivo no es suficiente para tener una buena diversidad en la búsqueda y explorar bien (en muchas direcciones), los mismos autores se proponen en [122] investigar la dependencia de los dos métodos en el marco de referencia en el que está



(a) PSO lineal:$(gBest - X_i) \nparallel (pBest_i - X_i)$ (b) PSO lineal:$(gBest - X_i) \parallel (pBest_i - X_i)$

(c) PSO clásico:$(gBest - X_i) \nparallel (pBest_i - X_i)$ (d) PSO clásico:$(gBest - X_i) \parallel (pBest_i - X_i)$

Figura 3.4: Representación gráfica del dominio instantáneo de búsqueda de una partícula ($\mathscr{D}_i$) del PSO lineal ((a)-(b)) vs PSO estándar ((c)-(d)) para un problema bidimensional. Se aprecia como en el PSO lineal cuando los vectores social y cognitivo son paralelos (b), se pierde diversidad direccional mientras que en el PSO estándar (d), no.

definido el problema de optimización. El marco de referencia en el que se define un problema es aquel que sirve para representar las soluciones del mismo y se refiere al sistema de coordenadas escogido para ello. Todo problema de optimización tiene asociado consigo un marco de referencia en el cual se establece la función objetivo y el cual se utiliza para buscar la solución. Como ejemplo, la figura 3.5 ilustra la situación en la que hay dos posibles marcos de referencia $\boldsymbol{x}$ y $\hat{\boldsymbol{x}}$ relacionados por un factor de escala s, un vector de traslación $\vec{t}$ y una matriz de rotación Q,[10] de tal modo que $\hat{\boldsymbol{x}} = \vec{t} + sQ\boldsymbol{x}$. La misma función objetivo es expresada en cada uno de los marcos de referencia, $\hat{f}$ en $\hat{\boldsymbol{x}}$ y f en $\boldsymbol{x}$, de modo que $\hat{f}(\hat{\boldsymbol{x}}) = f(\hat{\boldsymbol{x}}) = f(\vec{t} + sQ\boldsymbol{x}) = \hat{f}(\vec{t} + sQ\boldsymbol{x})$. Y como un cambio entre sistemas de referencia equivale a un cambio en la función, esto también puede interpretarse como que f y $\hat{f}$ son dos funciones distintas descritas en un mismo marco de referencia $\boldsymbol{x}$ ($\hat{f}(\boldsymbol{x}) = f(\vec{t} + sQ\boldsymbol{x})$). Sea como fuere, el hecho de elegir uno u otro marco de referencia para afrontar la optimización puede resultar decisivo en el rendimiento del algoritmo optimizador. Esto sucede porque algunos algoritmos dependen del marco de referencia. Se usa la palabra 'invarianza' para describir cuándo un algoritmo de optimización es independiente de las transformaciones que se puedan hacer en el marco de referencia. Así, se dice que un algoritmo es invariante en escala si su rendimiento no se ve comprometido (es independiente) por el escalamiento uniforme de todas las variables o es rotacionalmente invarian-

[10]Una matriz de rotación $Q \in Orth^+$ es una matriz ortogonal ($Q^{\mathrm{T}} = Q^{-1}$) cuyo determinante es igual a 1.

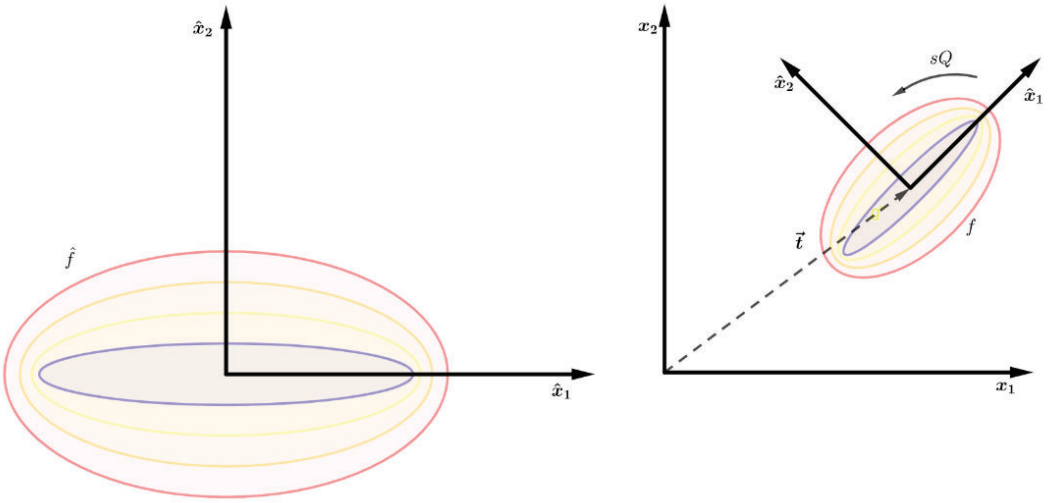


Figura 3.5: Gráfico de contorno de una función f en dos marcos de referencia distintos x y $\hat{x}$. Los marcos están trasladados, escalados y rotados uno con respecto del otro.

te si no se ve afectado por la rotación del sistema de referencia. Esta característica de los algoritmos de optimización (invarianza en el marco de referencia) es generalmente deseable ya que les hace ser aplicables a más clases de problemas diferentes (son más generales). Por su parte los algoritmos dependientes del marco implican una preferencia por algún marco de referencia específico, lo cual hace que estén sesgados hacia una subclase de problemas que presenten un favoritismo claro por ser resueltos en ese marco de referencia. Como en general no se puede saber cuándo un problema está mejor preparado para resolverse en un marco de referencia particular ni en cuál, siempre suele ser mejor optar por un algoritmo marco-invariante. Aquí conviene mencionar, como ejemplo de problemas con predilección por resolverse en un marco de referencia concreto, que existe una clase muy extendida de problemas que son descomponibles por dimensiones, es decir, son problemas d-dimensionales compuestos simplemente por la suma de d problemas unidimensionales (sus variables son completamente independientes). A las funciones objetivo descomponibles también se les conoce como *separables* y se sabe que las funciones de test (las que se emplean para probar nuevos optimizadores) más populares son de este tipo [127]. En realidad se dice que una función objetivo f es separable si los es con respecto a cada coordenada y es separable con respecto a una coordenada j si el valor óptimo para dicha coordenada no depende de la elección de las demás.[11] Matemáticamente sea $\vec{e_j} \in \mathbb{R}^n$ el vector canónico j-ésimo, f es separable si:

$$\forall j = 1, ..., d \quad \forall \vec{y}.\vec{z} \in \mathbb{R}^n, \quad \underset{\mu \in \mathbb{R}}{\operatorname{argmin}} f(\vec{y} - y_j\vec{e_j} + \mu\vec{e_j}) = \underset{\mu \in \mathbb{R}}{\operatorname{argmin}} f(\vec{z} - z_j\vec{e_j} + \mu\vec{e_j})$$

Una buena manera de convertir una función separable en no separable es rotándola [127], es decir, tomando como nuevo marco de referencia uno rotado con respecto

[11] Siendo exactos, el concepto de separabilidad es más amplio que el de descomponibilidad. El conjunto de las funciones separables es superconjunto del de las descomponibles. La característica que nos incumbe a nosotros es la de separabilidad. Una función separable muestra una clara preferencia por ser resuelta en la base canónica.

al original y expresando la función en este nuevo marco de referencia.[12] De este modo, las nuevas variables son combinaciones de las anteriores y se acaba con la independencia entre ellas.

Dicho esto, Wilke et al.[122] estudian la marco-invarianza del PSO estándar y descubren que éste es rotacionalmente variante. La invarianza se prueba comprobando que el resultado de actualizar las partículas en un marco de referencia $\hat{\boldsymbol{x}}$ sea el mismo que el de hacerlo en otro $\boldsymbol{x}$. Para el PSO estándar, la prueba sería:

$$\begin{aligned}
\hat{X}_i^{(t+1)} &= \hat{X}_i^{(t)} + w\hat{V}_i^{(t)} + c_1\Phi_1(\hat{pBest}_i^{(t)} - \hat{X}_i^{(t)}) + c_2\Phi_2(\hat{gBest}^{(t)} - \hat{X}_i^{(t)}) \\
&= sQX_i^{(t)} + \vec{t} + wsQV_i^{(t)} + c_1\Phi_1(sQpBest_i^{(t)} + \vec{t} - sQX_i^{(t)} - \vec{t}) \\
&+ c_2\Phi_2(sQgBest^{(t)} + \vec{t} - sQX_i^{(t)} - \vec{t}) \\
&= sQX_i^{(t)} + \vec{t} + wsQV_i^{(t)} + sc_1\Phi_1Q(pBest_i^{(t)} - X_i^{(t)}) + sc_2\Phi_2Q(gBest^{(t)} - X_i^{(t)}) \\
&\neq sQX_i^{(t)} + \vec{t} + wsQV_i^{(t)} + sc_1Q\Phi_1(pBest_i^{(t)} - X_i^{(t)}) + sc_2Q\Phi_2(gBest^{(t)} - X_i^{(t)}) \\
&= sQ(X_i^{(t)} + wV_i^{(t)} + c_1\Phi_1(pBest_i^{(t)} - X_i^{(t)}) + c_2\Phi_2(gBest^{(t)} - X_i^{(t)})) + \vec{t} \\
&= sQX_i^{(t+1)} + \vec{t}
\end{aligned}$$

La prueba demuestra que en el PSO estándar importa el sistema de referencia escogido para realizar la actualización de las partículas ya que no se alcanza la misma posición si se hace desde un marco o desde otro. La culpa de que esto sea así la tiene la propiedad de no conmutatividad de matrices, pues si $\Phi_l Q = Q\Phi_l,\ l = 1, 2\ , \forall Q \in Orth^+$, entonces se cumpliría la marco-invarianza del PSO. Pero esa condición solo se da cuando $\Phi_l = a_l I_d,\ l = 1, 2$ con $a_l \in \mathbb{R}$ y I_d la matriz identidad de tamaño $d \times d$. Es decir, se da solo en el PSO lineal. Además demuestra que el PSO es invariante en escala y traslación y por lo único que deja de ser invariante es por la rotación. Puesto que el único PSO rotacionalmente invariante es el lineal y éste ya se sabía que carecía de diversidad direccional, los autores tratan de crear un método (*WPSO*) que pueda conciliar ambos aspectos aproximando el requisito de invarianza rotacional en un sentido estocástico (en media). Quieren que al menos el PSO sea 'aproximadamente invariante' y lo hacen permitiendo un Φ_l que sea muestreado en un pequeño rango en torno a I_d, o sea, imponiendo pequeñas perturbaciones en la dirección de los vectores social y cognitivo. El nuevo método separa el escalamiento aleatorio de la magnitud de los vectores del de la dirección de los mismos. Y las pequeñas perturbaciones direccionales las consiguen introduciendo matrices de rotación aleatorias e independientes R_l que son multiplicadas por cada uno de los vectores social y cognitivo:

$$V_i = w * V_i + c_1 * rand_1 * R_1 \cdot (pBest_i - X_i) + c_2 * rand_2 * R_2 \cdot (gBest - X_i) \tag{3.17}$$

La forma de construir dichas matrices de rotación aleatorias R_l es mediante mapas exponenciales con el objetivo de buscar la eficiencia computacional. Para ello emplean el método de las series de Taylor, cuya expansión general del mapa exponencial viene dada por:

$$e^W = \sum_{n=0}^{\infty} \frac{W^n}{n!}$$

[12] Nótese de nuevo que transformar la función con respecto al marco de referencia es equivalente a transformar el marco de referencia con respecto a la función.

Como lo que interesa es que el mapeo resulte en una matriz de rotación R_l, entonces W ha de ser una matriz antisimétrica.[13] Además, como solo se contemplan rotaciones pequeñas, la construcción del mapa exponencial se hace utilizando solo los dos primeros términos de la serie. De esta forma, las matrices de rotación R_l quedan definidas por:

$$R_l = I_d + W_l; \qquad W_l = \frac{\alpha\pi}{180}(A_l - A_l^{\mathrm{T}}) \tag{3.18}$$

con A_l matrices cuadradas $d \times d$ de números aleatorios uniformemente distribuidos en $[-0.5, 0.5]$ ($\{A_l\}_{i,j} \sim \mathrm{U}(-0.5, 0.5)$) y α el factor de escala direccional. Así, se logra una aproximación lineal a una matriz de rotación, que es válida solamente para perturbaciones pequeñas pero cuya complejidad computacional es escasa. Las nuevas matrices de rotación aproximadas permiten rotar los vectores social y cognitivo en planos arbitrarios una cantidad α (en grados) de forma aproximada siempre y cuando α no sea grande.

Los trabajos de Wilke et al. [120, 122] no son los únicos en abordar la varianza rotacional del PSO y su relación con la diversidad direccional. Por ejemplo ya en [123] Clerc expuso que cuando el $pBest$ o el $gBest$ compartían una misma coordenada con el vector posición X, entonces el componente cognitivo $c_1 * Rand_1 \circ (pBest - X)$ o el social $c_2 * Rand_2 \circ (gBest - X)$ perdían esa dimensión. En [125], Bonjadi et al. primero investigan las propiedades de la multiplicación matricial entre Φ_l y los vectores cognitivo y social respectivamente. Resumen las propiedades de la operación en cuatro puntos:

(1) Φ_l no preserva la longitud de los vectores

(2) Φ_l no preserva la dirección de los vectores

(3) A medida que los vectores se acercan a un eje de coordenadas, la probabilidad de que los cambios en sus direcciones sean menores se incrementa rápidamente. Hasta el punto de no rotar nada si el vector está exactamente en un eje.

(4) El vector resultante de aplicar Φ_l reside en el mismo cuadrante (ortante en espacios multidimensionales) que el vector original, ya que los valores aleatorios de Φ_l son positivos.

Estas propiedades no hacen más que aportar más detalles sobre un hecho que ya sabemos: la varianza rotacional del PSO. Lo que se deduce de ellas es que la probabilidad de rotar los vectores social $\vec{g} = (gBest - X)$ o cognitivo $\vec{p} = (pBest - X)$ menos de θ grados en el PSO ($\mathbb{P}(\alpha < \theta)$ con $\alpha = \angle(\vec{g}, \Phi_2\vec{g})$ o $\alpha = \angle(\vec{p}, \Phi_1\vec{p})$) depende de los vectores $\vec{g}$ o $\vec{p}$ y no es controlable. Su solución es similar a la de [122], construyendo un método (*RotmPSO*) con la misma regla de actualización dada por la expresión (3.17) y difiriendo solamente en la manera de definir las matrices de rotación R_l. Su manera de hacerlo (la de [125]) es utilizando la definición de matriz de rotación euclídea en un plano principal $x_a x_b$ por un ángulo α de [124]:

$$Rot_{a,b}(\alpha) = \left\{ \begin{array}{ll} rot_{a,a} = rot_{b,b} = \cos(\alpha) & \\ rot_{a,b} = -\sin(\alpha) & \\ rot_{b,a} = \sin(\alpha) & \\ rot_{j,j} = 1, & j \neq a,\ j \neq b \\ rot_{i,j} = 0, & en\ caso\ contrario \end{array} \right\} \tag{3.19}$$

[13] Una matriz antisimétrica es una matriz cuadrada W cuya traspuesta es igual a su negativa, es decir $W^{\mathrm{T}} = -W$.

Esta expresión permite rotar un vector en alguno de los planos formados por dos de los ejes de coordenadas. Para poder rotar en cualquier plano, que es lo que se pretende, los autores multiplican todas las matrices de rotación correspondientes a cada plano principal entre ellas. Así definen las matrices R_l del nuevo PSO:

$$R_l = Rot(\mu, \sigma) = \prod_{1 \leq i < j \leq d} Rot_{i,j}(\alpha_{i,j}) \tag{3.20}$$

donde $Rot_{i,j}(\alpha_{i,j})$ es la matriz de rotación en el plano $x_i x_j$ dada por (3.19) y cuyo ángulo de rotación $\alpha_{i,j}$ es muestreado de una distribución normal de probabilidad con media μ y desviación típica σ ($\alpha_{i,j} \sim \mathcal{N}(\mu, \sigma)$). Este método, a diferencia del de Wilke (WPSO[122]), no usa ninguna aproximación en el cálculo de las R_l y por tanto es preciso. Además introduce aleatoriedad en la elección del ángulo de rotación, el cual pasa a ser una variable aleatoria más del método. Sin embargo, el coste computacional a priori es mucho mayor ya que al haber $\binom{d}{2} = \frac{d(d-1)}{2}$ planos principales, la generación de R_l entraña la multiplicación matricial de ese elevado número de matrices y supone una complejidad de $\mathcal{O}(d^5)$. Frente a esto los autores proponen una implementación que reduce la complejidad a $\mathcal{O}(d^2)$, que es la misma que la de [122] y sin sacrificar precisión. Esta implementación se vale de que las matrices $Rot_{i,j}(\alpha_{i,j})$ de (3.20) solamente contienen cuatro valores no nulos distintos de 1 (ver (3.19)) y al multiplicarlas por un vector, el resultado solo cambia dos componentes del vector, por lo que la complejidad de la multiplicación matricial se reduce a $\mathcal{O}(1)$.

También en [126] se señala que el PSO tiene un claro bias (sesgo) en la dirección de movimiento de sus partículas que depende de la dirección de los ejes de coordenadas elegidos y por tanto lo que hace que éste no sea rotacionalmente invariante. El bias es hacia las direcciones paralelas a los ejes. Esto lo constata [128], tratando de aportar una explicación teórica a este hecho. Su explicación es que los cambios de dirección en el vector aceleración (suma de los vectores social y cognitivo) provocados por la aleatoriedad, se caracterizan por dos propiedades principales: una es que las partículas tienden de media a alejarse de las direcciones paralelas a los ejes y acercarse a direcciones paralelas a las diagonales, pero la otra es que la varianza de estos cambios es mucho menor para direcciones cercanas a las de los ejes (del vector aceleración) que para las cercanas a las diagonales. A pesar de la propiedad primera, su combinación con la segunda hace que las direcciones paralelas a los ejes sean las preferidas y exista un claro bias hacia ellas. En realidad esta explicación está relacionada con la propiedad 3 de [125] que dice que a medida que la dirección del vector se acerca a la del eje de coordenadas, la probabilidad de que sufra un cambio menor en su dirección se incrementa rápidamente. Y a este bias en la dirección de movimiento de las partículas hay que añadirle también un bias en la magnitud tal y como apunta [128]. Este bias también depende de la dirección del vector aceleración y por tanto de la base de coordenadas. En concreto, de media las partículas se mueven más rápido cuando sus vectores aceleración son paralelos a una diagonal (los cambios en sus magnitudes al aplicárseles la aleatoriedad son menores).

La inclinación que muestra el PSO por explorar en direcciones paralelas a los ejes coordenados y que demuestra su varianza rotacional, tiene repercusiones obvias en el proceso de optimización. Como ya explicamos al introducir el concepto de marco-invarianza, los algoritmos no invariantes (como el PSO) rinden de forma diferente dependiendo del sistema de referencia y por tanto la elección del mismo se vuelve fundamental sobre todo para cierta clase de funciones que exhiben mayor facilidad

por ser optimizadas en un marco de referencia específico. Es el caso de las funciones separables, que muestran predilección por la base canónica como base del sistema de coordenadas. El efecto que tiene la no invarianza rotacional del PSO en la optimización de funciones separables es investigado en [129]. Allí llegan a la conclusión de que el PSO rinde muy bien en funciones separables donde las variables son independientes entre sí, pero su rendimiento decae destacablemente con rotaciones del sistema de coordenadas que hacen que las variables se vuelvan dependientes (y por ende las funciones no separables). En funciones no separables los costes de la búsqueda del PSO se incrementan aproximadamente de manera lineal con el número de condición de la función objetivo para números superiores a 100. En definitiva, como era de suponer, al PSO le resulta mucho más complicado optimizar funciones no separables que separables y esto lógicamente se debe a su sesgo hacia los ejes coordenados.

La evidente degradación del método cuando ha de optimizar funciones que no son separables, no hace más que motivar la creación de nuevas variantes PSO rotacionalmente invariantes. En [130] proponen un método (bautizado como *normLinked PSO*) que actualice las velocidades de las partículas de forma conjunta para todas las dimensiones en lugar de hacerlo por separado. Este método determina la dirección de movimiento según el signo de los elementos de los vectores social y cognitivo (según el ortante en el que residen) y no según sus direcciones específicas. De este modo dicen los autores acabar con la constricción a los ejes coordenados del PSO estándar ya que la distribución del ángulo del vector velocidad no está sesgado cuando estos vectores están próximos a un eje. La regla de actualización de la velocidad es la siguiente:

$$V_i = w * V_i + c_1 * \|pBest_i - X_i\|_2 * \Phi_1 \cdot \mathrm{sgn}(pBest_i - X_i) + c_2 * \|gBest - X_i\|_2 * \Phi_2 \cdot \mathrm{sgn}(gBest - X_i) \tag{3.21}$$

donde sgn es la función signo aplicada a cada elemento vectorial (devuelve el vector de signos de los componentes vectoriales) y $\|\cdot\|_2$ representa la norma euclídea.

Por su parte en [131] presentan una de las versiones de PSO invariante más conocidas, la denominada *SPSO2011*. En ella, en lugar de simular una atracción aleatoria de las partículas hacia las mejores posiciones encontradas para determinar la siguiente posición, lo que se hace es muestrear directamente en una hiperesfera $\mathcal{H}$ con centro C en la media de las influencias social, cognitiva y la actual posición. El radio de la hiperesfera es la distancia entre su centro y la posición actual de la partícula. Así, la hiperesfera correspondiente a la partícula i se define como

$$\begin{aligned} &\mathcal{H}_i(C_i, \|C_i - X_i\|) \\ &C_i = \frac{X_i + (X_i + c(pBest_i - X_i)) + (X_i + c(gBest - X_i))}{3} = X_i + c\frac{pBest_i + gBest - 2X_i}{3} \end{aligned} \tag{3.22}$$

y la nueva regla de actualización de posiciones queda:

$$X_i = wV_i + X_i' \tag{3.23}$$

con X_i' el punto aleatorio muestreado de la hiperesfera $\mathcal{H}_i$. De acuerdo al autor, dicho punto no necesariamente ha de ser muestreado de manera uniforme (siguiendo una distribución esférica uniforme). De hecho, por defecto se propone que el muestreo se realice a partir de una dirección con distribución uniforme y un radio también

distribuido uniformemente, es decir, mediante el siguiente algoritmo:

Algoritmo 3: Generar $X_i' \in \mathcal{H}_i(C_i, \|C_i - X_i\|)$

Input: $C_i, \|C_i - X_i\|$
Output: X_i'
1 Extraer muestra $\vec{z} \sim \mathcal{N}(\vec{0}, I)$
2 $\vec{z'} = \frac{\vec{z}}{\|\vec{z}\|}$
3 Extraer muestra $r \sim \mathrm{U}(0, \|C_i - X_i\|)$
4 $X_i' = C_i + r\vec{z'}$

Si quisiéramos que el muestreo fuese uniforme en la esfera (todo punto de la misma pudiera ser escogido con igual probabilidad), tendríamos que emplear la distribución $(\mathrm{U}(0, \|C_i - X_i\|))^{\frac{1}{d}}$ para el radio de muestreo r. De la forma del algoritmo 3, la probabilidad está más concentrada en las regiones cercanas al centro. La idea del método se ve perfectamente reflejada en la figura 3.6.

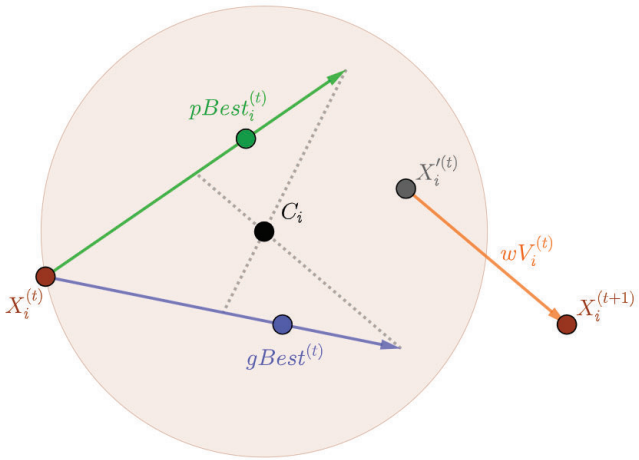


Figura 3.6: Representación gráfica del mecanismo de actualización de posiciones en el PSO propuesto por [131] para un problema bidimensional. El método se basa en la elección aleatoria de un punto X_i' dentro de una hiperesfera $\mathcal{H}_i$ (circunferencia en dos dimensiones) de centro C_i igual a la media de la posición actual, una posición influenciada por la componente social y otra posición influenciada por la componente cognitiva y de radio igual a la distancia entre la posición actual y el centro. A dicho punto se le suma también la componente inercial para constituir la siguiente posición de la partícula.

Aunque la versión original del SPSO2011 fuera la expresada en las ecuaciones (3.22) y (3.23), más tarde esté método ha sido presentado de otras maneras, matizando la definición del centro de la hiperesfera C_i de la fórmula (3.22). El matiz tiene que ver con los puntos que se toman para realizar el cálculo del centro. Siempre se toman el punto actual, un punto vinculado a la componente cognitiva y otro punto vinculado a la componente social. Ahora bien, si nos fijamos en (3.22), las componentes social y

cognitiva que se emplean en el cálculo carecen de elementos aleatorios como sí los tienen en el PSO estándar. Y es por eso que es habitual ver una presentación alternativa del método como la dada en [132], que sí incluya los elementos de aleatoriedad:

$$C_i = \frac{X_i + (X_i + c_1\Phi_1(pBest_i - X_i)) + (X_i + c_2\Phi_2(gBest - X_i))}{3} \tag{3.24}$$

Este aspecto es discutido en [133], donde argumentan que la diversidad del SPSO2011 original es pobre por no incluir elementos aleatorios en (3.22) y cuando los incluye en (3.24), lo hace reduciendo a la vez el radio de la hiperesfera ya que el rango de éstos es de 0 a 1. Esta reducción radial conlleva una menor exploración. Para dar solución a la falta de diversidad del SPSO2011, en [133] proponen una nueva regla:

$$C_i = \frac{X_i + (X_i + c_1\Psi(pBest_i - X_i)) + (X_i + c_2(2I - \Psi)(gBest - X_i))}{3} \tag{3.25}$$

con Ψ una matriz diagonal de números aleatorios uniformes en $[0, 2]$. Lo que cambia principalmente en esta nueva versión es que se amplía el rango de las variables aleatorias al doble (los elementos de Φ están en el intervalo $[0, 1]$ y los de Ψ en el $[0, 2]$) y que la relación entre la influencia social y la cognitiva pasa a ser complementaria, esto es, si la una resulta baja, la otra se vuelve alta y viceversa. Así el radio de búsqueda se mantiene a la par que se añade diversidad en la selección del centro debido a la inclusión de la aleatoriedad en su fórmula.

Otro PSO rotacionalmente invariante, surgido además para solucionar otros problemas del PSO estándar como el estancamiento o la convergencia local (a óptimos locales) no garantizada, es el propuesto por [134] (*LcRiPSO*). En él, se reemplazan los puntos de atracción $pBest$ y $gBest$ por puntos aleatorios en la vecindad de los mismos. Estos puntos son muestras extraídas de una distribución de probabilidad normal de media $pBest$ y $gBest$ respectivamente y varianzas σ_1^2 y σ_2^2. La regla modificada de actualización queda:

$$V_i = wV_i + c_1 * rand_1\left(\mathcal{N}(pBest_i, \sigma_1^2 I) - X_i\right) + c_2 * rand_2\left(\mathcal{N}(gBest, \sigma_2^2 I) - X_i\right) \tag{3.26}$$

En este método, al igual que en el PSO lineal, las matrices aleatorias Φ_1 y Φ_2 son sustituidas por números, regulando así únicamente la magnitud de los vectores social y cognitivo. A cambio, la diversidad direccional la consiguen perturbando aleatoriamente los atractores social y cognitivo estándar. Esta perturbación está dirigida por una distribución normal centrada en los atractores ($pBest$ y $gBest$), lo cual quiere decir que todas las posiciones que disten lo mismo de ellos, tienen la misma probabilidad de ser muestreadas. Dicho de otro modo, la probabilidad depende exclusivamente de la magnitud del vector que une el atractor estándar con la muestra y no de su dirección.[14] Además el valor de las varianzas σ^2 afecta al área de muestreo, con un valor alto resultando en una mayor probabilidad de generar la muestra lejos del atractor estándar y al revés.

Por último, siguiendo en la línea de [134] de perturbar aleatoriamente los puntos de atracción convencionales $pBest$ y $gBest$, en [135, 136] lo hacen de un modo diferente. En vez de emplear una distribución de probabilidad para extraer muestras aleatorias en sus vecindades, emplean vectores diferencia entre mejores posiciones individuales

[14] Esto es así porque la distribución de probabilidad usada es una normal multivariada cuyas variables son independientes entre sí y por tanto tiene la propiedad de ser esféricamente simétrica respecto a su media (su centro).

de partículas aleatorias. Así, la forma de la regla de actualización es la misma que en [134] expresando los nuevos atractores como $pBest'_i$ y $gBest'$:

$$V_i = wV_i + c_1 * rand_1(pBest'_i - X_i) + c_2 * rand_2(gBest' - X_i) \tag{3.27}$$

pero lo que cambia es la definición de los mismos, siendo en este último:

$$\begin{aligned} pBest'_i &= pBest_i + c_d(pBest_{i_1} - pBest_{i_2}) \\ gBest' &= gBest + c_d(pBest_{i_3} - pBest_{i_4}) \end{aligned} \tag{3.28}$$

donde c_d es una constante en el intervalo $[0, 1]$ y i_1, i_2, i_3, i_4 son índices aleatorios de partículas del enjambre tales que $i \neq i_1 \neq i_2 \neq i_3 \neq i_4$. A este método lo hemos denominado *HRiPSO*.

3.4.5. Breve reflexión final

Recapitulando lo visto en esta sección, podemos concluir que la presencia de aleatoriedad es clave en el funcionamiento del PSO. Ya sabíamos por los análisis de convergencia que era fundamental en su rendimiento, pero desconocíamos las razones, así como el papel que juega en él. En esta sección hemos analizado el método en su conjunto, lo que nos ha permitido dar respuesta a estas cuestiones.

En primer lugar, hemos comprobado que los componentes aleatorios afectan tanto a la influencia que tienen los puntos de atracción $pBest$ y $gBest$ en el movimiento de las partículas (y por tanto en la búsqueda) como al nivel de exploración que se tiene. Pero además, su mera presencia implica una mayor diversidad en la búsqueda, que a la postre hemos descubierto que es debida a que el escalamiento aleatorio se realiza de manera independiente para cada dimensión. Esta forma de introducir la aleatoriedad tiene un gran impacto en el comportamiento del PSO. Por un lado, provoca que exista diversidad, es decir, que se pueda buscar en más regiones diferentes del espacio y por otro, que el algoritmo sea rotacionalmente variante. Esto último significa que la búsqueda está condicionada por la orientación del sistema de coordenadas elegido. Sus implicaciones básicamente son que se pierde diversidad cuando la dirección de búsqueda (dirección de la velocidad de la partícula) es paralela a alguno de los ejes coordenados. Esto supone que el PSO muestre una clara inclinación por explorar en direcciones paralelas a los ejes, lo que se traduce en una búsqueda por dimensiones, en la que se actualizan los componentes dimensionales por separado. Es obvio que este sistema favorece la optimización de funciones separables. De hecho comentamos cómo su rendimiento decae considerablemente cuando se trata de optimizar funciones no separables [129].

Por otro lado, algunos artículos apuntaban a la capacidad que tiene la aleatoriedad de solucionar el problema del *'two steps forward, one step back'*. Sin embargo, reflexionando a posteriori es facil darse cuenta de que este problema asume implícitamente que las funciones objetivo sean separables, puesto que el problema se basa en el tratamiento de cada componente dimensional de manera ajena e independiente al resto. Este problema asume que el óptimo de una función se averigua hallando el óptimo de cada coordenada por separado. Es decir, asume que se puede llevar a cabo una búsqueda por dimensiones y que, si no, puede suceder que al actualizar varias coordenadas de golpe, alguna de ellas empeore aunque en general se mejore la solución. No obstante, todo ello se da solamente cuando lo que se está optimizando es una función separable. Y decir que la aleatoriedad del PSO ayuda a solucionar el

problema del *'two steps forward, one step back'*, en realidad es decir que ayuda a que el PSO explore el espacio dimensión a dimensión, o sea que lo haga actualizando las partículas con movimientos paralelos a los ejes coordenados. Precisamente esto concuerda con los análisis de invarianza rotacional que vimos después. Con lo cual, los métodos que revisamos para mejorar la respuesta del PSO al problema del *'two steps forward, one step back'*, son solamente válidos para optimizar funciones separables, pues tratan de explotar esta característica del PSO estándar de rendir muy bien en funciones separables.

En definitiva, esta sección nos ha ayudado a comprender mejor cómo funciona el PSO y qué es lo que hace que tenga tan buen rendimiento. El artífice es la aleatoriedad. La aleatoriedad perturba la influencia, magnitud y dirección de los vectores social y cognitivo, lo cual tiene un gran impacto en la diversidad de la búsqueda. Sin embargo, el buen rendimiento del algoritmo es destacable en funciones separables y lo es por algo que no estaba planeado por sus autores en [6]. Además, como se destacaba en [127], la mayoría de funciones de test utilizadas para comparar optimizadores son separables, por lo que el PSO sale muy bien parado y sus resultados son potenciados. Podríamos afirmar pues que gran parte del éxito del PSO es accidental y algo sobrevalorado. En realidad, la aleatoriedad fue introducida en el método original [6] para aportar más diversidad y no para optimizar por dimensiones cuando, en cambio, resulta ser esto último lo que más ensalza su rendimiento. Para una mayor comprensión si cabe de las claves positivas de funcionamiento del PSO para cualquier función no necesariamente separable (y que por tanto obedece más a la idea original del método), quizá podamos profundizar aún más en el análisis centrándonos más en el aspecto de la diversidad. Será una de las cosas que hagamos en el siguiente capítulo.

Capítulo 4

Análisis propio y construcción de nuevo método

En el capítulo anterior hicimos un repaso de la literatura sobre las distintas formas de análisis que existen acerca del PSO. Comenzamos estudiando los análisis de convergencia o estabilidad, que arrojan luz sobre el rendimiento del método según su configuración paramétrica y que por tanto, son de gran utilidad a la hora de seleccionar los hiperparámetros. Después también nos interesamos por el rol que tienen estos hiperparámetros en el funcionamiento del método, lo que nos ayuda aún más con su configuración y nos permite incluso realizar configuraciones dinámicas que se adaptan en cada momento a las necesidades de la búsqueda de optimización. Finalmente estudiamos las reglas que hacen funcionar al método y qué mecanismos determinan su comportamiento. Estos últimos análisis se centran sobre todo en el papel de la aleatoriedad y son los que dejan una idea más clara de cómo funciona realmente el PSO y los que más contribuyen a su comprensión. Sin embargo, ninguno de los análisis es del todo completo, pues se centran en aspectos concretos del método sin llegar a contemplar simultáneamente los efectos de los parámetros con los de la aleatoriedad, etc. Además, dentro de los análisis más generales que hemos visto (los de la sección 3.4), la mayoría se preocupa por destacar las claves de su buen rendimiento para una clase muy específica de problemas (los separables), obviando algo las claves de funcionamiento para todo tipo de problemas, que es para lo que realmente está concebido el PSO.

En este capítulo el objetivo es doble. Por un lado, el de llevar a cabo un análisis completo y detallado (o al menos más que los existentes) del funcionamiento interno del PSO estándar y por otro, dar con las claves, a través del análisis, que hacen que el PSO sea una buena metaheurística para cualquier tipo de problema (que es la idea de su creación en [6]). Un análisis completo nos permitirá definitivamente alcanzar un grado de conocimiento muy elevado del PSO, que nos servirá para realizar correcta y fácilmente el ajuste paramétrico, diseñar nuevas estrategias adaptativas o juzgar las actuales, planificar nuevas variantes PSO, etc. El análisis nos conducirá a cumplir con el segundo objetivo de detectar los puntos críticos del algoritmo responsables de su valía como optimizador de propósito general. También nos permitirá detectar los aspectos más negativos. Todo ello contribuirá a la construcción de un nuevo método PSO que explote las ventajas (cualidades) detectadas y minimice los defectos. El capítulo concluirá así con una nueva propuesta de PSO que aspira a superar en

rendimiento al estándar y cuyo funcionamiento sea bien conocido y el rol de sus parámetros sea claro, deshaciéndose con ello de gran parte de las dificultades que se encuentran en el PSO clásico, que son las que motivan este trabajo.

4.1. Bases del análisis

El análisis está enfocado en entender el funcionamiento de la fórmula de actualización de soluciones del PSO. Y es que, este es el tipo de análisis que consideramos más completo puesto que abarca de alguna manera al resto. Esto es porque entendiendo bien cómo funciona la regla que rige el método, se entiende cómo funciona el propio método en general y por ende, cómo funcionan sus parámetros y su convergencia, es decir, engloba todos los tipos de análisis contemplados durante el proyecto. En el análisis se tendrán en cuenta los hiperparámetros específicos y su encaje en la fórmula, la aleatoriedad, etc. Se procurará tener una visión lo más general posible del método, que abarque el mayor número de componentes y sus relaciones. Lógicamente muchas de las observaciones que se realicen ya han sido comentadas en el capítulo anterior. Otras, serán añadidas y presentadas por primera vez. En resumen, el análisis será una combinación ordenada de análisis estudiados junto con nuevas aportaciones, que permitirán formarse una idea clara de la estructura global del PSO y que darán origen a una nueva variante PSO basada en el análisis.

Para llevar a cabo el análisis, se adoptará el formato vectorial del PSO dado por las expresiones (3.13) y (3.14). La razón es que de esta forma se visualiza mejor el movimiento de las partículas por el espacio de búsqueda, que no entiende de dimensiones separadas e independientes sino que las considera (a las dimensiones del espacio) de manera conjunta. Y es que en realidad, las dimensiones no existen por separado (los cuerpos se mueven por espacios d-dimensionales) aunque utilicemos coordenadas (una por dimensión) para representar eventos físicos. Creemos pues que el contemplar la actualización de las componentes dimensionales (coordenadas) a la vez y de forma conjunta, favorece la interpretación y comprensión de las reglas de movimiento y de la fórmula de actualización del PSO. Esta es una convención puramente formal que no afecta en nada al método, tan solo se emplea para enfocar el análisis.

Por otro lado, la fórmula de actualización será analizada por partes, es decir, componente a componente se irá desgranando el método hasta que al final, cuando ya tengamos claro el rol y el funcionamiento por separado de cada componente, estudiaremos la regla compuesta. Hacerlo de esta manera nos permite entender la importancia de cada componente de la fórmula (el social, el cognitivo y el inercial) y de cada hiperparámetro específico. Además durante la ejecución del PSO puede y suele ser probable que a veces se anule algún componente y todo el peso recaiga en los restantes, por lo que comprenderlos de forma aislada es más que conveniente. Al ir reagrupando las componentes para acabar volviendo a la fórmula base (la fórmula completa de actualización), podremos analizar su encaje, las relaciones entre componentes y, lo más importante, el comportamiento conjunto que se alcanza, es decir, el funcionamiento completo del método. Comenzaremos analizando la componente social ya que posee el mayor nivel de importancia, pues contiene la información más relevante y clave para la metaheurística. Es la responsable de que se comparta información entre partículas y la que más valor tiene a la larga, pues es la que fuerza a todas las partículas a converger idealmente a la mejor solución posible.

El análisis irá muchas veces acompañado y apoyado por pruebas empíricas. Es necesario apuntar en este sentido que, con el fin de ceñirnos exclusivamente a la fórmula de actualización de soluciones del PSO, cualquier tipo de heurística normalmente empleada en el método ha sido descartada. Muchas veces se suelen emplear pequeñas estrategias inteligentes para inicializar las partículas, para limitar sus velocidades o sus posiciones, etc. Son este tipo de estrategias a las que nos referimos y usarlas podría dar lugar a conclusiones erróneas, ya que pueden confundir a la hora de atribuir comportamientos a ciertos elementos de la fórmula que en realidad se deban a dichas heurísticas. Para evitar estas confusiones en el análisis, el método será el básico expresado en el algoritmo 1, donde la inicialización de posiciones de las partículas se hará de forma aleatoria uniforme dentro de los límites del espacio de búsqueda ($x_{i,j} \sim \mathrm{U}(l_j, u_j)$) y las velocidades se inicializarán a 0. La gestión de salidas de los límites consistirá en ignorarlas, eso sí, ignorando también la evaluación de las soluciones inválidas. De esta forma lo que se pretende es alterar lo mínimo posible el curso natural de las partículas dictaminado por sus reglas de movimiento. Para las pruebas, salvo que se indique lo contrario, se trabajará con 50 partículas ($N = 50$) y el número máximo de iteraciones será de 500 ($maxIter = 500$). Además, los resultados de cada prueba serán promediados de un total de 20 ejecuciones. Y dicho esto, ahora sí, estamos listos para comenzar el análisis.

4.2. Componente social

Comenzaremos analizando la componente social de la fórmula: $c_2 * Rand_2 \circ (gBest - X)$. La componente social es la responsable de atraer a las partículas hacia la mejor posición global encontrada hasta el momento ($gBest$) y hacer que éstas centren la búsqueda en su entorno (en la vecindad del $gBest$). Es el único nexo en común de las partículas, que acaba consiguiendo, de ir bien la optimización, que todas converjan a un único punto. Es por tanto probablemente la guía más importante de movimiento que tengan las partículas. Por este motivo es preciso detenernos más en ella. Además, en muchas ocasiones durante la ejecución pasa que las partículas mejoran su mejor posición individual $pBest$ y en la siguiente iteración se encuentran con que $pBest = X$ y su componente cognitiva $c_1 * Rand_1 \circ (pBest - X)$ queda anulada, por lo que la única atracción de la partícula es hacia el $gBest$. La frecuencia con la que esto sucede depende entre otras cosas de la calidad de la optimización, pudiendo y generalmente llegando a ser muy habitual. Con esto queremos decir que la componente social aparece muy habitualmente en la fórmula sin la presencia de la cognitiva, lo cual refuerza que sea estudiada por separado.

Una regla de movimiento compuesta solamente por la componente social hace que las partículas se muevan en torno al $gBest$ y exploren su vecindad (búsqueda local). Como además todas ellas exploran la misma región, decimos que la componente favorece la búsqueda local. Lógicamente esto va asociado a una mayor explotación (que si estuvieran el resto de componentes presentes). Es por eso que para las pruebas empíricas que llevemos a cabo en esta sección utilizaremos una función objetivo sencilla e ideal para algoritmos explotativos como es la función *Sphere*. La función Sphere $f(X) = \sum_{j=1}^{d} x_j^2$ es convexa y unimodal, solo tiene un óptimo (el global) en $X^* = (0, ..., 0)$ cuya imagen es $f(X^*) = 0$. La representación gráfica de su versión bidimensional se muestra en la figura 4.1. Como se puede apreciar en la figura, es una función ideal para evaluar cualquier metaheurística explotativa. Nosotros trabajare-

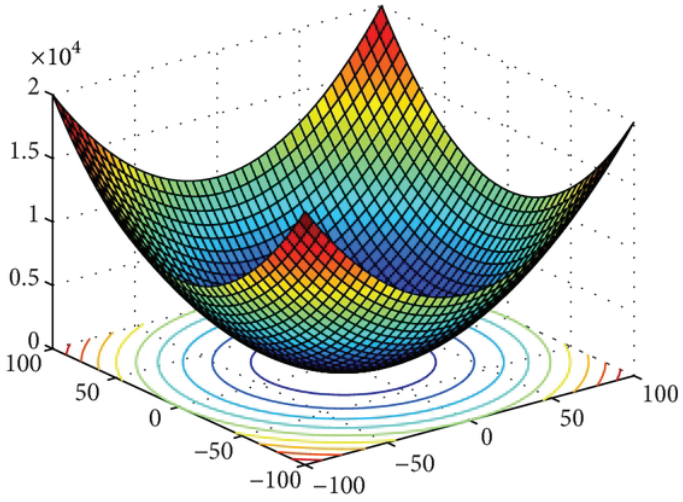


Figura 4.1: función Sphere de dos variables.

mos con su versión 30-dimensional, que suele ser un número de dimensiones muy común en la práctica y en las pruebas experimentales de evaluación de metaheurísticos. Optimizaremos las variables en el rango $[-500, 500]$, muy típico de la función Sphere.

Antes de considerar la componente social íntegra, conviene desglosarla. Primero examinaremos la componente sin los elementos aleatorios (determinista) para posteriormente irlos añadiendo progresivamente hasta llegar a la expresión completa original. Esperamos de esta manera apreciar mejor los cambios y aprender más sobre la propia componente.

4.2.1. $c_2(gBest - X)$

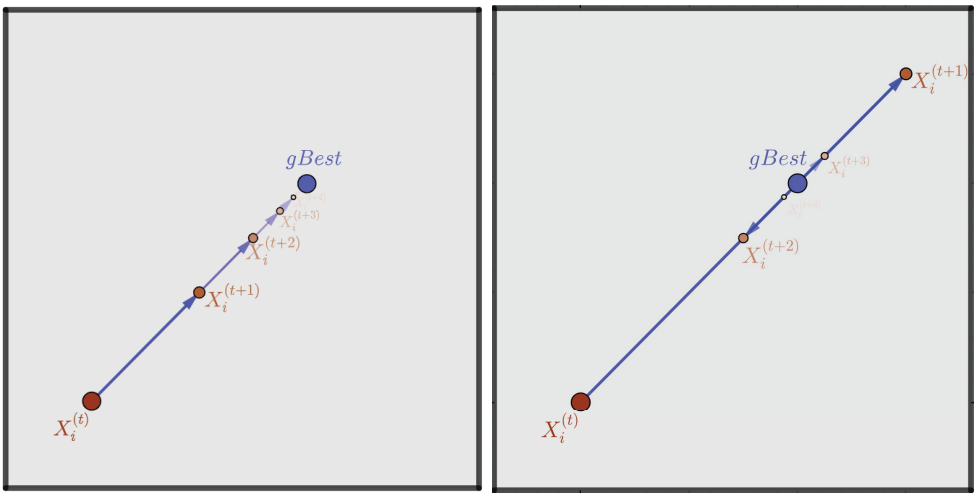
Sin aleatoriedad, la componente social queda reducida a la expresión $c_2(gBest - X)$. Es claro que en este caso las partículas se mueven todo el rato sobre un mismo eje cada una, que es la línea que une la posición de la partícula con el $gBest$. Al menos lo hacen hasta que se encuentra un nuevo $gBest$ y entonces cambian de eje. Esto es porque el vector social $(gBest - X)$ aparece multiplicado por un escalar c_2 que no altera su dirección, sino solo su módulo. Como todo ello conforma el vector velocidad V, que no es otra cosa que el desplazamiento al que va a ser sometida la partícula, ésta se desplaza siempre en la misma dirección (en dirección al $gBest$). El parámetro c_2 sirve para regular la magnitud del paso o desplazamiento de la partícula hacia el $gBest$ y lo hace de manera proporcional a la distancia entre X (posición de la partícula) y el $gBest$:

$$\|V\| = \|c_2(gBest - X)\| = c_2\|gBest - X\|$$

También implícitamente decide la proporción entre la nueva y la anterior distancia de la partícula al $gBest$ (asumiendo que éste no cambie):

$$\frac{\|(gBest - X) - c_2(gBest - X)\|}{\|gBest - X\|} = \frac{\|(1 - c_2)(gBest - X)\|}{\|gBest - X\|} = |1 - c_2|\frac{\|gBest - X\|}{\|gBest - X\|} = |1 - c_2|$$

La figura 4.2 ilustra el movimiento de una partícula en esta situación.



(a) $c_2 = 0.5$ (b) $c_2 = 1.5$

Figura 4.2: Representación gráfica bidimensional del PSO con regla de movimiento $V = c_2(gBest - X)$ tanto para el caso $c_2 < 1$ (a) como para el caso $c_2 > 1$ (b).

Con lo cual, gracias a c_2, las partículas por un lado evitan caer directamente en el objetivo y quedarse atrapadas ahí y por otro, deciden el ritmo con el que se acercan al mismo. Lógicamente cuanto mayor sea ese ritmo, más rápido y pronto convergerán al $gBest$ (mayor será la tasa de convergencia) y por tanto estarán explotando más. Por el contrario, si las partículas se acercan muy lentamente al objetivo, estarán visitando más posiciones del espacio y explorando más (búsqueda más exhaustiva). Este 'ritmo' o tasa de convergencia está vinculado a la distancia proporcional $|1 - c_2|$.[1] En concreto es su inversa. Cuando la distancia proporcional es grande, significa que nos acercamos lentamente al objetivo (tasa baja) y viceversa. Esto quiere decir que cuanto mayor sea la distancia proporcional, mayor exploración tendremos y cuanto menor, mayor explotación.[2] En función del valor de c_2, podemos distinguir entre cinco situaciones:

- $(\boldsymbol{c_2 = 1})$ **:** La distancia proporcional es 0. Las partículas convergen directamente y quedan atrapadas en el $gBest$ sin poder explorar nada. Por mucho que cambie el valor de c_2, será imposible hacer desconverger a las partículas.
- $(\boldsymbol{c_2 < 1})$ **:** La distancia proporcional es $(1 - c_2) < 1$. Las partículas convergen poco a poco según su tasa, aproximándose al objetivo siempre por defecto (no sobrepasándolo), tal y como ilustra la figura 4.2a. Cuanto menor sea c_2, menor será la tasa de convergencia y mayor la exploración en torno al $gBest$ y viceversa.
- $(\boldsymbol{c_2 \in (1, 2)})$ **:** La distancia proporcional es $(c_2 - 1) < 1$. Las partículas conver-

[1] En adelante llamaremos *distancia proporcional* al ratio entre las distancias nueva y previa de una partícula a su objetivo original cuando su movimiento es debido a una sola componente. Es decir, cuando la regla de movimiento solo contiene la componente social y el objetivo es $gBest$ (caso de la actual sección), la distancia proporcional se referirá a la magnitud $\frac{\|gBest^{(t)} - X^{(t+1)}\|}{\|gBest^{(t)} - X^{(t)}\|}$.

[2] Nótese que de acuerdo a lo que apuntamos en el capítulo preliminar, pese a que conceptualmente podemos considerar que estamos realizando una búsqueda local en torno al $gBest$ y por tanto explotando, contemplamos distintos niveles de exploración/explotación dependiendo del tamaño del vecindario del $gBest$ en el que buscamos (amplitud del espacio abarcado) ya que esto determina la capacidad de encontrar nuevas áreas prometedoras. Una búsqueda muy exhaustiva y con tasa de convergencia lenta en torno al $gBest$, se concentra en un espacio mucho más amplio que una con una tasa mayor.

gen poco a poco según su tasa, aproximándose al objetivo siempre por exceso (sobrepasándolo), tal y como ilustra la figura 4.2b. Cuanto mayor sea c_2, menor será la tasa de convergencia y mayor la exploración en torno al $gBest$ y viceversa.

- $(\boldsymbol{c_2 = 2})$ **:** La distancia proporcional es 1. Las partículas no convergen sino que oscilan alrededor del $gBest$, moviéndose a un lado y al otro del objetivo por su eje de movimiento. Al final siempre están visitando los mismos dos puntos y por tanto su nivel de exploración es prácticamente nulo.
- $(\boldsymbol{c_2 > 2})$ **:** La distancia proporcional es $(c_2 - 1) > 1$. Las partículas desconvergen del objetivo alejándose cada vez más de él con movimientos oscilatorios. Cuanto mayor sea c_2, mayor será la tasa con la que desconverjan. De mantenerse este estado de explosión del sistema, se puede acabar pronto con las partículas fuera de los límites del espacio de búsqueda (las distancias aumentan exponencialmente), lo cual es una situación totalmente indeseable.

Es evidente que el algoritmo solo funciona cuando $c_2 \in \{(0,1) \cup (1,2)\}$. En los demás casos o las partículas convergen instantáneamente sin posibilidad de escapar del objetivo o divergen. La distancia proporcional al $gBest$ indica el nivel de exploración que hay en torno a él y tanto valores de c_2 cercanos a 0 como cercanos a 2 presentan una gran exploración, mientras que los cercanos a 1, una gran explotación. Sin embargo, cuando $c_2 > 1$ las partículas exploran por exceso oscilando alrededor del objetivo, lo cual les hace explorar en una región más extensa (abarcan más espacio de búsqueda) que cuando exploran por defecto ($c_2 < 1$). Este hecho se puede apreciar comparando las figuras 4.2a y 4.2b. Es razonable pensar que la explotación del $gBest$ es mejor en el primer caso ($c_2 > 1$), pues siendo igual de exhaustivo en la búsqueda (misma tasa de convergencia), se explora una región mayor (mayor diversidad).

Para probar lo afirmado, ejecutamos el método para distintos valores del parámetro c_2. Los resultados se presentan en la siguiente tabla de figuras:

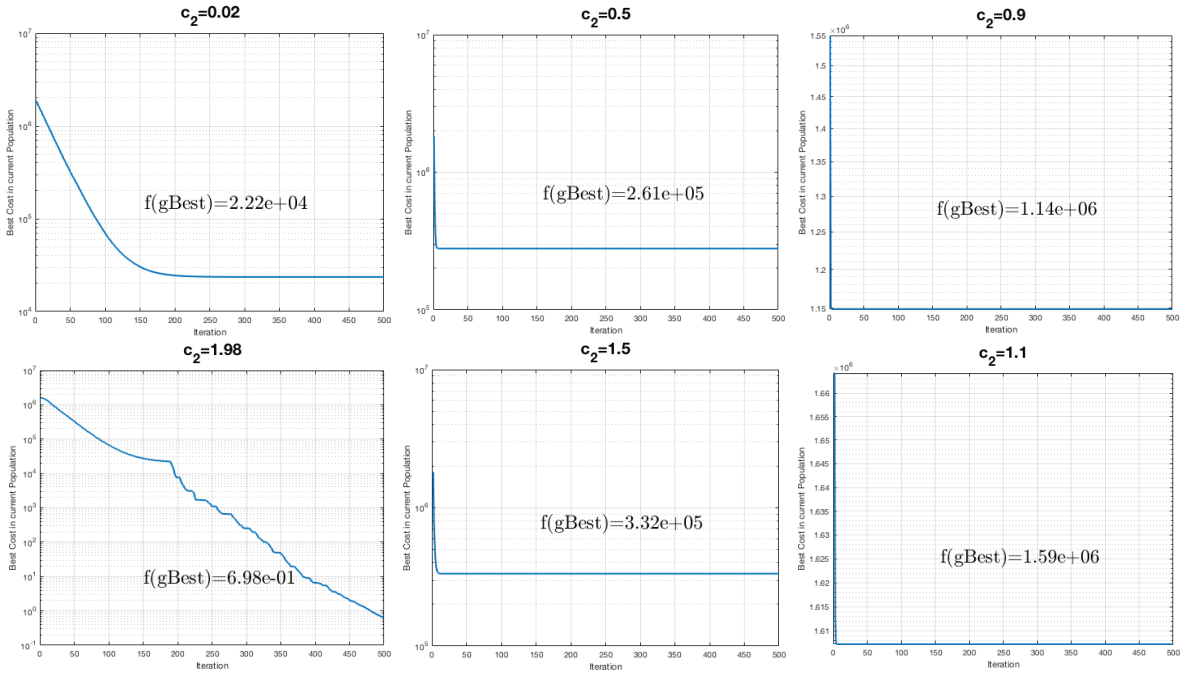


Figura 4.3: Resultados de PSO con $V = c_2(gBest - X)$ para distintos valores de c_2.

Las gráficas de la figura representan el valor del mejor fitness (eje y) encontrado hasta la iteración t (eje x) $\forall t \in \{1, ..., maxIter\}$. También aparece remarcado el mejor fitness encontrado durante la ejecución completa, es decir, $f(gBest^{(maxIter)})$. Interpretar este tipo de gráficas es muy sencillo. Hay que fijarse en la pendiente de la gráfica para saber el nivel de progreso que atraviesa el algoritmo. Cuando la gráfica tiene pendiente 0 significa que no hay mejora con el paso de las iteraciones y el algoritmo se ha estancado. Si esta situación se mantiene durante un largo número de iteraciones hasta el final de la ejecución, es un indicativo de que seguramente el algoritmo haya convergido. Hay que tener en cuenta que la mayoría de estas gráficas están en escala logarítmica para el eje vertical y y en escala lineal para el horizontal x, para así reflejar no tanto la magnitud de las mejoras que se dan $f(gBest^{(t)}) - f(gBest^{(t+\Delta t)})$ sino las proporciones de esas mejoras (mejoras relativas) $\frac{f(gBest^{(t)}) - f(gBest^{(t+\Delta t)})}{f(gBest^{(t)})}$, que es lo que marca el buen o mal rendimiento de un optimizador. Lógicamente si las mejoras relativas de $f(gBest^{(t)})$ se mantienen constantes (gráficas logarítmicas con pendiente lineal), es síntoma de que el optimizador está funcionando bien. Si no, puede que no sea posible por la propia función o puede que sea culpa del optimizador, aunque de todos modos no necesariamente implica un mejor rendimiento. En este caso como la función (la Sphere) tiene fitness óptimo 0, sí que se puede mantener infinitamente la misma tasa de mejora relativa.

Juzgando las gráficas, observamos que los resultados son los esperados de acuerdo a nuestro análisis. Primero vemos que cuanto mayor es la distancia proporcional de las partículas al $gBest$ ($|1 - c_2|$), más se explora y mejores resultados se obtienen. Los peores resultados se dan cuando esta distancia es mínima. En esos casos la convergencia se produce de forma muy prematura prácticamente nada más comenzar la ejecución. Por eso, sus resultados finales dependen mucho de la inicialización de las partículas. Por ejemplo, el valor esperado de la función Sphere tras la primera iteración es menor cuando $c_2 < 1$ y las partículas se aproximan por defecto al $gBest$ que cuando $c_2 > 1$ y se aproximan por exceso, simplemente porque, tal y como son inicializadas, se espera que caigan más cerca del punto $(0, ..., 0)$. Entonces no tiene demasiado sentido analizar los resultados de estos métodos tan explotativos cuando en verdad no son fruto de la búsqueda que realiza el algoritmo. Lo que sí tiene sentido es comparar los resultados de los dos métodos más exploratívos que hemos probado ($c_2 = 0.02$ y $c_2 = 1.98$). Aquí vemos que mientras que el algoritmo de aproximación por defecto no es capaz de acabar evitando la convergencia prematura, el de aproximación por exceso sí la evita y mantiene un progreso bastante lineal durante el proceso. Sin duda es debido a su mayor capacidad para moverse por regiones más extensas, que le hacen descubrir más fácilmente nuevas regiones prometedoras. Y así aunque la tasa de explotación sea igual en las dos configuraciones, la calidad de la misma es muy superior en la segunda y queda patente en los resultados de 4.3.

4.2.2. $c_2 rand_2(gBest - X)$

Antes de continuar con la introducción total de la aleatoriedad del PSO estándar, veremos qué sucede si únicamente la introducimos parcialmente tal y como se hace en el PSO lineal. Y es que además suele ser común encontrarse en la literatura confusiones del PSO lineal con el estándar. La regla de actualización que tenemos ahora es $V = c_2 rand_2(gBest - X)$.

Lo que cambia con la introducción del número aleatorio $rand_2$ es que el escalamiento

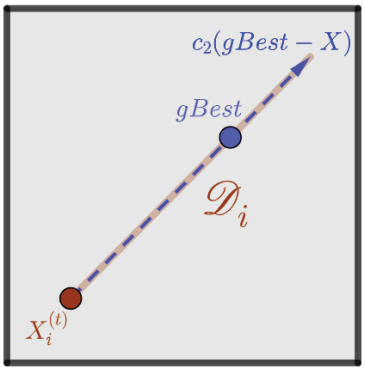


Figura 4.4: Representación gráfica bidimensional del dominio instantáneo de búsqueda en el PSO con regla $V = c_2 rand_2(gBest - X)$ cuando $c_2 = 1.5$.

del módulo del vector social $(gBest - X)$ ya no depende exclusivamente de c_2 sino ahora también de $rand_2$. Tanto c_2 como $rand_2$ son los encargados ahora de regular la magnitud relativa del paso o desplazamiento de la partícula al $gBest$ y con ello, su distancia proporcional $|1 - c_2 rand_2|$. Por lo demás, las partículas siguen 'ancladas' en trayectorias con direcciones fijas que las obligan a buscar en líneas o ejes que atraviesan el $gBest$. Básicamente el análisis anterior sigue siendo válido ya que $rand_2$ es un escalar como lo es c_2 y lo es también su producto $c_2 rand_2$. Con lo cual, prácticamente todo lo que decíamos antes para c_2 sigue siendo válido para $c_2 rand_2$, que es ahora el escalar que multiplica al vector social $(gBest - X)$. Cierto es que ahora ya no tenemos un escalamiento constante y controlado, sino que está gobernado por la aleatoriedad. Como $c_2 rand_2 \sim \mathrm{U}(0, c_2)$, cualquier valor en el intervalo $[0, c_2]$ puede hacer las veces de lo que antes era c_2. El dominio instantáneo de búsqueda pasa de ser un punto a ser un intervalo (ver figura 4.4). Esto por supuesto introduce incertidumbre en el proceso y con ello descontrol para el usuario. Ya no se puede saber cuál va a ser la siguiente posición de la partícula ni su distancia proporcional al $gBest$, pues pasa a formar parte del azar. Eso sí, cada vez que tomemos la muestra aleatoria y tengamos una instancia de $c_2 rand_2$, los mismos cinco estados de 4.2.1 aplican para la instancia. Así podemos determinar si un paso/movimiento concreto de una partícula es explorativo o es explotativo. El problema es que ahora las distancias proporcionales ya no se mantienen constantes con el tiempo y también hay alternancia entre estados. Estos cambios continuos en los niveles de exploración/explotación es necesario analizar cómo afectan al rendimiento del algoritmo.

Que el movimiento de la partícula esté gobernado por la aleatoriedad implica que a veces los pasos que dan sean perjudiciales y condicionen completamente el devenir del algoritmo, pues las decisiones del pasado tienen un gran impacto en el futuro. Tomar una sola decisión errónea en un momento dado, puede conllevar la pérdida de oportunidades de acercarse a regiones mejores y obtener mejores resultados. Esto se debe a que cuando explotamos, vamos concentrándonos cada vez en áreas más reducidas y descartando regiones de alrededor y que si sobreexplotamos, estaremos descartando regiones que son prometedoras y que podrían guiar mejor la explotación. Todos los movimientos cuentan y no pueden ser vistos simplemente desde una perspectiva promedio. Por eso, una tasa de convergencia constante garantiza un rendimiento consistente, independiente del tamaño (escala) del área de búsqueda y del número de iteración, mientras que una aleatoria, no. La aleatoria, por muy explo-

rativa que sea de media, puede ver arruinados sus resultados solo con que alguna instanciación salga perjudicial. Idealmente, lo mejor sería una tasa variable pero no aleatoria que se ajustase a las necesidades de cada instante. Es por lo que existen las técnicas adaptativas.

Para que el algoritmo aleatorio no sufra el problema comentado, es necesario asegurarse de que para todas las situaciones, cualquiera de los valores probables que puede tomar $c_2 rand_2$, vaya a resultar en un movimiento no perjudicial (no demasiado explotativo como para inducir una convergencia prematura). Ello significa que cualquiera de los valores probables podría establecerse constante durante toda la ejecución (algoritmo determinista) y causar un buen rendimiento. Pero dado el caso, no todos los valores arrojarían el mismo resultado, sino que algunos serían más explorativos que otros y viceversa. En concreto siempre habría algún valor c_2^* que destacase positivamente sobre el resto al tener asociado un nivel de exploración óptimo (el mejor equilibrio para el problema a tratar). Esto quiere decir que en reglas generales, un movimiento realizado con c_2^* es mejor que uno realizado con cualquiera de los demás valores del rango de la variable aleatoria $c_2 rand_2$. Así, alternar aleatoriamente entre todos los valores del rango, implica que en general (de media), los movimientos sean peores y el rendimiento del algoritmo sea también peor que utilizando el valor constante y determinista c_2^* (asumimos regularidad en la topografía de la función). Esto demuestra que los cambios continuos y aleatorios en los niveles de exploración/explotación no benefician al método y que es más sensato optar por unos controlados y deterministas. A su vez, examinando los valores del rango de $c_2 rand_2$, es fácil darse cuenta de que cuando $c_2 > 1$ o es muy cercano a 1, están incluidos valores que pueden poner en riesgo el rendimiento por ser demasiado explotativos e inducir a convergencia prematura, que es el problema del que hablamos.

En resumen, ese descontrol que aporta $rand_2$ al escalamiento modular $c_2 rand_2$ de $(gBest - X)$, no es teóricamente muy positivo para el método. Aún así, dentro del descontrol, lo que se sigue pudiendo controlar a través del parámetro c_2 es el extremo del intervalo $[0, c_2]$ que define el dominio instantáneo de búsqueda. Cuanto mayor sea c_2, mayor será el abanico de posiciones candidatas de la partícula y por tanto también menor el control del proceso por parte del usuario. Pero sobre todo, a través de c_2 se puede controlar el valor esperado de la distancia proporcional $|1 - c_2 rand_2|$ y por consiguiente, la tendencia media de exploración/explotación del algoritmo:

$$
\begin{aligned}
\mathbb{E}\left(|1 - c_2 rand_2|\right) &= \int_0^1 |1 - c_2 rand_2| \; d(rand_2) \\
&= \int_0^{\text{mín}(1, \frac{1}{c_2})} 1 - c_2 rand_2 \; d(rand_2) \; + \; \int_{\text{mín}(1, \frac{1}{c_2})}^1 -(1 - c_2 rand_2) \; d(rand_2) \\
&= \left(rand_2 - c_2 \frac{rand_2^2}{2} \right) \Big|_0^{\text{mín}\left(1, \frac{1}{c_2}\right)} \; - \; \left(rand_2 - c_2 \frac{rand_2^2}{2} \right) \Big|_{\text{mín}\left(1, \frac{1}{c_2}\right)}^1 \\
&= 2 \left(\text{mín}\left(1, \frac{1}{c_2}\right) - c_2 \frac{\text{mín}\left(1, \frac{1}{c_2}\right)^2}{2} \right) - \left(1 - \frac{c_2}{2}\right) \\
&= \begin{cases} 1 - \dfrac{c_2}{2}, & si \quad c_2 < 1 \\ \dfrac{(c_2 - 1)^2 + 1}{2c_2}, & en \; caso \; contrario \end{cases}
\end{aligned}
$$

Esta es la única medida que disponemos para saber el nivel de exploración de nuestro algoritmo. Lógicamente cuanto mayor sea, más explorativo será nuestro algoritmo. Fijándonos en la fórmula, esto sucede, en el caso de que $c_2 < \sqrt{2}$, cuanto menor sea c_2 y, en el caso de $c_2 > \sqrt{2}$, cuanto mayor sea. Además ahora c_2 puede ser mayor que 2 y la distancia proporcional esperada menor que 1, es decir, las partículas pueden acometer movimientos divergentes, pero convergiendo en promedio. Es más, por lo que hemos comentado antes, incluso con una distancia proporcional esperada mayor que 1, las partículas podrían terminar convergiendo.

Para ordenar ideas, lo mejor será probar empíricamente el rendimiento del método para distintos valores de c_2. Y como gran parte del análisis se basa en la comparación del método con el anterior sin aleatoriedad, hemos primado elegir valores de c_2 para los cuales la distancia proporcional esperada coincida con alguna de las de la figura 4.3. Los resultados son los siguientes:

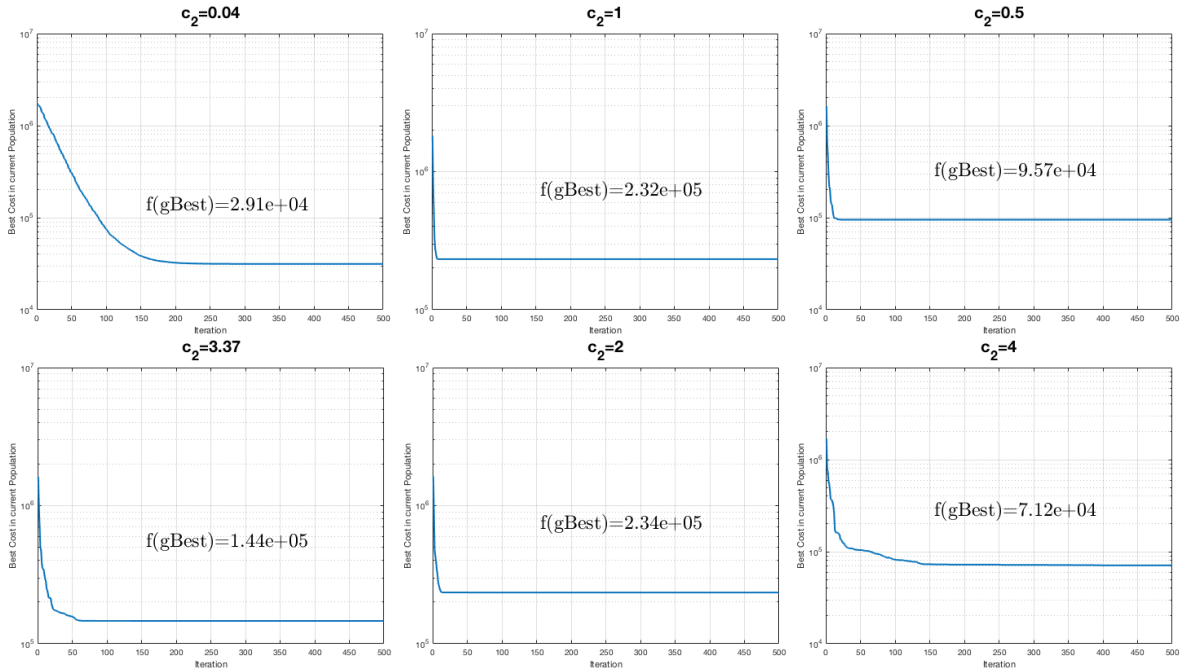


Figura 4.5: Resultados de PSO con $V = c_2 rand_2(gBest - X)$ para distintos valores de c_2.

Se puede ver que los resultados en general son peores que los de 4.3. Ninguna de las configuraciones logra evitar la convergencia prematura sin ser divergente. De hecho, comparando los dos resultados que quedamos que eran más representativos de 4.3 (los menos dependientes de la inicialización) con sus homólogos en 4.5 (los de igual distancia proporcional esperada), vemos que estos últimos son peores. Todo esto refuerza lo que decíamos de que no es lo mismo mantener un nivel de exploración/explotación constante que uno variable y aleatorio. El aleatorio no se puede juzgar solo por su nivel medio sino también por su rango. Pues como dijimos, cada paso del algoritmo es importante y decisivo para su devenir, por lo que es importante que el rango de niveles de exploración que admita el algoritmo aleatorio sea adecuado para que no se permita que se den pasos demasiado explotativos que arruinen la búsqueda. Y claro, puestos a tener que elegir un rango de niveles que siempre sean beneficio-

sos para nuestro problema, mejor elegir directamente el nivel que consideremos más beneficioso. Es por eso que lo normal es que se obtenga mejor rendimiento con el algoritmo determinista que con el aleatorio.[3] De hecho, si nos fijamos, el resultado de la configuración $c_2 = 0.04$ es mejor que el de la configuración $c_2 = 3.37$, teniendo las dos la misma distancia proporcional esperada de 0.98. Claramente lo que marca aquí la diferencia es que el rango de distancias proporcionales de la primera configuración ($c_2 = 0.04$), que es $[0.96, 1]$, no incluye valores muy explotativos que puedan deteriorar tanto su rendimiento con respecto a la versión determinista de 4.3. Mientras, en la segunda configuración ($c_2 = 3.37$) el rango es de $[0, 2.37]$, que sí incluye valores muy explotativos que dan al traste con el rendimiento. Todo es fruto del descontrol que aporta la aleatoriedad.

Por otra parte sí que se observa que cuanto mayor es el valor esperado de la distancia proporcional, mayor es la exploración del algoritmo y mejores resultados se obtienen. Claro que, llega un punto en el que las partículas en vez de converger, divergen y sus resultados empeoran ($c_2 = 4$). Igualmente, la posibilidad que otorga la aleatoriedad de dar pasos divergentes de vez en cuando ($c_2 > 2$) mientras se converge de media, no resulta ser ventajosa. Lo que sí consigue la aleatoriedad es reducir la sensibilidad del algoritmo en su hiperparámetro c_2. Dicho de otra manera, los resultados que se obtienen para las distintas configuraciones no varían tanto entre sí (son más parecidos entre ellos que en 4.3), lo que quiere decir que el algoritmo es menos dependiente del parámetro c_2 o que no se ve tan afectado por cambios en él.

En conclusión, la aleatoriedad en sí y la incertidumbre que proporciona a la hora de ampliar el abanico de opciones que tiene una partícula para moverse (el dominio instantáneo de búsqueda), no es lo que otorga exploración al PSO. La exploración la marca el tipo de movimientos que finalmente acaba realizando, es decir, las instancias de la variable aleatoria velocidad V. Y en este apartado hemos probado que precisamente la introducción de la aleatoriedad modular no favorece la exploración ni la calidad de la explotación en torno al $gBest$. La aleatoriedad modular no resulta beneficiosa en el rendimiento del método, más bien al contrario, resulta perjudicial. Es más, resta capacidad de control del proceso al usuario y minimiza su influencia a través de la selección del hiperparámetro c_2, que se vuelve bastante menos decisivo.

4.2.3. $c_2 Rand_2 \circ (gBest - X)$

Ya por fin podemos analizar la fórmula completa de la componente social con la introducción de la aleatoriedad clásica. La expresión de la nueva regla queda $V = c_2 Rand_2 \circ (gBest - X)$.

La diferencia de esta expresión con la anterior se encuentra en el elemento aleatorio. En la fórmula anterior el elemento aleatorio $rand_2$ era un número escalar y en esta, un vector ($Rand_2$). Así, mientras que antes todas las componentes del vector social $(gBest - X)$ eran multiplicadas por un mismo valor $rand_2$, ahora cada componente se escala de manera independiente (es multiplicada por un valor distinto). Esta diferencia entre el mapeo aleatorio lineal y el clásico es recogida por la siguiente expresión:

[3] Aquí matizamos de nuevo que asumimos cierta regularidad en la topografía de la función, es decir, el terreno de búsqueda no cambia drásticamente con la escala. Si lo hiciera, la aleatoriedad podría resultar beneficiosa al permitir cambios en la exploración pero más bien por su naturaleza dinámica. En ese caso, sería más conveniente recurrir a técnicas adaptativas que sí seleccionan a conciencia el nivel de exploración adecuado a cada fase de búsqueda.

$$\begin{pmatrix} gbest_1 - x_1 \\ \vdots \\ gbest_j - x_j \\ \vdots \\ gbest_d - x_d \end{pmatrix} \longmapsto \begin{pmatrix} rand_2(gbest_1 - x_1) \\ \vdots \\ rand_2(gbest_j - x_j) \\ \vdots \\ rand_2(gbest_d - x_d) \end{pmatrix}$$

(a) Mapeo lineal

$$\begin{pmatrix} gbest_1 - x_1 \\ \vdots \\ gbest_j - x_j \\ \vdots \\ gbest_d - x_d \end{pmatrix} \longmapsto \begin{pmatrix} \{rand_2\}_1(gbest_1 - x_1) \\ \vdots \\ \{rand_2\}_j(gbest_j - x_j) \\ \vdots \\ \{rand_2\}_d(gbest_d - x_d) \end{pmatrix}$$

(b) Mapeo clásico

Como podemos observar, el mapeo lineal afecta solamente al módulo del vector y no a su dirección. El clásico en cambio sí modifica también la dirección del vector social, pues al multiplicarse cada componente vectorial por un número diferente, las proporciones que existen entre ellas (las componentes) se ven alteradas. Esta alteración en las proporciones del vector equivale a un cambio en su dirección. Podemos decir que el mapeo aleatorio clásico hace 'girar' al vector social $(gBest - X)$ además de manipular también su módulo como hace el mapeo lineal. En consecuencia, el movimiento de las partículas deja de estar 'atado' a un eje como lo estaba hasta ahora con el mapeo lineal. Las partículas pueden escapar de él cambiando de dirección (girando) y de esta manera dejar de buscar todo el rato en una misma línea espacial. Con esto consiguen aumentar considerablemente la diversidad en la búsqueda debido a que son capaces de abarcar más regiones diferentes ocupando mejor el espacio. En resumidas cuentas, este hecho contribuye a mejorar la búsqueda en torno al $gBest$. El nuevo dominio instantáneo de búsqueda es bastante mayor. En lugar de ser un segmento, ahora es un hiperplano del espacio. Su representación gráfica bidimensional es la de la figura 4.6.

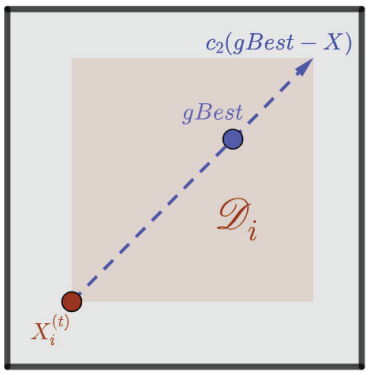


Figura 4.6: Representación gráfica bidimensional del dominio instantáneo de búsqueda en el PSO con regla $V = c_2 Rand_2 \circ (gBest - X)$ cuando $c_2 = 1.5$.

Más adelante profundizaremos en las características de los giros[4] y entraremos a analizar más detalladamente el método. Pero antes, vamos a comprobar que efectivamente estos cambios direccionales en el vector social son muy beneficiosos para el PSO. Para ello, ejecutamos el método para las mismas configuraciones paramétricas

[4]Utilizamos la palabra 'giro' para referirnos en verdad al cambio direccional entre el vector original y el resultante. No andamos interesados en el movimiento de giro sino en la diferencia final en las direcciones de ambos vectores, en el ángulo que hay entre ellos. Así, cuando hablamos de 'giro' hacemos alusión al giro virtual sobre el plano que comparten los dos vectores (el plano de rotación contiene al vector original).

que en 4.5 con el fin de comparar resultados entre los dos tipos de mapeo aleatorio (lineal vs clásico) y evaluar el efecto que tiene la introducción de la aleatoriedad direccional en el clásico. Los resultados son los que siguen:

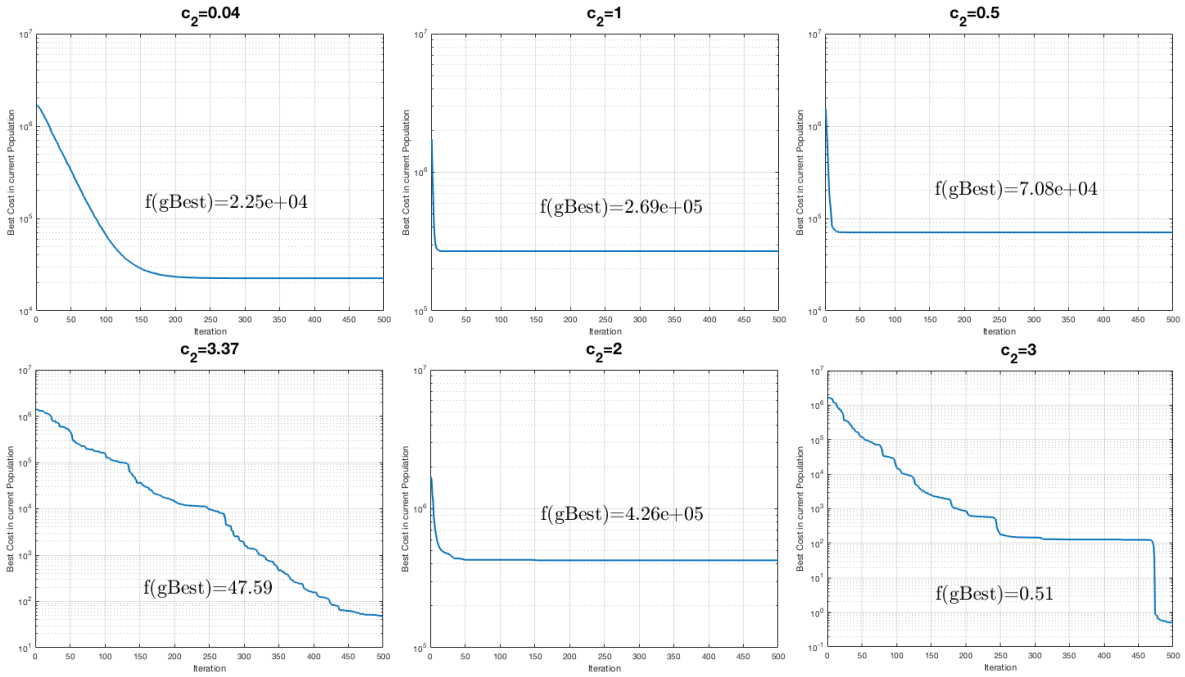


Figura 4.7: Resultados de PSO con $V = c_2 Rand_2 \circ (gBest - X)$ para distintos valores de c_2.

Los resultados confirman la mejoría general del método con la inclusión de la aleatoriedad direccional. Sobre todo la mejora queda patente en las configuraciones paramétricas más exploratives (las de menor tasa de convergencia). En las explotativas en cambio, apenas hay diferencia con los resultados de 4.5. Esto es comprensible debido a la reflexión que hicimos en su momento de que al ser configuraciones tan explotativas, las partículas convergen muy rápidamente sin darles tiempo a explorar y por tanto, realmente no llega a haber un proceso de búsqueda como para ser tenido en consideración. Es por eso que cualquier modificación del proceso de búsqueda apenas va a afectarles y sus resultados van a ser muy parecidos. Y si nos fijamos en las configuraciones más exploratives, que son las que realmente sirven para valorar el método, vemos como las mejoras más importantes se dan cuando los valores de c_2 son grandes. Esto ya nos indica que el impacto de la aleatoriedad direccional en el método no es ajeno al escalamiento modular del vector social y que a mayor magnitud relativa del paso de una partícula, mayor es el efecto de la perturbación direccional. Este efecto es muy grande y beneficioso. Gracias a él la explotación del $gBest$ es mucho mejor al lograr aumentar la diversidad de la búsqueda. Esta es la razón por la cual, mientras antes se acababa convergiendo prematuramente al no encontrar regiones prometedoras (4.5), ahora ya no. De hecho, el resultado para $c_2 = 3$ es el mejor de los obtenidos hasta este punto del análisis.

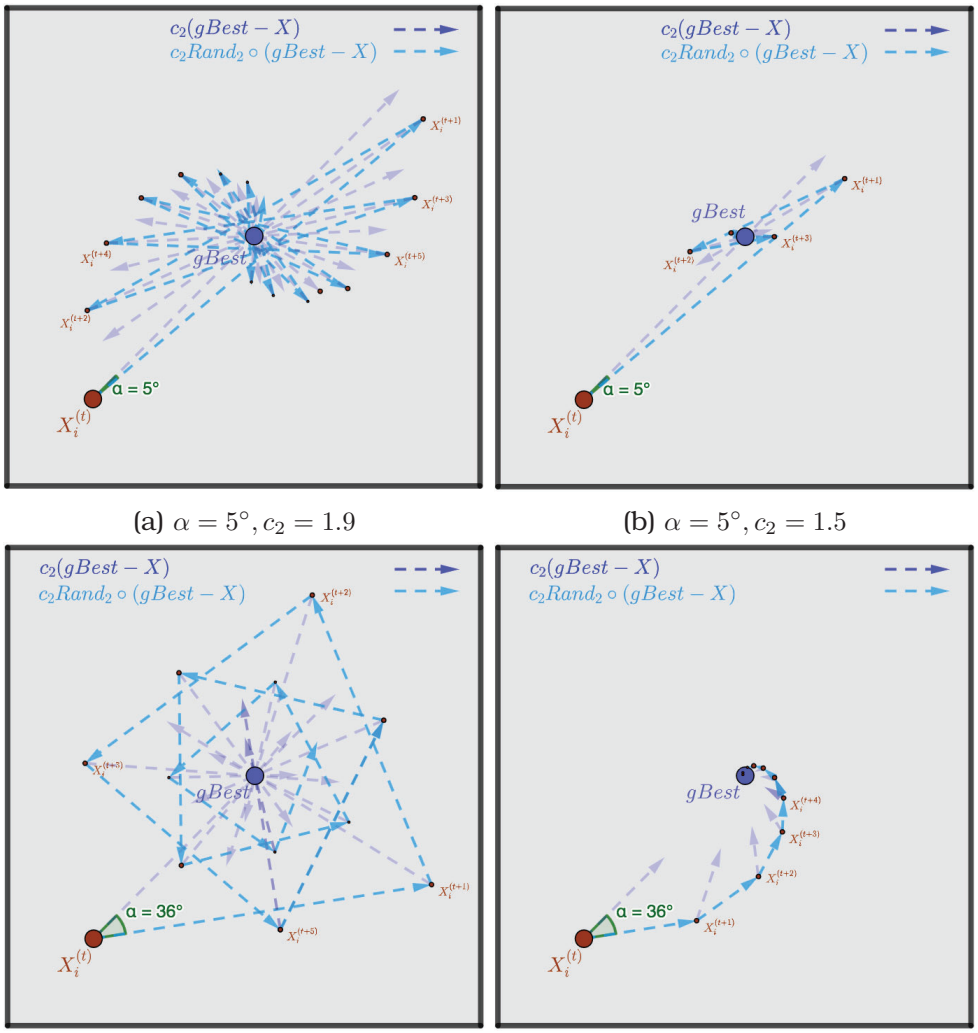
Que el impacto de los giros del vector social esté relacionado con su escalamiento modular, no es de extrañar. Pues al final, por mucho que exista un giro, si luego el avance (tamaño del paso) no es grande, no se puede llegar a apreciar demasiado el

giro. Digamos que el módulo influye en la importancia que le queremos dar a ese cambio de dirección. Cuanta más distancia se recorra en la nueva dirección, más notaremos el efecto del giro. En realidad, andamos interesados en la combinación giro+módulo. A esta combinación la llamaremos 'desvío' ya que representa el desvío que se produce del camino recto al $gBest$, o sea, del eje de búsqueda en el que antes las partículas estaban 'encerradas' y que ahora tratamos de evitar. Es lógico que verdaderamente lo que hace que las partículas dejen de buscar todo el rato en una misma línea espacial y que aumenten la diversidad logrando mucho mejores resultados, es el desvío, pues el movimiento al final viene dado por el giro+módulo y es el que dicta si escapamos o no de la línea espacial de búsqueda.

Cuanto mayor es el módulo, mayor el desvío, más se distancia la partícula del eje de búsqueda y mayor es su capacidad de abarcar regiones más amplias del espacio (mayor diversidad). Este hecho se ilustra en la figura 4.8. La figura representa cuatro situaciones distintas en las que la partícula mantiene un desvío constante a lo largo de la ejecución con el fin de ilustrar el efecto del desvío en la explotación del $gBest$. Cada situación viene representada por una subfigura diferente. Las subfiguras de la primera fila muestran el movimiento de una partícula que gira $5°$ con respecto al vector social $(gBest - X)$ y las de la segunda, el de una partícula que gira $36°$. Las subfiguras de la primera columna exhiben un tamaño relativo del paso (escalamiento modular) mayor que las de la segunda. Comparando las dos situaciones de cada fila, nos damos cuenta de que cuanto mayor es el escalamiento modular c_2, más acentuado es el giro de la partícula y la búsqueda se concentra en un radio mayor alrededor del $gBest$. Es decir, la partícula es capaz de cubrir un espacio mayor al no estar su trayectoria tan ceñida al eje de búsqueda. Esto se traduce en una mayor diversidad y un mejor rendimiento del algoritmo.

Prácticamente lo mismo podríamos decir del ángulo de giro α. Cuanto mayor es el ángulo, mayor el desvío y mayor la diversidad de la búsqueda. Podemos comprobar esto observando la comparación entre las figuras 4.8a y 4.8c y entre las figuras 4.8b y 4.8d. A pesar de que las figuras de la segunda fila tienen una magnitud relativa del paso menor (menor c_2) que sus respectivas de la primera fila, su desvío es mayor al disponer de un giro mucho mayor. Ahora bien, hasta ahora hemos hecho hincapié en que ante dos situaciones con mismo α pero distinto c_2, la que tuviera el mayor c_2 era la que mejor exploraba. Y ante dos situaciones con mismo c_2 pero distinto α, la que tiene el mayor α es la que mejor explora. Sin embargo, comparar dos situaciones con distinto α y distinto c_2 es algo más complejo. En ese caso no solamente entra en juego el cálculo del desvío que hay en cada situación, sino también la conclusión que extraímos en 4.2.1 de que los pasos grandes (los de aproximación por exceso: $c_2 > 1$) aportaban mayor diversidad que los cortos (de aproximación por defecto: $c_2 < 1$). Esto último se ve reflejado si comparamos las figuras 4.8a y 4.8d. El desvío es mayor en la situación segunda (figura 4.8d) que en la primera (4.8a) pero en cambio, en la primera se abarca una mayor región de búsqueda porque los pasos grandes que da la partícula le permiten ir visitando puntos más distantes de diferentes regiones que no pueden ser alcanzadas en la segunda situación. Si los pasos son pequeños, sea cual sea el desvío, las partículas estarán visitando posiciones muy parecidas del espacio y sus trayectorias estarán más concentradas y menos esparcidas por el espacio, ocupando un área menor.

Lo que está claro es que el desvío es el responsable de una explotación eficiente en el PSO. Los movimientos con desvío garantizan que, con el paso de las iteraciones,



(a) $\alpha = 5°, c_2 = 1.9$ (b) $\alpha = 5°, c_2 = 1.5$

(c) $\alpha = 36°, c_2 = 1.5$ (d) $\alpha = 36°, c_2 = 0.5$

Figura 4.8: Representación gráfica de la influencia del escalamiento modular en el desvío de una partícula de su camino al $gBest$. Ante un mismo ángulo de giro α, a mayor módulo del vector alterado $c_2 Rand_2 \circ (gBest - X)$, mayor es el desvío y mayor es la diversidad que se alcanza a lo largo de la ejecución. Con fines didácticos, hemos establecido constante y determinista la variable aleatoria $Rand_2$ para que así las alteraciones del vector social $(gBest - X)$ sean iguales en todas las iteraciones y se aprecie claramente el efecto comentado.

las partículas recorran de forma inteligente el espacio que rodea al $gBest$, pues son la llave para acceder a regiones más distintas y ganar en diversidad. Ya no es solo que gracias a los desvíos las partículas lleguen a regiones que antes eran inalcanzables (4.6 vs 4.4), sino que guían bien el proceso alternando entre gran variedad de localizaciones diferentes. Esto se debe a que un desvío cambia el eje de la búsqueda en torno al $gBest$. Y como el proceso de búsqueda ha de ser convergente, es importante que en todo momento haya desvíos para poder conservar la diversidad. Podemos afirmar que el desviarse resulta mucho más positivo que el no hacerlo porque facilita el descubrimiento de nuevas regiones prometedoras del espacio de búsqueda. El objetivo pues de las partículas es desviarse del $gBest$ ya que si no lo hacen, el sistema no progresará y se volverá estacionario. Es decir, si las partículas no se desvían del $gBest$, el algoritmo seguramente acabe convergiendo prematuramente y su rendimiento no sea bueno (figura 4.5), sobre todo si tratamos con problemas de elevada dimensionalidad.

Una vez destacado el gran valor del desvío en el método, lo siguiente que podemos hacer es desgranarlo por partes (giro+módulo). El elemento aleatorio de la fórmula $Rand_2$ se encarga tanto de la modificación direccional como de la modular. Lo que pretendemos ahora es separar ambas funcionalidades. Queremos por un lado tener la parte direccional y, por otro, la modular. Haciendo

$$V_D = \frac{Rand_2 \circ (gBest - X)}{\|Rand_2 \circ (gBest - X)\|} * \|gBest - X\|$$

$$V_M = c_2 * rand_2$$

$$V = V_M * V_D$$

logramos diferenciar las dos partes, con V_D siendo la encargada de los cambios direccionales y V_M, de los cambios en el módulo. Así ahora en vez de tener una única variable aleatoria $Rand_2$ que acapare ambas funciones, tenemos dos (una para cada tarea) como son $Rand_2$ y $rand_2$. Esto nos permite analizar por separado el rol del escalamiento modular y el direccional y en especial, el papel de la aleatoriedad en cada uno.

No obstante, la nueva formulación de V no es exactamente equivalente a la anterior. Y es que, en la nueva, al estar separado el escalamiento modular del direccional, significa que ambos son independientes, es decir, la distribución de probabilidad del módulo es independiente de la del giro, lo cual quiere decir que la probabilidad de que el vector social modificado V tenga una magnitud determinada es la misma para todos los giros. Antes esto no era así ya que los giros grandes iban asociados con un módulo esperado menor que los pequeños. Esto sucedía porque cuanto mayor es el giro, mayor es la alteración en las proporciones entre los componentes vectoriales, los cuales necesariamente disminuyen al ser multiplicados por valores en $[0, 1]$, lo que implica que de media $\|Rand_2 \circ (gBest - X)\|$ sea menor. Gráficamente esto se puede comprobar observando la figura 4.6. La figura muestra que el dominio instantáneo de búsqueda tiene forma rectangular (en espacios multidimensionales sería un hiperrectángulo) con diagonal el vector social. Esto indica que los giros del vector van asociados con una disminución en su magnitud, mayor cuanto más grande es el giro.

La nueva formulación del método arroja los siguientes resultados, esta vez presentados en formato tabla:

$c_2 = 0.04$	$c_2 = 0.5$	$c_2 = 1$	$c_2 = 2$	$c_2 = 3$	$c_2 = 3.37$
$2.33e+04$	$6.29e+04$	$2.12e+05$	31.28	0.03	0.81

Cuadro 4.1: Resultados de PSO con $V = c_2 rand_2 \|gBest - X\| \frac{Rand_2 \circ (gBest - X)}{\|Rand_2 \circ (gBest - X)\|}$.

Como era de prever, los resultados son mejores bajo la nueva formulación. Lo único que ha cambiado es lo que destacábamos: el aumento del módulo esperado para los giros. Y como ya sabíamos, esto es muy beneficioso para el algoritmo puesto que incrementa los desvíos. Ante los mismos giros, tenemos ahora módulos mayores en promedio, que es justamente lo que advertíamos que aumentaba la diversidad y mejoraba la explotación del $gBest$. Antes digamos que el método desaprovechaba gran parte de su capacidad no potenciando los desvíos, que son pieza fundamental en su éxito. Por eso la mejora con respecto a 4.7 es generalizada. Sobre todo a partir de $c_2 = 2$ ($c_2 > 2$) es cuando se nota realmente. Las demás configuraciones o son demasiado explotativas o c_2 es tan pequeño que el efecto del desvío no se puede apreciar. Sin embargo, antes c_2 tenía que ser mayor que 3 para que el algoritmo no convergiera prematuramente y ahora, le basta con ser superior a 2. Gracias al desvío se introduce diversidad y ello permite que el nivel de exploración (tasa de convergencia) sea menor. La mejor de las configuraciones de 4.1 sigue siendo $c_2 = 3$ aunque si probamos configuraciones de $2 < c_2 < 3$, sus resultados no distan demasiado entre sí. Este hecho ya lo demostramos en el apartado 4.2.2 para $V = c_2 rand_2(gBest - X)$ cuando apuntamos que la aleatoriedad modular rebaja ostensiblemente la sensibilidad del método en su hiperparámetro c_2. En cualquier caso, este resultado se convierte en el mejor de largo encontrado hasta el momento.

4.2.3.1. Escalamiento modular

Ahora que hemos conseguido la independencia entre el escalamiento modular y el direccional del vector social, podemos por fin analizar cada uno de forma individual. Respecto al escalamiento modular, podemos aplicar prácticamente el mismo análisis que hicimos en 4.2.2, solo que ahora, al introducirse los cambios direccionales, adquiere aún una mayor relevancia al ser responsable de los desvíos que se dan y que afectan directamente a la diversidad de la búsqueda. Así como en 4.2.2 ya vimos que los movimientos pequeños ($c_2 < 1$) eran peores que los grandes ($c_2 > 1$) precisamente por conllevar una menor diversidad, en este apartado hemos concluido que los movimientos con desvíos menores son peores. Y así como comprobamos en 4.2.2 que en este sentido la presencia de la aleatoriedad no ayudaba y que resultaba más sensato escoger un mismo movimiento para toda la ejecución (determinista y constante) que combinar varios al azar, aquí podemos aplicar la misma lógica. Todo parece indicar que la situación es muy parecida a la que teníamos en 4.2.2 y que por tanto, la decisión coherente es eliminar la aleatoriedad modular también este caso. Así, V_M pasa a ser simplemente c_2. Los nuevos resultados son:

$c_2 = 0.02$	$c_2 = 1.3$	$c_2 = 1.4$	$c_2 = 1.5$	$c_2 = 1.6$	$c_2 = 1.7$
$2.25e+04$	454.17	0.20	$2.05e-04$	0.004	41.47

Cuadro 4.2: Resultados de PSO con $V = c_2 \|gBest - X\| \frac{Rand_2 \circ (gBest - X)}{\|Rand_2 \circ (gBest - X)\|}$.

En esta ocasión hemos optado por no incluir las configuraciones excesivamente ex-

plotativas, ya que sabemos que sus resultados no van a ser significativos. Por lo demás, la ventana de configuraciones aceptables se reduce drásticamente con respecto a 4.1. Como ya sabíamos, ganar en control implica una mayor dependencia de la configuración elegida y la necesidad de una mayor precisión en la elección. El rango de elección se reduce como consecuencia de esta mayor sensibilidad paramétrica. Pero a cambio, la elección óptima arroja mejores resultados en ausencia de aleatoriedad tal y como esperábamos. Aún así las diferencias entre escalamiento aleatorio y determinista no son tan notables ahora como lo eran en 4.2.2, debido a que ahora seguimos incluyendo aleatoriedad en la parte direccional (no la hemos eliminado del todo). Sea como fuere, podemos confirmar que la aleatoriedad modular no es necesaria para la correcta explotación en el PSO. A partir de ahora continuaremos nuestro análisis obviando la aleatoriedad modular.

4.2.3.2. Escalamiento direccional

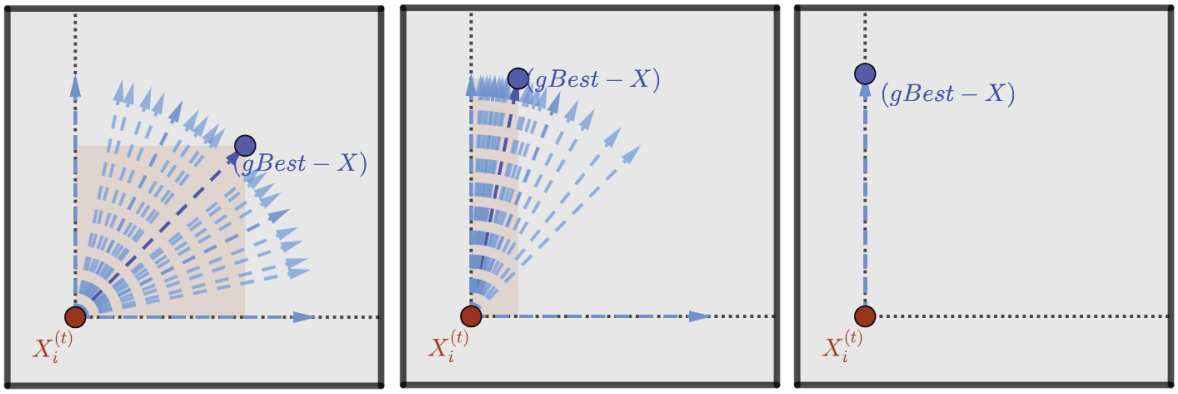
Es momento de centrarnos en el escalamiento direccional del método. De entrada, podemos ver que éste está completamente controlado por la aleatoriedad (variable $Rand_2$). Así como el escalamiento modular sí que está regulado por el usuario mediante la variable de control c_2, el direccional, no. El escalamiento direccional depende de entera y exclusivamente de la aleatoriedad y el usuario no puede intervenir en él. Y dado que su importancia en el método es, según hemos analizado, incluso mayor que la del módulo, no se entiende como no existe ningún mecanismo de control sobre él por parte del usuario. Lo ideal sería disponer de un hiperparámetro que permitiera al usuario regular de algún modo los giros del vector social como pasa con c_2 y la magnitud. Como no es así, debemos aún con más motivo investigar las características de los giros en el PSO para alcanzar un mayor nivel de comprensión sobre su funcionamiento. Tras hacerlo, nuestro objetivo debería ser introducir algún tipo de herramienta que liberara algo esta operación del azar.

El sistema de giro, como ya avanzamos al principio del apartado, consiste en la alteración de las proporciones de los componentes vectoriales mediante la multiplicación directa de cada componente por un número aleatorio independiente y uniforme en $[0, 1]$ ($Rand_2 \circ (gBest - X)$). Si nos fijamos, la alteración es proporcional tanto a las proporciones del vector original $(gBest - X)$ como a las del vector aleatorio $Rand_2$:

$$\frac{\{rand_2\}_{j_1} * (gbest_{j_1} - x_{j_1})}{\{rand_2\}_{j_2} * (gbest_{j_2} - x_{j_2})} = \frac{\{rand_2\}_{j_1}}{\{rand_2\}_{j_2}} * \frac{(gbest_{j_1} - x_{j_1})}{(gbest_{j_2} - x_{j_2})}$$

Esto quiere decirnos que el giro viene marcado por la aleatoriedad pero también por la dirección del vector a girar. En otras palabras, el giro depende de las coordenadas del vector y ante una misma instancia de $Rand_2$, puede existir más o menos giro en función de la dirección del vector social (de las proporciones entre sus componentes). Por ejemplo, no es lo mismo que el vector social sea el $(1, ..., 1)$, en cuyo caso es $Rand_2$ el que decide la nueva dirección (el giro depende de $Rand_2$), que que sea el $(1, 0, 0, ..., 0)$, en cuyo caso no importa el valor de $Rand_2$ y la dirección no puede ser alterada. En general, si el vector social es paralelo a alguno de los ejes de coordenadas, entonces el sistema de giro no puede actuar sobre él y permanecerá intacto (sin girar). Si es diagonal, entonces el giro viene dado por la dirección del vector aleatorio. Estos dos casos son presentados en las figuras 4.9a, 4.9c.

Claramente este sistema de giro del PSO es dependiente del sistema de coordenadas. Los giros dependen de la orientación de los ejes coordenados, es decir, de la base de



(a) (b) (c)

Figura 4.9: Representación gráfica de los giros en el PSO según la dirección del vector social. Conforme más se acerca el vector a uno de los ejes coordenados, mayor es su rango de giro pero menor su dispersión. La distribución de la probabilidad queda cada vez más compactada en torno al vector original hasta llegar al caso extremo (c) donde el vector no tiene posibilidad alguna de giro.

coordenadas escogida. A priori esto no parece deseado ya que los sistemas de coordenadas se utilizan para representar una realidad desde una perspectiva concreta, pero no existen, tan solo son una herramienta artificial que necesitamos para describir eventos en el espacio. Y que la posición absoluta de una partícula interfiera en su movimiento va completamente en contra de la intuición física que queremos recrear en el PSO. El movimiento de las partículas debería ser independiente del observador y su sistema de referencia de acuerdo con las leyes físicas para evitar la existencia de un sistema de referencia privilegiado. Aquí estamos inspirándonos en una metáfora física y aunque no respetar este principio no es erróneo, sí que es algo incoherente. Para que la búsqueda fuera fiel al modelo real, lo único que debería influir en las decisiones sobre el movimiento de las partículas serían sus posiciones relativas (en este caso sus posiciones con respecto al $gBest$). Si no, las partículas están mostrando una preferencia por ciertas direcciones del espacio cuando todas deberían ser equivalentes, más si estamos buscando soluciones homogéneamente repartidas por el espacio.

Esta dependencia en la orientación del sistema de coordenadas ya la estudiamos en 3.4.4 durante la review de la sección anterior. La denominamos 'varianza rotacional'. Y por seguir aportando más detalles que la confirman, podemos recordar las propiedades del sistema de giro de 3.4.4. Una de ellas es que el vector girado reside siempre en el mismo ortante que el original. La otra, es una generalización de los casos mencionados de 4.9a, 4.9c (la propiedad 3). Esta última recoge que a medida que la dirección del vector original se aproxima a la de uno de los ejes coordenados, la probabilidad de sus giros decae hasta llegar a ser 0 cuando son paralelos. Es decir, cuanto menor sea el ángulo entre el vector social y uno de los ejes coordenados, mayor será su resistencia a girar (véase figura 4.9b). Sin embargo, su rango de giro será mayor (puede girar más). Y a esto podemos añadir que en la versión original $V = c_2 Rand_2 \circ (gBest - X)$ el módulo del vector también se ve afectado por el sistema de coordenadas. Como apuntamos antes, a mayor giro, menor es el módulo esperado y como los giros son dependientes del marco de referencia, también lo son los módulos. Esto lo podemos volver a constatar comparando los dominios instantáneos

de búsqueda de los casos de 4.9a y 4.9b. Los giros en vectores próximos a los ejes (4.9b) van acompañados de una reducción modular mayor que la de los próximos a las diagonales (4.9a). Además, algo que tampoco se ha dicho es que la probabilidad del giro se concentra más en torno a la diagonal del dominio instantáneo, o sea, independientemente de la dirección del vector social, siempre existe una mayor probabilidad de que el giro sea pequeño. La probabilidad del giro es siempre decreciente con su magnitud. Esto es gracias a que los elementos de $Rand_2$ siguen distribuciones uniformes independientes, lo que hace que sea más probable que las proporciones entre ellos sean cercanas a 1 que no lejanas (probabilidad direccional no uniforme sesgada hacia la diagonal $(1,...,1)_u$). Esto por ejemplo no sucedería si $Rand_2$ siguiese una distribución normal multivariada $\mathcal{N}(\vec{0}, I)$, que es esféricamente simétrica (con distribución de probabilidad ajena a la dirección del vector).

Todo ello deja de manifiesto que, aparte de no existir un criterio consistente y justificado de giro en el PSO, éste (el giro) aparenta no ser tenido demasiado en consideración. Y es que, por lo que hemos podido analizar, en la mayoría de situaciones los giros son escasos y se da más preferencia a no girar o a hacerlo muy poco. Y encima, cuando se hace con notoriedad, suele venir acompañado de una reducción muy grande del módulo, lo cual no ayuda en absoluto a fomentar el desvío que tan beneficioso resulta. Esto último ya lo solventamos eliminando la aleatoriedad modular del método y ahora nos hemos de poner con los problemas relativos a la parte direccional V_D. Así pues, retomando $V = c_2\|gBest - X\|\frac{Rand_2 \circ (gBest-X)}{\|Rand_2 \circ (gBest-X)\|}$, demostremos empíricamente la validez de nuestro análisis recabando estadísticos sobre la distribución del ángulo de giro α:

Media	**Desviación típica**	**Mediana**	**Rango**
$28.19°$	$12.95°$	$26.17°$	$87.42°$

Cuadro 4.3: Estadísticas sobre el ángulo de giro α en el PSO con $V = c_2\|gBest - X\|\frac{Rand_2 \circ (gBest-X)}{\|Rand_2 \circ (gBest-X)\|}$ para $c_2 = 1.5$.

Los resultados respaldan nuestra apreciación de que los giros en el PSO son cortos. El ángulo medio de giro es bajo y la probabilidad se concentra en valores aún más bajos que la media (la mediana es menor que la media). La desviación típica no es tampoco demasiado grande comparada con el rango. Esto explica que tengamos que utilizar un escalamiento modular relativamente elevado (c_2 alto) para compensar el desvío que ha de estar presente para explotar mejor el $gBest$ y que el algoritmo pueda funcionar bien y no converja a un valor pobre. Aún y todo, hay ocasiones en las que, por muy grande que sea el módulo, si el ángulo de giro es muy corto, el desvío no podrá ser grande y se resentirá la búsqueda. El desvío está condicionado por el giro. Que el sistema de giro sea inmutable y no adaptable a las necesidades de la optimización impide que podamos adoptar una estrategia óptima, pues supedita nuestras decisiones. Los giros son cortos y esto nos obliga a establecer un c_2 alto, cuando quizá, no fuera la decisión idónea ya que repercute a otros aspectos del método (tasa de convergencia, etc). Es el momento pues de abordar el siguiente objetivo que nos pusimos al principio: el de introducir algún tipo de mecanismo que agregue capacidad de control al usuario sobre el sistema de giro. Además, así aprovechamos también para corregir esa varianza rotacional del sistema de giro. La idea es proponer un nuevo sistema de giro que funcione igual para todo tipo de situaciones y que esté

parametrizado con el fin de poder otorgar poder de decisión al usuario sobre el giro de las partículas.

4.2.3.3. Propuesta de método de giro

Lo que buscamos es un método de giro de vectores para el PSO que sea rotacionalmente invariante y que permita al usuario elegir el ángulo del giro. Bajo esta descripción, ya vimos que en la literatura existen principalmente dos propuestas, que son las de Wilke (WPSO[122]) y Bonjadi (RotmPSO[125]). Las dos emplean matrices de rotación R_l para girar los vectores social y cognitivo y se diferencian en la manera en la que definen dichas matrices. En [122] siguen la definición dada por 3.18 y en [125] la dada por las expresiones 3.19 y 3.20.

Aunque estas dos propuestas son muy parecidas a lo que buscamos, no son exactamente lo que queremos. Y es que, recordemos, nuestra concepción de giro es restrictiva. Nosotros en todo momento por giro nos referimos al cambio de dirección que experimenta el vector original tras la operación. El proceso de transformación que sigue el vector original no es lo relevante. Lo relevante es la diferencia que existe entre la nueva dirección del vector resultante y la antigua del original. Lo que va a decidir el desvío de las partículas va a ser el ángulo final entre el vector original y el girado y no cuál haya sido el ángulo de giro. En cambio, una matriz de rotación sirve para rotar un vector en un plano cualquiera, lo que significa que el ángulo entre el vector rotado y el original no es igual al ángulo de rotación. Los vectores pueden rotar mucho (ángulo de rotación grande) sin modificar demasiado su dirección (ángulo pequeño entre vector original y rotado). Esto no pasa cuando el plano de rotación contiene al vector original. En este caso, la rotación da como resultado un nuevo vector contenido también en ese plano y el ángulo de rotación coincide con el ángulo entre los dos vectores. Es a este tipo concreto de giros a los que nos referíamos durante el análisis. Obviamente es una subclase dentro de la clase completa de rotaciones que pueden darse es un espacio d-dimensional. Por eso no estamos interesados en el conjunto completo de las matrices de rotación $Orth^+$ sino tan solo en un subconjunto de las mismas $\subset Orth^+$ definido para cada vector a girar (la rotación depende del vector).

Resumidamente, nosotros queremos un método `Gira(`$\vec{g}$`,`α`)` que dado un ángulo de giro α, devuelva un vector rotado aleatoriamente $\vec{a}$ que forme α grados con $\vec{g}$. Matemáticamente buscamos un vector $\vec{a}$ que satisfaga la siguiente ecuación:

$$\langle \vec{g}, \vec{a} \rangle = \sum_{j=1}^{d} g_j a_j = \cos\alpha \|\vec{g}\| \|\vec{a}\|$$

con $\|\vec{a}\| = \|\vec{g}\|$. Haciendo esta sustitución en la ecuación y operando con ella tenemos:

$$\sum_{j=1}^{d} g_j a_j = \cos\alpha \|\vec{g}\|^2 \implies \sum_{j=1}^{d} g_j a_j - \|\vec{g}\|^2 \cos\alpha = 0 \implies \sum_{j=1}^{d} g_j a_j - \left(\sum_{j=1}^{d} g_j^2 \right) \cos\alpha = 0 \implies$$

$$\implies \sum_{j=1}^{d} g_j a_j - \sum_{j=1}^{d} g_j^2 \cos\alpha = 0 \implies \sum_{j=1}^{d} \left(g_j a_j - g_j^2 \cos\alpha \right) = 0 \implies \sum_{j=1}^{d} \underbrace{(a_j - g_j \cos\alpha)}_{r_j} g_j = 0$$

Definiendo un nuevo vector $\vec{r}$ cuyos elementos $r_j = a_j - g_j \cos\alpha$, tenemos que $\vec{r}$ es perpendicular a $\vec{g}$ ($\vec{r} \perp \vec{g}$). Reescribiendo $\vec{a}$ en función de $\vec{r}$:

$$a_j = r_j + g_j \cos\alpha \ \mid \ \vec{r} \perp \vec{g}$$

Además como $\|\vec{a}\| = \|\vec{g}\|$:

$$\|\vec{a}\|^2 = \|\vec{g}\|^2 \implies \sum_{j=1}^{d} a_j^2 = \|\vec{g}\|^2 \implies \sum_{j=1}^{d} (r_j + g_j \cos\alpha)^2 = \|\vec{g}\|^2 \implies$$

$$\implies \sum_{j=1}^{d} \left(r_j^2 + g_j^2 \cos^2\alpha + 2 r_j g_j \cos\alpha\right) = \|\vec{g}\|^2 \implies$$

$$\implies \sum_{j=1}^{d} r_j^2 + \sum_{j=1}^{d} g_j^2 \cos^2\alpha + \sum_{j=1}^{d} 2 r_j g_j \cos\alpha = \|\vec{g}\|^2 \implies$$

$$\implies \|\vec{r}\|^2 + \cos^2\alpha \|\vec{g}\|^2 + 2\cos\alpha \underbrace{\left(\sum_{j=1}^{d} r_j g_j\right)}_{\shortparallel\ 0} = \|\vec{g}\|^2 \implies$$

$$\implies \|\vec{r}\|^2 = \|\vec{g}\|^2 - \|\vec{g}\|^2 \cos^2\alpha \implies \|\vec{r}\|^2 = \|\vec{g}\|^2 \left(1 - \cos^2\alpha\right) \implies$$

$$\implies \|\vec{r}\|^2 = \|\vec{g}\|^2 \sin^2\alpha \implies \|\vec{r}\| = \|\vec{g}\| \sin\alpha$$

Podemos recoger las conclusiones en la siguiente proposición:

Proposición 1 *Sea* $\vec{a} = (a_1, ..., a_j, ..., a_d) : a_j = r_j \sin\alpha + g_j \cos\alpha \mid (\vec{r} \perp \vec{g}) \wedge (\|\vec{r}\| = \|\vec{g}\|) \xrightarrow{\text{entonces}} (\angle(\vec{a}, \vec{g}) = \alpha) \wedge (\|\vec{a}\| = \|\vec{g}\|)$.

Geométricamente la demostración de este resultado es trivial por definición de seno y coseno y se plasma en la figura 4.10. Todo lo que hay que saber es que $\sin\alpha = \frac{\text{cateto opuesto}}{\text{hipotenusa}}$ y que $\cos\alpha = \frac{\text{cateto contiguo}}{\text{hipotenusa}}$.

Básicamente para conseguir nuestro propósito de obtener un vector rotado $\vec{a}$ que forme α grados con el original $\vec{g}$, debemos encontrar primero un vector perpendicular a $\vec{g}$ y de igual módulo. Es decir, encontrar un $\vec{r} \mid (\langle \vec{r}, \vec{g} \rangle = 0) \wedge (\|\vec{r}\| = \|\vec{g}\|)$. Sin embargo, esto no es complicado. Existen multitud de técnicas que nos permiten hallar un vector perpendicular a otro. Nosotros hemos escogido una también basada en el concepto de proyección ortogonal de un vector sobre otro. Es la siguiente:

$$\vec{r} = \frac{\vec{u} - \text{proj}_{\vec{g}}\vec{u}}{\left\|\vec{u} - \text{proj}_{\vec{g}}\vec{u}\right\|}\|\vec{g}\| = \frac{\vec{u} - \frac{\langle \vec{u}, \vec{g} \rangle}{\|\vec{g}\|^2}\vec{g}}{\left\|\vec{u} - \frac{\langle \vec{u}, \vec{g} \rangle}{\|\vec{g}\|^2}\vec{g}\right\|}\|\vec{g}\| = \frac{\vec{u} - \frac{\sum_{j=1}^{d} u_j g_j}{\|\vec{g}\|^2}\vec{g}}{\left\|\vec{u} - \frac{\sum_{j=1}^{d} u_j g_j}{\|\vec{g}\|^2}\vec{g}\right\|}\|\vec{g}\|$$

donde el vector $\vec{u}$ es un vector aleatorio no paralelo a $\vec{g}$ que sigue una distribución normal multivariada $\mathcal{N}(\vec{0}, I_d)$.[5] Esta técnica logra obtener un vector perpendicular $\vec{r}$ al original $\vec{g}$ a través del cálculo de la proyección ortogonal de un vector aleatorio $\vec{u}$ sobre $\vec{g}$. Dado que el vector $\vec{u}$ sigue una distribución esféricamente simétrica, puede tomar cualquier dirección del espacio con igual probabilidad y ello conlleva que el muestreo del vector perpendicular $\vec{r}$ se haga a partir de una distribución de probabilidad uniforme sobre el conjunto de los vectores perpendiculares a $\vec{g}$. Esto implica que el giro de $\vec{g}$ puede darse en cualquier dirección del espacio d-dimensional por igual, que es lo que necesitamos de cara a la búsqueda del PSO.

[5] La condición de no paralelismo de $\vec{u}$ con $\vec{g}$ se supone por ser la distribución de probabilidad de $\vec{u}$ continua y por tanto la probabilidad de que se de nula ($= 0$).

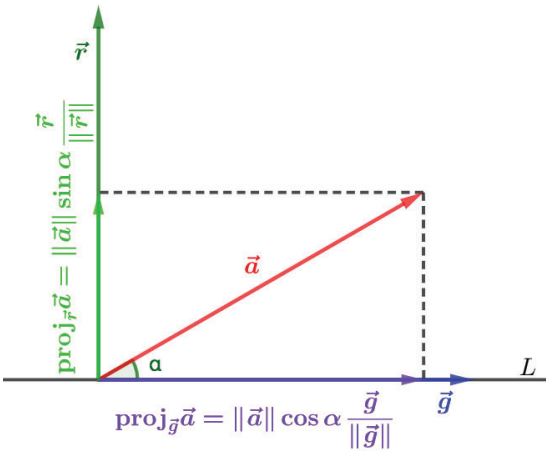


Figura 4.10: Representación geométrica bidimensional de la descomposición de un vector $\vec{a}$ en la suma de vectores ortogonales.

Finalmente, juntando todas las partes, nos queda que

$$\vec{a} = \frac{\vec{u} - \frac{\sum_{j=1}^{d} u_j g_j}{\|\vec{g}\|^2}\vec{g}}{\left\|\vec{u} - \frac{\sum_{j=1}^{d} u_j g_j}{\|\vec{g}\|^2}\vec{g}\right\|}\|\vec{g}\| \sin\alpha + \vec{g}\cos\alpha \tag{4.1}$$

El algoritmo del método de giro en pseudocódigo sería el siguiente:

Algoritmo 4: `Gira(`$\vec{g}$`,` α`)`: Girar vector $\vec{g}$ α grados de tal modo que el nuevo vector difiera del original en α grados.

1 **Function** *Gira* ($\vec{g}$,α)
  Input: $\vec{g}$,α
  Output: $\vec{a}$
2 Extraer muestra $\vec{u} \sim \mathcal{N}(\vec{0}, I_d)$
3 $\vec{r} = \vec{u} - \frac{\vec{u}^{\mathrm{T}}\vec{g}}{\|\vec{g}\|^2}\vec{g}$
4 $\vec{r} = \frac{\vec{r}}{\|\vec{r}\|}\|\vec{g}\|$
5 $\vec{a} = \vec{r}\sin\alpha + \vec{g}\cos\alpha$

El método, por lo que hemos comentado de que no muestra preferencia por ninguna dirección de giro en particular, es rotacionalmente invariante. De todas formas lo demostraremos. Para ello recordemos que tenemos que probar que el resultado de realizar la operación (en este caso el giro) en un marco de referencia $\hat{\boldsymbol{x}}$ sea el mismo que el de hacerlo en otro $\boldsymbol{x}$. Sin embargo aquí convendría puntualizar qué es lo que entendemos por 'el mismo'. En [122] es donde sientan el precedente de demostrar la no-invarianza rotacional del PSO y aportan las condiciones que sirven de estándar para que el método sea rotacionalmente invariante. Determinan que para que esto se cumpla se ha de dar que $\Phi_l Q = Q\Phi_l,\ l = 1, 2\ , \forall Q \in Orth^+$, hecho que solo se da cuando $\Phi_l = a_l I_d,\ l = 1, 2$ con $a_l \in \mathbb{R}$. Es por eso que tratan de construir un método que aproxime estocásticamente al PSO lineal sustituyendo Φ_l por $R_l \approx I_d$ (Ec. 3.18). No obstante, toda su deducción y lógica de razonamiento se debe a una

interpretación particular de la igualdad en el resultado de la actualización del PSO. Su interpretación es demasiado estricta. Para ellos, la actualización llevada a cabo en ambos sistemas de referencia $\hat{\boldsymbol{x}}$ y $\boldsymbol{x}$ tiene que derivar exactamente en la misma posición para que el método sea considerado rotacionalmente invariante. Así la ecuación $\Phi Q = Q\Phi \ , \forall Q \in Orth^+$ ha de cumplirse en el sentido clásico. Pero lo que no tienen en cuenta es que en realidad las variables Φ_l son aleatorias y que por tanto, la importancia no reside en la instancia o valor concreto que pueda tomar en un momento dado sino en su distribución de probabilidad, que es lo que las define. Es decir, no tiene demasiado sentido comparar variables aleatorias como si fueran comunes (no aleatorias). Y por eso, quizá la forma más adecuada de afrontar la ecuación $\Phi Q = Q\Phi \ , \forall Q \in Orth^+$ sea desde un sentido probabilístico, considerando que la igualdad es en distribución. En este punto, sí que tendríamos que cualquier matriz de rotación aleatoria R con distribución de probabilidad uniforme en $Orth^+$ satisfaría $RQ \overset{\mathrm{d}}{=} QR \ , \forall Q \in Orth^+$, ya que por definición toda multiplicación de matrices de rotación es también matriz de rotación y como R es uniforme, también lo es cualquier rotación aplicada sobre R y viceversa ($QR \overset{\mathrm{d}}{=} R \overset{\mathrm{d}}{=} RQ \ , \forall Q \in Orth^+$).

De todos modos, no es necesario que la matriz de rotación R siga una distribución uniforme en $Orth^+$ para cumplir $RQ \overset{\mathrm{d}}{=} QR \ , \forall Q \in Orth^+$. Basta con que su distribución sea independiente del plano de rotación, o lo que es lo mismo, que todos los ejes de giro tengan la misma probabilidad de ocurrir y sean tratados por igual. Es decir, que a fin de cuentas lo único que importe sea la orientación relativa de los ejes con respecto a la del vector y no (respecto) al sistema de referencia. Si esto se da, es indiferente la orientación (dirección) del vector original, pues la distribución de probabilidad de su rotación no variará al no distinguir entre planos de rotación y el método será rotacionalmente invariante. En cambio si no se da y no todos los planos de rotación son igual considerados en la distribución de R, entonces el giro afectará diferente en función de la orientación del vector y el método tendrá varianza rotacional. El método de [125] da por supuesta esta condición de uniformidad en los ejes de rotación.

Nosotros sin embargo sí vamos a demostrar de forma rigurosa que nuestra propuesta es rotacionalmente invariante. Lo que tenemos que hacer es demostrar que

$$Q\,\texttt{Gira}\,(\vec{g},\alpha) \ \overset{\mathrm{d}}{=}\ \texttt{Gira}\,(Q\vec{g},\alpha)\,, \quad \forall Q \in Orth^+$$

Desarrollando la parte derecha de la ecuación:

$$\texttt{Gira}\,(Q\vec{g},\alpha) \ = \frac{\vec{u} - \frac{\vec{u}^{\mathrm{T}} Q\vec{g}}{\|Q\vec{g}\|^2} Q\vec{g}}{\left\|\vec{u} - \frac{\vec{u}^{\mathrm{T}} Q\vec{g}}{\|Q\vec{g}\|^2} Q\vec{g}\right\|} \|Q\vec{g}\| \sin\alpha + Q\vec{g}\cos\alpha$$

Teniendo en cuenta que $\vec{u} \sim \mathcal{N}(\vec{0}, I_d)$ y que por tanto sigue una distribución esféricamente simétrica, uniforme en las direcciones espaciales (todas las direcciones tienen la misma probabilidad, i.e. la distribución de probabilidad de los vectores es ajena a su orientación espacial), $\vec{u}$ es invariante ante rotaciones y $Q\vec{u} \overset{\mathrm{d}}{=} \vec{u} \overset{\mathrm{d}}{=} \vec{u}Q \ , \forall Q \in Orth^+$. Además, sabiendo que la multiplicación de un vector $\vec{g}$ por una matriz de rotación Q no altera el módulo del vector sino tan solo su dirección ($\|Q\vec{g}\| = \|\vec{g}\|$) y que por

definición $Q^{\mathrm{T}}Q = I$:

$$\begin{aligned}
\texttt{Gira}(Q\vec{g},\alpha) &= \frac{\vec{u} - \frac{\vec{u}^{\mathrm{T}}Q\vec{g}}{\|\vec{g}\|^2}Q\vec{g}}{\left\|\vec{u} - \frac{\vec{u}^{\mathrm{T}}Q\vec{g}}{\|\vec{g}\|^2}Q\vec{g}\right\|}\|\vec{g}\|\sin\alpha + Q\vec{g}\cos\alpha \overset{\mathrm{d}}{=} \frac{Q\vec{u} - \frac{(Q\vec{u})^{\mathrm{T}}Q\vec{g}}{\|\vec{g}\|^2}Q\vec{g}}{\left\|Q\vec{u} - \frac{(Q\vec{u})^{\mathrm{T}}Q\vec{g}}{\|\vec{g}\|^2}Q\vec{g}\right\|}\|\vec{g}\|\sin\alpha + Q\vec{g}\cos\alpha \\
&= \frac{Q\vec{u} - \frac{\vec{u}^{\mathrm{T}}Q^{\mathrm{T}}Q\vec{g}}{\|\vec{g}\|^2}Q\vec{g}}{\left\|Q\vec{u} - \frac{\vec{u}^{\mathrm{T}}Q^{\mathrm{T}}Q\vec{g}}{\|\vec{g}\|^2}Q\vec{g}\right\|}\|\vec{g}\|\sin\alpha + Q\vec{g}\cos\alpha = \frac{Q\vec{u} - \frac{\vec{u}^{\mathrm{T}}\vec{g}}{\|\vec{g}\|^2}Q\vec{g}}{\left\|Q\vec{u} - \frac{\vec{u}^{\mathrm{T}}\vec{g}}{\|\vec{g}\|^2}Q\vec{g}\right\|}\|\vec{g}\|\sin\alpha + Q\vec{g}\cos\alpha \\
&= \frac{Q\left(\vec{u} - \frac{\vec{u}^{\mathrm{T}}\vec{g}}{\|\vec{g}\|^2}\vec{g}\right)}{\left\|Q\left(\vec{u} - \frac{\vec{u}^{\mathrm{T}}\vec{g}}{\|\vec{g}\|^2}\vec{g}\right)\right\|}\|\vec{g}\|\sin\alpha + Q\vec{g}\cos\alpha = Q\frac{\vec{u} - \frac{\vec{u}^{\mathrm{T}}\vec{g}}{\|\vec{g}\|^2}\vec{g}}{\left\|\vec{u} - \frac{\vec{u}^{\mathrm{T}}\vec{g}}{\|\vec{g}\|^2}\vec{g}\right\|}\|\vec{g}\|\sin\alpha + Q\vec{g}\cos\alpha \\
&= Q\left(\frac{\vec{u} - \frac{\vec{u}^{\mathrm{T}}\vec{g}}{\|\vec{g}\|^2}\vec{g}}{\left\|\vec{u} - \frac{\vec{u}^{\mathrm{T}}\vec{g}}{\|\vec{g}\|^2}\vec{g}\right\|}\|\vec{g}\|\sin\alpha + \vec{g}\cos\alpha\right) = Q\,\texttt{Gira}(\vec{g},\alpha)
\end{aligned}$$

Acabamos de demostrar que nuestra propuesta de giro sí es rotacionalmente invariante. Esto significa que es consistente y funciona igual para cualquier tipo de situación, independientemente del sistema de referencia que se elija y del vector a girar. Aparte goza de más ventajas. Por ejemplo si comparamos nuestra propuesta con la de Wilke (WPSO[122]), vemos que la nuestra es un método de giro exacto mientras que la de Wilke (Ec. 3.18) es un método de aproximación. Su método consiste en generar matrices de rotación aproximadas, por lo que de entrada, los giros ya no son precisos sino que contienen errores de aproximación. Además al no ser rotaciones exactas, la operación de giro también altera el módulo de los vectores. Y su aproximación solo es válida para valores pequeños del ángulo de giro α, pues los errores de aproximación crecen con α y para valores grandes de α, el error es demasiado grande y el método no sirve. Con lo cual, α es controlable por el usuario pero lo es solo en un pequeño rango, por lo que el poder de decisión sobre él es muy reducido.

Por otra parte, comparando nuestra propuesta con la de Bonjadi (RotmPSO[125]), podemos comprobar que la de Bonjadi obliga a que el ángulo de giro α sea aleatorio mientras la nuestra, no. Su construcción de la matriz de rotación dada por la expresión 3.20 requiere que los ángulos de rotación por los planos principales, $\alpha_{i,j}$, sean aleatorios para que la rotación pueda ocurrir en cualquier plano. Pero al mismo tiempo, esto implica que el ángulo final del giro es gobernado por la aleatoriedad. Por tanto aunque su método de giro es exacto al igual que el nuestro, no es controlable más que en distribución. Tan solo podemos controlar la media μ y desviación típica σ de la distribución gaussiana que usa para determinar los ángulos de giro por cada plano principal de rotación ($\alpha_{i,j}$). El nuestro en cambio permite al usuario seleccionar con exactitud el ángulo total del giro α. Aparte como ya dijimos, nuestro método no consiente cualquier plano de rotación, solamente permite aquellos que dan lugar a rotaciones efectivas en cuanto a cambios direccionales, por lo que la selección exacta del ángulo de giro equivale a la selección exacta del cambio direccional del vector, que es lo realmente importante en el PSO. El RotmPSO no genera rotaciones uniformemente distribuidas ([137]) y a algunos cambios direccionales se les asigna mayor probabilidad que a otros. En concreto imperan los cambios pequeños de dirección, que es justo lo que tratábamos de corregir del PSO original. En definitiva, el método de giro de Bonjadi (RotmPSO[125], Ec. 3.20) no admite demasiado control del usuario ni una buena precisión en sus decisiones.

Mención aparte podríamos hacer sobre la complejidad computacional. El WPSO (Ec. 3.18) genera matrices cuadradas $d \times d$ de números aleatorios, lo cual tiene una complejidad $\mathcal{O}(d^2)$. Después realiza dos sumas de matrices $d \times d$, cuya complejidad también es $\mathcal{O}(d^2)$. En total, el orden de complejidad de su algoritmo es de $\mathcal{O}(d^2)$. Por su parte, el RotmPSO realiza $\binom{d}{2} - 1 = \frac{d(d-1)}{2} - 1$ productos matriciales. De normal cada producto matricial tiene un coste del orden de $\mathcal{O}(d^3)$, por lo que la complejidad total del algoritmo ascendería a $\mathcal{O}(d^5)$. Sin embargo, los autores aprovechándose de la forma de las matrices (Ec. 3.19), logran proponer una implementación que reduce la complejidad del producto matricial a $\mathcal{O}(1)$, con lo que el coste final de su implementación es de $\mathcal{O}(d^2)$. Y por lo que respecta a nuestro método, nosotros generamos vectores aleatorios de dimensión d ($\mathcal{O}(d)$) y luego el resto de operaciones son del orden de $\mathcal{O}(d)$ como el producto escalar de vectores d-dimensionales, la norma de vectores d-dimensionales, la multiplicación de vectores d-dimensionales por escalares o las sumas entre esos vectores. Así pues, nuestro método tiene un coste computacional del orden de $\mathcal{O}(d)$. Comparado con los otros dos métodos de giro, el nuestro tiene un orden menor de complejidad. Mientras los otros tienen un orden cuadrático, el nuestro es lineal. Este hecho remarca aún más la bondad de nuestra propuesta con respecto a las existentes.

4.2.3.4. Integración del nuevo sistema de giro

Por fin disponemos de un sistema de giro independiente de la orientación de los vectores y parametrizado, para que sea el usuario el que decida en todo momento la magnitud de los giros. Ya no nos vamos a ver forzados a elegir una determinada configuración paramétrica sino que nuestro abanico de opciones se va a incrementar, ya que ahora podremos tener un mayor control sobre los desvíos y esto presumiblemente supondrá una mejora en el rendimiento del método. Este nuevo sistema de giro nos va a permitir seguir analizando el impacto del escalamiento direccional en el método.

Volviendo al punto donde lo dejamos del análisis, nos habíamos quedado en que $V_M = c_2$ y $V_D = \|gBest - X\| \frac{Rand_2 \circ (gBest - X)}{\|Rand_2 \circ (gBest - X)\|}$. De ahora en adelante, en cambio, el escalamiento direccional vendrá dado por $V_D = \texttt{Gira}\,((gBest - X), \alpha)$. Con este nuevo escalamiento direccional, podemos probar cómo se comporta el método ante distintos valores de α. En la siguiente tabla se recogen algunos resultados del PSO con $V = c_2\texttt{Gira}\,((gBest - X), \alpha)$ para distintos valores de c_2 y α:

	$\boldsymbol{c_2 = 0.1}$	$\boldsymbol{c_2 = 0.9}$	$\boldsymbol{c_2 = 1.1}$	$\boldsymbol{c_2 = 1.3}$	$\boldsymbol{c_2 = 1.5}$	$\boldsymbol{c_2 = 1.7}$
$\boldsymbol{\alpha = 18°}$	$2.64e+04$	$8.80e+05$	$1.17e+06$	$7.63e+05$	$2.27e+05$	$4.79e-18$
$\boldsymbol{\alpha = 28°}$	$2.82e+04$	$5.32e+05$	$5.47e+05$	$2.75e+04$	$1.02e-14$	127.31
$\boldsymbol{\alpha = 38°}$	$2.40e+04$	$1.16e+05$	$6.37e-27$	$1.07e-10$	205.10	$1.62e+06$
$\boldsymbol{\alpha = 48°}$	$2.15e+04$	$1.21e-19$	$1.44e-05$	$2.91e+04$	$1.63e+06$	$1.63e+06$

Cuadro 4.4: Resultados de PSO con $V = c_2\texttt{Gira}\,((gBest - X), \alpha)$.

En general se ven mucho mejores resultados que antes, incluso para valores de α parecidos a los que se dan en promedio en el PSO estándar. Sin duda influye mucho que ahora α ya no depende de la orientación del vector. El desvío de las partículas es el mismo sean cuales sean sus posiciones con respecto al $gBest$. Así, la calidad de

los movimientos de las partículas no tiene que ver con sus situaciones geográficas y la búsqueda mejora. Por otro lado, vemos que ante un mismo c_2, un mayor α da un mayor desvío y una mayor exploración. Lo mismo pasa con c_2 una vez fijado α. Además es claro que la flexibilidad que otorga el poder seleccionar el ángulo de giro α resulta muy beneficiosa.

Lo cierto es que los resultados son suponiendo un α determinista que no cambia durante la ejecución, cuando en el PSO clásico, el ángulo de giro es aleatorio. Sin embargo, siguiendo la coherencia en nuestro análisis, tal y como hicimos con el escalamiento modular, eliminar la aleatoriedad de los desvíos es positivo, mantener un desvío óptimo constante da mejor resultado que hacerlo al azar. Y esto aplica también a los giros, por lo que lo adecuado es eliminar la aleatoriedad de α. Como muestra de ello, en la siguiente tabla se presentan los resultados análogos a los de 4.4, solo que sustituyendo α por una variable aleatoria uniforme con media α. La comparación con 4.4 deja a las claras lo expuesto:

	$c_2 = 0.1$	$c_2 = 0.9$	$c_2 = 1.1$	$c_2 = 1.3$	$c_2 = 1.5$	$c_2 = 1.7$
$\alpha \sim \mathbf{U(0°, 36°)}$	$2.29e+04$	$9.24e+05$	$1.20e+06$	$8.81e+05$	$2.85e+05$	0.37
$\alpha \sim \mathbf{U(0°, 56°)}$	$2.46e+04$	$8.28e+05$	$9.52e+05$	$4.69e+05$	673.56	640.53
$\alpha \sim \mathbf{U(0°, 76°)}$	$2.44e+04$	$6.45e+05$	$7.67e+05$	$4.67e+03$	2.09	$1.56e+06$
$\alpha \sim \mathbf{U(0°, 96°)}$	$2.61e+04$	$6.13e+05$	$2.83e+05$	$1.30e-03$	$1.13e+06$	$1.53e+06$

Cuadro 4.5: Resultados de PSO con $V = c_2\texttt{Gira}\,((gBest - X), \alpha)$.

Durante todo este apartado, los desvíos han acaparado toda nuestra atención debido a su gran relevancia a la hora de explotar bien la región del $gBest$. Hemos convenido en todo momento que los desvíos grandes permitían aumentar la diversidad de la búsqueda mejorándola. Sin embargo, hemos dejado de lado la importancia del nivel de exploración o tasa de convergencia del algoritmo. Esta tasa ya acordamos que venía dada por la distancia proporcional al $gBest$ (la nueva distancia relativa a la anterior). Lógicamente, el desvío y la distancia proporcional van de la mano, están relacionados. Normalmente un desvío grande está asociado a una distancia proporcional también grande. Por eso muchas veces cuando hablamos de incrementar el desvío, implícitamente estamos incrementando el nivel de exploración de nuestro algoritmo (reduciendo la tasa de convergencia). Al hacerlo, las partículas aumentan la diversidad de la búsqueda y encima, tienen más tiempo para explorar ya que se aproximan más lentamente al $gBest$. Pero el efecto de demorar más la convergencia (subir el nivel explorativo), muchas veces juega en contra porque hace que el algoritmo sobreexplore. Es decir, incrementar el desvío es positivo siempre y cuando nos mantengamos en unos niveles de exploración adecuados. Si el algoritmo es demasiado explorativo, entonces por mucho que ganemos en diversidad, los resultados no serán buenos por la lentitud del proceso de búsqueda (esto se ve en algunos resultados de 4.4).

Mediante los escalamientos modular y direccional controlamos tanto el nivel de exploración a través de la distancia proporcional como la diversidad a través principalmente del desvío. Por eso con el sistema de giro del PSO estándar, que no nos dejaba controlar α, estábamos obligados a seleccionar un c_2 relativamente alto para tener diversidad, lo cual nos forzaba a un nivel de exploración más grande del que quizá era necesario para el problema. A veces incluso nos era imposible alcanzar un nivel de diversidad adecuado sin que el algoritmo fuera divergente. Con la inclusión del

hiperparámetro α tenemos más libertad para poder decidir el nivel de exploración y el desvío por separado. Expresemos ambos matemáticamente. En primer lugar, el nivel de exploración dado por la distancia proporcional, que denotaremos por λ_1, en esta ocasión sería:

$$\lambda_1 = \frac{\|gBest - (X + c_2\texttt{Gira}\,((gBest - X), \alpha)\,)\|}{\|gBest - X\|} = \frac{\|(gBest - X) - c_2\texttt{Gira}\,((gBest - X), \alpha)\,\|}{\|gBest - X\|}$$

Sustituyendo los vectores $(gBest - X)$ y $\texttt{Gira}\,((gBest - X), \alpha)$ por $\vec{g}$ y $\vec{a}$ respectivamente con el fin de clarificar la expresión:

$$\lambda_1 = \frac{\|\vec{g} - c_2\vec{a}\|}{\|\vec{g}\|} = \frac{\sqrt{\sum_{j=1}^{d}(g_j - c_2 a_j)^2}}{\|\vec{g}\|} = \frac{\sqrt{\sum_{j=1}^{d}(g_j^2 + c_2^2 a_j^2 - 2c_2 g_j a_j)}}{\|\vec{g}\|} =$$

$$= \frac{\sqrt{\sum_{j=1}^{d} g_j^2 + c_2^2 \sum_{j=1}^{d} a_j^2 - 2c_2 \sum_{j=1}^{d} g_j a_j}}{\|\vec{g}\|} = \frac{\sqrt{\|\vec{g}\|^2 + c_2^2\|\vec{a}\|^2 - 2c_2\|\vec{g}\|\|\vec{a}\|\cos\angle(\vec{g},\vec{a})}}{\|\vec{g}\|}$$

Como $\|\vec{a}\| = \|\vec{g}\|$ y $\angle(\vec{a},\vec{g}) = \alpha$:

$$\lambda_1 = \frac{\sqrt{\|\vec{g}\|^2 + c_2^2\|\vec{g}\|^2 - 2c_2\|\vec{g}\|^2\cos\alpha}}{\|\vec{g}\|} = \frac{\sqrt{\|\vec{g}\|^2(1 + c_2^2 - 2c_2\cos\alpha)}}{\|\vec{g}\|} = \frac{\|\vec{g}\|\sqrt{1 + c_2^2 - 2c_2\cos\alpha}}{\|\vec{g}\|}$$

$$= \sqrt{1 + c_2^2 - 2c_2\cos\alpha}$$

Y en segundo lugar, el desvío, que denotaremos por λ_2, es igual a la distancia entre la posición futura de la partícula y la recta que une la posición actual con el $gBest$. Esta distancia es igual a la norma del vector que va desde la posición futura $X + c_2\texttt{Gira}\,((gBest - X), \alpha)$ hasta la proyección ortogonal de la misma sobre el vector social $(gBest - X)$. O lo que es lo mismo, es igual a $\|c_2\texttt{Gira}\,((gBest - X), \alpha) - \text{proj}_{(gBest-X)}c_2\texttt{Gira}\,((gBest - X), \alpha)\,\|$. No obstante, necesitamos que la medida del desvío sea genérica, por lo que tomaremos también su valor proporcional a la distancia al $gBest$. Observando la figura 4.10:

$$\lambda_2 \;=\; \frac{\|\|c_2\texttt{Gira}\,((gBest - X), \alpha)\,\|\sin\alpha\|}{\|gBest - X\|} \;=\; \frac{c_2\|gBest - X\|\sin\alpha}{\|gBest - X\|} \;=\; c_2\sin\alpha$$

Gracias a estas nuevas variables λ_1 y λ_2, podemos realizar el ajuste de hiperparámetros de tal modo que satisfaga nuestras preferencias para el nivel de exploración y diversidad del algoritmo metaheurístico. Dado un nivel de exploración concreto, existen múltiples opciones para el nivel de desvío tal y como se representa en la figura 4.11. Es más, dado un nivel de exploración y un nivel de desvío concretos, puede haber dos opciones de configuración de c_2 y α (ver 4.11). En esa situación, tal y como se ha dicho en más de una ocasión durante el análisis, la opción de mayor c_2 resulta en una diversidad mayor. En cualquier caso, de acuerdo a nuestro análisis siempre nos va a interesar escoger la mayor diversidad posible dentro del nivel de exploración que manejemos, ya que hace que la búsqueda sea más efectiva y eficiente. Y por lo estudiado, la mayor diversidad se alcanza con el mayor desvío. Observando 4.11, el mayor desvío únicamente admite una sola configuración paramétrica. Calculémosla.

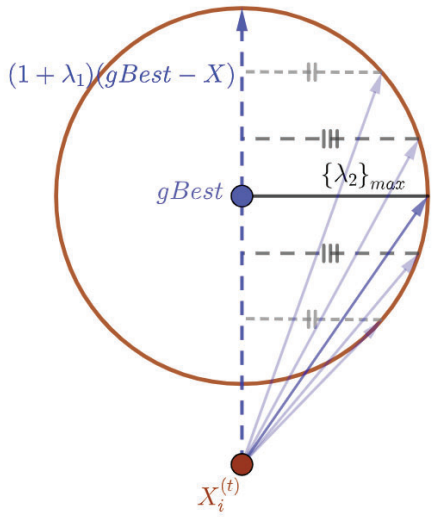


Figura 4.11: Representación gráfica bidimensional de los distintos niveles de desvío λ_2 que existen para un nivel de exploración λ_1 concreto.

Tenemos que maximizar λ_2 sujeta a λ_1. Expresando λ_2 en términos de λ_1:

$$\begin{cases} \lambda_1^2 = 1 + c_2^2 - 2c_2 \cos\alpha \\ \lambda_2 = c_2 \sin\alpha \end{cases} \implies \cos\alpha = \frac{\lambda_1^2 - 1 - c_2^2}{-2c_2} \implies \sin\alpha = \sqrt{1 - \left(\frac{1 + c_2^2 - \lambda_1^2}{2c_2}\right)^2}$$

$$\implies \lambda_2 = c_2\sqrt{1 - \left(\frac{1 + c_2^2 - \lambda_1^2}{2c_2}\right)^2}$$

Y ahora queda resolver un problema de maximización de λ_2, con λ_1 constante:

$$\frac{\mathrm{d}\lambda_2}{\mathrm{d}c_2} = 0 \implies c_2 * \frac{1}{2} * \frac{1}{\sqrt{1 - \left(\frac{1+c_2^2-\lambda_1^2}{2c_2}\right)^2}} * \left(-2\left(\frac{1 + c_2^2 - \lambda_1^2}{2c_2}\right)\left(\frac{2c_2 * 2c_2 - 2(1 + c_2^2 - \lambda_1^2)}{(2c_2)^2}\right)\right)$$

$$+ \sqrt{1 - \left(\frac{1 + c_2^2 - \lambda_1^2}{2c_2}\right)^2} = 0$$

$$\implies -\left(\frac{1 + c_2^2 - \lambda_1^2}{2}\right)\left(\frac{4c_2^2 - 2(1 + c_2^2 - \lambda_1^2)}{4c_2^2}\right) + 1 - \left(\frac{1 + c_2^2 - \lambda_1^2}{2c_2}\right)^2 = 0$$

$$\implies 1 - \left(\frac{1 + c_2^2 - \lambda_1^2}{2c_2}\right)^2 - \left(\frac{1 + c_2^2 - \lambda_1^2}{2}\right)\left(1 - \frac{1 + c_2^2 - \lambda_1^2}{2c_2^2}\right) = 0$$

$$\implies 1 - \frac{(1 + c_2^2 - \lambda_1^2)^2}{4c_2^2} - \left(\frac{1 + c_2^2 - \lambda_1^2}{2}\right) + \frac{(1 + c_2^2 - \lambda_1^2)^2}{4c_2^2} = 0$$

$$\implies 1 - \frac{1 + c_2^2 - \lambda_1^2}{2} = 0 \implies 1 + c_2^2 - \lambda_1^2 = 2 \implies c_2^2 = 1 + \lambda_1^2$$

$$\implies c_2 = \sqrt{1 + \lambda_1^2}$$

Sustituyendo de vuelta el valor de c_2 en la expresión de λ_1 para hallar α:

$$\lambda_1^2 = 1 + \left(\sqrt{1+\lambda_1^2}\right)^2 - 2\sqrt{1+\lambda_1^2}\cos\alpha \implies 1 - \sqrt{1+\lambda_1^2}\cos\alpha = 0 \implies \cos\alpha = \frac{1}{\sqrt{1+\lambda_1^2}}$$
$$\implies \alpha = \arccos\frac{1}{\sqrt{1+\lambda_1^2}} = \arccos\frac{1}{c_2}$$

Realmente, si nos fijamos en 4.11, vemos que el mayor desvío siempre se produce cuando el vector que va desde la nueva posición $X + c_2\texttt{Gira}\,((gBest - X), \alpha)$ al $gBest$, es perpendicular al vector social $(gBest - X)$. Esta observación nos ahorra cálculos, pues nos permite averiguar la nueva posición directamente haciendo

$$X^{(t+1)} = gBest^{(t)} + \lambda_1\vec{r} \quad | \quad (\vec{r} \perp (gBest^{(t)} - X^{(t)}) \ \wedge \ \|\vec{r}\| = \|gBest - X\|) \tag{4.2}$$

Operando como lo haríamos de normal, llegamos al mismo resultado:

$$X + c_2\texttt{Gira}\,((gBest - X), \alpha) \ = \ X + c_2(\vec{r}\sin\alpha + (gBest - X)\cos\alpha) \ =$$
$$= X + c_2\left(\vec{r}\sqrt{1 - \left(\frac{1}{c_2}\right)^2} + (gBest - X)\frac{1}{c_2}\right) \ = \ X + \vec{r}c_2\frac{\sqrt{c_2^2 - 1}}{c_2} + (gBest - X)c_2\frac{1}{c_2} =$$
$$= X + \vec{r}\sqrt{1 + \lambda_1^2 - 1} + gBest - X \ = \ \vec{r}\lambda_1 + gBest$$

Pero es que además, en altas dimensiones (d grandes), dos vectores aleatorios muestreados de forma independiente de acuerdo a una distribución esféricamente simétrica, tienden a ser perpendiculares entre sí. Para demostrarlo, sin pérdida de generalidad podemos fijar uno de ellos como $\vec{e_1} = (1, 0, ..., 0)$ y calcular para un $\vec{u} \sim \mathcal{N}(\vec{0}, I_d)$:

$$\mathbb{E}\left(\cos\angle(\vec{e_1}, \vec{u})\right) \ = \ \mathbb{E}\left(\frac{\langle\vec{e_1}, \vec{u}\rangle}{\|\vec{u}\|}\right) \ = \ \mathbb{E}\left(\frac{u_1}{\|\vec{u}\|}\right) = 0$$

por simetría de $\frac{u_1}{\|\vec{u}\|}$ con respecto al 0 (invarianza de su distribución ante la transformación $\frac{u_1}{\|\vec{u}\|} \mapsto -\frac{u_1}{\|\vec{u}\|}$). Calculando también la varianza:

$$\sigma^2\left(\frac{u_1}{\|\vec{u}\|}\right) \ = \ \mathbb{E}\left(\left(\frac{u_1}{\|\vec{u}\|}\right)^2\right) - \mathbb{E}\left(\frac{u_1}{\|\vec{u}\|}\right)^2 \ = \ \mathbb{E}\left(\frac{u_1^2}{\|\vec{u}\|^2}\right) - 0 \ = \ \mathbb{E}\left(\frac{u_1^2}{\|\vec{u}\|^2}\right)$$

Teniendo en cuenta que todas las variables u_j son independientes e idénticamente distribuidas (i.i.d.), $\frac{u_{j_1}^2}{\|\vec{u}\|^2} \overset{\text{d}}{=} \frac{u_{j_2}^2}{\|\vec{u}\|^2},\ 1 \le j_1 < j_2 \le d$ y

$$\mathbb{E}\left(\frac{u_1^2}{\|\vec{u}\|^2}\right) \ = \ \frac{1}{d}\sum_{j=1}^{d}\mathbb{E}\left(\frac{u_j^2}{\|\vec{u}\|^2}\right) \ = \ \frac{1}{d}\,\mathbb{E}\left(\sum_{j=1}^{d}\frac{u_j^2}{\|\vec{u}\|^2}\right) \ = \ \frac{1}{d}\,\mathbb{E}\left(\frac{\sum_{j=1}^{d}u_j^2}{\|\vec{u}\|^2}\right) \ = \ \frac{1}{d}\,\mathbb{E}\left(\frac{\|\vec{u}\|^2}{\|\vec{u}\|^2}\right)$$
$$= \ \frac{1}{d}\,\mathbb{E}\,(1) \ = \ \frac{1}{d}$$

Así pues:

$$\lim_{d\to+\infty}\mathbb{E}\left(\frac{u_1}{\|\vec{u}\|}\right) = 0 \ \wedge \ \lim_{d\to+\infty}\sigma^2\left(\frac{u_1}{\|\vec{u}\|}\right) = \lim_{d\to+\infty}\frac{1}{d} = 0 \implies \lim_{d\to+\infty}\frac{u_1}{\|\vec{u}\|} = 0$$

lo cual era más que intuible, ya que cuantos más elementos tenga el vector $\vec{u}$, mayor será su norma y por ende, menor la proporción $\frac{u_1}{\|\vec{u}\|}$.

Lo que viene a decirnos este resultado es que efectivamente cuanto mayor es la dimensión d con la que trabajamos, mayor es la probabilidad de que un vector aleatorio uniformemente distribuido $\vec{u} \sim \mathcal{N}(\vec{0}, I_d)$ sea perpendicular a otro cualquiera en $\mathbb{R}^d$. Esta probabilidad se puede expresar usando la inecuación de Chebyshev:

$$\mathbb{P}\left(\frac{u_1}{\|\vec{u}\|} \geq \epsilon\right) \leq \frac{1}{d\epsilon^2}$$

Por supuesto esto significa que en dimensiones elevadas, el vector perpendicular al vector social que generamos en `Gira`$((gBest - X), \alpha)$ $(\vec{r})$ puede ser reemplazado de forma aproximada por el vector aleatorio $\vec{u}$ que empleamos para el proceso, puesto que éste será aproximadamente perpendicular a $(gBest - X)$. De este modo podemos ahorrar aún más en cómputo. Y más, si optamos por nuestra propuesta de desvíos máximos, de acuerdo a la expresión 4.2 nos quedaría:

$$X^{(t+1)} = gBest^{(t)} + \lambda_1 \frac{\vec{u}}{\|\vec{u}\|}\|gBest - X\| \quad | \quad \vec{u} \sim \mathcal{N}(\vec{0}, I_d) \tag{4.3}$$

Nótese que esta última ecuación es la de muestreo uniforme en la superficie de una hiperesfera de radio $\lambda_1\|gBest - X\|$. Es decir, tomar muestras aleatorias distribuidas uniformemente en la superficie de una hiperesfera implica, en dimensiones elevadas, que las partículas se desvíen lo máximo posible dentro de su nivel de exploración λ_1. Esto se ve claro si calculamos λ_2 en este algoritmo de muestreo uniforme en una $(d-1)$-esfera:

$$\begin{aligned}
\lambda_2 \;&=\; \frac{\|\vec{a}\|\sin\alpha}{\|\vec{g}\|} \;=\; \frac{\|\vec{a}\|\sqrt{1 - \left(\frac{\langle \vec{a}, \vec{g}\rangle}{\|\vec{a}\|\|\vec{g}\|}\right)^2}}{\|\vec{g}\|} \;=\; \frac{\sqrt{\|\vec{a}\|^2\|\vec{g}\|^2 - \left(\sum_{j=1}^d a_j g_j\right)^2}}{\|\vec{g}\|^2} \\
&= \frac{\sqrt{\left\|\vec{g} + \lambda_1 \frac{\vec{u}}{\|\vec{u}\|}\|\vec{g}\|\right\|^2\|\vec{g}\|^2 - \left(\sum_{j=1}^d \left(g_j + \lambda_1 \frac{u_j}{\|\vec{u}\|}^{0}\|\vec{g}\|\right) g_j\right)^2}}{\|\vec{g}\|^2} \\
&\longrightarrow \frac{\sqrt{\left(\|\vec{g}\|^2 + \lambda_1^2\|\vec{g}\|^2 \sum_{j=1}^d \frac{u_j^2}{\|\vec{u}\|^2} + \sum_{j=1}^d 2\lambda_1\|\vec{g}\| \frac{u_j}{\|\vec{u}\|}^{0} g_j\right)\|\vec{g}\|^2 - (\|\vec{g}\|^2)^2}}{\|\vec{g}\|^2} \\
&\longrightarrow \frac{\sqrt{\|\vec{g}\|^4 + \lambda_1^2\|\vec{g}\|^4 - \|\vec{g}\|^4}}{\|\vec{g}\|^2} \;=\; \frac{\lambda_1\|\vec{g}\|^2}{\|\vec{g}\|^2} \;=\; \lambda_1 \;=\; \{\lambda_2\}_{max}
\end{aligned}$$

El algoritmo es un buen aproximador de nuestro método recomendado de máximos desvíos. Para un número de dimensiones suficientemente grande como el que estamos tratando en este análisis ($d = 30$), las diferencias entre el método exacto y éste son prácticamente inapreciables.

La conclusión de este análisis es que el hiperparámetro c_2 es el responsable tanto del nivel de exploración alrededor del objetivo ($gBest$) como de la diversidad del PSO. En la versión original del método, este hiperparámetro viene multiplicado por elementos

aleatorios, que por tanto introducen aleatoriedad en estas dos características (nivel de exploración y diversidad). Durante el análisis hemos probado que esto no es necesario ni positivo y que es mejor que sea el usuario el que decida en todo momento el nivel de exploración y diversidad en función del problema de optimización a tratar. Además, hemos deducido que lo mejor es optar por una diversidad máxima dentro de las posibilidades del nivel de exploración que decidamos. En este sentido hemos modificado el método para hacerlo posible. Así tan solo nos debemos de preocupar por seleccionar bien el balance adecuado de exploracion/explotación para nuestro problema.

La explicación detrás de nuestra propuesta es la siguiente. El objetivo es hallar el óptimo de una función con la mayor precisión posible. Para lograrlo, es necesario converger a un ritmo lo suficientemente alto como para ir mejorando en precisión (reduciendo escalas del espacio de búsqueda), esto es, explotar. Sin embargo, al querer realizar la explotación demasiado rápido existe el riesgo evidente de que las partículas no den con la región apropiada durante el proceso y por tanto se converja a una región no óptima, indeseada. Podemos regular esta tasa de convergencia mediante λ_1: distancia proporcional al $gBest$ que han de recorrer las partículas. Y nótese que si conseguimos obtener un proceso de convergencia en el que las partículas se distribuyen de manera eficiente por el espacio, abarcando la mayor parte de éste y recorriendo regiones distintas todo el tiempo (mayor diversidad), entonces tendremos sin lugar a dudas una probabilidad superior de encontrar regiones prometedoras (mejor exploración). Con una mejor disposición de las partículas en el espacio, nos podemos permitir reducir λ_1 y converger más rápido (aumentar precisión más rápido), ya que en ese caso, el riesgo de no encontrar la región más prometedora en cada ocasión disminuye y por tanto podemos acelerar el proceso. Esta mejor disposición de la que hablamos se consigue gracias al desvío. El desvío hace que las partículas cambien de región constantemente, visitando durante el proceso nuevas zonas del espacio y haciendo que el enjambre en conjunto cubra mejor el entorno del punto de referencia al que converger ($gBest$ en este caso). Sin desvíos las partículas están peor repartidas alrededor del objetivo y la búsqueda empeora. El objetivo digamos que es el de reducir escalas de búsqueda lo antes posible sin dejar pasar las mejores regiones y la conclusión es que un buen desvío nos lo permite. Los desvíos grandes nos permiten reducir λ_1 y con ello aumentar el ritmo de convergencia (explotar más), logrando un resultado muy preciso. Y esto es lo que explica que queramos siempre maximizar el desvío.

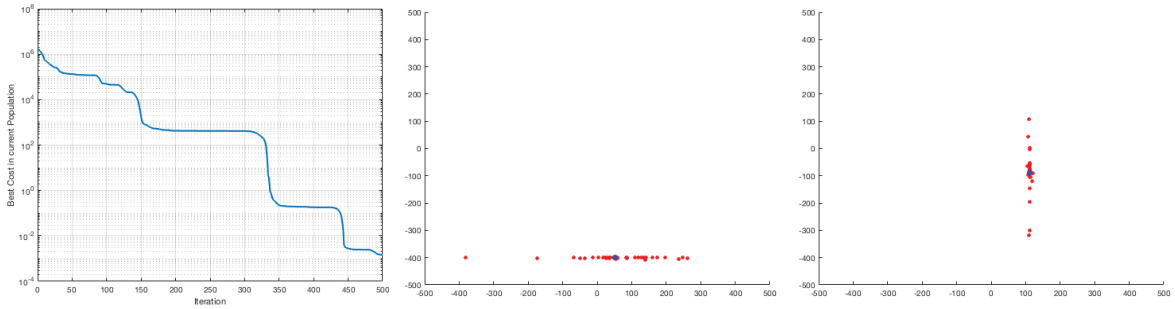
4.2.3.5. Diferencias con el sistema de giro original

El sistema de giro propuesto tiene la ventaja de adaptarse en cada momento a los requisitos del usuario. El usuario puede seleccionar el ángulo de giro de cada movimiento si así lo quiere. Por tanto, es un sistema de giro más genérico que el original del PSO clásico. No obstante, si el usuario no interviene activamente decidiendo cada giro en función de la posición de la partícula, el sistema de por sí es independiente del sistema de referencia, tal y como hemos demostrado recientemente. El original en cambio ya sabemos que no lo es. Luego por mucho que pretendamos replicar el sistema original a través del propuesto (en teoría se puede porque es más genérico), no lo conseguiremos a menos que tengamos muy en cuenta este hecho y gestionemos cada giro basándonos en la posición de cada partícula con respecto al $gBest$ y de la misma forma que lo hace el original.

Como bien sabemos, la forma de hacerlo del original es otorgando mayor probabilidad de giros cortos cuanto más parecido sea el vector social a alguno de los ejes coordenados. Es decir, de media cuanto menor sea el ángulo del vector social con cualquiera de los ejes coordenados, menor será su giro. En otras palabras, los giros decrecen con el ángulo entre el vector social y el eje de coordenadas más próximo, hasta llegar el punto en el que si son completamente paralelos, entonces la probabilidad de girar es exactamente 0. Esta propiedad del sistema de giro original claramente incita a que las partículas se queden atrapadas en los ejes paralelos a los del sistema de coordenadas con origen en el $gBest$. Provoca que las partículas busquen en ejes como lo harían sin desvíos, aunque con la particularidad de que dichos ejes sean paralelos a los ejes coordenados. Esto conlleva que las partículas exploren y converjan por dimensiones. Lógicamente favorece la optimización de funciones separables. Todo esto ya había sido expuesto en la literatura y contado durante nuestra review en 3.4.4.

Pero lo que no se ha dicho en la literatura es que las dimensiones que se exploran no son casuales, sino que dependen del estado en que se encuentra el algoritmo. Cuando una componente del vector social destaca (es muy grande con respecto al resto), es cuando la partícula se queda atrapada en esa dimensión. Y que destaque no es casual, al final depende mucho de cómo va cambiando el $gBest$. Si el cambio se da más en una dimensión que en otras, entonces esa dimensión acabará siendo la que más atrape las partículas. Y en definitiva, todas o casi todas las partículas se quedarán atrapadas en una misma dimensión. Esa dimensión va cambiando durante la ejecución ya que en realidad la convergencia exacta del enjambre en una dimensión no existe. Podríamos quedarnos pues con que la búsqueda se realiza por dimensiones pero de forma conjunta como enjambre. Es decir, no es que cada partícula busque individualmente en alguna dimensión, sino que practicamente todas en conjunto buscan en la misma dimensión. Así se va alternando la búsqueda por dimensiones diferentes, pero siempre por cada una, es todo el enjambre el que participa en ella. Y la alternancia entre dimensiones tiene que ver con el progreso de la optimización. Cuando el $gBest$ deja de mejorar, las partículas cambian la dimensión en la que buscar por aquella que ofrezca mayor progreso.

La figura 4.12 es la demostración de esta convergencia secuencial por dimensiones. En la figura 4.12a se aprecia cómo el progreso de la optimización no es lineal sino que hay 'baches' en los cuales no hay mejoras o las que hay no siguen el mismo ritmo que en el resto de la ejecución. El algoritmo se estanca en esas etapas 'bache'. Y el estancamiento coincide con la ausencia de mejoras en la búsqueda de una dimensión y por tanto, con la convergencia del enjambre en esa dimensión y su transición hacia la exploración de una nueva con mayor margen de mejora. De este modo, cuando el algoritmo cambia la dimensión de búsqueda, las mejoras que se producen son muy rápidas hasta que poco a poco se van estabilizando. El progreso no es constante porque todas las partículas están volcadas en la misma tarea. Visualmente las figuras 4.12b y 4.12c ayudan a entender esta circunstancia que comentamos. La situación de la figura 4.12b sucede antes en el tiempo que la de 4.12c. Es observando ambas cuando se aprecia la transición en la búsqueda de dimensiones y se ve cómo funciona el algoritmo.



(a) Gráfica $f(gBest^{(t)})$ (b) $X_i^{(t_1)} \quad \forall i$ (c) $X_i^{(t_2)}$, $t_2 > t_1 \quad \forall i$

Figura 4.12: Ilustración de la exploración por dimensiones del PSO con $V = c_2\|gBest - X\|\frac{Rand_2 \circ (gBest - X)}{\|Rand_2 \circ (gBest - X)\|}$ para $c_2 = 1.5$. En (a) se muestra la gráfica de progreso de la optimización, que tiene forma de escalera debido a la exploración asíncrona por dimensiones que impide que las mejoras relativas sean constantes. Tanto en (b) como en (c) se representa la situación de las partículas del enjambre (en rojo) alrededor del objetivo $gBest$ (en azul) en dos puntos temporales diferentes (primero (b) y después (c)).

Esta manera de explorar solamente tiene sentido si estamos optimizando funciones separables. Pero, dado el caso, tampoco es la mejor manera de hacerlo. Y no lo es por dos motivos principales. El primero, porque realmente es una aproximación no exacta a la búsqueda por dimensiones y además indirecta, que no se da de primeras ni se tiene por qué mantener permanentemente en el tiempo. De hecho la idea original del método no era esta, es bastante casual. Y en segundo lugar, el hecho de que todas las partículas se dediquen a lo mismo, a explorar la misma dimensión todas o casi todas ellas, de algún modo es desaprovecharlas. Es desaprovechar esa capacidad de paralelización que pueden aportar las partículas porque al tener todas la misma tarea, su utilidad individual decae. Si cada una de ellas tuviera una tarea diferente como la de explorar una dimensión distinta cada una, entonces se ahorraría tiempo ya que las partículas pasarían a ser más útiles y necesarias y podrían adelantar el trabajo que iban a tener que hacer sí o sí de explorar todas las dimensiones del espacio. Puede que este sea uno de los motivos por los que la selección del hiperparámetro N permita tanta flexibilidad y no existan grandes diferencias entre un valor bajo de N y uno alto (el número de partículas no es demasiado relevante en el PSO estándar), tal y como estudiamos en 3.3.3.

La solución a estos defectos pasa por diseñar un método de búsqueda exacta por dimensiones que explore simultáneamente a través de sus partículas las diferentes dimensiones del espacio. Así, deben ser las partículas individualmente las que decidan en cada iteración la dimensión en la que buscar. Puede ser aleatoriamente o puede quizá diseñarse alguna heurística más sofisticada que les guíe en su decisión. Aquí vamos a proponer un método PSO para funciones separables en el que las partículas seleccionan aleatoriamente la dimensión en la cual moverse alrededor del $gBest$, basándonos en el concepto de distancia proporcional de las partículas al $gBest$ para mantener una tasa de convergencia (nivel de exploración) deseada, tal y como insistimos en la versión general del método. La idea es que cada partícula pueda caer en cualquier eje paralelo al coordenado respetando la distancia proporcional estipulada λ_1. Esto es, por cada partícula se debe seleccionar aleatoriamente un eje de coordenadas y un sentido de movimiento (todos parten con igual probabilidad) y automáticamente su posición quedará determinada por el único punto que cumple

con la distancia proporcional al $gBest$ (λ_1) y se encuentra en el eje y lado (respecto al $gBest$) seleccionados. De esta forma el progreso será constante porque todas las dimensiones se trabajan a la vez y no se avanza a ritmos diferentes como se hacía antes (4.12a). Además, trasladamos la filosofía de nuestro análisis a este método e imponemos que el usuario decida en todo momento su nivel de exploración (eliminamos el azar por defecto). Los detalles del nuevo PSO para funciones separables (PSO-separable por abreviar) se presentan en el siguiente bloque de pseudocódigo:

Algoritmo 5: Método propuesto de PSO por dimensiones separadas

para *cada partícula* $i = 1, ..., N$ **hacer**
  Inicializar la posición X_i de la partícula dentro de los límites: $x_{i,j} \in [l_j, u_j]$
  si $f(X_i) < f(gBest)$ **entonces**
    Actualizar la mejor posición colectiva: $gBest = X_i$
  fin
  Inicializar la velocidad V_i de la partícula
fin
mientras *No se cumpla el criterio de terminación* **hacer**
  $gBest^{(t+1)} = gBest^{(t)}$
  para *cada partícula* $i = 1, ..., N$ **hacer**
    Seleccionar aleatoriamente una dimensión $k \sim \text{Uniforme}(1, \ldots, d)$ y un signo $s \sim \text{Uniforme}(1, -1)$
    Construir vector $\vec{r}$ = k-ésimo vector canónico: $\vec{r} = (0, ..., 1, ..., 0) \mid \ r_k = 1 \ \wedge \ r_{j \neq k} = 0$
    Hacer $X_i = gBest^{(t)} + s\vec{r}\|gBest^{(t)} - X_i\| * \lambda_1$
    Comprobar y gestionar si es necesario la factibilidad de las posiciones X_i
    $X_i' = \vec{r} \circ X_i + (1 - \vec{r}) \circ gBest^{(t+1)}$
    si $f(X_i') < f(gBest^{(t+1)})$ **entonces**
      Actualizar la futura mejor posición colectiva: $gBest^{(t+1)} = X_i'$
    fin
  fin
fin

El algoritmo también puede modificarse para que en lugar de seleccionar aleatoriamente las dimensiones de búsqueda de cada partícula, se vayan recorriendo todas equitativamente, siendo cada partícula asignada a una dimensión concreta y cada dimensión cubierta por grupos de igual tamaño. Realmente en esencia este último enfoque es equivalente al que vimos en la literatura de [118] (2). Todos estos métodos PSO se basan en explorar por dimensiones teniendo como punto de referencia al $gBest$. Es decir, parten del $gBest$ y solo buscan en los ejes paralelos a los de referencia. Las posiciones de búsqueda solamente se diferencian en una componente del $gBest$. En este sentido son fieles a la manera de funcionar del PSO estándar que recién hemos explicado. Pero la diferencia está en que, aparte de que la búsqueda por dimensiones es exacta y no aproximada, las búsquedas por las distintas dimensiones son simultáneas, ocurren a la vez. Más adelante tendremos la oportunidad de com-

parar nuestro método 5 con el de [118] (2). Aparte de la manera en que seleccionan las dimensiones cada uno de los dos métodos, también se distinguen por la regla de actualización de posiciones que emplean. El nuestro, basándose en los resultados del presente análisis, opta por una actualización determinista en la que es el usuario el que decide en todo momento el nivel de exploración del algoritmo.

4.3. Componente cognitiva

La componente cognitiva de la fórmula de actualización del PSO es $c_1 Rand_1 \circ (pBest - X)$. Tiene la misma estructura que la componente social, solo que en vez de tener como atractor a la mejor posición histórica global (*gBest*), su atractor es el $pBest$, que es la mejor posición individual. Pero básicamente el análisis de esta componente es el mismo que el que hemos hecho de la social. Lo único que cambia es que la regla de movimiento compuesta únicamente por la componente cognitiva guía a las partículas a explotar la región del $pBest$ y no la del $gBest$ como ocurría antes con la regla social (la formada exclusivamente por la componente social). Con lo cual el análisis es exactamente el mismo solo que intercambiando los papeles de $gBest$ por $pBest$ y c_2 por c_1. Así, la exploración de las partículas en torno a estos dos puntos es idéntica. Desde el punto de vista individual de las partículas, sus búsquedas son locales en torno al $pBest$. Y desde el punto de vista global se tiene más exploración ahora, ya que simultáneamente se contemplan más regiones distintas al centrarse cada partícula en una región diferente, independiente de las del resto de partículas. También hay que decir que la componente cognitiva por sí sola, aislada del resto de componentes, no es funcional, puesto que a nada que las partículas encuentran una posición histórica mejor que la que tenían (mejoren su $pBest$), se quedan atrapadas en esa posición sin poder escapar de ella para lo que resta de ejecución.

4.4. Integración de las componentes social y cognitiva

Las dos componentes principales del PSO, en las cuales reside más la heurística, son la social y la cognitiva. Ya las hemos analizado por separado y es momento de integrarlas y analizar el método con ellas dos presentes. La diferencia es que ya no tenemos que converger a un punto de atracción, sea $gBest$ o sea $pBest$, sino que ahora lo hacemos a un punto influenciado por ambos. Es decir, las partículas ya no se mueven en torno a uno de esos puntos de referencia, ahora se mueven por regiones intermedias dependientes de la influencia de cada punto en las partículas. En el PSO original esta influencia está determinada por la proporción entre c_1 y c_2. La proporción es la que marca si queremos sentirnos más atraídos por el $gBest$ o por el $pBest$ (dar más importancia a explotación global o local). Un c_2 grande en proporción a c_1 hace que las partículas se muevan por regiones más cercanas al $gBest$ que al $pBest$ y por tanto, que se explote más (partículas concentradas en regiones parecidas) que si c_1 es mayor que c_2 y las partículas llevan a cabo búsquedas más independientes (más exploración). Luego los hiperparámetros c_1 y c_2 no solo regulan el nivel de exploración que se realiza alrededor de los atractores sino que también deciden la importancia de cada uno de ellos en el movimiento de las partículas. Esto se ve claro si recordamos

la expresión 3.3 para el PSO determinista dada por Clerc [43]:

$$V = V + (c_1 + c_2)\left(\underbrace{\frac{c_1 pBest + c_2 gBest}{c_1 + c_2}}_{P} - X\right)$$

donde P era el punto de convergencia, que claramente depende de la media ponderada de los dos atractores $gBest$ y $pBest$, cuyos pesos vienen dados por los hiperparámetros c_1 y c_2. Lo cual quiere decir que efectivamente estos hiperparámetros regulan la influencia de cada atractor en el movimiento de las partículas. También la suma de ellos ($c_1 + c_2$) marca la exploración alrededor de P. Realmente la expresión también puede verse desde otra perspectiva equivalente:

$$V = V + \frac{c_1}{c_1 + c_2}(c_1 + c_2)(pBest - X) + \frac{c_2}{c_1 + c_2}(c_1 + c_2)(gBest - X)$$

donde el punto de atracción ya no es P sino que seguimos teniendo los atractores clásicos $pBest$ y $gBest$, en torno a los cuales se muestrea un punto, fruto de la atracción de cada uno de ellos. El movimiento final viene dado por la ponderación de estos dos puntos. El enfoque anterior exploraba sobre la ponderación de los atractores clásicos y éste, pondera las exploraciones sobre los mismos.

Sin embargo esas expresiones son en ausencia de aleatoriedad. En la versión estocástica, ya no tenemos como tal un punto de atracción fijo P sobre el que explotar. Ahora P' es aleatorio:

$$V = V + \left(c_1 \frac{\|Rand_1 \circ \vec{p}\|}{\|\vec{p}\|} + c_2 \frac{\|Rand_2 \circ \vec{g}\|}{\|\vec{g}\|}\right) * \tag{4.4}$$

$$* \left(\underbrace{\frac{c_1 \frac{\|Rand_1 \circ \vec{p}\|}{\|\vec{p}\|} \frac{Rand_1 \circ \vec{p}}{\|Rand_1 \circ \vec{p}\|} \|\vec{p}\| + c_2 \frac{\|Rand_2 \circ \vec{g}\|}{\|\vec{g}\|} \frac{Rand_2 \circ \vec{g}}{\|Rand_2 \circ \vec{g}\|} \|\vec{g}\|}{c_1 \frac{\|Rand_1 \circ \vec{p}\|}{\|\vec{p}\|} + c_2 \frac{\|Rand_2 \circ \vec{g}\|}{\|\vec{g}\|}} + X}_{P'} - X\right) \tag{4.5}$$

Como $\mathbb{E}(Rand_1) = \mathbb{E}(Rand_2) = (0.5, ..., 0.5) = 0.5^d$, entonces $\mathbb{E}(P') = \frac{c_1 pBest + c_2 gBest}{c_1 + c_2} = P$, tal y como ya sabíamos por los análisis de convergencia. Es decir, el atractor esperado es el mismo que el de la versión determinista del método, que es el punto al que acaban convergiendo las posiciones esperadas de las partículas (3.5). Al final sí podemos decir que las partículas exploran en torno a P, pero se mueven guiadas por un atractor cambiante y aleatorio P'. Y debido al escalamiento direccional del PSO, este atractor no es la ponderación de los clásicos $pBest$ y $gBest$, sino la de sus correspondientes puntos asociados al giro $\frac{Rand_1 \circ (pBest - X)}{\|Rand_1 \circ (pBest - X)\|} \|pBest - X\|$ y $\frac{Rand_2 \circ (gBest - X)}{\|Rand_2 \circ (gBest - X)\|} \|gBest - X\|$. Esto nos obliga a adoptar la segunda perspectiva, en la que la exploración se realiza para cada atractor clásico ($pBest$ y $gBest$) y después el resultado de ambas exploraciones (el movimiento asociado a cada una) se pondera.

Sea como fuere, en el PSO clásico, los hiperparámetros c_1 y c_2 determinan tanto las exploraciones sobre los $pBest$ y $gBest$ (nivel de exploración, desvíos, etc) como el grado de influencia de cada una de ellas (de cada exploración) en el movimiento final de las partículas. La influencia la determinan a través de sus valores relativos (proporciones entre c_1 y c_2) y la exploración a través de sus magnitudes en sí. Quizá sería más fácil y práctico tener ambas funcionalidades separadas y contar con diferentes parámetros

para cada una. Así por un lado tendríamos los parámetros propios de la exploración sobre cada atractor $pBest$ y $gBest$ y por otro, un parámetro que mida la importancia de cada exploración (de cada muestra en torno al $gBest$ y el $pBest$). Para esto último podemos diseñar un parámetro, que denotaremos por β, que tome un valor entre 0 y 1 ($\beta \in [0,1]$) y que signifique el grado de importancia relativa que se le da a la explotación local (explotación del $pBest$) en comparación con la global (explotación del $gBest$). Un β alto (> 0.5) querrá decir que se le da más importancia al $pBest$ y a la explotación local que al $gBest$ y a la global, lo cual implica una mayor exploración que si fuera al revés. Para la parte de exploración sobre cada atractor, podemos aplicar los conocimientos extraídos durante el análisis. El método compuesto por las dos componentes social y cognitiva quedaría:

$$V = \beta c_1 \texttt{Gira}\,((pBest - X), \alpha_1) + (1-\beta) c_2 \texttt{Gira}\,((gBest - X), \alpha_2) \tag{4.6}$$

con c_1, α_1 y c_2, α_2 escogidos en función del nivel de exploración y de desvío λ_1, λ_2 deseados. La figura 4.13 muestra el movimiento de una partícula bajo este método.

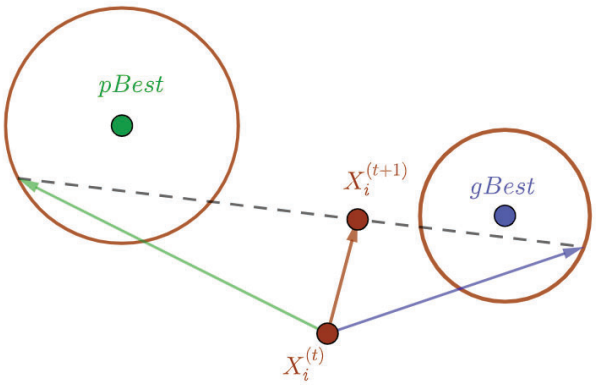


Figura 4.13: Representación gráfica bidimensional del PSO con $V = \beta c_1 \texttt{Gira}\,((pBest - X), \alpha_1) + (1-\beta) c_2 \texttt{Gira}\,((gBest - X), \alpha_2)$ y $\beta = 0.6$.

Viendo la expresión 4.4, la influencia de cada atractor es también aleatoria en el PSO clásico. Sin embargo, nosotros tras algunas pruebas hemos decidido que β sea determinista, también siguiendo la línea de este análisis de reducir al máximo la aleatoriedad para dotar de más control al usuario. Además como apunte, cuando $pBest = X$, la única atracción que existe es hacia el $gBest$, por lo que no tiene sentido ponderar atracciones en este caso. Y recordamos que esta situación se da bastante a menudo durante la ejecución si la optimización va bien. Así que en estas situaciones estableceremos β a 0.

Con la combinación de las componentes social y cognitiva cabe deducir que se reduce el efecto del sesgo hacia direcciones paralelas a los ejes coordenados que teníamos antes cuando solo explorábamos sobre una de ellas, ya que ahora a menos que el $pBest$ y el $gBest$ se encuentren alineados sobre un mismo eje paralelo a uno de coordenadas, las partículas no se pueden quedar atrapadas en ellos al ser atraídas hacia direcciones diferentes. No obstante, el sesgo sigue existiendo, más si las partículas mejoran sus $pBest$ con facilidad.

4.5. Componente inercial

Tan solo queda por reintegrar al método la componente inercial wV. Esta componente a priori no aporta información heurística al método, ya que la información acerca de la velocidad anterior de las partículas no tiene por qué revelar direcciones de búsqueda prometedoras. Es más bien una componente que agrega cierta aleatoriedad al integrar un movimiento no estratégico, sin fundamento heurístico aparente (no promete una guía inteligente de búsqueda) [6] al resto de movimientos de atracción cognitiva y social que sí son responsables de orientar la búsqueda hacia regiones prometedoras. Podríamos decir pues que es una componente puramente explorativa, como de hecho ya sugerían los análisis parámetricos que repasamos en 3.3.1. Sin embargo, esto tampoco es del todo cierto. Si observamos la gráfica 3.2 sobre la estabilidad orden-2 del PSO estándar, vemos como no siempre que incrementamos w para un valor determinado de los parámetros c_1 y c_2 el nivel de exploración aumenta. Se ve como para valores pequeños de w, un aumento del parámetro (y por tanto un aumento en la influencia de la componente inercial) significa tasas de convergencia más altas y más explotación, hasta que llega un punto en que ocurre lo contrario y el algoritmo empieza a crecer en exploración con el aumento de w. Como decimos, este comportamiento depende de cada configuración de c_1 y c_2, pero nos muestra que la asociación de inercia con exploración no siempre es acertada. En algunos casos la presencia de esta componente añadirá explotación y en otros, exploración. En cualquier caso, más allá de ser otro elemento regulador del balance exploración/explotación del método, no parece que su funcionalidad pueda ir mucho más lejos. Y nosotros, tras las modificaciones que hemos incluido, no solo hemos procurado que el balance explorativo esté en teoría bien cubierto, sino que además nos hemos preocupado mucho por su calidad (calidad de la exploración, diversidad) revisando el escalamiento direccional de los vectores social y cognitivo. Luego por tanto, si nuestra lectura sobre el rol de esta componente es correcta y no es más que un elemento regulador de la exploración, entonces no tiene sentido incorporarla en el método. Nuestro método PSO carecerá pues de componente inercial.

En el PSO estándar, el papel que juega la componente inercial es esencial, como así lo indican muchos de los trabajos estudiados durante la review paramétrica de 3.3.1. Y pensando el motivo por el que podría necesitar tanto el método de esta componente cuando realmente puede regular su exploración a través de los hiperparámetros c_1 y c_2, nos damos cuenta de una cosa. Cuando $pBest = X$, la componente cognitiva queda anulada y la única atracción de la partícula es hacia el $gBest$ mediante la componente social. Pero claro, la exploración que se realiza cuando las dos componentes participan no es la misma que cuando solamente participa una de ellas. La configuración adecuada en cada caso cambia y el c_2 escogido para el primer caso no sirve para el segundo. Es así que habría que modificar su valor en estas situaciones, de forma parecida a como hicimos en la sección anterior estableciendo un $\beta = 0$ para que $(1 - \beta)c_2$, que hace las veces de c_2 en el PSO estándar, se adapte al caso en que $pBest = X$ y el atractor solo sea $gBest$. Pero a falta de esta regulación extra en el PSO estándar, es la componente inercial la que puede aportar esta exploración

[6] Ciertamente convendría profundizar en el análisis para afirmar con rotundidad este hecho, pues sí que es probable que la definición recursiva de la velocidad tenga relevancia para escapar con más celeridad de una región de búsqueda a otra por la capacidad de poder acumular velocidad de iteraciones anteriores, útil más que nada en situaciones de mejoras permanentes de los $pBest$, indicativos de la necesidad de cambios bruscos de región de búsqueda.

necesaria en esos casos. Claro que, para el resto de casos, esta componente no solo sería innecesaria sino que aportaría desequilibrio en el balance exploración/explotación del método (generalmente aportaría sobreexploración). Sería conveniente probar pues una versión del PSO estándar sin componente inercial pero con regulación de c_2 (incremento) para los casos en que $pBest = X$. Que esta versión funcionara igual o mejor que la original, nos daría la razón en cuanto al papel de la componente inercial y ratificaría nuestra decisión de prescindir de ella. Esta prueba la realizaremos próximamente en el capítulo experimental.

4.6. Método final

Una vez analizados todos los componentes de la fórmula de actualización del PSO tanto por separado como de forma conjunta, podemos por fin terminar de concretar nuestro método PSO extraído del análisis. Su fórmula de actualización sería la de la ecuación (4.6) pero estableciendo los valores de c_1, α_1 y c_2, α_2 para que los desvíos $\{\lambda_2\}_1$ y $\{\lambda_2\}_2$ siempre sean máximos dentro de los niveles de exploración $\{\lambda_1\}_1$ y $\{\lambda_1\}_2$ deseados. Tal y como advertimos en 4.2.3.4, el método de máximos desvíos se puede simplificar mediante la ecuación (4.2), por lo que la fórmula final puede convertirse en:

$$X = \beta(pBest + \{\lambda_1\}_1\vec{r_1}) + (1-\beta)(gBest + \{\lambda_1\}_2\vec{r_2}) \tag{4.7}$$

Por supuesto también hemos de tener en cuenta lo que comentamos de que β ha de ser dependiente de si $pBest = X$ o no, siendo 0 en el primer caso. En resumen, la nueva posición de la partícula se decide de la manera en que se ilustra en la figura 4.13. La filosofía es la de ponderar dos atracciones guiadas por cada punto de referencia $pBest$ y $gBest$, siendo cada atracción parametrizada en función del nivel de exploración que se quiera en torno a cada punto. En realidad lo más adecuado sería establecer los dos niveles de exploración iguales $\{\lambda_1\}_1 = \{\lambda_1\}_2$.

Nuestro método posee tres hiperparámetros: $\{\lambda_1\}_1, \{\lambda_1\}_2$ y β. Es el mismo número que el del PSO original con w, c_1 y c_2, pero a diferencia de éste, en el nuestro se conoce el rol de cada uno porque han sido diseñados específicamente de acuerdo a su rol. Así es más fácil la tarea de selección de hiperparámetros, el diseño de nuevos métodos o estrategias derivadas como las adaptativas, etc. Ahora está claro el efecto que tiene el incrementar o decrementar alguno de los parámetros, su impacto en el balance de exploración/explotación que se pretende conseguir. En concreto, los tres hiperparámetros tienen una relación de proporcionalidad directa con el grado de exploración del algoritmo. Cuanto más grandes son, mayor es la exploración que se logra. Además, los tres están restringidos a un dominio útil en el cual rinden bien. El parámetro β está restringido al intervalo $[0, 1]$ por definición, ya que representa una proporción y sería inválido que tomara valores fuera de dicho intervalo. Y los otros dos parámetros $\{\lambda_1\}_1$ y $\{\lambda_1\}_2$, en realidad pueden tomar cualquier valor en $[0, \infty)$, pero si toman valores mayores que 1, la convergencia no está asegurada (el algoritmo podría desconverger) y eso no nos interesa, sobre todo si estos valores se mantienen constantes durante la ejecución. Y de todos modos, las ventajas de un algoritmo que converge y desconverge durante la ejecución no son claras como pudimos de hecho discutir en 4.2.2. Así pues, el dominio adecuado para estos dos parámetros también es $[0, 1]$, aunque siendo aún más minuciosos, tras varias pruebas podemos llegar a la conclusión de que su ventana útil es $[0.6, 1]$. Es decir, cuando sus valores son menores que 0.6, el método no rinde nunca o casi nunca bien. De todos modos, el hecho de que

los tres hiperparámetros del método tengan un dominio acotado y con una relación directa entre sus valores y el grado de exploración, nos sugiere que podríamos ir incluso un paso más allá y agruparlos para así acabar con un solo hiperparámetro que regule el grado de exploración total del algoritmo. A este nuevo hiperparámetro lo llamaremos `exploration_level` y la idea es que pueda tomar valores en $[0, 1]$ que representen el grado de exploración del algoritmo, siendo 0 el grado mínimo y 1 el máximo. Su rango es correspondido linealmente con los rangos útiles de cada uno de los tres hiperparámetros del método:

$$\begin{cases} \{\lambda_1\}_1 = \{\lambda_1\}_2 = 0.6 + 0.4 * \texttt{exploration\_level} \\ \beta = \texttt{exploration\_level} \end{cases}$$

Hemos logrado reducir el número de hiperparámetros a tan solo uno, que es el mínimo para que el usuario pueda decidir el grado de exploración del algoritmo en función del problema a tratar en cada momento. Esto ha sido gracias a que el análisis que hemos llevado a cabo sobre el PSO nos ha desvelado sus mecanismos de funcionamiento y nos ha permitido sustituir los parámetros originales por otros cuyos roles eran mucho más claros y comprensibles. A raíz de esto nos ha sido posible crear un parámetro central encargado de configurar estos parámetros del método para que se adapten a nuestros requisitos. La demostración de que efectivamente nuestro método está bien diseñado se puede conseguir probando experimentalmente que a mayor valor de la variable `exploration_level`, se alcance un mayor grado de exploración en el algoritmo. Se tendría que dar una correspondencia lineal entre el grado de exploración y el valor de la variable `exploration_level` para que podamos afirmar la calidad de nuestra propuesta.

El nuevo método definitivo queda descrito en el bloque de pseudocódigo del Algoritmo 6. El método es un modelo del que se pueden hacer muchas modificaciones como por ejemplo hacer que tenga convergencia local garantizada siguiendo la propuesta de [61] de mutar la mejor partícula del enjambre, que en nuestro método siempre se encuentra estancada, sin moverse. Simplemente es un modelo sustitutivo del estándar básico. Aunque a diferencia de él, hay que decir que es un método completamente marco-invariante que funciona igual para cualquier tipo de función y en cualquier sistema de referencia escogido. El estándar ya sabemos que muestra un claro favoritismo por las funciones separables. Así que por un lado se espera que nuestro método mejore al estándar en la optimización de funciones no separables y por otro, que no lo haga con funciones separables, a pesar de que su rendimiento en ambos casos sea el mismo. De todos modos, también hemos venido destacando que el PSO estándar tampoco es el mejor método para optimizar funciones separables y que había propuestas de variantes PSO más efectivas para esta labor específica. De hecho nosotros hemos propuesto una (5). Es así que podríamos plantearnos un método híbrido PSO en el que parte de las partículas del enjambre siguiesen el método 5 y la otra parte, el método 6. De esta manera siempre se conseguiría superar al PSO estándar, ya que el método híbrido estaría preparado para destacar tanto en funciones separables como en no separables.

Por último cabe recordar que existe la posibilidad de adoptar una versión aproximada del algoritmo 6 cuando d es lo suficientemente grande ($\frac{1}{d}$ cercano a 0). Esta versión ahorra coste computacional y consistiría en utilizar la expresión (4.3) en lugar de (4.2) para la exploración alrededor de un atractor. Esto supondría reemplazar

Algoritmo 6: Método propuesto de PSO

para *cada partícula* $i = 1, ..., N$ **hacer**
  Inicializar la posición X_i de la partícula dentro de los límites: $x_{i,j} \in [l_j, u_j]$
  Inicializar la mejor posición individual $pBest_i$ a la posición inicial: $pBest_i = X_i$
  si $f(pBest_i) < f(gBest)$ **entonces**
    Actualizar la mejor posición colectiva: $gBest = pBest_i$
  fin
fin
$\{\lambda_1\}_1 = 0.6 + 0.4 * \texttt{exploration\_level}$
$\{\lambda_1\}_2 = 0.6 + 0.4 * \texttt{exploration\_level}$
mientras *No se cumpla el criterio de terminación* **hacer**
  para *cada partícula* $i = 1, ..., N$ **hacer**
    $\beta = \texttt{exploration\_level} * (pBest_i \neq X_i)$
    $\vec{u}_1 \sim \mathcal{N}(\vec{0}, I_d)$
    $\vec{u}_2 \sim \mathcal{N}(\vec{0}, I_d)$
    $\vec{r}_1 = \vec{u}_1 - \frac{\vec{u}_1^{\mathrm{T}}(pBest_i - X_i)}{\|pBest_i - X_i\|^2}(pBest_i - X_i)$
    $\vec{r}_1 = \frac{\vec{r}_1}{\|\vec{r}_1\|}\|pBest_i - X_i\|$
    $\vec{r}_2 = \vec{u}_2 - \frac{\vec{u}_2^{\mathrm{T}}(gBest - X_i)}{\|gBest - X_i\|^2}(gBest - X_i)$
    $\vec{r}_2 = \frac{\vec{r}_2}{\|\vec{r}_2\|}\|gBest - X_i\|$
    $X_i = \beta(pBest_i + \{\lambda_1\}_1\vec{r}_1) + (1 - \beta)(gBest + \{\lambda_1\}_2\vec{r}_2)$
    Comprobar y gestionar si es necesario la factibilidad de las posiciones X_i
  fin
  para *cada partícula* $i = 1, ..., N$ **hacer**
    si $f(X_i) < f(pBest_i)$ **entonces**
      Actualizar la mejor posición individual: $pBest_i = X_i$
      si $f(pBest_i) < f(gBest)$ **entonces**
        Actualizar la mejor posición colectiva: $gBest = pBest_i$
      fin
    fin
  fin
fin

la actualización de X_i en el algoritmo 6 (dada por (4.7)) por la expresión

$$X_i = \beta \left(pBest_i + \{\lambda_1\}_1 \frac{\vec{u}_1}{\|\vec{u}_1\|} \|pBest_i - X_i\| \right) + (1-\beta) \left(gBest + \{\lambda_1\}_2 \frac{\vec{u}_2}{\|\vec{u}_2\|} \|gBest - X_i\| \right) \tag{4.8}$$

Podríamos hacer el análisis de convergencia de nuestro método final, ya sea empleando la ecuación (4.8) o la ecuación (4.7). Por simplicidad vamos a hacerlo con la primera (4.8), aunque hacerlo con la segunda sería equivalente, pues el reemplazamiento de $\frac{\vec{u}_1}{\|\vec{u}_1\|}$ y $\frac{\vec{u}_2}{\|\vec{u}_2\|}$ por $\frac{\vec{r}_1}{\|\vec{r}_1\|}$ y $\frac{\vec{r}_2}{\|\vec{r}_2\|}$ no altera el resultado del análisis al tener todas estas variables la misma esperanza matemática y la misma varianza. Asimismo, sabemos por [58] que asumir la convergencia en distribución de $pBest$ y $gBest$ es condición necesaria para la estabilidad de órdenes- 1 y 2 del PSO. Nosotros seguiremos esta asunción ya que es la forma más débil de estancamiento que existe en la literatura para analizar la estabilidad. También asumiremos constantes los hiperparámetros del método, a pesar de que el hiperparámetro β sí cambie de vez en cuando en situaciones concretas. Con todo, comenzamos con el análisis de orden- 1, calculando a partir de (4.8) el valor esperado de $X^{(t+1)}$:

$$\begin{aligned} \mathbb{E}\left(X^{(t+1)}\right) = \beta \left(\mathbb{E}\left(pBest^{(t)}\right) + \{\lambda_1\}_1 \mathbb{E}\left(\frac{\vec{u}_1}{\|\vec{u}_1\|} \right) \mathbb{E}\left(\|pBest^{(t)} - X^{(t)}\| \right) \right) + \\ + (1-\beta) \left(\mathbb{E}\left(gBest^{(t)}\right) + \{\lambda_1\}_2 \mathbb{E}\left(\frac{\vec{u}_2}{\|\vec{u}_2\|} \right) \mathbb{E}\left(\|gBest^{(t)} - X^{(t)}\| \right) \right) \end{aligned}$$

Pero como sabemos del apartado 4.2.3.4, $\mathbb{E}\left(\frac{\vec{u}_1}{\|\vec{u}_1\|}\right) = \vec{0}$ y $\mathbb{E}\left(\frac{\vec{u}_2}{\|\vec{u}_2\|}\right) = \vec{0}$, por lo que la expresión anterior queda reducida a:

$$\mathbb{E}\left(X^{(t+1)}\right) = \beta\, \mathbb{E}\left(pBest^{(t)}\right) + (1-\beta)\, \mathbb{E}\left(gBest^{(t)}\right)$$

de la cual se deduce que

$$\exists \lim_{t\to+\infty} \mathbb{E}(X^{(t)}) \iff \exists \lim_{t\to+\infty} \mathbb{E}(pBest^{(t)}) \wedge \exists \lim_{t\to+\infty} \mathbb{E}(gBest^{(t)})$$

lo que quiere decir que no importan los valores de los hiperparámetros del método $\beta, \{\lambda_1\}_1, \{\lambda_1\}_2$ para la convergencia de $\mathbb{E}\left(X^{(t)}\right)$. Tan solo es condición suficiente y necesaria que converjan $\mathbb{E}\left(pBest^{(t)}\right)$ y $\mathbb{E}\left(gBest^{(t)}\right)$, lo cual es asunción de nuestro análisis. Por tanto, $\mathbb{E}\left(X^{(t)}\right)$ converge siempre y lo hace a:

$$\lim_{t\to+\infty} \mathbb{E}(X^{(t)}) = \beta \lim_{t\to+\infty} \mathbb{E}(pBest^{(t)}) + (1-\beta) \lim_{t\to+\infty} \mathbb{E}(gBest^{(t)})$$

Este resultado era de suponer, ya que precisamente diseñamos nuestro método con vistas a separar la influencia de cada atractor en la región de convergencia (mediante β) del nivel de exploración (mediante $\{\lambda_1\}_1$ y $\{\lambda_1\}_2$). En nuestro método β hace las veces de $\frac{c_1}{c_1+c_2}$, por lo que el punto de convergencia es análogo al del PSO original (P).

Respecto a la estabilidad orden- 2, como $\sigma^2(X^{(t)}) = \mathbb{E}\left(X^{(t)^2}\right) - \mathbb{E}\left(X^{(t)}\right)^2$ y $\mathbb{E}\left(X^{(t)}\right)$ siempre converge (y por tanto también lo hace $\mathbb{E}\left(X^{(t)}\right)^2$):

$$\exists \lim_{t\to+\infty} \sigma^2(X^{(t)}) \iff \exists \lim_{t\to+\infty} \mathbb{E}\left(X^{(t)^2}\right)$$

Es decir, para analizar la estabilidad orden- 2 basta con analizar la convergencia de $\mathbb{E}\left(X^{(t)^2}\right)$. Así pues comenzaremos calculando la ecuación de $X^{(t+1)^2}$:

$$
\begin{aligned}
X^{(t+1)^2} = & \left(\beta \left(pBest^{(t)} + \{\lambda_1\}_1 \frac{\vec{u}_1}{\|\vec{u}_1\|} \left\| pBest^{(t)} - X^{(t)} \right\| \right) + \right. \\
& \left. + (1-\beta) \left(gBest^{(t)} + \{\lambda_1\}_2 \frac{\vec{u}_2}{\|\vec{u}_2\|} \left\| gBest^{(t)} - X^{(t)} \right\| \right) \right)^2 \\
= & \, \beta^2 \left(pBest^{(t)} + \{\lambda_1\}_1 \frac{\vec{u}_1}{\|\vec{u}_1\|} \left\| pBest^{(t)} - X^{(t)} \right\| \right)^2 + \\
& + (1-\beta)^2 \left(gBest^{(t)} + \{\lambda_1\}_2 \frac{\vec{u}_2}{\|\vec{u}_2\|} \left\| gBest^{(t)} - X^{(t)} \right\| \right)^2 + \\
& + 2\beta(1-\beta) \left(pBest^{(t)} + \{\lambda_1\}_1 \frac{\vec{u}_1}{\|\vec{u}_1\|} \left\| pBest^{(t)} - X^{(t)} \right\| \right) * \\
& * \left(gBest^{(t)} + \{\lambda_1\}_2 \frac{\vec{u}_2}{\|\vec{u}_2\|} \left\| gBest^{(t)} - X^{(t)} \right\| \right) \\
= & \, \beta^2 \left(pBest^{(t)^2} + \{\lambda_1\}_1^2 \left(\frac{\vec{u}_1}{\|\vec{u}_1\|} \right)^2 \left\| pBest^{(t)} - X^{(t)} \right\|^2 + \right. \\
& \left. + 2 pBest^{(t)} \{\lambda_1\}_1 \frac{\vec{u}_1}{\|\vec{u}_1\|} \left\| pBest^{(t)} - X^{(t)} \right\| \right) + \\
& + (1-\beta)^2 \left(gBest^{(t)^2} + \{\lambda_1\}_2^2 \left(\frac{\vec{u}_2}{\|\vec{u}_2\|} \right)^2 \left\| gBest^{(t)} - X^{(t)} \right\|^2 + \right. \\
& \left. + 2 gBest^{(t)} \{\lambda_1\}_2 \frac{\vec{u}_2}{\|\vec{u}_2\|} \left\| gBest^{(t)} - X^{(t)} \right\| \right) + \\
& + 2\beta(1-\beta) \left(pBest^{(t)} + \{\lambda_1\}_1 \frac{\vec{u}_1}{\|\vec{u}_1\|} \left\| pBest^{(t)} - X^{(t)} \right\| \right) * \\
& * \left(gBest^{(t)} + \{\lambda_1\}_2 \frac{\vec{u}_2}{\|\vec{u}_2\|} \left\| gBest^{(t)} - X^{(t)} \right\| \right)
\end{aligned}
$$

Introduciendo ahora el operador de esperanza matemática:

$$
\begin{aligned}
\mathbb{E}\left(X^{(t+1)^2}\right) = \beta^2 & \left(\mathbb{E}\left(pBest^{(t)^2} \right) + \{\lambda_1\}_1^2 \, \mathbb{E}\left(\left(\frac{\vec{u}_1}{\|\vec{u}_1\|} \right)^2 \right) \mathbb{E}\left(\left\| pBest^{(t)} - X^{(t)} \right\|^2 \right) + \right. \\
& \left. + 2 \, \mathbb{E}\left(pBest^{(t)} \right) \{\lambda_1\}_1 \, \mathbb{E}\left(\frac{\vec{u}_1}{\|\vec{u}_1\|} \right) \mathbb{E}\left(\left\| pBest^{(t)} - X^{(t)} \right\| \right) \right) + \\
+ (1-\beta)^2 & \left(\mathbb{E}\left(gBest^{(t)^2} \right) + \{\lambda_1\}_2^2 \, \mathbb{E}\left(\left(\frac{\vec{u}_2}{\|\vec{u}_2\|} \right)^2 \right) \mathbb{E}\left(\left\| gBest^{(t)} - X^{(t)} \right\|^2 \right) + \right. \\
& \left. + 2 \, \mathbb{E}\left(gBest^{(t)} \right) \{\lambda_1\}_2 \, \mathbb{E}\left(\frac{\vec{u}_2}{\|\vec{u}_2\|} \right) \mathbb{E}\left(\left\| gBest^{(t)} - X^{(t)} \right\| \right) \right) + \\
+ 2\beta(1-\beta) & \left(\mathbb{E}\left(pBest^{(t)} \right) + \{\lambda_1\}_1 \, \mathbb{E}\left(\frac{\vec{u}_1}{\|\vec{u}_1\|} \right) \mathbb{E}\left(\left\| pBest^{(t)} - X^{(t)} \right\| \right) \right) * \\
& * \left(\mathbb{E}\left(gBest^{(t)} \right) + \{\lambda_1\}_2 \, \mathbb{E}\left(\frac{\vec{u}_2}{\|\vec{u}_2\|} \right) \mathbb{E}\left(\left\| gBest^{(t)} - X^{(t)} \right\| \right) \right)
\end{aligned}
$$

Aplicando de nuevo la igualdad $\mathbb{E}\left(\frac{\vec{u}_1}{\|\vec{u}_1\|}\right) = \vec{0}$ y $\mathbb{E}\left(\frac{\vec{u}_2}{\|\vec{u}_2\|}\right) = \vec{0}$:

$$\begin{aligned}\mathbb{E}\left(X^{(t+1)^2}\right) = \beta^2 &\left(\mathbb{E}\left(pBest^{(t)^2}\right) + \{\lambda_1\}_1^2\,\mathbb{E}\left(\left(\frac{\vec{u}_1}{\|\vec{u}_1\|}\right)^2\right)\mathbb{E}\left(\left\|pBest^{(t)} - X^{(t)}\right\|^2\right)\right) + \\ &+ (1-\beta)^2\left(\mathbb{E}\left(gBest^{(t)^2}\right) + \{\lambda_1\}_2^2\,\mathbb{E}\left(\left(\frac{\vec{u}_2}{\|\vec{u}_2\|}\right)^2\right)\mathbb{E}\left(\left\|gBest^{(t)} - X^{(t)}\right\|^2\right)\right) + \\ &+ 2\beta(1-\beta)\,\mathbb{E}\left(pBest^{(t)}\right)\mathbb{E}\left(gBest^{(t)}\right)\end{aligned}$$

Podemos calcular $\mathbb{E}\left(\left\|pBest^{(t)} - X^{(t)}\right\|^2\right)$:

$$\begin{aligned}\mathbb{E}\left(\left\|pBest^{(t)} - X^{(t)}\right\|^2\right) &= \mathbb{E}\left(\sum_{j=1}^{d}\left(pbest_j^{(t)} - x_j^{(t)}\right)^2\right) = \\ &= \mathbb{E}\left(\sum_{j=1}^{d}\left(pbest_j^{(t)^2} + x_j^{(t)^2} - 2pbest_j^{(t)}x_j^{(t)}\right)\right) = \\ &= \mathbb{E}\left(\sum_{j=1}^{d} pbest_j^{(t)^2} + \sum_{j=1}^{d} x_j^{(t)^2} - 2\sum_{j=1}^{d} pbest_j^{(t)}x_j^{(t)}\right) = \\ &= \mathbb{E}\left(\left\|pBest^{(t)}\right\|^2\right) + \sum_{j=1}^{d}\mathbb{E}\left(x_j^{(t)^2}\right) - 2\sum_{j=1}^{d}\mathbb{E}\left(pbest_j^{(t)}x_j^{(t)}\right)\end{aligned}$$

Como la fórmula de $\mathbb{E}\left(X^{(t)^2}\right)$ es la misma para todas las dimensiones $1 \leq j \leq d$, tenemos que $\mathbb{E}\left(x_{j_1}^{(t)^2}\right) = \mathbb{E}\left(x_{j_2}^{(t)^2}\right) \;\; \forall j_1, j_2 \in 1, ..., d$. Esto es, todas las componentes dimensionales del vector $X^{(t)^2}$ tienen la misma esperanza matemática. Lo cual viene a decir que $\sum_{j=1}^{d}\mathbb{E}\,x_j^{(t)^2} = d * \mathbb{E}\left(x_{j_1}^{(t)^2}\right)$. Haciendo esta sustitución en la expresión anterior

y aplicando también la ecuación de $x_j^{(t)}$:

$$\begin{aligned}
\mathbb{E}\left(\left\|pBest^{(t)} - X^{(t)}\right\|^2\right) = \; & \mathbb{E}\left(\left\|pBest^{(t)}\right\|^2\right) + d\,\mathbb{E}\left(x_{j_1}^{(t)^2}\right) - \\
& - 2\sum_{j=1}^{d}\mathbb{E}\left(pbest_j^{(t)}\left(\beta pbest_j^{(t-1)} + (1-\beta)gbest_j^{(t-1)} + \right.\right. \\
& \qquad + \beta\{\lambda_1\}_1\frac{\{\vec{u}_1\}_j}{\|\vec{u}_1\|}\|pBest^{(t-1)} - X^{(t-1)}\| + \\
& \qquad \left.\left. + (1-\beta)\{\lambda_1\}_2\frac{\{\vec{u}_2\}_j}{\|\vec{u}_2\|}\|gBest^{(t-1)} - X^{(t-1)}\|\right)\right) \\
= \; & \mathbb{E}\left(\left\|pBest^{(t)}\right\|^2\right) + d\,\mathbb{E}\left(x_{j_1}^{(t)^2}\right) - 2\sum_{j=1}^{d}\left(\mathbb{E}\left(\beta pbest_j^{(t)}pbest_j^{(t-1)}\right) + \right. \\
& + \mathbb{E}\left((1-\beta)pbest_j^{(t)}gbest_j^{(t-1)}\right)\Big) \\
= \; & \mathbb{E}\left(\left\|pBest^{(t)}\right\|^2\right) + d\,\mathbb{E}\left(x_{j_1}^{(t)^2}\right) - 2\beta\,\mathbb{E}\left(\langle pBest^{(t)}, pBest^{(t-1)}\rangle\right) - \\
& - 2(1-\beta)\,\mathbb{E}\left(\langle pBest^{(t)}, gBest^{(t-1)}\rangle\right)
\end{aligned}$$

Y procediendo de la misma forma para hallar $\mathbb{E}\left(\left\|gBest^{(t)} - X^{(t)}\right\|^2\right)$:

$$\begin{aligned}
\mathbb{E}\left(\left\|gBest^{(t)} - X^{(t)}\right\|^2\right) = \; & \mathbb{E}\left(\left\|gBest^{(t)}\right\|^2\right) + d\,\mathbb{E}\left(x_{j_1}^{(t)^2}\right) - \\
& - 2(1-\beta)\,\mathbb{E}\left(\langle gBest^{(t)}, gBest^{(t-1)}\rangle\right) - 2\beta\,\mathbb{E}\left(\langle pBest^{(t)}, gBest^{(t)}\rangle\right)
\end{aligned}$$

Y volviendo a la ecuación principal de $\mathbb{E}\left(X^{(t+1)^2}\right)$, si incorporamos estas últimas igualdades y recordamos también de 4.2.3.4 que $\mathbb{E}\left(\left(\frac{\vec{u}_1}{\|\vec{u}_1\|}\right)^2\right) = \frac{1}{d} * \vec{1}$ y $\mathbb{E}\left(\left(\frac{\vec{u}_2}{\|\vec{u}_2\|}\right)^2\right) = \frac{1}{d} * \vec{1}$:

$$\begin{aligned}
\mathbb{E}\left(X^{(t+1)^2}\right) = \beta^2\Bigg(& \mathbb{E}\left(pBest^{(t)^2}\right) + \{\lambda_1\}_1^2\frac{1}{d}\Bigg(\mathbb{E}\left(\left\|pBest^{(t)}\right\|^2\right) + d\,\mathbb{E}\left(X^{(t)^2}\right) - \\
& - 2\beta\,\mathbb{E}\left(\langle pBest^{(t)}, pBest^{(t-1)}\rangle\right) - 2(1-\beta)\,\mathbb{E}\left(\langle pBest^{(t)}, gBest^{(t)}\rangle\right)\Bigg)\Bigg) \\
+ (1-\beta)^2\Bigg(& \mathbb{E}\left(gBest^{(t)^2}\right) + \{\lambda_1\}_2^2\frac{1}{d}\Bigg(\mathbb{E}\left(\left\|gBest^{(t)}\right\|^2\right) + d\,\mathbb{E}\left(X^{(t)^2}\right) - \\
& - 2(1-\beta)\,\mathbb{E}\left(\langle gBest^{(t)}, gBest^{(t-1)}\rangle\right) - 2\beta\,\mathbb{E}\left(\langle pBest^{(t)}, gBest^{(t)}\rangle\right)\Bigg)\Bigg) \\
+ 2\beta(1-\beta)\,\mathbb{E}\left(pBest^{(t)}\right)\mathbb{E}\left(gBest^{(t)}\right)
\end{aligned}$$

La relación de recurrencia es lineal no homogénea y de grado 1. Es de la forma

$$\mathbb{E}\left(X^{(t+1)^2}\right) - \left(\beta^2\{\lambda_1\}_1^2 + (1-\beta)^2\{\lambda_1\}_2^2\right)\mathbb{E}\left(X^{(t)^2}\right) + b = 0$$

con b la parte no homogénea de la recurrencia, que sabemos por asunción que es convergente. Por tanto también sabemos que el sistema es estable si y solo si todas las raíces del polinomio característico asociado a la recurrencia tienen un módulo menor que 1. El polinomio característico en este caso es

$$e - \left(\beta^2\{\lambda_1\}_1^2 + (1-\beta)^2\{\lambda_1\}_2^2\right)$$

que, como es de grado 1, solamente tiene una raíz real en $e = \left(\beta^2\{\lambda_1\}_1^2 + (1-\beta)^2\{\lambda_1\}_2^2\right)$. Con lo cual, el sistema recurrente converge cuando $\left|\beta^2\{\lambda_1\}_1^2 + (1-\beta)^2\{\lambda_1\}_2^2\right| < 1$. Como $\left(\beta^2\{\lambda_1\}_1^2 + (1-\beta)^2\{\lambda_1\}_2^2\right) > 0$, la condición de convergencia queda:

$$\beta^2\{\lambda_1\}_1^2 + (1-\beta)^2\{\lambda_1\}_2^2 < 1 \tag{4.9}$$

Y si consideramos como en la versión final del algoritmo 6 que $\{\lambda_1\}_1 = \{\lambda_1\}_2 = \lambda_1$:

$$\beta^2\lambda_1^2 + (1-\beta)^2\lambda_1^2 < 1 \implies \lambda_1^2 < \frac{1}{\beta^2 + (1-\beta)^2} \implies \lambda_1 < \sqrt{\frac{1}{\beta^2 + (1-\beta)^2}} \tag{4.10}$$

Es más, si finalmente incorporamos la variable `exploration_level`, con $\{\lambda_1\}_1 = \{\lambda_1\}_2 = 0.6 + 0.4 * \texttt{exploration\_level}$ y $\beta = \texttt{exploration\_level}$, entonces la región de convergencia dependería por completo de `exploration_level` (`exp_level` por abreviar) y sería:

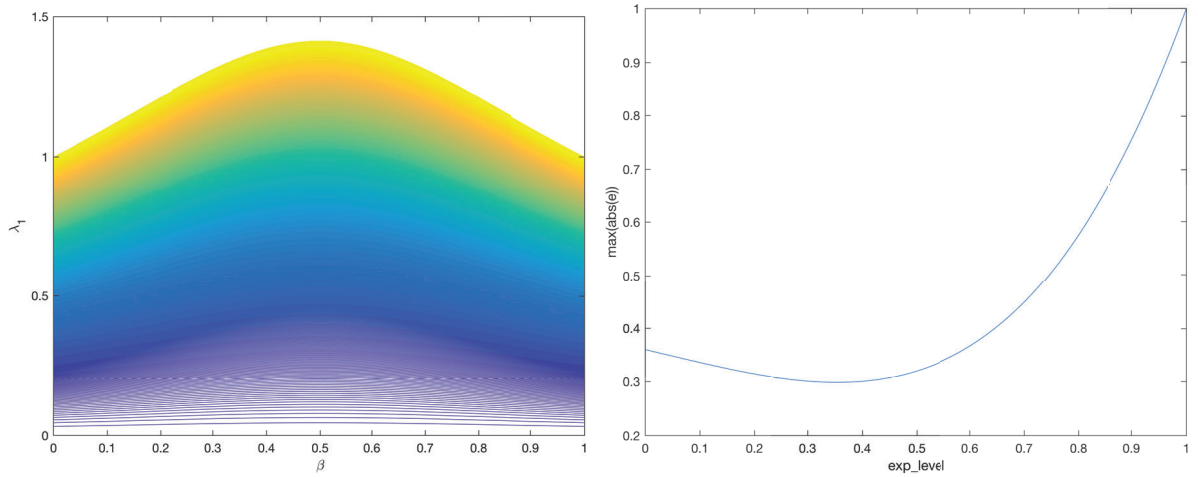
$$\texttt{exp\_level}^2(0.6 + 0.4 * \texttt{exp\_level})^2 + (1 - \texttt{exp\_level})^2(0.6 + 0.4 * \texttt{exp\_level})^2 < 1$$

Operando la inecuación llegamos a:

$$0.32 * \texttt{exp\_level}^4 + 0.64 * \texttt{exp\_level}^3 - 0.08 * \texttt{exp\_level}^2 - 0.24 * \texttt{exp\_level} - 0.64 < 0$$

La inecuación es de cuarto grado. Resolviéndola obtenemos que $\texttt{exp\_level} \in (-2.1545, 1.0)$. Sin embargo, por definición tenemos que $\texttt{exp\_level} > 0$, por lo que la región de convergencia queda restringida a $\texttt{exp\_level} \in (0, 1)$. Esto es justo lo que buscábamos con la confección de este parámetro, que el rango válido de la variable `exp_level` fuera el $(0, 1)$. El límite de la región de convergencia está en $\texttt{exp\_level} = 1$, lo cual quiere decir que éste es su valor más explorativo (el asociado a una raíz del polinomio característico mayor). Este hecho demuestra de manera teórica la validez de nuestro método.

Las gráficas de las regiones de convergencia tanto para el caso en que consideremos por separado las variables β y $\{\lambda_1\}_1 = \{\lambda_1\}_2 = \lambda_1$ como para el caso en que las consideremos agrupadas por la variable `exploration_level` (de la forma en que se hace en el algoritmo 6), aparecen presentadas en la figura 4.14. Fijándonos en la figura 4.14a, así como en la expresión (4.10), vemos que realmente la tasa de convergencia depende de ambos parámetros β y λ_1 y que el rango útil de λ_1 (en el cual el algoritmo es convergente) varía con β, pudiendo ser mayor que 1 (su rango mínimo) y alcanzando un rango máximo de $\sqrt{2}$ en $\beta = 0.5$. Lo que es claro es que ante un mismo valor de λ_1, el nivel de exploración del algoritmo será diferente en función de β. Esta observación quizá nos sugiera redefinir el parámetro `exploration_level`



(a) (b)

Figura 4.14: Región paramétrica de convergencia del PSO propuesto (estabilidad orden-2). Se muestra para cada configuración paramétrica la tasa de convergencia de $\{\sigma^2(X^{(t)})\}_{t=0}^{+\infty}$. En el caso de (a) se hace mediante un mapa de contorno. (a) representa la gráfica 3-dimensional de la función que hace corresponder a cada dupla de valores paramétricos $(\beta, \{\lambda_1\}_1 = \{\lambda_1\}_2 = \lambda_1)$, la tasa de convergencia de $\{\sigma^2(X^{(t)})\}_{t=0}^{+\infty}$. (b) es una representación de (a) cuando $\beta = \texttt{exploration\_level}$ y $\lambda_1 = 0.6 + 0.4 * \texttt{exploration\_level}$.

para que se ajuste al rango útil variable de λ_1 del siguiente modo:

$$\begin{cases} \{\lambda_1\}_1 = \{\lambda_1\}_2 = \sqrt{\dfrac{1}{\beta^2 + (1-\beta)^2}}\,(0.6 + 0.4 * \texttt{exploration\_level}) \\ \beta = \texttt{exploration\_level} \end{cases}$$

No obstante, hay que tener presente también que el análisis no tiene en cuenta la relación entre los hiperparámetros y las distribuciones de probabilidad de los atractores $pBest$ y $gBest$ (asume independencia), cuando es obvio que sobre todo β, al decidir sobre la región a explorar, tiene una gran incidencia en ellas. Cabe esperar que cuanto mayor sea β, los cambios en el $gBest$ y en el $pBest$ sean menos frecuentes y que la proximidad entre estos atractores sea menor (se aproximen más lentamente), favoreciendo así una convergencia más lenta y una mayor exploración, y viceversa. Al no tener en cuenta esta información tan importante para la convergencia total de las partículas, las tasas de convergencia mostradas en la figura 4.14 no son del todo exactas. Además como es lógico, estos análisis de convergencia tampoco desvelan la calidad de la exploración para cada configuración paramétrica. Es decir, aunque haya dos configuraciones con una igual tasa de convergencia, sus rendimientos no tienen por qué ser iguales. En este caso, el rendimiento obtenido subiendo β y manteniendo λ_1 no tiene por qué ser el mismo que subiendo λ_1 y manteniendo β, por más que las tasas de convergencia de ambas configuraciones sean idénticas. Es por eso que el establecimiento de la variable `exploration_level` se haya realizado desde un principio de manera más empírica y en contra de lo que sugiere el análisis de estabilidad. De todos modos, en base a esto queda patente que la mejor opción sería configurar cada parámetro λ_1 y β por separado. La opción de hacerlo conjuntamente mediante

`exploration_level`, aunque seguramente no pueda aspirar a un rendimiento tan alto, consiga que la diferencia no sea significativa, con la gran ventaja de ahorrar el ajuste de un parámetro al usuario. En lo que resta de trabajo nosotros nos decantaremos por esta última opción, ya que es la más práctica de cara a la aplicabilidad del método por parte de cualquier usuario medio. Eso sí, de cara a cualquier investigación futura o confección de alguna variante adaptativa sobre el método, siempre será más recomendable partir de la opción primera.

Por último, con respecto al punto fijo de la varianza hay que decir que no se anula salvo cuando $\lim_{t\to+\infty} gBest^{(t)} = \lim_{t\to+\infty} pBest^{(t)}$. Es decir, solamente se alcanza la convergencia plena determinista (convergencia en sentido clásico) cuando las variables $pBest$ y $gBest$ convergen a la misma variable aleatoria. En ese caso, todas las partículas (sus posiciones X) convergen a esa misma variable. Si no, las partículas siguen en movimiento alrededor del punto medio ponderado de sus atractores, solamente convergiendo en valor esperado y en varianza (en distribución) pero continuando así con la exploración. Es lo mismo que sucede en el PSO clásico.

Una vez estudiado el rendimiento teórico de nuestro método, el siguiente paso será ya hacerlo de manera práctica.

Capítulo 5

Experimentación

Tras el análisis realizado, toca poner a prueba todas las deducciones y resultados teóricos obtenidos en él. Para ello, contamos con este capítulo experimental, que servirá para constatar empíricamente muchas de las afirmaciones y conclusiones realizadas durante el trabajo. Principalmente su objetivo es el de testear el nuevo método PSO propuesto, que recoge buena parte del aprendizaje desarrollado durante el análisis. También intentaremos demostrar ciertas características del PSO estándar que han sido reveladas durante el trabajo. Además compararemos rendimientos de distintos métodos PSO para situar lo competitiva que en la práctica es nuestra propuesta. Todo ello requiere de unas bases bien definidas sobre las que se sustentarán los experimentos y que detallaremos a continuación. También explicaremos más en profundidad cuáles son las pruebas experimentales que vamos a llevar a cabo.

5.1. Bases experimentales

Toda la experimentación se ha llevado a cabo bajo el entorno MatlabR2016b. Las implementaciones de las distintas variantes que forman parte de las pruebas experimentales se han hecho siempre en la medida de lo posible de manera vectorizada, tal y como permite el lenguaje Matlab. Así, los cálculos correspondientes a las distintas partículas y distintas dimensiones son paralelizados, sobre todo con el fin de reducir costes temporales de ejecución, aprovechando las características tanto del algoritmo como del lenguaje. Por supuesto hemos comprobado que las implementaciones sean correctas evaluando la concordancia de sus resultados con los presentados en los artículos de referencia.

Respecto a las variantes PSO implementadas hay que decir que solo nos hemos ceñido a su regla de actualización y hemos descartado cualquier estrategia inteligente que acompañe al método ya sea para gestionar límites de velocidades o posiciones, inicializaciones, etc. Esto lo hacemos para garantizar que las comparaciones entre métodos sean justas y los resultados obtenidos de las mismas sean únicamente debidos a la fórmula en sí del método y no a heurísticas que la acompañan, tal y como también hicimos en el análisis del Capítulo 4. En concreto, las inicializaciones de las posiciones de las partículas se han realizado aleatoriamente siguiendo una distribución de probabilidad uniforme en los límites del espacio de búsqueda ($x_{i,j} \sim \mathrm{U}(l_j, u_j)$) y las velocidades han sido inicializadas a 0. La gestión de salidas de los límites ha consistido en ignorarlas, ignorando también la evaluación de las soluciones inválidas.

La experimentación consistirá en comparar distintas variantes PSO, por lo que necesitamos establecer unas bases y condiciones justas para las comparativas. Por ejemplo, se ha de seleccionar una medida temporal para garantizar que todos los métodos compitan en igualdad de condiciones. El tiempo de ejecución no sería una medida justa ya que depende mucho de la implementación. Tampoco lo sería el número de iteraciones, ya que la carga de trabajo que realizan algunos métodos en una iteración no es comparable a la de otros. En estos casos se suele utilizar el número de evaluaciones (FEs) como medida temporal y así lo haremos nosotros. Se utiliza ya que la relación entre esta medida y el tiempo de ejecución es fuerte cuanto más compleja se vuelve la función y por tanto, proporciona una comparativa en igualdad cuando la función a optimizar es muy compleja y el coste de una sola evaluación es largo con respecto al resto de operaciones del PSO cuyos costes son despreciables. Un estándar muy común es que el número de evaluaciones sea de 2×10^5. Este será el número de FEs que emplearemos para todas nuestras pruebas experimentales.

Las pruebas comparativas entre métodos se harán, con el fin de ser completas y representativas, para distintas dimensiones y para distintos tamaños de población. Se tomará como resultado de cada prueba el error entre el mejor fitness encontrado al término de la ejecución y el fitness óptimo de la función objetivo ($error = f(gBest^{(maxIter)}) - f(X^*)$). Cada prueba será repetida hasta un total de 30 veces y será el resultado promedio de esas 30 ejecuciones el que finalmente asociemos a dicha prueba. Las funciones que emplearemos para las pruebas son las habituales que solemos encontrar en la literatura sobre el PSO y otras metaheurísticas para evaluar rendimientos de nuevas propuestas y/o realizar comparaciones entre métodos [138, 139]. Dichas funciones quedan recogidas en la tabla 5.1.

Cuadro 5.1: Funciones de test empleadas para la experimentación

Nombre	Definición	Rango de búsqueda	$f(X^*)$	Notación	Notación rotada
Sphere	$\sum_{j=1}^{d} x_j^2$	$[-500, 500]^d$	0	f_1	f_{10}
Rosenbrock	$\sum_{j=1}^{d-1} \left(100(x_{j+1} - x_j^2)^2 + (x_j - 1)^2\right)$	$[-2.048, 2.048]^d$	0	f_2	f_{11}
Ackley	$-20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{j=1}^{d} x_j^2}\right) - \exp\left(\frac{1}{d}\cos\left(2\pi x_j\right)\right) + 20 + \exp\left(1\right)$	$[-32, 32]^d$	0	f_3	f_{12}
Griewank	$\frac{1}{4000}\sum_{j=1}^{d} x_j^2 - \prod_{j=1}^{d} \cos\left(\frac{x_j}{\sqrt{j}}\right) + 1$	$[-600, 600]^d$	0	f_4	f_{13}
Rastrigin	$\sum_{j=1}^{d} \left(x_j^2 - 10\cos\left(2\pi x_j\right) + 10\right)$	$[-5.12, 5.12]^d$	0	f_5	f_{14}
Schwefel 2.22	$\sum_{j=1}^{d} \lvert x_j\rvert + \prod_{j=1}^{d} \lvert x_j\rvert$	$[-10, 10]^d$	0	f_6	f_{15}
Weierstrass	$\sum_{j=1}^{d}\sum_{k=0}^{20} 0.5^k \cos\left(2\pi 3^k (x_j + 0.5)\right) - d\sum_{k=0}^{20} 0.5^k \cos\left(\pi 3^k\right)$	$[-0.5, 0.5]^d$	0	f_7	f_{16}
Alpine	$\sum_{j=1}^{d} \lvert x_j \sin x_j + 0.1 x_j\rvert$	$[-10, 10]^d$	0	f_8	f_{17}
Generalized Penalized	$\frac{\pi}{d}\Big(10\sin^2\left(\pi + \frac{\pi}{4}(x_1 + 1)\right) + \sum_{j=1}^{d-1}\left(\frac{1}{4}(x_j + 1)\right)^2 * \\ * \left(1 + 10\sin^2\left(\pi + \frac{\pi}{4}(x_{j+1} + 1)\right)\right) + \left(\frac{1}{4}(x_d + 1)\right)^2\Big) + \\ + \sum_{j=1}^{d}\Big(100(x_j - 10)^4 (x_j > 10) + 100(-x_j - 10)^4 (x_j < -10)\Big)$	$[-50, 50]^d$	0	f_9	f_{18}

Tal y como se refleja en la tabla, no solo vamos a recurrir a las versiones estándar de las funciones (las definiciones originales de las mismas descritas en la segunda columna), sino que también trabajaremos con sus versiones rotadas. Recordemos que la rotación de la función es una manera sencilla de convertirla en una función no-separable, que es algo que nos interesa para ver lo generales que son los métodos probados, para ver si realmente son buenos métodos de búsqueda o simplemente están explotando una característica de la función que la hace ser optimizable por buscadores simples. Para conseguir la versión rotada de cada función se ha seguido el método de Salomon [127], que cosiste en generar una matriz ortogonal $Q \in Orth$ (para lo cual se ha seguido el procedimiento Gram Schmidt) y reemplazar las variables de entrada X por $Q \cdot X$, es decir, realizar un cambio de base de coordenadas y expresar las variables en la nueva base. Cada una de las 30 ejecuciones que se realizan por prueba contará con su propia matriz ortogonal Q generada de forma independiente para así cerciorarnos de que no exista un sesgo hacia ninguna base determinada y garantizar que la verificación sobre la sensibilidad o no del método en la base de coordenadas escogida sea válida.

Aparte de las definiciones de cada función y las notaciones que vamos a emplear para referirnos a ellas en los sucesivos experimentos, también se muestra en la tabla el rango de búsqueda válido (fronteras del espacio de búsqueda) para cada una de las funciones y sus valores mínimos. En todas ellas el valor mínimo es 0, por lo que realmente el error que queremos representar es el mismo valor mínimo encontrado por el método ($error = f(gBest^{(maxIter)}) - f(X^*) = f(gBest^{(maxIter)})$). Hay que decir que aunque no aparezca en la tabla, la mayoría de estas funciones tienen su óptimo en el centro de coordenadas (en el $(0, ..., 0)$), que es el centro del espacio de búsqueda. Ante esta situación convendría también considerar versiones desplazadas de las funciones para evitar la supremacía de algunos métodos con tendencia a concentrar la búsqueda en el centro del espacio de búsqueda. Sin embargo, en base a lo estudiado durante el proyecto, el PSO en principio no sufre de este sesgo y ninguna variante del mismo debería de hacerlo, por lo que nos hemos ahorrado esta carga experimental extra. Por lo demás, las funciones han sido escogidas de tal modo que exhiban características diferentes entre sí. Tenemos alguna función unimodal y el resto son multimodales, que son en teoría las más difíciles de optimizar, tenemos funciones convexas, no convexas, etc. Asimismo la mayoría de funciones son separables y es porque, como mencionamos en su momento, casi todas las funciones de test que se usan para comparar metaheurísticos son separables. Por eso en parte es más que conveniente su rotación, para así tener un benchmark más representativo y es que, en un futuro, la aplicación práctica de estos métodos muy probablemente involucre funciones no-separables, que son las que más aparecen en problemas reales (funciones de coste para entrenamiento de redes neuronales, etc).

Ahora sí, ya estamos listos para dar paso a las distintas pruebas experimentales. Pero antes las introduciremos brevemente. Son un total de cinco experimentos. En cada experimento se detallará el objetivo y las condiciones específicas. Quedará especificado el número de dimensiones d y tamaño de enjambre N en el que se enmarca cada prueba. Recordemos que por cada experimento se realizarán pruebas para distintos valores de N y d. Con especificar el N asociado a cada prueba, podemos deducir el número de iteraciones $maxIter$, puesto que ambos parámetros determinan el número de FEs, que es un parámetro que consideramos fijo e igual a 2×10^5. Los experimentos son los siguientes:

- Comparar PSO con componente inercial vs PSO sin componente inercial siguiendo la intuición aportada en 4.5
- Comparar experimentalmente los distintos métodos de giro discutidos en el apartado 4.2.3.3
- Comprobar empíricamente el correcto funcionamiento del método propuesto en el Algoritmo 6 mediante la comparación entre diferentes configuraciones paramétricas
- Comparar distintos métodos PSO con el propuesto en el Algoritmo 6
- Comparar métodos PSO específicos para funciones separables, entre los que se incluye el propuesto en el Algoritmo 5

5.2. Experimentos

En esta sección presentaremos cada uno de los experimentos recién mencionados al final de la anterior y desarrollaremos los resultados obtenidos, haciendo algún pequeño apunte sobre los mismos.

5.2.1. Esencialidad de la componente inercial

En este apartado pretendemos evaluar la validez empírica de nuestra observación en 4.5, donde pusimos encima de la mesa que el hecho de que la componente inercial fuese considerada imprescindible podría tener mucho que ver con la pérdida de nivel de exploración cuando $pBest = X$. Establecimos que sería interesante constatarlo probando con una versión del PSO estándar sin componente inercial pero con incrementos de c_2 para los casos en que $pBest = X$. Comparando esa versión con la original, nos daremos cuenta de si efectivamente la componente inercial aporta ventajas que van más allá de este hecho o si por el contrario, todo su valor se debe a lo expuesto, lo cual significaría arrojar luz sobre el verdadero rol del hiperparámetro w que tanta atención atrae por parte de los investigadores como vimos en 3.3.1.

Para ello, contaremos con la configuración más adoptada por la comunidad para el PSO estándar. Es la configuración que según múltiples estudios arroja mejores resultados [44] y consiste en un factor inercial $w = 0.72$ y unos factores social y cognitivo $c_1 = c_2 = 1.49$. Este PSO será comparado con su versión sin inercia ($w = 0$) y con su versión sin inercia pero con incremento de c_2 cuando $pBest = X$ ($w = 0$, $c_2 = \{c_2\}_{(pBest=X)} * (pBest = X) + \{c_2\}_{(pBest \neq X)} * (pBest \neq X)$). Con el fin de ser justos, sabiendo que la inercia también afecta al balance explorativo del método, las versiones sin inercia que vamos a utilizar han de ajustar el valor de los hiperparámetros c_1 y c_2 para adaptarse al balance de la configuración elegida. Concretamente han de incrementar algo sus valores, ya que en este caso, en esta configuración, la componente inercial aporta exploración (ver figura 3.2). Este ajuste se ha realizado de tal modo que la tasa de convergencia de la varianza para la versión sin inercia ($w = 0$) coincida con la de la versión con inercia ($w = 0.72$). Con un poco de álgebra llegamos a que $c_1 = c_2 = 1.68$ es la configuración equivalente en balance explorativo. Así, en la versión sin inercia pero con c_2 adaptable, cuando $pBest \neq X$ tendremos que $\{c_2\}_{(pBest \neq X)} = 1.68$, mientras que cuando $pBest = X$, el valor que tomará c_2 ha sido escogido empíricamente decidiendo que sea igual a 2.7 ($\{c_2\}_{(pBest=X)} = 2.7$). En definitiva, las tres versiones a comparar

se resumen, junto con la notación que emplearemos para referirnos a ellas, en el siguiente listado:

- PSO1: $c_1 = 1.49$, $c_2 = 1.49$, $w = 0.72$
- PSO2: $c_1 = 1.68$, $c_2 = 1.68$, $w = 0$
- PSO3: $c_1 = 1.68$, $c_2 = 2.7 * (pBest = X) + 1.68 * (pBest \neq X)$, $w = 0$

La comparación se ha realizado para las dimensiones $d = 30$ y $d = 50$, como suele ser habitual en este tipo de comparativas. Por cada una, se ha probado con los tamaños de enjambre $N = 30$ y $N = 50$. Como los métodos a comparar usan un número de evaluaciones de función (FEs) por iteración igual a N, entonces el número de iteraciones es igual para los tres y es $maxIter = \frac{2 \times 10^5}{N}$, ya que como bien especificamos en las bases experimentales, el número de FEs va a ser 2×10^5 para todas las pruebas que vayamos a realizar durante la experimentación. Para este test en primera instancia no hemos considerado las versiones rotadas de las funciones, pues el objetivo no es poner a prueba el rendimiento del PSO estándar (lo haremos más adelante), sino tan solo comprobar la necesariedad de la componente inercial en el método. Así las cosas, en las siguientes Tablas 5.2 y 5.3 se presentan los resultados obtenidos. Como también será un hecho habitual en el capítulo, los mejores resultados de cada comparación (para cada función por cada tamaño poblacional N y cada dimensión d) aparecen resaltados en negrita.

Cuadro 5.2: Comparativa entre modelos PSO con y sin inercia para un tamaño de dimensión $d = 30$

Función	$N = 30$			$N = 50$		
	PSO1	PSO2	PSO3	PSO1	PSO2	PSO3
f_1	$7.48e-94$	$7.70e+04$	$\mathbf{7.54e-120}$	$9.42e-82$	$2.44e+04$	$\mathbf{4.32e-85}$
f_2	$\mathbf{5.86e+00}$	$2.48e+02$	$9.72e+00$	$\mathbf{9.88e+00}$	$1.78e+02$	$1.16e+01$
f_3	$2.60e+00$	$1.34e+01$	$\mathbf{9.64e-01}$	$1.24e+00$	$1.16e+01$	$\mathbf{2.11e-01}$
f_4	$5.82e-02$	$2.54e+01$	$\mathbf{2.50e-02}$	$2.61e-02$	$9.70e+00$	$\mathbf{1.52e-02}$
f_5	$\mathbf{6.66e+01}$	$1.20e+02$	$6.80e+01$	$5.52e+01$	$1.03e+02$	$\mathbf{5.20e+01}$
f_6	$3.25e-03$	$2.80e+01$	$\mathbf{2.06e-74}$	$1.82e-14$	$1.45e+01$	$\mathbf{1.52e-54}$
f_7	$6.58e+00$	$2.22e+01$	$\mathbf{4.12e+00}$	$5.20e+00$	$1.99e+01$	$\mathbf{2.09e+00}$
f_8	$3.36e-05$	$7.89e+00$	$\mathbf{4.78e-09}$	$6.09e-12$	$4.90e+00$	$\mathbf{7.36e-14}$
f_9	$3.63e-01$	$2.01e+05$	$\mathbf{1.45e-01}$	$3.61e-01$	$6.63e+03$	$\mathbf{1.11e-01}$

Cuadro 5.3: Comparativa entre modelos PSO con y sin inercia para un tamaño de dimensión $d = 50$

Función	$N = 30$			$N = 50$		
	PSO1	PSO2	PSO3	PSO1	PSO2	PSO3
f_1	$3.93e-22$	$1.09e+05$	$\mathbf{1.13e-52}$	$3.96e-27$	$3.30e+04$	$\mathbf{2.96e-38}$
f_2	$\mathbf{4.44e+01}$	$4.66e+02$	$4.80e+01$	$\mathbf{4.65e+01}$	$2.76e+02$	$4.73e+01$
f_3	$5.77e+00$	$1.62e+01$	$\mathbf{1.69e+00}$	$3.15e+00$	$1.56e+01$	$\mathbf{3.72e-01}$
f_4	$1.43e-01$	$5.05e+01$	$\mathbf{2.91e-02}$	$4.75e-02$	$1.29e+01$	$\mathbf{1.90e-02}$
f_5	$1.64e+02$	$2.40e+02$	$\mathbf{1.38e+02}$	$1.51e+02$	$1.97e+02$	$\mathbf{1.18e+02}$
f_6	$9.09e-03$	$7.21e+01$	$\mathbf{6.03e-33}$	$3.70e-03$	$4.48e+01$	$\mathbf{5.42e-26}$
f_7	$2.64e+01$	$4.53e+01$	$\mathbf{1.07e+01}$	$2.13e+01$	$4.13e+01$	$\mathbf{5.78e+00}$
f_8	$2.12e-03$	$1.73e+01$	$\mathbf{6.14e-11}$	$8.63e-04$	$1.17e+01$	$\mathbf{9.08e-13}$
f_9	$3.17e-01$	$3.62e+05$	$\mathbf{1.87e-01}$	$1.77e-01$	$9.10e+03$	$\mathbf{9.34e-02}$

Definitivamente los resultados confirman nuestra teoría. En la inmensa mayoría de comparaciones, el PSO sin inercia pero con incrementos de c_2 cuando $pBest = X$ (PSO3) resulta victorioso. Por otra parte, el PSO sin inercia y con c_2 fijo (PSO2) está muy por debajo en rendimiento de los otros dos. Esto quiere decir que efectivamente tiene mucho que ver en el impacto positivo de la componente inercial su influencia en la gestión de las situaciones en que $pBest = X$. En general, comparando el PSO1 (PSO estándar) con el PSO3, vemos que los resultados son muy parecidos y que ambos métodos son prácticamente equivalentes, como pretendíamos al diseñar el experimento. Es más, salvo en una función (la Rosenbrock f_2), los resultados son mejores en el PSO3. En el caso de la única función que muestra un rendimiento superior del PSO1, realmente la diferencia de resultados con el PSO3 no parece relevante, podríamos decir que prácticamente sus rendimientos son iguales. No obstante, puede que sí (de hecho casi seguro) que la diversidad que introduce la componente inercial cuando $pBest = X$ es mayor que la que se tiene cuando simplemente se incrementa c_2 como en el PSO3, ya que como recordamos del análisis, la diversidad la aportaban los desvíos y los desvíos se logran sobre todo aumentando los giros con respecto al vector social, que es precisamente lo que puede aportar la inercia en estos casos. Cuando $pBest \neq X$ este hecho ya no sería diferencial porque ya existe otro vector (el cognitivo) que introduce una dirección diferente y que va a ayudar a que las direcciones de las partículas sean más diversas. Pero este extra de diversidad que aporta la inercia cuando $pBest = X$ podría ser clave para que el PSO1 rindiese mejor que el

PSO3 en algunas funciones. Y además en especial la función Rosenbrock f_2 es de las pocas no-separables, lo cual implica que su optimización va a depender mucho más de la diversidad en la búsqueda. Y prácticamente lo contrario podríamos decir para las funciones separables, que se benefician más de una búsqueda por dimensiones como la que induce el PSO sin desvíos externos y que por tanto se ven perjudicadas con la inclusión de nuevos atractores que diversifican las direcciones de búsqueda de las partículas y que dificultan que éstas se queden atrapadas en los ejes paralelos a los de coordenadas. Esta lectura es muy importante para comprender por qué el PSO3 funciona mejor que el PSO1 para funciones separables, hecho que se refleja en las tablas de resultados.

Podríamos ahora realizar la misma comparativa pero con las funciones rotadas. Así veremos si el impacto de la inercia en la diversidad de la búsqueda es muy importante o no. Los resultados de esta nueva prueba se muestran en las Tablas 5.4 y 5.5.

Cuadro 5.4: Comparativa entre modelos PSO con y sin inercia para un tamaño de dimensión $d = 30$

Función	$N = 30$			$N = 50$		
	PSO1	PSO2	PSO3	PSO1	PSO2	PSO3
f_{10}	$8.34e-102$	$7.13e+04$	$\mathbf{1.70e-119}$	$3.08e-82$	$2.30e+04$	$\mathbf{2.16e-85}$
f_{11}	$2.10e+01$	$2.69e+02$	$\mathbf{1.99e+01}$	$\mathbf{1.97e+01}$	$1.86e+02$	$2.38e+01$
f_{12}	$3.79e+00$	$1.35e+01$	$\mathbf{3.05e+00}$	$\mathbf{2.62e+00}$	$1.21e+01$	$2.73e+00$
f_{13}	$1.53e-02$	$2.62e+01$	$\mathbf{9.44e-03}$	$\mathbf{1.04e-02}$	$1.02e+01$	$1.09e-02$
f_{14}	$9.58e+01$	$1.21e+02$	$\mathbf{8.46e+01}$	$9.48e+01$	$1.17e+02$	$\mathbf{8.56e+01}$
f_{15}	$\mathbf{6.13e+00}$	$6.31e+01$	$6.72e+00$	$\mathbf{5.43e+00}$	$4.70e+01$	$5.59e+00$
f_{16}	$2.32e+01$	$2.96e+01$	$\mathbf{2.29e+01}$	$2.02e+01$	$2.90e+01$	$\mathbf{1.96e+01}$
f_{17}	$\mathbf{7.04e+00}$	$1.94e+01$	$9.63e+00$	$\mathbf{6.74e+00}$	$1.63e+01$	$7.71e+00$
f_{18}	$8.92e+00$	$2.89e+05$	$\mathbf{8.40e+00}$	$\mathbf{3.87e+00}$	$4.98e+04$	$4.14e+00$

Cuadro 5.5: Comparativa entre modelos PSO con y sin inercia para un tamaño de dimensión $d = 50$

Función	$N = 30$			$N = 50$		
	PSO1	PSO2	PSO3	PSO1	PSO2	PSO3
f_{10}	$1.01e-25$	$1.28e+05$	$\mathbf{5.71e-53}$	$2.91e-20$	$3.29e+04$	$\mathbf{3.97e-38}$
f_{11}	$\mathbf{4.56e+01}$	$4.36e+02$	$5.06e+01$	$\mathbf{4.64e+01}$	$2.67e+02$	$4.99e+01$
f_{12}	$5.90e+00$	$1.64e+01$	$\mathbf{5.31e+00}$	$4.66e+00$	$1.52e+01$	$\mathbf{3.32e+00}$
f_{13}	$\mathbf{6.24e-03}$	$4.67e+01$	$6.90e-03$	$5.91e-03$	$1.27e+01$	$\mathbf{5.83e-03}$
f_{14}	$2.26e+02$	$2.70e+02$	$\mathbf{1.90e+02}$	$2.23e+02$	$2.21e+02$	$\mathbf{1.72e+02}$
f_{15}	$1.29e+02$	$1.46e+02$	$\mathbf{3.16e+01}$	$6.10e+01$	$1.22e+02$	$\mathbf{1.98e+01}$
f_{16}	$5.48e+01$	$5.46e+01$	$\mathbf{4.92e+01}$	$5.19e+01$	$5.52e+01$	$\mathbf{4.79e+01}$
f_{17}	$\mathbf{2.21e+01}$	$4.26e+01$	$2.23e+01$	$2.15e+01$	$4.05e+01$	$\mathbf{1.77e+01}$
f_{18}	$1.85e+01$	$1.09e+06$	$\mathbf{1.54e+01}$	$1.59e+01$	$8.23e+04$	$\mathbf{1.34e+01}$

De nuevo obtenemos unos resultados muy similares a los de las funciones sin rotar. La diferencia entre los métodos PSO1 y PSO3 es prácticamente inapreciable en todos los casos contemplados y en todas las funciones. Las diferencias son tan pequeñas que da la sensación de que son debidas a la aleatoriedad más que a otra cosa. Y nada más lejos de la realidad, la diversidad que hablábamos que podía introducir la inercia y que no podría replicar el PSO3 es totalmente imperceptible, pues no solo el PSO3 obtiene casi los mismos resultados sino que además donde más diferencia encontramos es en la función f_10 y la diferencia es en favor del PSO3, lo cual significa que su diversidad está al nivel o incluso por encima de la del PSO1. Esto también podría deberse al impacto negativo que provocaría la inercia cuando $pBest \neq X$ al restar protagonismo a las otras dos componentes (la social y la cognitiva) que guían la heurística del método.

Sea como fuere, lo que se pretendía mediante este experimento era primero desvelar las claves ya teorizadas en 3.3.1 sobre papel tan decisivo que juega la componente inercial en el PSO, qué es lo que hace que sea tan importante y tenga un impacto tan beneficioso en su rendimiento. Y después, demostrar que esta componente no es necesaria si los demás parámetros del método (c_1 y c_2) son seleccionados con criterio. Hemos conseguido ambas cosas. Por un lado hemos entendido que la gestión de la situación específica en la que $pBest = X$ es la que pone en valor a w y la que hace que la componente inercial sea pieza fundamental en el PSO. Por otro, también

hemos aprendido que no es la única manera de hacerlo y que gestionando la situación de otra forma podemos ahorrarnos una componente estéril con su correspondiente hiperparámetro. Y que de hecho, gestionándola, como decimos, de otra forma (PSO3), tenemos más libertad a la hora de decidir el balance exploración/explotación general del método y es que al gestionar las situaciones $pBest = X$ y $pBest \neq X$ de forma independiente, el PSO3 tiene más margen de maniobra y no condiciona la gestión de $pBest = X$ al resto de situaciones, es decir, no condiciona el balance explorativo general del algoritmo como sí hace la componente inercial. Por supuesto esto no hace sino que ratificar nuestra decisión de no incluir componente inercial en nuestro método final propuesto en 4.6, ya que es que además en el nuestro aparte de haber una gestión diferente para los casos $pBest = X$ y $pBest \neq X$, la diversidad ya está muy bien cubierta siempre y no necesita de una componente extra para ello.

5.2.2. Comparativa de métodos de giro

Otro asunto que podíamos tener pendiente a raíz de nuestro análisis del Capítulo 4 es la comparación experimental en la práctica de los tres métodos de giro para el PSO discutidos en el apartado 4.2.3.3. Estos tres métodos son, recordamos, el WPSO (Ec. 3.18), el RotmPSO (Ec. 3.20 y 3.19) y el nuestro (Algoritmo 4). Los dos primeros se basan en la construcción de matrices de rotación y difieren en la manera de hacerlo. El nuestro en cambio está basado en un sistema diferente. En el apartado 4.2.3.3 ya detallamos y discutimos las características teóricas de cada uno de los métodos, pero quedó en el aire la comparación práctica.

Para llevar a cabo la comparativa práctica, lo primero hay que aclarar ciertos aspectos relativos a la implementación de estos métodos. Como ya expusimos en la discusión teórica del apartado 4.2.3.3, tanto el WPSO como el RotmPSO, al trabajar con matrices de rotación, su complejidad computacional crece muchísimo. La sola generación de la matriz ya tiene un coste mínimo de $\mathcal{O}(d^2)$, a lo que hay que añadir su multiplicación por el vector a rotar, también del mismo orden de complejidad. Pero no solo eso, al situarse dentro del marco del PSO según la regla 3.17, este proceso de generación-multiplicación hay que realizarlo dos veces por partícula (para rotar el vector social por un lado y el cognitivo por otro), que a su vez multiplicado por el número de partículas N resulta en un coste elevadísimo. Eso sin contar que por ejemplo en el caso del RotmPSO el código se vuelve menos paralelizable (la expresión 3.20 cuenta con muchas operaciones secuenciales que no permiten si quiera aprovechar las ventajas de Matlab para paralelizar operaciones y ahorrar tiempo de ejecución). En definitiva, el tiempo de ejecución del algoritmo completo, incluso en dimensiones bajas como $d = 30$ dista en más de un órden de magnitud del de la versión estándar con el sistema de giro clásico. Esto es un problema no solo para la practicidad de los métodos sino también de cara a la comparativa que queremos realizar. Estamos hablando de que de normal la evaluación de una función tiene un orden de complejidad menor que la generación de la matriz de rotación, luego por tanto en este caso el criterio establecido en las bases experimentales de medir el coste en FEs no es válido. Si queremos que realmente la comparativa sea justa necesitamos que los tiempos que consumen sean parecidos al menos para las condiciones en las que se va a realizar la comparativa (los tamaños de dimensión y enjambre). Y echando un vistazo a los apartados experimentales de los artículos originales, rápidamente nos damos cuenta de que la solución que toman para rebajar el coste computacional de sus propuestas es implementarlas utilizando solamente una única matriz de rotación que sea com-

partida para la rotación de todos los vectores, reduciendo así ostensiblemente los costes por generación y multiplicación de matrices $d \times d$.

Nosotros generaremos una única matriz de rotación y la usaremos para los dos vectores social y cognitivo de cada una de las N partículas del enjambre. Podemos permitirnos compartir la matriz de rotación debido a que ésta no afecta igual a los distintos vectores que van a ser rotados, sino que las rotaciones que produce dependen de dichos vectores y sus relaciones con el eje de rotación (este hecho también fue discutido en su momento), algunos efectuando mayores cambios direccionales que otros y sobre todo manteniendo direcciones de movimiento diferentes (quedamos que era la clave de los giros), conservando así en todo momento la diversidad en la búsqueda que es lo que se pretende con los giros. De hecho hemos podido comprobarlo comparando las dos implementaciones (la que genera una matriz por cada partícula y vector y la que genera una única matriz para todas las partículas y vectores) y viendo que la diferencia es ínfima. Esto en cambio no sucede en nuestro método, que necesita repetir la operación de giro para cada partícula de manera individualizada para conseguir esta diversidad. Así las cosas, puesto que estamos tratando con tamaños de dimensión relativamente bajos, los costes temporales de cada método se igualan y podemos compararlos en equidad.

Una vez resuelto el tema de la implementación de los métodos de giro, tenemos que concretar sus hiperparámetros. No es difícil ya que cada método solo tiene uno. En el WPSO el hiperparámetro es α, que es el ángulo de rotación en grados y que los autores establecen desde un primer momento a 3. Mientras, en el RotmPSO el hiperparámetro es σ, que es la desviación típica de los ángulos de rotación en grados por los diferentes planos principales. Basándonos en el valor que utilizan en [125] para la experimentación del RotmPSO, diríamos que $\sigma = 2$ es el valor elegido. Por tanto siguiendo el criterio de sus autores, nosotros hemos realizado la misma selección. Y para nuestro método de giro cuyo hiperparámetro es α representando el ángulo de giro (y de cambio direccional) en grados, hemos seleccionado un valor de 10 tras un testeo manual rápido con varios valores. La lista final de configuraciones es:

- WPSO: $\alpha = 3$
- RotmPSO: $\sigma = 2$
- Propuesta: $\alpha = 10$

Todos estos métodos de giro los vamos a probar sobre la regla estándar de actualización, o sea, sustituyendo el procedimiento de giro estándar del PSO (el escalamiento direccional) por el de cada uno de los tres métodos $\texttt{Gira}_k$ a comparar, con k el índice que identifica a cada uno:

$$V = wV + c_1 rand_1 \texttt{Gira}_k(pBest - X) + c_2 rand_2 \texttt{Gira}_k(gBest - X)$$

La configuración de este PSO será la comunmente establecida y que hemos utilizado en el primer experimento, que consiste en un $w = 0.72$ y $c_1 = c_2 = 1.49$. Teníamos que decantarnos por un PSO para establecer la comparativa y lo hemos hecho por el más típico y extendido, pero es necesario que tengamos en cuenta que los resultados no tienen por qué generalizarse a otros PSO diferentes, en los cuales quizá destaquen otros métodos distintos. Es decir, que un método obtenga peores resultados en esta comparativa no quiere decir que sea peor método de giro, sino que se adecúa peor a la versión estándar del PSO que estamos considerando, si bien es cierto que los

tres teóricamente persiguen el mismo objetivo aunque hacerlo de maneras distintas pueda cambiar sus rendimientos.

La comparación se ha llevado a cabo para los tamaños de dimensión $d = 30$ y $d = 50$ y los de enjambre $N = 30$ y $N = 50$. Cada una de las comparaciones se corresponde con una de las siguientes Tablas de resultados 5.6 y 5.7.

Los resultados indican que hay bastante igualdad entre los tres métodos. Salvo en muy contadas excepciones, ningún método destaca de forma muy clara sobre el resto y el orden de magnitud de los resultados suele ser el mismo para los tres. Ahora bien, si nos ponemos a contar el número de funciones en las que cada método es mejor (obtiene resultados superiores a los otros dos), vemos como el nuestro gana en las cuatro comparaciones (para todas las combinaciones (d, N) probadas). Y sobre todo, es más superior cuando $N = 50$, lo que nos hace pensar que cuantas más partículas haya, más potencial tiene nuestra propuesta de giro. De todos modos, esta igualdad en resultados deja de manifiesto que en el PSO estándar no es demasiado importante la precisión de los giros ya que, entre otras cosas, aparte de existir la inercia que ya aporta algo de diversidad y resta importancia a la diversidad introducida a través de los otros componentes, el algoritmo está repleto de aleatoriedad. Por eso es que al final los métodos de giro no destacan tanto y se ven más o menos iguales. Probablemente en un PSO en el que la aleatoriedad no jugase un papel tan importante se podrían analizar más diferencias entre los métodos de giro. Aún así, insistimos, nuestra propuesta de giro no solo es competente y equiparable a las demás sino que incluso podemos decir que obtiene ligeramente mejores resultados.

Cuadro 5.6: Comparativa entre métodos de giro del PSO estándar para un tamaño de dimensión $d = 30$

Función	$N = 30$			$N = 50$		
	WPSO	RotmPSO	Propuesta	WPSO	RotmPSO	Propuesta
f_1	$2.06e-123$	$7.07e-147$	$\mathbf{1.03e-161}$	$1.59e-96$	$8.46e-105$	$\mathbf{4.20e-119}$
f_2	$1.36e+01$	$1.57e+01$	$\mathbf{9.73e+00}$	$1.05e+01$	$1.31e+01$	$\mathbf{7.44e+00}$
f_3	$6.86e+00$	$\mathbf{3.81e+00}$	$4.18e+00$	$5.43e+00$	$\mathbf{3.19e+00}$	$3.73e+00$
f_4	$9.92e-03$	$\mathbf{8.77e-03}$	$1.04e-02$	$1.43e-02$	$\mathbf{8.13e-03}$	$1.23e-02$
f_5	$\mathbf{6.47e+01}$	$7.23e+01$	$7.46e+01$	$\mathbf{5.76e+01}$	$6.60e+01$	$6.13e+01$
f_6	$\mathbf{1.93e+00}$	$4.16e+00$	$2.20e+00$	$\mathbf{1.29e+00}$	$2.01e+00$	$1.64e+00$
f_7	$2.04e+01$	$1.88e+01$	$\mathbf{1.81e+01}$	$1.83e+01$	$1.68e+01$	$\mathbf{1.67e+01}$
f_8	$\mathbf{1.62e+00}$	$2.47e+00$	$1.67e+00$	$\mathbf{5.26e-01}$	$1.51e+00$	$1.14e+00$
f_9	$8.35e+00$	$\mathbf{6.60e+00}$	$6.69e+00$	$4.30e+00$	$3.65e+00$	$\mathbf{3.00e+00}$
f_{10}	$8.98e-124$	$4.08e-146$	$\mathbf{7.72e-162}$	$2.09e-95$	$3.20e-105$	$\mathbf{1.50e-119}$
f_{11}	$1.37e+01$	$1.39e+01$	$\mathbf{9.52e+00}$	$1.11e+01$	$1.44e+01$	$\mathbf{7.87e+00}$
f_{12}	$6.70e+00$	$\mathbf{3.59e+00}$	$4.54e+00$	$5.14e+00$	$\mathbf{3.32e+00}$	$3.54e+00$
f_{13}	$1.59e-02$	$1.40e-02$	$\mathbf{1.39e-02}$	$1.39e-02$	$1.49e-02$	$\mathbf{1.25e-02}$
f_{14}	$\mathbf{6.48e+01}$	$7.00e+01$	$6.62e+01$	$6.61e+01$	$6.75e+01$	$\mathbf{6.58e+01}$
f_{15}	$\mathbf{2.32e+00}$	$4.70e+00$	$2.90e+00$	$\mathbf{1.08e+00}$	$2.56e+00$	$2.02e+00$
f_{16}	$1.93e+01$	$\mathbf{1.91e+01}$	$1.97e+01$	$1.93e+01$	$1.87e+01$	$\mathbf{1.82e+01}$
f_{17}	$1.87e+00$	$3.91e+00$	$\mathbf{1.73e+00}$	$\mathbf{1.17e+00}$	$2.97e+00$	$1.62e+00$
f_{18}	$7.36e+00$	$8.05e+00$	$\mathbf{4.37e+00}$	$3.06e+00$	$6.24e+00$	$\mathbf{2.70e+00}$

Cuadro 5.7: Comparativa entre métodos de giro del PSO estándar para un tamaño de dimensión $d = 50$

Función	$N = 30$			$N = 50$		
	WPSO	RotmPSO	Propuesta	WPSO	RotmPSO	Propuesta
f_1	$9.26e-78$	$7.71e-79$	$\mathbf{1.30e-95}$	$9.65e-59$	$1.85e-55$	$\mathbf{1.83e-72}$
f_2	$4.17e+01$	$4.19e+01$	$\mathbf{3.95e+01}$	$4.26e+01$	$4.23e+01$	$\mathbf{3.96e+01}$
f_3	$8.15e+00$	$\mathbf{4.14e+00}$	$7.23e+00$	$6.42e+00$	$\mathbf{3.71e+00}$	$5.93e+00$
f_4	$9.18e-03$	$\mathbf{3.78e-03}$	$6.15e-03$	$8.29e-03$	$\mathbf{6.89e-03}$	$7.06e-03$
f_5	$\mathbf{1.17e+02}$	$1.41e+02$	$1.17e+02$	$\mathbf{1.16e+02}$	$1.31e+02$	$1.18e+02$
f_6	$\mathbf{7.32e+00}$	$1.25e+01$	$8.40e+00$	$\mathbf{6.11e+00}$	$1.04e+01$	$6.12e+00$
f_7	$3.95e+01$	$\mathbf{3.82e+01}$	$3.87e+01$	$3.71e+01$	$\mathbf{3.58e+01}$	$3.69e+01$
f_8	$7.93e+00$	$1.33e+01$	$\mathbf{7.18e+00}$	$5.66e+00$	$8.93e+00$	$\mathbf{4.85e+00}$
f_9	$1.39e+01$	$1.46e+01$	$\mathbf{1.31e+01}$	$\mathbf{1.13e+01}$	$1.13e+01$	$1.15e+01$
f_{10}	$1.62e-78$	$1.36e-78$	$\mathbf{1.27e-95}$	$4.42e-59$	$6.10e-56$	$\mathbf{4.57e-72}$
f_{11}	$4.32e+01$	$4.12e+01$	$\mathbf{3.93e+01}$	$4.45e+01$	$4.40e+01$	$\mathbf{3.98e+01}$
f_{12}	$7.66e+00$	$\mathbf{4.04e+00}$	$6.81e+00$	$6.30e+00$	$\mathbf{3.91e+00}$	$6.23e+00$
f_{13}	$\mathbf{4.27e-03}$	$8.70e-03$	$7.63e-03$	$5.91e-03$	$\mathbf{3.45e-03}$	$8.29e-03$
f_{14}	$\mathbf{1.16e+02}$	$1.34e+02$	$1.26e+02$	$\mathbf{1.10e+02}$	$1.19e+02$	$1.11e+02$
f_{15}	$\mathbf{9.40e+00}$	$1.26e+01$	$9.84e+00$	$7.45e+00$	$1.23e+01$	$\mathbf{6.81e+00}$
f_{16}	$4.07e+01$	$\mathbf{3.72e+01}$	$3.95e+01$	$3.65e+01$	$3.64e+01$	$\mathbf{3.63e+01}$
f_{17}	$\mathbf{6.69e+00}$	$1.26e+01$	$7.41e+00$	$5.91e+00$	$1.00e+01$	$\mathbf{5.44e+00}$
f_{18}	$1.48e+01$	$1.53e+01$	$\mathbf{1.33e+01}$	$1.12e+01$	$1.11e+01$	$\mathbf{9.62e+00}$

5.2.3. Configuración de nuestra propuesta

Ahora vamos a probar los resultados de nuestra propuesta PSO dada por el Algoritmo 6. El objetivo es comprobar que el método propuesto funciona correctamente según lo previsto y confirmar así la validez de su confección. Para ello, probaremos con distintas configuraciones paramétricas. En esencia, lo que tenemos que ver es que los resultados obtenidos para los distintas configuraciones son coherentes con lo esperado. En otras palabras, probaremos con distintos valores de `exploration_level` para ver si efectivamente el nivel de exploración guarda correlación directa con este parámetro. Si es así, en el fondo se estará confirmando que la lógica que hay detrás del método es la adecuada.

El método implementado en Matlab (código del Algoritmo 6) que usaremos en este experimento y en el siguiente se adjunta en el Anexo .1. Los valores del hiperparámetro `exploration_level` que han sido elegidos para el experimento son $\{0.3, 0.5, 0.8, 0.9, 0.95\}$. Se han elegido estos valores por dos motivos. Primero porque cubren bastante bien el espacio paramétrico (representan bien el dominio de `exploration_level`) y por tanto permiten observar la incidencia del hiperparámetro en el método y después porque el buen rendimiento en todos los PSO incluido el nuestro, suele alcanzarse con niveles de exploración altos, por lo que la distribución de los valores seleccionados para el experimento está centrada en valores altos de `exploration_level`, que en la mayoría de los casos serán los únicos a tener en consideración. De este modo podemos también extraer una guía de este experimento para realizar la selección paramétrica.

El objetivo en este caso no es hacer una comparativa exhaustiva entre las distintas configuraciones paramétricas seleccionadas, nos vale con ver cómo se comporta el método en cada una. Aun así queremos evaluarlo en distintas situaciones para tener una radiografía completa de su funcionamiento. Por ello, seguiremos probando con distintos tamaños poblacionales N ($N = 30$ y $N = 50$) y con distintas dimensiones d, de nuevo tomaremos $d = 30$ y $d = 50$. Los resultados del experimento se recogen en las Tablas 5.8, 5.9, 5.10 y 5.11.

Cuadro 5.8: Comparativa entre configuraciones paramétricas de nuestra propuesta PSO para un tamaño de dimensión $d = 30$ y de población $N = 30$

Función	exp_level = 0.3	exp_level = 0.5	exp_level = 0.8	exp_level = 0.9	exp_level = 0.95
f_1	$\mathbf{7.41e-323}$	$3.05e-291$	$3.29e-74$	$8.86e-27$	$2.91e-09$
f_2	$1.79e+01$	$\mathbf{1.36e+01}$	$2.10e+01$	$2.42e+01$	$2.61e+01$
f_3	$2.83e+00$	$1.02e+00$	$\mathbf{7.88e-15}$	$1.18e-14$	$5.22e-06$
f_4	$8.35e-03$	$7.39e-03$	$3.28e-03$	$\mathbf{3.70e-18}$	$2.36e-07$
f_5	$8.88e+01$	$5.90e+01$	$\mathbf{2.83e+01}$	$1.15e+02$	$1.52e+02$
f_6	$1.28e+01$	$1.75e+00$	$7.82e-08$	$\mathbf{2.04e-13}$	$7.87e-05$
f_7	$1.99e+01$	$1.12e+01$	$2.08e+00$	$6.36e-01$	$\mathbf{4.15e-01}$
f_8	$8.17e+00$	$3.81e+00$	$2.00e-01$	$\mathbf{1.93e-01}$	$9.22e+00$
f_9	$9.19e+00$	$5.78e-01$	$3.46e-03$	$\mathbf{9.08e-23}$	$2.86e-08$
f_{10}	$\mathbf{8.40e-323}$	$1.09e-290$	$4.34e-74$	$1.12e-26$	$2.77e-09$
f_{11}	$1.81e+01$	$\mathbf{1.37e+01}$	$2.11e+01$	$2.44e+01$	$2.63e+01$
f_{12}	$3.17e+00$	$7.08e-01$	$\mathbf{7.76e-15}$	$1.17e-14$	$5.08e-06$
f_{13}	$7.63e-03$	$5.34e-03$	$1.89e-03$	$9.86e-04$	$\mathbf{2.49e-07}$
f_{14}	$8.77e+01$	$7.13e+01$	$\mathbf{3.12e+01}$	$1.00e+02$	$1.56e+02$
f_{15}	$1.58e+01$	$3.13e+00$	$2.47e-01$	$\mathbf{2.63e-13}$	$1.00e-04$
f_{16}	$1.95e+01$	$1.30e+01$	$2.39e+00$	$\mathbf{2.88e-01}$	$9.50e-01$
f_{17}	$8.46e+00$	$4.06e+00$	$\mathbf{7.00e-02}$	$1.74e-01$	$1.06e+01$
f_{18}	$9.06e+00$	$8.80e-01$	$\mathbf{1.63e-32}$	$6.09e-23$	$5.03e-08$

Cuadro 5.9: Comparativa entre configuraciones paramétricas de nuestra propuesta PSO para un tamaño de dimensión $d = 30$ y de población $N = 50$

Función	exp_level = 0.3	exp_level = 0.5	exp_level = 0.8	exp_level = 0.9	exp_level = 0.95
f_1	$\mathbf{1.69e-280}$	$3.13e-189$	$6.92e-45$	$1.07e-14$	$8.98e-04$
f_2	$1.86e+01$	$\mathbf{1.65e+01}$	$2.29e+01$	$2.58e+01$	$2.69e+01$
f_3	$1.69e+00$	$5.34e-01$	$\mathbf{7.76e-15}$	$7.07e-09$	$3.01e-03$
f_4	$9.02e-03$	$5.17e-03$	$1.48e-03$	$\mathbf{2.47e-04}$	$1.85e-03$
f_5	$8.88e+01$	$6.12e+01$	$\mathbf{2.99e+01}$	$1.35e+02$	$1.67e+02$
f_6	$8.94e+00$	$9.75e-01$	$\mathbf{6.01e-11}$	$6.70e-08$	$1.97e-02$
f_7	$1.65e+01$	$9.07e+00$	$1.78e+00$	$\mathbf{1.08e-01}$	$1.74e+00$
f_8	$6.16e+00$	$2.41e+00$	$\mathbf{4.09e-02}$	$4.37e-02$	$9.50e+00$
f_9	$4.71e+00$	$4.69e-01$	$\mathbf{7.46e-30}$	$7.88e-14$	$1.42e-04$
f_{10}	$\mathbf{1.12e-279}$	$5.06e-189$	$6.41e-45$	$9.84e-15$	$8.56e-04$
f_{11}	$1.92e+01$	$\mathbf{1.62e+01}$	$2.28e+01$	$2.60e+01$	$2.72e+01$
f_{12}	$2.03e+00$	$2.42e-01$	$\mathbf{7.76e-15}$	$6.77e-09$	$2.96e-03$
f_{13}	$7.22e-03$	$7.47e-03$	$2.46e-03$	$\mathbf{2.47e-04}$	$1.49e-03$
f_{14}	$7.72e+01$	$5.92e+01$	$\mathbf{2.52e+01}$	$1.28e+02$	$1.62e+02$
f_{15}	$1.08e+01$	$2.26e+00$	$1.97e-02$	$\mathbf{8.07e-08}$	$2.32e-02$
f_{16}	$1.71e+01$	$9.27e+00$	$2.10e+00$	$\mathbf{3.38e-01}$	$2.49e+00$
f_{17}	$6.12e+00$	$2.20e+00$	$\mathbf{2.40e-02}$	$1.08e-01$	$1.06e+01$
f_{18}	$5.37e+00$	$5.32e-01$	$\mathbf{1.68e-32}$	$5.26e-14$	$1.96e-04$

Cuadro 5.10: Comparativa entre configuraciones paramétricas de nuestra propuesta PSO para un tamaño de dimensión $d = 50$ y de población $N = 30$

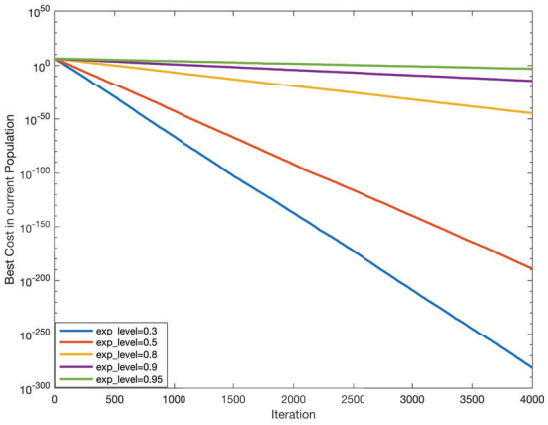
Función	exp_level = 0.3	exp_level = 0.5	exp_level = 0.8	exp_level = 0.9	exp_level = 0.95
f_1	$\mathbf{1.04e-242}$	$4.75e-189$	$1.34e-48$	$4.92e-15$	$5.94e-03$
f_2	$4.00e+01$	$\mathbf{3.71e+01}$	$4.44e+01$	$4.69e+01$	$4.74e+01$
f_3	$6.64e+00$	$2.19e+00$	$7.14e-02$	$\mathbf{3.18e-09}$	$4.64e-03$
f_4	$4.92e-03$	$3.78e-03$	$\mathbf{3.33e-16}$	$2.47e-04$	$2.32e-03$
f_5	$1.52e+02$	$1.16e+02$	$\mathbf{5.98e+01}$	$2.46e+02$	$3.38e+02$
f_6	$3.20e+01$	$9.47e+00$	$8.73e-01$	$4.78e-02$	$\mathbf{4.08e-02}$
f_7	$4.15e+01$	$2.67e+01$	$6.62e+00$	$\mathbf{1.77e+00}$	$2.16e+00$
f_8	$2.17e+01$	$1.05e+01$	$7.18e-01$	$\mathbf{8.69e-02}$	$2.03e+01$
f_9	$1.95e+01$	$8.70e+00$	$6.22e-03$	$\mathbf{8.09e-16}$	$6.75e-05$
f_{10}	$\mathbf{5.44e-241}$	$5.92e-189$	$8.13e-49$	$4.90e-15$	$5.83e-03$
f_{11}	$4.02e+01$	$\mathbf{3.96e+01}$	$4.40e+01$	$4.66e+01$	$4.75e+01$
f_{12}	$6.98e+00$	$2.11e+00$	$\mathbf{1.07e-14}$	$3.15e-09$	$4.47e-03$
f_{13}	$3.61e-03$	$5.50e-03$	$1.23e-03$	$\mathbf{2.41e-11}$	$2.53e-03$
f_{14}	$1.50e+02$	$1.06e+02$	$\mathbf{5.85e+01}$	$2.34e+02$	$3.32e+02$
f_{15}	$3.28e+01$	$1.10e+01$	$1.49e+00$	$\mathbf{2.20e-02}$	$4.81e-02$
f_{16}	$4.06e+01$	$2.66e+01$	$7.89e+00$	$\mathbf{2.83e+00}$	$3.35e+00$
f_{17}	$2.09e+01$	$1.09e+01$	$9.84e-01$	$\mathbf{2.40e-01}$	$2.12e+01$
f_{18}	$1.86e+01$	$6.85e+00$	$2.07e-03$	$\mathbf{8.18e-16}$	$9.33e-05$

Cuadro 5.11: Comparativa entre configuraciones paramétricas de nuestra propuesta PSO para un tamaño de dimensión $d = 50$ y de población $N = 50$

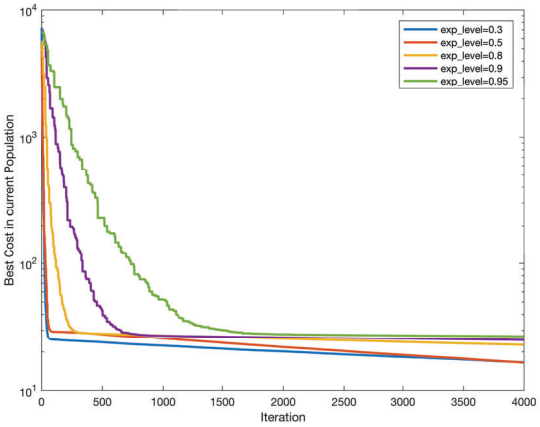
Función	exp_level = 0.3	exp_level = 0.5	exp_level = 0.8	exp_level = 0.9	exp_level = 0.95
f_1	$\mathbf{1.20e-174}$	$4.24e-126$	$6.29e-29$	$3.27e-07$	$1.14e+01$
f_2	$4.19e+01$	$\mathbf{4.02e+01}$	$4.59e+01$	$4.72e+01$	$4.78e+01$
f_3	$3.63e+00$	$1.42e+00$	$\mathbf{1.04e-14}$	$2.43e-05$	$3.24e-01$
f_4	$6.89e-03$	$3.61e-03$	$\mathbf{2.22e-16}$	$9.20e-07$	$4.83e-01$
f_5	$1.40e+02$	$1.05e+02$	$\mathbf{5.68e+01}$	$2.83e+02$	$3.52e+02$
f_6	$2.28e+01$	$6.24e+00$	$4.46e-01$	$\mathbf{1.75e-04}$	$1.17e+00$
f_7	$3.59e+01$	$2.28e+01$	$4.90e+00$	$\mathbf{1.19e+00}$	$7.66e+00$
f_8	$1.37e+01$	$7.57e+00$	$3.03e-01$	$\mathbf{8.08e-02}$	$1.94e+01$
f_9	$1.53e+01$	$5.03e+00$	$2.07e-03$	$\mathbf{1.84e-09}$	$3.15e-02$
f_{10}	$\mathbf{3.52e-174}$	$2.04e-126$	$7.32e-29$	$2.77e-07$	$1.11e+01$
f_{11}	$4.14e+01$	$\mathbf{3.98e+01}$	$4.58e+01$	$4.73e+01$	$4.79e+01$
f_{12}	$3.47e+00$	$1.41e+00$	$\mathbf{1.01e-14}$	$2.39e-05$	$3.27e-01$
f_{13}	$5.67e-03$	$5.34e-03$	$1.64e-03$	$\mathbf{8.23e-04}$	$4.96e-01$
f_{14}	$1.48e+02$	$1.05e+02$	$\mathbf{5.66e+01}$	$2.83e+02$	$3.53e+02$
f_{15}	$2.41e+01$	$6.23e+00$	$2.13e-01$	$\mathbf{1.87e-04}$	$1.29e+00$
f_{16}	$3.64e+01$	$2.21e+01$	$5.36e+00$	$\mathbf{1.96e+00}$	$9.18e+00$
f_{17}	$1.60e+01$	$7.81e+00$	$4.07e-01$	$\mathbf{7.51e-02}$	$2.25e+01$
f_{18}	$1.40e+01$	$4.08e+00$	$6.22e-03$	$\mathbf{1.63e-09}$	$3.10e-02$

Los resultados parecen evidenciar, lo primero, que nuestro método es rotacionalmente invariante (lo sabemos por el análisis teórico) al comparar la equidad entre los resultados obtenidos para las funciones sin rotar y sus versiones rotadas. Y más allá de eso, también confirman la hipótesis del experimento, ya que los resultados varían con `exploration_level`, siendo en general pobres en los dos valores extremos probados para el hiperparámetro, en un caso por sobreexplotación y en otro, por sobreexploración. Realmente esto lo podemos constatar ya que por ejemplo la función Sphere (f_1), que es una función muy sencilla de optimizar (unimodal, convexa), da siempre el mejor resultado con el valor más bajo de `exploration_level` (0.3) y conforme se incrementa el parámetro, sus resultados decaen. Sin embargo, las funciones más complejas con múltiples óptimos locales, que son la gran mayoría, arrojan los mejores resultados con valores más altos del hiperparámetro. Asimismo, en general el valor óptimo del hiperparámetro crece con el número de dimensiones, pues vemos que los mejores resultados obtenidos en la Tabla 5.11 requieren de media de un `exploration_level` algo más elevado que los mejores de la Tabla 5.9. Los mejores resultados se obtienen generalmente con configuraciones de `exploration_level` entre 0.8 y 0.9. Es lo que hablábamos de que los métodos PSO encuentran el rendimiento sobre todo en las configuraciones más exploradoras. Pero el máximo nivel de exploración tampoco es deseable, tiene que existir un equilibrio como así demuestran los resultados obtenidos para `exploration_level` $= 0.95$.

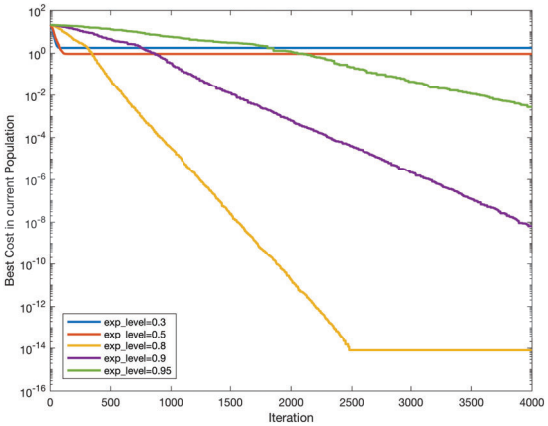
Para quizá reflejar más claramente el estado de exploración/explotación de cada configuración, hemos decidido incorporar algunas gráficas comparativas de la evolución del algoritmo para cada una de las cinco configuraciones. Así como hicimos en el Capítulo 4, estas gráficas representan el mejor valor obtenido por el algoritmo hasta cada iteración t, $1 \leq t \leq maxIter$. Esto es, $f(gBest^{(t)})$. Estas gráficas permiten comprobar fácilmente cuándo un método es explorativo y cuándo es explotativo. Como el método es claramente invariante a rotaciones, hemos decidido solamente representar las nueve primeras funciones. Además, en vista de la coherencia observada en los resultados para las distintas dimensiones y tamaños de enjambre, se ha decidido representar únicamente la dimensión $d = 30$ y el tamaño de enjambre $N = 50$ por considerarlos los valores más habituales en las comparativas. Las gráficas se han obtenido ejecutando el algoritmo varias veces y tomando solo aquellas ejecuciones más representativas y coherentes por supuesto con los resultados recogidos en la Tabla 5.9. Todas ellas se presentan en la Figura 5.1.



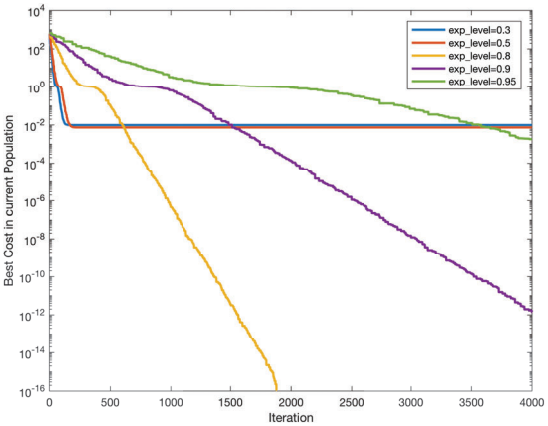
(a) Función Sphere f_1



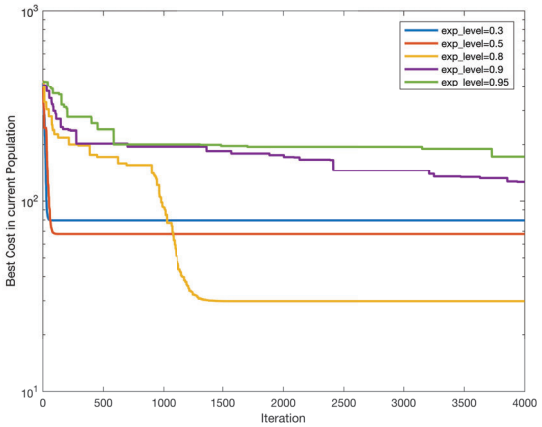
(b) Función Rosenbrock f_2



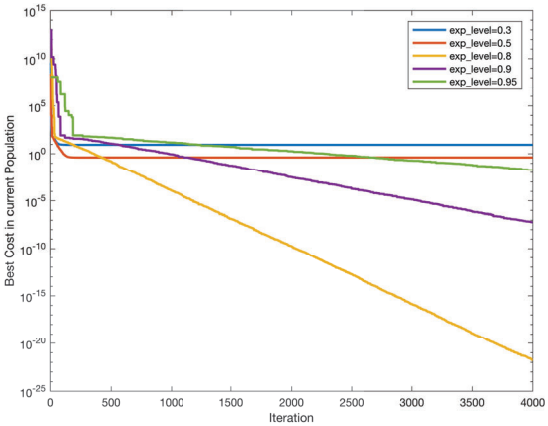
(c) Función Ackley f_3



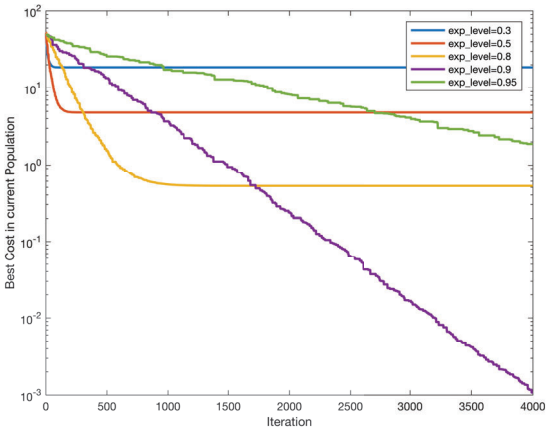
(d) Función Griewank f_4



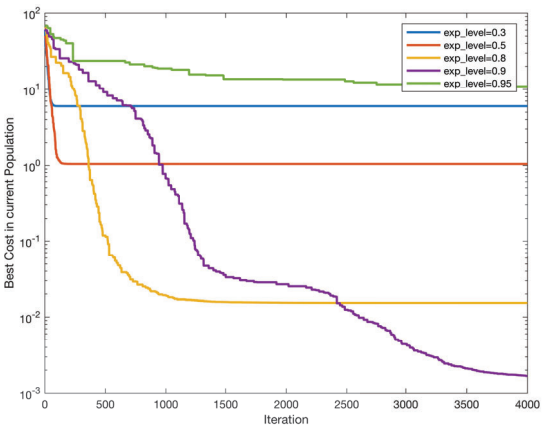
(e) Función Rastrigin f_5



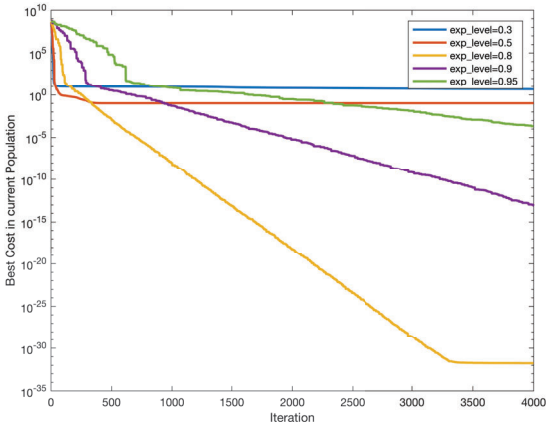
(f) Función Schwefel 2.22 f_6



(g) Función Weierstrass f_7



(h) Función Alpine f_8



(i) Función Generalized Penalized f_9

Figura 5.1: Representaciones gráficas de la evolución del mejor fitness encontrado por cada configuración paramétrica de nuestra propuesta PSO.

Como se puede ver en cada una de las figuras, las líneas de color verde y morado son las únicas que no convergen en ninguna de las funciones representadas. Son las curvas de las configuraciones $exploration\_level = 0.9$ y $exploration\_level = 0.95$. Esto ya nos indica que son las dos configuraciones más explorativas que hemos probado. Además la pendiente de la curva correspondiente a la configuración $exploration\_level = 0.95$ (la verde) siempre es la más pequeña de todas porque es la que más lentamente converge. Y podríamos aplicar la misma lógica con el resto de curvas y veremos que el orden del nivel de exploración de cada configuración se corresponde con el de la pendiente de su curva. Las configuraciones más explotativas convergen en muchos casos prematuramente, aunque eso no significa que sus resultados sean peores, como podemos apreciar por ejemplo con la función Rastrigin f_5, que a la vista está que es la función más complicada de optimizar. La gráfica de la función Rosenbrock f_2 es quizá la más curiosa de todas. Esta gráfica aparenta que todas las curvas han convergido. Sin embargo, si nos fijamos en detalle vemos que esto no es cierto, aunque las pendientes son muy pequeñas. Sabemos que esto es cosa de la forma de la función ya que precisamente las configuraciones que presentan una pendiente mayor son las más explotativas (por orden de mayor a menor nivel de explotación). La gráfica nos muestra como hay dos partes muy diferenciadas en las curvas. Hay una parte primera que dura muy pocas iteraciones iniciales y en la que la optimización

es muy sencilla y luego hay otra en la que el progreso es muy lento, aunque el $gBest$ está en constante movimiento, no se estanca pero avanza muy despacio. Quizá esta función al presentar dos fases de búsqueda tan distintas invitaría a diseñar algún mecanismo de ajuste dinámico del hiperparámetro. Por lo demás las gráficas confirman que los mejores resultados se obtienen generalmente con configuraciones de `exploration_level` entre 0.8 y 0.9. Hay que dejar claro que en este experimento no se ha tratado de seleccionar la mejor configuración para cada función, sino que se ha probado con configuraciones tipo fijas. Es decir, los resultados mostrados podrían ser aún mejores.

5.2.4. Comparativa de variantes PSO

Tras haber demostrado experimentalmente que nuestra propuesta (Algoritmo 6) funciona de acuerdo a lo previsto y que su único hiperparámetro tiene un sentido y una relación directa con el equilibrio explorativo del método que facilita enormemente la labor de su selección, ahora hemos de demostrar su eficacia. La demostraremos comparando nuestra propuesta con otras variantes PSO visitadas durante el trabajo en el Capítulo 3. En concreto serán siete las variantes que serán comparadas con la nuestra. Son las siguientes: PSO estándar (Ec. 2.2), BareBonesPSO (Ec. 3.6), normLinkedPSO (Ec. 3.21), SPSO11 (Ec. 3.24 y 3.23), MSPSO11 (Ec. 3.25 y 3.23), LcRiPSO (Ec. 3.26) y HRiPSO (Ec. 3.27 y 3.28). Sobre todo se han buscado variantes que tengan un perfil parecido a la nuestra, que hayan surgido del análisis de las características del PSO, casi todas concebidas para solucionar la varianza rotacional del PSO estándar. También se ha seleccionado el propio PSO estándar para tener una referencia en la comparativa.

Los hiperparámetros asociados a cada una de las variantes se han escogido en la medida de lo posible siguiendo las indicaciones de los correspondientes artículos en los que fueron presentadas. Ahora bien, a la hora de implementar las variantes, nos hemos dado cuenta de que algunas están construidas sobre otras versiones PSO que no son la estándar, véase versiones que emplean topologías diferentes para la estructura del enjambre, versiones asíncronas, etc. Por supuesto muchas están construidas sobre PSOs con heurísticas para inicialización de las partículas o gestión de límites de posiciones o velocidades, etc. Nosotros, como ya explicamos en la Sección 5.1, con el fin de ser justos y comparar todas las variantes en igualdad de condiciones y que la comparativa solo se ciña a la regla de actualización de posiciones, tenemos que eliminar toda clase de heurísticas añadidas y construir las variantes sobre las bases del PSO estándar. Lo que ocurre es que entonces las recomendaciones paramétricas de los autores de las variantes ya no son válidas en algunos de esos casos. Para esas situaciones, se ha llevado a cabo un ajuste paramétrico manual para dar con los hiperparámetros que logran los mejores resultados para las funciones de test que estamos considerando. Hay que decir que las variantes que han requerido de este ajuste han sido una minoría ya que casi todas conservaban su rendimiento al trasladarse al marco estándar exigido para el experimento. El resumen de las configuraciones finales utilizadas por cada variante se presenta en la Tabla 5.12. Nótese que la configuración elegida para el PSO estándar es la utilizada en el primero de los experimentos por ser una de las más comunes y utilizadas (probablemente la más empleada) y que es la equivalente a la variante PSO constrictivo de la ecuación 3.4 con $c_1 = c_2 = 2.05$, la cual es muy popular. Así en el fondo la comparativa la estamos realizando también con esta variante. Por otra parte, la configuración elegida para

nuestra propuesta se basa en los resultados del experimento anterior, decidiéndose que `exploration_level` $= 0.9$.

Cuadro 5.12: Configuraciones paramétricas escogidas para las distintas variantes que forman parte de la comparativa

Variante	Configuración paramétrica
PSO	$w = 0.72$ $c_1 = c_2 = 1.49$
BareBonesPSO	$c_1 = c_2$
normLinkedPSO	$w = 0.6$ $c_1 = c_2 = 0.194$
SPSO11	$w = 1/(2 * \ln 2)$ $c_1 = c_2 = 2$
MSPSO11	$w = 1/(2 * \ln 2)$ $c_1 = c_2 = 0.5 + \ln 2$
LcRiPSO	$w = 0.72$ $c_1 = c_2 = 1.49$ $\sigma_1 = (pBest = X) * (l * \sigma_1) + (pBest \neq X) * (l * \|pBest - X\|)$ $\sigma_2 = (gBest = X) * (l * \sigma_2) + (gBest \neq X) * (l * \|gBest - X\|)$ $l = 0.91 * 0.51/(N^{0.21} * d^{0.58})$
HRiPSO	$w = 0.6$ $c_1 = c_2 = 1.49$ $c_d = 0.5$
Propuesta	`exploration_level` $= 0.9$

En esta ocasión incluimos también la dimensionalidad $d = 10$ a la comparativa, por lo que en total se tratan las dimensiones $d = 10$, $d = 30$ y $d = 50$. Los tamaños de población siguen siendo $N = 30$ y $N = 50$. De nuevo, todos los métodos de la comparativa emplean un número de evaluaciones de función (FEs) por iteración igual a N, de modo que $maxIter = \frac{2 \times 10^5}{N}$ para todos ellos. Los resultados de la comparativa para cada dimensión y cada tamaño de enjambre examinados se muestran en las Tablas 5.13, 5.14, 5.15, 5.16, 5.17 y 5.18.

Cuadro 5.13: Comparativa entre variantes PSO para un tamaño de dimensión $d = 10$ y de población $N = 30$

Función	PSO	BareBonesPSO	normLinkedPSO	SPSO11	MSPSO11	LcRiPSO	HRiPSO	Propuesta
f_1	$\mathbf{0.00e+00}$	$\mathbf{0.00e+00}$	$4.94e-324$	$\mathbf{0.00e+00}$	$\mathbf{0.00e+00}$	$4.94e-324$	$3.06e-137$	$2.09e-110$
f_2	$6.84e-01$	$2.81e-01$	$1.21e+01$	$\mathbf{2.68e-01}$	$1.20e+00$	$6.65e-01$	$5.32e-01$	$6.19e-01$
f_3	$\mathbf{4.32e-15}$	$7.70e-02$	$4.78e-01$	$1.99e-01$	$2.09e-01$	$8.20e-01$	$2.54e-01$	$4.44e-15$
f_4	$8.59e-02$	$\mathbf{5.49e-02}$	$1.46e-01$	$1.28e-01$	$9.33e-02$	$1.55e-01$	$4.66e-01$	$1.21e-01$
f_5	$7.10e+00$	$\mathbf{6.14e+00}$	$1.33e+01$	$1.50e+01$	$1.93e+01$	$1.79e+01$	$2.59e+01$	$1.32e+01$
f_6	$4.30e-180$	$\mathbf{6.21e-300}$	$2.00e-01$	$4.01e-164$	$1.32e-164$	$3.69e-126$	$1.06e-34$	$2.66e-48$
f_7	$8.53e-02$	$1.17e-01$	$1.64e+00$	$1.42e+00$	$1.39e+00$	$2.49e+00$	$2.72e-01$	$\mathbf{0.00e+00}$
f_8	$2.13e-15$	$3.95e-15$	$1.78e-01$	$8.61e-02$	$6.39e-02$	$1.28e-02$	$\mathbf{1.54e-16}$	$1.88e-01$
f_9	$\mathbf{4.71e-32}$	$3.11e-02$	$1.01e+00$	$5.18e-02$	$1.04e-02$	$1.24e-01$	$1.04e-02$	$\mathbf{4.71e-32}$
f_{10}	$\mathbf{0.00e+00}$	$\mathbf{0.00e+00}$	$\mathbf{0.00e+00}$	$\mathbf{0.00e+00}$	$\mathbf{0.00e+00}$	$4.94e-324$	$8.78e-136$	$6.67e-110$
f_{11}	$6.77e-01$	$1.06e+00$	$7.92e+00$	$7.28e-01$	$1.27e+00$	$8.24e-01$	$\mathbf{2.76e-01}$	$5.51e-01$
f_{12}	$6.84e-01$	$9.97e-01$	$9.76e-01$	$2.76e-01$	$2.25e-01$	$6.13e-01$	$2.03e-01$	$\mathbf{4.44e-15}$
f_{13}	$\mathbf{1.13e-01}$	$1.52e-01$	$1.50e-01$	$1.28e-01$	$1.17e-01$	$1.27e-01$	$4.72e-01$	$1.17e-01$
f_{14}	$1.72e+01$	$2.08e+01$	$1.71e+01$	$1.70e+01$	$1.61e+01$	$1.81e+01$	$2.44e+01$	$\mathbf{1.33e+01}$
f_{15}	$1.10e-10$	$1.12e+00$	$8.64e-01$	$1.40e-01$	$3.70e-04$	$\mathbf{1.23e-122}$	$2.71e-27$	$4.71e-48$
f_{16}	$3.06e+00$	$3.62e+00$	$2.83e+00$	$2.54e+00$	$2.39e+00$	$2.70e+00$	$1.73e-01$	$\mathbf{5.15e-02}$
f_{17}	$1.31e-01$	$6.36e-01$	$5.62e-01$	$2.95e-01$	$1.35e-01$	$2.30e-02$	$\mathbf{1.82e-16}$	$1.13e-01$
f_{18}	$9.33e-02$	$7.26e-02$	$9.88e-01$	$4.72e-32$	$1.04e-02$	$8.29e-02$	$4.72e-32$	$\mathbf{4.71e-32}$

Cuadro 5.14: Comparativa entre variantes PSO para un tamaño de dimensión $d = 10$ y de población $N = 50$

Función	PSO	BareBonesPSO	normLinkedPSO	SPSO11	MSPSO11	LcRiPSO	HRiPSO	Propuesta
f_1	$1.13e-219$	$\mathbf{0.00e+00}$	$\mathbf{0.00e+00}$	$6.37e-242$	$3.89e-265$	$3.00e-225$	$3.10e-78$	$3.98e-68$
f_2	$4.33e-01$	$4.84e-01$	$7.82e+00$	$9.36e-01$	$9.51e-01$	$2.66e-01$	$\mathbf{2.66e-01}$	$1.52e+00$
f_3	$\mathbf{4.09e-15}$	$4.44e-15$	$4.24e-01$	$3.85e-02$	$4.20e-15$	$7.44e-01$	$4.44e-15$	$4.44e-15$
f_4	$7.45e-02$	$\mathbf{5.83e-02}$	$1.64e-01$	$1.10e-01$	$1.41e-01$	$1.38e-01$	$5.08e-01$	$8.26e-02$
f_5	$\mathbf{4.18e+00}$	$4.64e+00$	$1.57e+01$	$1.69e+01$	$1.68e+01$	$1.46e+01$	$2.85e+01$	$1.28e+01$
f_6	$3.86e-117$	$\mathbf{7.98e-209}$	$8.73e-03$	$8.73e-120$	$2.66e-129$	$1.70e-96$	$2.31e-36$	$3.35e-30$
f_7	$5.06e-02$	$4.69e-03$	$1.43e+00$	$1.16e+00$	$1.06e+00$	$2.04e+00$	$\mathbf{0.00e+00}$	$1.14e-06$
f_8	$\mathbf{1.14e-15}$	$2.64e-15$	$3.10e-01$	$4.74e-02$	$2.84e-02$	$1.13e-02$	$2.99e-04$	$8.99e-02$
f_9	$\mathbf{4.71e-32}$	$1.04e-02$	$1.74e-01$	$\mathbf{4.71e-32}$	$\mathbf{4.71e-32}$	$1.66e-01$	$\mathbf{4.71e-32}$	$4.80e-32$
f_{10}	$9.44e-220$	$\mathbf{0.00e+00}$	$\mathbf{0.00e+00}$	$6.52e-242$	$3.18e-266$	$1.82e-226$	$4.66e-78$	$1.95e-68$
f_{11}	$1.27e+00$	$1.06e+00$	$9.81e+00$	$5.81e-01$	$4.89e-01$	$7.15e-01$	$\mathbf{3.18e-02}$	$1.53e+00$
f_{12}	$2.76e-01$	$6.08e-01$	$2.54e-01$	$1.87e-01$	$1.70e-01$	$8.25e-01$	$\mathbf{4.44e-15}$	$\mathbf{4.44e-15}$
f_{13}	$1.33e-01$	$1.51e-01$	$1.38e-01$	$1.37e-01$	$1.28e-01$	$1.45e-01$	$5.04e-01$	$\mathbf{8.80e-02}$
f_{14}	$1.72e+01$	$2.10e+01$	$1.71e+01$	$1.54e+01$	$1.50e+01$	$1.85e+01$	$2.71e+01$	$\mathbf{1.21e+01}$
f_{15}	$\mathbf{1.54e-96}$	$5.88e-01$	$7.35e-02$	$9.21e-07$	$1.02e-21$	$8.97e-95$	$3.63e-36$	$4.88e-30$
f_{16}	$2.94e+00$	$3.30e+00$	$1.86e+00$	$2.19e+00$	$2.45e+00$	$1.85e+00$	$\mathbf{0.00e+00}$	$2.44e-05$
f_{17}	$1.50e-01$	$5.12e-01$	$3.61e-01$	$1.73e-01$	$1.58e-01$	$1.12e-02$	$\mathbf{2.58e-04}$	$1.36e-01$
f_{18}	$5.18e-02$	$5.18e-02$	$3.13e-01$	$1.04e-02$	$\mathbf{4.71e-32}$	$2.28e-01$	$4.72e-32$	$4.74e-32$

Cuadro 5.15: Comparativa entre variantes PSO para un tamaño de dimensión $d = 30$ y de población $N = 30$

Función	PSO	BareBonesPSO	normLinkedPSO	SPSO11	MSPSO11	LcRiPSO	HRiPSO	Propuesta
f_1	$7.48e-94$	$3.14e-101$	$1.15e-152$	$\mathbf{8.38e-213}$	$1.96e-210$	$4.80e-156$	$6.84e+04$	$8.86e-27$
f_2	$\mathbf{5.86e+00}$	$1.39e+01$	$1.93e+01$	$1.65e+01$	$9.35e+00$	$9.98e+00$	$1.58e+02$	$2.42e+01$
f_3	$2.60e+00$	$4.84e-01$	$8.21e-01$	$1.90e+00$	$2.18e+00$	$4.60e+00$	$1.01e+01$	$\mathbf{1.18e-14}$
f_4	$5.82e-02$	$1.16e-02$	$6.64e-03$	$9.11e-03$	$8.45e-03$	$1.51e-02$	$2.59e+01$	$\mathbf{3.70e-18}$
f_5	$6.66e+01$	$\mathbf{5.96e+01}$	$1.01e+02$	$7.90e+01$	$8.66e+01$	$6.71e+01$	$1.57e+02$	$1.15e+02$
f_6	$3.25e-03$	$\mathbf{1.58e-71}$	$4.04e-01$	$1.88e+00$	$1.63e+00$	$2.75e+00$	$1.97e+01$	$2.04e-13$
f_7	$6.58e+00$	$3.36e+00$	$1.37e+01$	$1.67e+01$	$1.75e+01$	$1.93e+01$	$1.56e+01$	$\mathbf{6.36e-01}$
f_8	$3.36e-05$	$\mathbf{1.01e-13}$	$4.82e+00$	$5.58e+00$	$5.45e+00$	$8.42e-01$	$7.24e+00$	$1.93e-01$
f_9	$3.63e-01$	$1.31e-01$	$5.40e-01$	$3.78e+00$	$2.77e+00$	$4.42e+00$	$2.50e+01$	$\mathbf{9.08e-23}$
f_{10}	$8.34e-102$	$2.37e-101$	$3.42e-149$	$\mathbf{2.51e-213}$	$1.86e-210$	$2.04e-156$	$7.68e+04$	$1.12e-26$
f_{11}	$2.10e+01$	$2.55e+01$	$1.71e+01$	$1.88e+01$	$1.09e+01$	$\mathbf{1.04e+01}$	$1.60e+02$	$2.44e+01$
f_{12}	$3.79e+00$	$2.08e+00$	$9.55e-01$	$2.08e+00$	$2.04e+00$	$4.08e+00$	$1.03e+01$	$\mathbf{1.17e-14}$
f_{13}	$1.53e-02$	$5.67e-03$	$8.37e-03$	$1.32e-02$	$1.07e-02$	$1.36e-02$	$2.57e+01$	$\mathbf{9.86e-04}$
f_{14}	$9.58e+01$	$1.15e+02$	$7.89e+01$	$7.77e+01$	$8.56e+01$	$\mathbf{7.22e+01}$	$1.55e+02$	$1.00e+02$
f_{15}	$6.13e+00$	$1.24e+01$	$1.24e+00$	$4.97e+00$	$4.96e+00$	$3.34e+00$	$2.18e+01$	$\mathbf{2.63e-13}$
f_{16}	$2.32e+01$	$2.60e+01$	$1.52e+01$	$1.87e+01$	$1.92e+01$	$1.97e+01$	$1.60e+01$	$\mathbf{2.88e-01}$
f_{17}	$7.04e+00$	$1.31e+01$	$5.97e+00$	$6.56e+00$	$8.25e+00$	$3.07e+00$	$7.23e+00$	$\mathbf{1.74e-01}$
f_{18}	$8.92e+00$	$7.89e+00$	$2.71e-01$	$3.53e+00$	$2.63e+00$	$5.67e+00$	$4.39e+02$	$\mathbf{6.09e-23}$

Cuadro 5.16: Comparativa entre variantes PSO para un tamaño de dimensión $d = 30$ y de población $N = 50$

Función	PSO	BareBonesPSO	normLinkedPSO	SPSO11	MSPSO11	LcRiPSO	HRiPSO	Propuesta
f_1	$9.42e-82$	$1.28e-66$	$2.33e-96$	$\mathbf{5.92e-145}$	$3.10e-144$	$6.40e-116$	$3.80e+04$	$1.07e-14$
f_2	$9.88e+00$	$2.26e+01$	$2.34e+01$	$1.60e+01$	$1.18e+01$	$\mathbf{5.50e+00}$	$9.20e+01$	$2.58e+01$
f_3	$1.24e+00$	$\mathbf{7.76e-15}$	$8.50e-01$	$2.00e+00$	$1.83e+00$	$4.10e+00$	$8.26e+00$	$7.07e-09$
f_4	$2.61e-02$	$8.78e-03$	$7.06e-03$	$1.46e-02$	$6.16e-03$	$1.24e-02$	$1.53e+01$	$\mathbf{2.47e-04}$
f_5	$5.52e+01$	$\mathbf{4.58e+01}$	$6.79e+01$	$7.55e+01$	$9.07e+01$	$6.27e+01$	$1.80e+02$	$1.35e+02$
f_6	$1.82e-14$	$\mathbf{8.81e-48}$	$5.57e-01$	$1.42e+00$	$6.55e-01$	$9.38e-01$	$1.47e+01$	$6.70e-08$
f_7	$5.20e+00$	$1.69e+00$	$1.31e+01$	$1.44e+01$	$1.43e+01$	$1.83e+01$	$1.13e+01$	$\mathbf{1.08e-01}$
f_8	$6.09e-12$	$\mathbf{6.19e-14}$	$4.93e+00$	$4.46e+00$	$4.50e+00$	$5.35e-01$	$4.27e+00$	$4.37e-02$
f_9	$3.61e-01$	$1.45e-01$	$1.30e-01$	$8.81e-01$	$1.52e+00$	$4.54e+00$	$5.00e+00$	$\mathbf{7.88e-14}$
f_{10}	$3.08e-82$	$9.06e-70$	$1.40e-97$	$\mathbf{4.05e-145}$	$4.68e-144$	$5.40e-115$	$3.82e+04$	$9.84e-15$
f_{11}	$1.97e+01$	$2.57e+01$	$1.92e+01$	$1.99e+01$	$1.43e+01$	$\mathbf{7.17e+00}$	$8.61e+01$	$2.60e+01$
f_{12}	$2.62e+00$	$2.11e+00$	$9.07e-01$	$1.99e+00$	$1.85e+00$	$4.00e+00$	$8.03e+00$	$\mathbf{6.77e-09}$
f_{13}	$1.04e-02$	$1.18e-02$	$8.53e-03$	$9.60e-03$	$8.53e-03$	$1.30e-02$	$1.64e+01$	$\mathbf{2.47e-04}$
f_{14}	$9.48e+01$	$1.13e+02$	$7.52e+01$	$7.40e+01$	$7.87e+01$	$\mathbf{6.95e+01}$	$1.81e+02$	$1.28e+02$
f_{15}	$5.43e+00$	$1.27e+01$	$1.34e+00$	$3.54e+00$	$3.29e+00$	$1.49e+00$	$1.43e+01$	$\mathbf{8.07e-08}$
f_{16}	$2.02e+01$	$2.51e+01$	$1.54e+01$	$1.67e+01$	$1.73e+01$	$1.87e+01$	$1.13e+01$	$\mathbf{3.38e-01}$
f_{17}	$6.74e+00$	$1.13e+01$	$5.26e+00$	$5.79e+00$	$5.56e+00$	$1.60e+00$	$4.30e+00$	$\mathbf{1.08e-01}$
f_{18}	$3.87e+00$	$6.00e+00$	$8.30e-02$	$2.10e+00$	$1.85e+00$	$3.11e+00$	$4.34e+00$	$\mathbf{5.26e-14}$

Cuadro 5.17: Comparativa entre variantes PSO para un tamaño de dimensión $d = 50$ y de población $N = 30$

Función	PSO	BareBonesPSO	normLinkedPSO	SPSO11	MSPSO11	LcRiPSO	HRiPSO	Propuesta
f_1	$3.93e-22$	$3.72e+05$	$3.31e+06$	$\mathbf{8.12e-138}$	$2.38e-133$	$2.71e-91$	$1.68e+05$	$4.92e-15$
f_2	$4.44e+01$	$3.89e+03$	$1.59e+04$	$4.15e+01$	$\mathbf{3.43e+01}$	$3.98e+01$	$3.29e+02$	$4.69e+01$
f_3	$5.77e+00$	$1.17e+01$	$2.07e+01$	$3.40e+00$	$3.19e+00$	$7.56e+00$	$1.17e+01$	$\mathbf{3.18e-09}$
f_4	$1.43e-01$	$1.42e+02$	$1.08e+03$	$7.47e-03$	$4.19e-03$	$6.08e-03$	$6.39e+01$	$\mathbf{2.47e-04}$
f_5	$1.64e+02$	$3.82e+02$	$7.86e+02$	$1.56e+02$	$1.73e+02$	$\mathbf{1.15e+02}$	$2.76e+02$	$2.46e+02$
f_6	$\mathbf{9.09e-03}$	$1.05e+24$	$7.03e+22$	$8.32e+00$	$5.74e+00$	$7.60e+00$	$4.15e+01$	$4.78e-02$
f_7	$2.64e+01$	$4.56e+01$	$8.92e+01$	$3.74e+01$	$3.98e+01$	$3.99e+01$	$3.23e+01$	$\mathbf{1.77e+00}$
f_8	$\mathbf{2.12e-03}$	$6.31e+01$	$1.25e+02$	$1.53e+01$	$1.48e+01$	$5.69e+00$	$2.10e+01$	$8.69e-02$
f_9	$3.17e-01$	$1.11e+08$	$1.12e+09$	$1.15e+01$	$9.00e+00$	$1.24e+01$	$4.81e+03$	$\mathbf{8.09e-16}$
f_{10}	$1.01e-25$	$1.94e+05$	$3.36e+06$	$\mathbf{2.31e-137}$	$1.93e-133$	$2.30e-91$	$1.67e+05$	$4.90e-15$
f_{11}	$4.56e+01$	$3.69e+03$	$1.66e+04$	$4.15e+01$	$\mathbf{3.74e+01}$	$4.04e+01$	$3.53e+02$	$4.66e+01$
f_{12}	$5.90e+00$	$1.32e+01$	$2.08e+01$	$3.40e+00$	$2.84e+00$	$7.22e+00$	$1.14e+01$	$\mathbf{3.15e-09}$
f_{13}	$6.24e-03$	$1.49e+02$	$1.04e+03$	$4.93e-03$	$7.22e-03$	$7.63e-03$	$5.77e+01$	$\mathbf{2.41e-11}$
f_{14}	$2.26e+02$	$5.55e+02$	$7.77e+02$	$1.51e+02$	$1.75e+02$	$\mathbf{1.13e+02}$	$2.80e+02$	$2.34e+02$
f_{15}	$1.29e+02$	$2.56e+19$	$2.54e+19$	$1.28e+01$	$1.27e+01$	$7.60e+00$	$4.43e+01$	$\mathbf{2.20e-02}$
f_{16}	$5.48e+01$	$7.14e+01$	$8.27e+01$	$4.18e+01$	$4.27e+01$	$4.04e+01$	$3.25e+01$	$\mathbf{2.83e+00}$
f_{17}	$2.21e+01$	$8.91e+01$	$1.22e+02$	$1.79e+01$	$1.86e+01$	$6.87e+00$	$1.96e+01$	$\mathbf{2.40e-01}$
f_{18}	$1.85e+01$	$2.81e+08$	$1.95e+09$	$1.25e+01$	$1.04e+01$	$1.15e+01$	$8.04e+03$	$\mathbf{8.18e-16}$

Cuadro 5.18: Comparativa entre variantes PSO para un tamaño de dimensión $d = 50$ y de población $N = 50$

Función	PSO	BareBonesPSO	normLinkedPSO	SPSO11	MSPSO11	LcRiPSO	HRiPSO	Propuesta
f_1	$3.96e-27$	$7.72e+04$	$3.16e+06$	$\mathbf{2.18e-96}$	$2.09e-93$	$2.33e-66$	$1.09e+05$	$3.27e-07$
f_2	$4.65e+01$	$2.23e+03$	$1.49e+04$	$3.93e+01$	$\mathbf{3.81e+01}$	$3.91e+01$	$2.33e+02$	$4.72e+01$
f_3	$3.15e+00$	$8.07e+00$	$2.09e+01$	$2.65e+00$	$2.68e+00$	$7.05e+00$	$9.72e+00$	$\mathbf{2.43e-05}$
f_4	$4.75e-02$	$4.72e+01$	$1.19e+03$	$5.25e-03$	$4.59e-03$	$6.73e-03$	$4.10e+01$	$\mathbf{9.20e-07}$
f_5	$1.51e+02$	$4.65e+02$	$7.68e+02$	$1.47e+02$	$1.59e+02$	$\mathbf{1.03e+02}$	$3.52e+02$	$2.83e+02$
f_6	$3.70e-03$	$1.73e+22$	$1.54e+20$	$6.03e+00$	$5.39e+00$	$5.38e+00$	$3.63e+01$	$\mathbf{1.75e-04}$
f_7	$2.13e+01$	$2.85e+01$	$8.75e+01$	$3.47e+01$	$3.48e+01$	$3.69e+01$	$2.66e+01$	$\mathbf{1.19e+00}$
f_8	$\mathbf{8.63e-04}$	$6.78e+01$	$1.13e+02$	$1.17e+01$	$1.22e+01$	$4.17e+00$	$1.46e+01$	$8.08e-02$
f_9	$1.77e-01$	$2.29e+08$	$1.08e+09$	$9.34e+00$	$8.98e+00$	$8.40e+00$	$2.73e+01$	$\mathbf{1.84e-09}$
f_{10}	$2.91e-20$	$2.48e+05$	$3.27e+06$	$\mathbf{2.49e-96}$	$3.22e-93$	$2.59e-66$	$1.22e+05$	$2.77e-07$
f_{11}	$4.64e+01$	$4.42e+03$	$1.72e+04$	$4.45e+01$	$\mathbf{3.84e+01}$	$3.96e+01$	$2.12e+02$	$4.73e+01$
f_{12}	$4.66e+00$	$1.24e+01$	$2.09e+01$	$2.86e+00$	$2.84e+00$	$7.65e+00$	$1.01e+01$	$\mathbf{2.39e-05}$
f_{13}	$5.91e-03$	$4.61e+01$	$1.32e+03$	$7.46e-03$	$6.65e-03$	$6.40e-03$	$4.12e+01$	$\mathbf{8.23e-04}$
f_{14}	$2.23e+02$	$5.53e+02$	$7.20e+02$	$1.44e+02$	$1.62e+02$	$\mathbf{1.08e+02}$	$3.57e+02$	$2.83e+02$
f_{15}	$6.10e+01$	$1.27e+18$	$1.05e+18$	$1.01e+01$	$1.66e+01$	$4.88e+00$	$3.48e+01$	$\mathbf{1.87e-04}$
f_{16}	$5.19e+01$	$7.00e+01$	$7.76e+01$	$3.79e+01$	$3.80e+01$	$3.65e+01$	$2.62e+01$	$\mathbf{1.96e+00}$
f_{17}	$2.15e+01$	$9.03e+01$	$1.13e+02$	$1.46e+01$	$1.61e+01$	$3.55e+00$	$1.46e+01$	$\mathbf{7.51e-02}$
f_{18}	$1.59e+01$	$3.36e+08$	$1.65e+09$	$1.20e+01$	$9.92e+00$	$9.47e+00$	$1.63e+01$	$\mathbf{1.63e-09}$

Los resultados demuestran la eficacia de nuestra propuesta. Que hayamos elegido una configuración bastante explorativa hace que nuestro método no destaque demasiado en ninguna función obteniendo valores muy bajos, pero a cambio hace que tenga un comportamiento muy regular y sus resultados sean más homogéneos manteniéndose siempre o casi siempre en valores bajos. Es así que por ejemplo para la función Sphere nuestra propuesta obtiene unos resultados mucho más discretos que la mayoría (aún así buenos) y sin embargo, para las funciones más complejas como la Ackley, Alpine, Weierstrass, etc sus resultados son sensiblemente superiores a los del resto de variantes, las cuales convergen prematuramente en la gran mayoría de situaciones. También se ve esto muy claro examinando los resultados para dimensiones bajas como $d = 10$, donde nuestro método no destaca tanto al tener una configuración más explorativa que la que se requiere para un espacio dimensional tan pequeño. Cuando $d = 10$ todos los métodos comparados exhiben muy buenos resultados salvo quizá en las funciones Rastrigin y Weierstrass, siendo nuestro método siempre destacado en esta última. Pero esto viene a decirnos que la optimización en este tamaño de dimensión en general no supone ningún reto mayúsculo y que por tanto es más interesante prestar nuestra atención en dimensiones más altas. Es justo ahí (en las dimensiones más altas) donde más sobresale nuestro método. Conforme más aumentamos de dimensión, más fuerte se hace nuestro método en relación al resto. En todas las comparativas de $d = 30$ y $d = 50$, nuestra propuesta consigue los mejores resultados para un mínimo de nueve de las dieciocho funciones que hay, es decir, como mínimo la mitad de las veces nuestro método gana a los otros siete. Y aquí no estamos contando que dos de los métodos (el PSO y el BareBones PSO) son claramente marco-dependientes y que tienen una ventaja añadida sobre el resto para optimizar funciones separables, con lo que es muy difícil ganarles en este tipo de funciones. Aún así en bastantes ocasiones nuestro método lo hace. Dicho esto, el verdadero potencial de nuestro método se ve si solo nos fijamos en las versiones rotadas de las funciones $f_{10} - f_{18}$, que en realidad son un indicador más fiable de lo bueno que es un optimizador al no presentar sesgos de ningún tipo que puedan beneficiar no tanto al que mejor búsqueda realiza sino al que mejor los explota.

Al hilo de la invarianza rotacional, ciertamente se observa que casi todos los métodos la cumplen excepto, claro, los dos primeros. Tanto el PSO como el BareBones PSO sufren una drástica caída en rendimiento cuando se trata de optimizar las funciones rotadas con respecto a las originales. Esta diferencia no es tan notoria (o pasa más desapercibida) en $d = 10$ ya que aunque sigue existiendo, los resultados para las funciones rotadas siguen siendo competitivos debido por supuesto a la sencillez de las funciones. Pero la diferencia se acrecienta con $d = 30$ y más aún con $d = 50$. Para estas dimensiones, ambos convergen prematuramente en todas las funciones excepto en dos: la Sphere y la Griewank. Estas dos funciones tienen la particularidad de ser rotacionalmente invariantes o de serlo aproximadamente en el caso de la segunda ya que $(QX)^{\mathrm{T}}QX = X^{\mathrm{T}}Q^{\mathrm{T}}QX = X^{\mathrm{T}}X \; \forall Q \in Orth^{+}$. La Griewank, aunque se compone también de otro sumando, éste es un productorio de números en $[-1, 1]$ que se vuelve despreciable a la mínima que el número de dimensión sea un poco elevado. Es por eso que las versiones rotadas de ambas funciones siguen siendo separables o prácticamente separables y permiten a los métodos PSO y BareBones PSO rendir igual de bien que en las versiones originales. En el duelo particular entre estos dos métodos nos encontramos que el BareBones PSO es superior para $d = 30$, mientras que el PSO lo es para $d = 50$. En $d = 10$ ambos tienen un rendimiento bastante similar y en el cómputo global gana el PSO por ser más estable y regular en rendimiento,

sobre todo porque el BareBones PSO en $d = 50$ se viene abajo y sus resultados son muy malos. El PSO estándar también sufre del escalado dimensional pero no es tan bestial.

Otros métodos que también se resienten mucho con el aumento de la dimensión son el normLinkedPSO y el HRiPSO. Directamente el HRiPSO solo rinde bien para $d = 10$, así que parece diseñado para la optimización en esa dimensión como también lo demuestra que toda la experimentación realizada en su artículo original [135, 136] se hace sobre $d = 10$. Respecto al normLinkedPSO, su rendimiento no es malo hasta llegar a $d = 50$, donde se vuelve tan explorativo que nada más comenzar la optimización las partículas escapan de la región de búsqueda y no se produce optimización alguna. Preguntándonos por este cambio tan radical en el nivel de exploración de una dimensión a otra, encontramos un fallo importante en el método. Resulta que en su regla de actualización de velocidades (Ec. 3.21) ellos emplean el vector de signos de los vectores social y cognitivo para guiar el movimiento en vez de emplear los propios vectores. Al hacerlo, han de recuperar de alguna manera la información relativa a la magnitud del paso que deben dar las partículas y para ello toman el módulo de los vectores (social y cognitivo) y lo multiplican directamente por el vector de signos. Pero al hacer esto, lo que no están teniendo en cuenta es que el vector resultante va a tener una norma mayor que la de los vectores originales (social y cognitivo) porque su norma va a ser el producto de la del vector original por la de su vector de signos y la norma de los vectores de signos es mayor o igual que 1. En concreto, es igual a $\sqrt{d}$. Así, cuando tenemos un tamaño de dimensión alto como $d = 50$, esta norma resultante va a ser mucho más grande que la que debiera, con lo que las partículas desconvergen rápidamente. La solución sencilla a este problema es añadir el factor $\frac{1}{\sqrt{d}}$ a cada una de las componentes social y cognitiva de su regla de actualización de velocidades. De todos modos, en $d = 30$ no logra escapar de óptimos locales en las funciones multimodales más complejas, por lo que más allá de ese problema el método no parece tampoco demasiado prometedor.

Las demás variantes son más competitivas, sobre todo la LcRiPSO, quizá también porque juega con un hiperparámetro (l) que se adapta al número de dimensiones y tamaño de enjambre del problema a tratar, lo que hace que no sea tan sensible a los cambios de dimensión. El SPSO11 y MSPSO11 obtienen resultados muy similares en todas las pruebas realizadas y destacan sobre todo en la función Sphere, lo que nos indica que son métodos explotativos y por ello también con problemas para escapar de óptimos locales.

Finalmente, para concluir con el experimento hemos decidido resumir los resultados en una pequeña tabla que muestre para cada una de las variantes y cada una de las pruebas, el rendimiento promedio que han tenido. De esta forma será más sencillo corroborar todos los aspectos discutidos y nos aportará claridad visual de la diferencia en rendimiento entre métodos. Para ello lo que hemos hecho ha sido coger cada tabla de resultados relativos a una prueba (para un d y N concretos) y normalizarlos (normalización min-max) para cada función a través de los métodos. Así tenemos para cada función los resultados entre 0 y 1 que indican lo bueno o no que es cada método en relación a los demás. Un 1 indica que el método es el peor de los ocho para esa función y un 0, que es el mejor. Los valores intermedios se determinan en función de la proximidad a estos resultados peor y mejor. Una vez tenemos los datos normalizados por función, lo que hacemos es promediarlos a través de las funciones, es decir, obtener el resultado normalizado promedio de cada método. Esta operación

la repetimos para las distintas pruebas y agrupamos todo en la Tabla 5.19. Ahora sí es claro que nuestra propuesta PSO es ciertamente muy prometedora.

Cuadro 5.19: Resumen de la comparativa entre variantes PSO, consistente en los resultados normalizados promedio de cada método.

	$d = 10$		$d = 30$		$d = 50$	
	$N = 30$	$N = 50$	$N = 30$	$N = 50$	$N = 30$	$N = 50$
PSO	**0.135**	**0.130**	0.173	0.208	0.108	0.094
BareBonesPSO	0.285	0.270	0.203	0.283	0.426	0.361
normLinkedPSO	0.646	0.563	0.183	0.208	1.000	1.000
SPSO11	0.219	0.175	0.228	0.265	0.091	0.082
MSPSO11	0.195	0.167	0.240	0.266	0.095	0.086
LcRiPSO	0.282	0.363	0.196	0.261	0.097	0.091
HRiPSO	0.261	0.222	0.942	0.890	0.163	0.153
Propuesta	0.209	0.183	**0.064**	**0.092**	**0.021**	**0.030**

5.2.5. Comparativa de variantes PSO separables

Si bien hemos estudiado experimentalmente variantes PSO genéricas que funcionan igual de bien para toda clase de funciones y problemas de optimización, ahora nos centraremos en aquellas que son diseñadas específicamente para resolver un tipo muy concreto de funciones: las separables. A estas variantes PSO las hemos llamado *PSO separables* para distinguirlas de las genéricas. El PSO estándar, a pesar de que no ha sido diseñado para optimizar funciones separables, sí podemos clasificarlo en esta categoría, puesto que, tal y como hemos señalado a lo largo del trabajo tanto durante el análisis como durante la experimentación, en realidad debe su éxito fundamentalmente a la optimización de este tipo de funciones (referirse al apartado 4.2.3.5). Estas variantes (las separables) destacan mucho en los benchmark que se suelen utilizar para evaluar el rendimiento de las metaheurísticas de optimización, ya que la mayoría de funciones de test son separables. Por eso mismo las lecturas y conclusiones extraídas suelen ser engañosas porque después no generalizan a los problemas de optimización que nos solemos encontrar en la vida real.

En este experimento pretendemos demostrar que esta fortaleza (virtud principal) por la que el PSO es considerado como uno de los mejores métodos metaheurísticos de optimización, se puede explotar aún más mediante otras variantes PSO construidas específicamente para ello- las separables. Además estas variantes realizan búsquedas

mucho más sencillas (unidimensionales) y permiten un progreso mucho más rápido al simultanearlas. Durante la review del Capítulo 3 (en el apartado 3.4.2) vimos que existen principalmente dos artículos con propuestas de estas características ([117] y [118]). Compararemos el PSO estándar con algunas de estas propuestas, entre las que incluimos también la nuestra dada en el apartado 4.2.3.5 (Algoritmo 5). De esta forma podremos ver qué variante separable es la más efectiva y lo bueno o no que es el PSO estándar en su faceta más destacada (la separabilidad) cuando realmente es comparado con otros métodos que han sido diseñados específicamente para explotarla.

En [117] realizan un total de tres propuestas distintas de PSO separables, tal y como estudiamos en el apartado 3.4.2. En el mismo artículo llevan a cabo una comparativa experimental de las tres y concluyen que la última de ellas es la más prometedora, la que destacadamente obtiene los mejores resultados. Con lo cual nosotros para nuestro experimento tan solo consideraremos esta variante última, la nombrada PSODDS (Ec. 3.12 y 3.11). El resto de variantes serán la propuesta en [118] (CPSO-S) que hemos descrito en el Algoritmo 2 y la nuestra, además del PSO estándar. Respecto a las configuraciones paramétricas de cada variante, se ha seguido el mismo criterio que en los anteriores experimentos, obedeciendo a las recomendaciones de los autores. En este caso no ha sido necesario en ninguna de ellas modificar la configuración recomendada puesto que sus rendimientos no se han visto afectados por tener que enmarcarse dentro de las bases estándar establecidas para la experimentación. Para el PSO estándar se han escogido dos configuraciones diferentes. Por un lado, la habitual que estamos utilizando para todos los experimentos (la homóloga al PSO constrictivo típico con los parámetros originales recomendados) y por otro, una configuración en la que solo exista la componente social ($w = 0$ y $c_1 = 0$), ya que según nuestro análisis de los apartados 4.2.3.5 y 4.4, la responsable de que surjan las búsquedas por dimensiones es esta componente y su efecto se reduce conforme más atractores introduzcamos en el método, que impiden el atrapamiento de las partículas en los ejes paralelos a los coordenados. El esquema de las configuraciones de cada método lo tenemos en la Tabla 5.20.

Cuadro 5.20: Configuraciones paramétricas escogidas para las distintas variantes que forman parte de la comparativa de PSOs separables

Variante	Configuración paramétrica
PSO1	$w = 0.72$ $c_1 = c_2 = 1.49$
PSO2	$w = 0$ $c_1 = 0$ $c_2 = 3.8$
PSODDS	$w = 0.72$ $c_1 = c_2 = 1.49$
CPSO-S	$w = (maxIter - t)/maxIter \implies w = 1 \longrightarrow 0$ $c_1 = c_2 = 1.49$
Propuesta	$\lambda_1 = 0.98$

La comparativa se ha llevado a cabo para las dimensiones $d = 30$ y $d = 50$. Las funciones de test no incluyen sus versiones rotadas, es decir, solo evaluaremos las funciones $f_1 - f_9$ puesto que los métodos están orientados a la optimización de funciones separables. En cuanto a los tamaños de enjambre, hay que decir que una de las variantes (la CPSO-S) usa un número de evaluaciones de función (FEs) por iteración mucho mayor que el resto al segmentar cada partícula en sus componentes dimensionales, los cuales trata como subpartículas diferentes dentro de los subenjambres correspondientes a cada dimensión. En otras palabras, en realidad trabaja con un tamaño poblacional $N * d$ y por tanto realiza un total de $N * d$ FEs por iteración, que es el número total de soluciones que se obtienen al modificar una sola componente por el valor de una de sus subpartículas. Así, esta variante requiere de un parámetro N diferente al resto y naturalmente también de un número de iteraciones diferente. Para elegir este parámetro recurrimos a los valores contemplados en su artículo original y que están en consonancia con los hallazgos de [140], que son $N = 10$ y $N = 20$. Así, su número máximo de iteraciones será $maxIter = \frac{2\times10^5}{10*d}$ y $maxIter = \frac{2\times10^5}{20*d}$ respectivamente. Lógicamente, por la constitución del resto de variantes, este número tan elevado de partículas ($N * d$) no es deseable y ellos cuentan con un $N = 30$ y $N = 50$ como hemos venido haciendo hasta ahora. Para estas variantes se aplica el mismo número de iteraciones que hemos empleado en todos los demás experimentos, esto es, $maxIter = \frac{2\times10^5}{N}$. Para poder comparar todas las variantes es necesario hacerlo de esta manera por las propias características de cada una, con el hándicap de que no exista una igualdad en número de iteraciones y tamaño de enjambre. El método CPSO-S trabaja con un número de partículas mucho mayor y un número de iteraciones mucho menor que el resto de variantes, cuando en general hemos aprendido durante el trabajo que lo opuesto es más deseable. Con todo, a continuación se presentan las Tablas de resultados 5.21, 5.22, 5.23 y 5.24.

Cuadro 5.21: Comparativa entre variantes PSO separables para un tamaño de dimensión $d = 30$ y de población $N = 30$ ($N = 10$ para el CPSO-S)

Función	PSO1	PSO2	PSODDS	CPSO-S	Propuesta
f_1	$7.48e-94$	$1.39e+03$	$5.60e-100$	$\mathbf{1.09e-132}$	$1.48e-23$
f_2	$5.86e+00$	$3.17e+01$	$\mathbf{1.27e+00}$	$3.29e+01$	$2.40e+01$
f_3	$2.60e+00$	$7.91e-01$	$3.86e-01$	$\mathbf{3.43e-14}$	$2.36e-13$
f_4	$5.82e-02$	$5.00e-02$	$\mathbf{1.35e-02}$	$2.23e-02$	$1.37e-02$
f_5	$6.66e+01$	$6.75e-01$	$6.90e+01$	$\mathbf{8.91e-14}$	$3.39e-13$
f_6	$3.25e-03$	$2.54e-01$	$3.62e-53$	$\mathbf{4.70e-77}$	$3.93e-13$
f_7	$6.58e+00$	$3.48e-01$	$4.16e+00$	$\mathbf{9.00e-15}$	$3.81e-11$
f_8	$3.36e-05$	$7.33e-02$	$1.66e-04$	$\mathbf{2.88e-13}$	$5.96e-11$
f_9	$3.63e-01$	$4.39e-02$	$1.42e-01$	$\mathbf{1.57e-32}$	$3.97e-26$

Cuadro 5.22: Comparativa entre variantes PSO separables para un tamaño de dimensión $d = 30$ y de población $N = 50$ ($N = 20$ para el CPSO-S)

Función	PSO1	PSO2	PSODDS	CPSO-S	Propuesta
f_1	$9.42e-82$	$2.01e+00$	$1.52e-64$	$\mathbf{8.50e-94}$	$3.56e-13$
f_2	$9.88e+00$	$2.93e+01$	$\mathbf{9.30e-01}$	$2.75e+01$	$3.02e+01$
f_3	$1.24e+00$	$3.03e-01$	$2.03e-01$	$\mathbf{3.08e-14}$	$1.65e-08$
f_4	$2.61e-02$	$4.69e-02$	$1.30e-02$	$1.91e-02$	$\mathbf{1.01e-02}$
f_5	$5.52e+01$	$2.67e-01$	$6.08e+01$	$\mathbf{1.33e-13}$	$3.32e-02$
f_6	$1.82e-14$	$1.77e-02$	$8.49e-35$	$\mathbf{6.96e-49}$	$5.92e-08$
f_7	$5.20e+00$	$4.79e-02$	$1.94e+00$	$\mathbf{1.04e-14}$	$1.32e-04$
f_8	$6.09e-12$	$4.10e-05$	$1.27e-03$	$\mathbf{4.31e-16}$	$4.68e-06$
f_9	$3.61e-01$	$2.75e-09$	$8.99e-02$	$\mathbf{1.57e-32}$	$1.91e-16$

Cuadro 5.23: Comparativa entre variantes PSO separables para un tamaño de dimensión $d = 50$ y de población $N = 30$ ($N = 10$ para el CPSO-S)

Función	PSO1	PSO2	PSODDS	CPSO-S	Propuesta
f_1	$3.93e-22$	$8.27e+04$	$1.30e-63$	$\mathbf{5.28e-92}$	$1.50e-09$
f_2	$4.44e+01$	$1.30e+02$	$\mathbf{2.11e+01}$	$6.77e+01$	$7.05e+01$
f_3	$5.77e+00$	$8.80e+00$	$1.80e+00$	$\mathbf{5.09e-14}$	$1.16e-06$
f_4	$1.43e-01$	$1.05e+01$	$\mathbf{4.93e-03}$	$1.94e-02$	$1.17e-02$
f_5	$1.64e+02$	$2.85e+01$	$1.48e+02$	$\mathbf{1.38e-13}$	$2.32e-01$
f_6	$9.09e-03$	$4.57e+00$	$1.36e-34$	$\mathbf{1.10e-47}$	$2.81e-06$
f_7	$2.64e+01$	$3.08e+00$	$1.48e+01$	$\mathbf{2.89e-14}$	$9.33e-03$
f_8	$2.12e-03$	$2.55e+00$	$1.25e-03$	$\mathbf{1.10e-09}$	$2.10e-04$
f_9	$3.17e-01$	$1.78e+07$	$2.70e-01$	$\mathbf{9.42e-33}$	$9.14e-12$

Cuadro 5.24: Comparativa entre variantes PSO separables para un tamaño de dimensión $d = 50$ y de población $N = 50$ ($N = 20$ para el CPSO-S)

Función	PSO1	PSO2	PSODDS	CPSO-S	Propuesta
f_1	$3.96e-27$	$1.20e+04$	$3.56e-41$	$\mathbf{2.32e-55}$	$3.06e-04$
f_2	$4.65e+01$	$1.01e+02$	$\mathbf{2.44e+01}$	$7.67e+01$	$7.40e+01$
f_3	$3.15e+00$	$4.67e+00$	$9.97e-01$	$\mathbf{4.88e-14}$	$8.18e-04$
f_4	$4.75e-02$	$6.52e+00$	$\mathbf{2.47e-03}$	$1.70e-02$	$8.74e-03$
f_5	$1.51e+02$	$1.01e+01$	$1.40e+02$	$\mathbf{1.29e-13}$	$1.65e-01$
f_6	$3.70e-03$	$1.53e+00$	$8.64e-23$	$\mathbf{1.87e-29}$	$1.42e-03$
f_7	$2.13e+01$	$1.15e+00$	$1.21e+01$	$\mathbf{3.65e-14}$	$1.95e-01$
f_8	$8.63e-04$	$6.77e-01$	$3.79e-03$	$\mathbf{1.16e-08}$	$1.59e-02$
f_9	$1.77e-01$	$2.33e+06$	$1.85e-01$	$\mathbf{9.42e-33}$	$4.66e-07$

Claramente los resultados nos muestran que sin importar el número de dimensiones d o el tamaño de población N, el método CPSO-S supera con holgura al resto. El segundo mejor método por lo general sería el nuestro, ya que a pesar de que no destaca nunca o casi nunca (solo en una ocasión ha sido el mejor), es bastante regular y consistente, al menos más que los demás competidores. Por su parte el PSO estándar en cualquiera de las dos configuraciones que hemos probado se queda bastante lejos de la mejor solución, lo que viene a demostrar que tampoco es un método demasiado aconsejable para optimizar funciones separables, los hay mejores. La segunda configuración probada (PSO2) arroja unos resultados muy pobres sobre todo para la dimensionalidad $d = 50$, pese a nuestra apreciación de que debería comportarse mejor que la primera. Además, se ha observado que muchas de las veces unos pocos resultados malos arruinan su media, que refleja un peor rendimiento que el que realmente tiene de normal. Juntando esto con su pobre rendimiento en $d = 50$, podemos deducir que lo que está pasando es que está siendo penalizado por la aleatoriedad. La aleatoriedad en el escalamiento modular del vector social hace que tengamos que escoger un c_2 muy alto como 3.8 para evitar la convergencia prematura en algunas funciones, lo cual también hace que en otras se sobreexplore y sus resultados a veces no sean buenos. Lógicamente este efecto se acentúa con el aumento de dimensión y así vemos como en funciones relativamente sencillas de optimizar en las que hay continuos cambios en el $gBest$ como en la f_1, este método PSO2 es el que peores resultados obtiene de largo. No obstante, para $d = 30$ sus resultados en las funciones separables más complejas como la f_3, f_5 o f_7 son mejores que los del PSO1 (converge prematuramente), que en cambio, rinde mejor en funciones más sencillas o en las que la separabilidad no es tan relevante (f_1, f_2), también en parte por la diversidad que le proporcionan el resto de componentes (inercial y cognitivo), que hace que la búsqueda por dimensiones no sea tan estricta como en el PSO2. Y de hecho esto se

puede ver en que un número mayor de partículas beneficia al PSO1 más que a ningún otro, lo cual es sinónimo de que las diversifica más, no están tan concentradas en ejes paralelos a los coordenados. El PSO1 es más explotativo y menos dependiente de la aleatoriedad al tener más componentes que la equilibran y por tanto es un método mucho más consistente que el PSO2. Aún así los dos decaen mucho en su rendimiento con el aumento de la dimensión.

Pero centrándonos de nuevo en el claro ganador de la comparativa, el CPSO-S, necesitamos entender por qué es tan superior. Realmente pueden haber varios motivos. Primero, es el único que utiliza una configuración paramétrica dinámica (w linealmente decreciente) y esto lógicamente puede influir en el rendimiento del método y resultar ventajoso. Y luego, lo que principalmente le distingue de los otros métodos, sobre todo del nuestro que es el más parecido, es que en vez de utilizar todas las partículas para todas las búsquedas dimensionales, emplea partículas específicas para cada dimensión. Esto podía pensarse como un inconveniente del método ya que requiere muchas más partículas que los demás métodos y por tanto, para cumplir con el mismo número de evaluaciones de función FEs dispone de muchas menos iteraciones. Sin embargo, a la vista está que acaba siendo una ventaja, pues no hay más que ver que el método es insensible o casi al aumento de la dimensionalidad. Y seguramente se deba a la notoria reducción de aleatoriedad que introduce, pues no depende del azar que la búsqueda se realice en una u otra dimensión o que la búsqueda por una dimensión le toque a una partícula más convergida que otra como puede ocurrir en nuestro método. Además permite que la búsqueda sea más homogénea, que se busque igual en unas zonas que en otras (por ejemplo en nuestro método el lado del gBest en el que cae la partícula es aleatorio y si a eso le añadimos que la dimensión también es aleatoria, entonces muy probablemente en dimensiones elevadas no se va a explorar igual en cada dimensión).

Habiendo aprendido que la mejor opción PSO para optimizar funciones separables es la de separar la búsqueda mediante enjambres independientes dedicados específicamente a cada dimensión, podemos aplicar nuestros conocimientos extraídos del análisis del apartado 4.2.2 y construir un método que realice d búsquedas por dimensiones pero siguiendo la regla sin aleatoriedad y solo con la componente social:

$$V = (1 + \lambda_1) * (gBest - X)$$

De este modo podemos comparar esta versión (Propuesta 2) con el CPSO-S y comprobar que efectivamente el éxito del método se debe a su estructura y no a la regla de actualización que utiliza. Para que la comparación esta vez sea equitativa, hemos establecido que el hiperparámetro λ_1 sea linealmente decreciente de 1 a 0 del mismo modo que hace el CPSO-S con w. Los resultados se presentan en la Tabla 5.25.

Cuadro 5.25: Comparativa entre variantes PSO separables

Función	$d = 30$				$d = 50$			
	$N = 10$		$N = 20$		$N = 10$		$N = 20$	
	CPSO-S	Propuesta 2	CPSO-S	Propuesta 2	CPSO-S	Propuesta 2	CPSO-S	Propuesta 2
f_1	$1.09e-132$	$\mathbf{6.26e-175}$	$8.50e-94$	$\mathbf{4.20e-122}$	$\mathbf{5.28e-92}$	$3.84e-87$	$2.32e-55$	$\mathbf{1.53e-73}$
f_2	$\mathbf{3.29e+01}$	$4.22e+01$	$\mathbf{2.75e+01}$	$4.99e+01$	$\mathbf{6.77e+01}$	$9.03e+01$	$7.67e+01$	$\mathbf{6.11e+01}$
f_3	$3.43e-14$	$\mathbf{3.17e-14}$	$3.08e-14$	$\mathbf{2.95e-14}$	$5.09e-14$	$\mathbf{4.07e-14}$	$\mathbf{4.88e-14}$	$5.25e-14$
f_4	$\mathbf{2.23e-02}$	$2.44e-02$	$1.91e-02$	$\mathbf{1.21e-02}$	$\mathbf{1.94e-02}$	$2.22e-02$	$1.70e-02$	$\mathbf{1.40e-02}$
f_5	$\mathbf{8.91e-14}$	$1.63e-13$	$1.33e-13$	$\mathbf{1.27e-13}$	$\mathbf{1.38e-13}$	$1.74e-13$	$1.29e-13$	$\mathbf{1.15e-13}$
f_6	$4.70e-77$	$\mathbf{7.97e-91}$	$6.96e-49$	$\mathbf{1.54e-59}$	$1.10e-47$	$\mathbf{1.17e-49}$	$1.87e-29$	$\mathbf{1.45e-34}$
f_7	$\mathbf{9.00e-15}$	$1.61e-14$	$1.04e-14$	$\mathbf{9.71e-15}$	$\mathbf{2.89e-14}$	$4.59e-14$	$\mathbf{3.65e-14}$	$4.03e-14$
f_8	$2.88e-13$	$\mathbf{4.96e-15}$	$\mathbf{4.31e-16}$	$4.32e-15$	$1.10e-09$	$\mathbf{7.80e-15}$	$1.16e-08$	$\mathbf{4.85e-15}$
f_9	$\mathbf{1.57e-32}$	$\mathbf{1.57e-32}$	$\mathbf{1.57e-32}$	$\mathbf{1.57e-32}$	$\mathbf{9.42e-33}$	$\mathbf{9.42e-33}$	$\mathbf{9.42e-33}$	$\mathbf{9.42e-33}$

Ahora sí que los resultados son prácticamente equivalentes. Esto demuestra que en verdad la fortaleza del CPSO-S no es su regla de actualización PSO sino la estructura de enjambre que lleva, es decir, emplear partículas fijas concentradas y asociadas a dimensiones específicas y no partículas generales que puedan dedicarse a la búsqueda por varias dimensiones al mismo tiempo. La regla de actualización no es demasiado relevante ya que las búsquedas son unidimensionales, con lo que son muy sencillas y no requieren apenas de diversidad.

No obstante que los resultados de nuestro primer método no fueran los mejores no quiere decir que el método sea desechable, y es que hay que tener en cuenta una cosa y es que el CPSO-S (el más sobresaliente), así como el último método propio propuesto que emplean N partículas fijas por dimensión, no son sostenibles a dimensiones muy elevadas, solamente son válidos para dimensiones pequeñas. Y no lo son porque mínimo requieren del mismo número de partículas que de dimensiones (en la práctica más, al menos el doble porque las búsquedas requieren de dos o más partículas), con lo que si la dimensión es muy alta el número de partículas es también muy alto y llega un punto en el que no es viable mantener tantísimas partículas durante

una ejecución del algoritmo. En esas, una comparación con nuestro primer método propuesto obligaría a que el número de iteraciones fuese muy desigual en favor del nuestro, lo que podría traducirse en un vuelco en los resultados que hiciera a nuestro método mucho más recomendable. El CPSO-S solo compensa para dimensiones bajas y para las altas es donde aparecen el resto de métodos de la comparativa, entre los cuales, el nuestro se postula como una de las mejores alternativas.

Capítulo 6

Conclusiones

Al ser el PSO una de las metaheurísticas de optimización más importantes y utilizadas, acapara gran atención por parte de la comunidad investigadora. Sin embargo, es bastante desconocido su funcionamiento interno y los mecanismos clave que hacen de él un optimizador tan potente. Esto sumado al no desdeñable número de hiperparámetros que posee, de los cuales es muy dependiente, hace que no sea tarea sencilla para el usurario medio confeccionar el método para la resolución de cualquier problema de optimización específico. Y para los investigadores tampoco resulta fácil diseñar nuevas variantes o estrategias que mejoren su rendimiento o hagan más cómodo el proceso de selección al usuario. Por todo esto el objetivo del presente trabajo era arrojar luz sobre el rol de cada componente del PSO, que sirviera para comprender mejor el método, facilitando de esta manera la labor tanto de selección de configuraciones parámetricas como de creación e investigación de futuras mejoras.

Para ello hemos comenzado estudiando los diferentes artículos de la literatura que tratan temas similares, relacionados con el análisis del método. En concreto hemos clasificado estos artículos en tres bloques o clases distintas. El primero sería el de los análisis de convergencia o estabilidad, que abordan el análisis del método desde una perspectiva matemática estudiando las tasas de convergencia asociadas a cada configuración paramétrica y los puntos de equilibrio o convergencia del sistema. Este tipo de análisis se basa estrictamente en el rendimiento esperado del método aunque requiere de una serie de suposiciones o asunciones no demasiado realistas. Además, solo puede predecir el nivel de exploración del algoritmo pero no la calidad del mismo (diversidad de la búsqueda, heurísticas que la guían y que contribuyen a que la búsqueda se lleve a cabo en áreas más o menos prometedoras, etc). Aún así, estos análisis son muy útiles para saber qué configuraciones son factibles (convergentes) y entender mejor el impacto que tienen los hiperparámetros en el nivel de exploración del PSO, así como para comprender un poco más el método. La segunda clase de análisis es el paramétrico, que trata de entender el rol de los hiperparámetros profundizando en su utilidad dentro del método desde un punto de vista heurístico. La gran mayoría de estos análisis vienen acompañados de alguna propuesta de PSO adaptativo que usa la información extraída del análisis para diseñarla. Hemos visitado muchos de ellos y aprendido que no siempre hay un consenso claro y global sobre el rol de los hiperparámetros y que, a pesar de que existan criterios bastante establecidos como el de considerar al factor inercial w como el parámetro más importante del método y principal responsable de la exploración, luego los resultados de las

comparativas demuestran que la mejor decisión suele ser optar por un w constante o incluso aleatorio, lo cual es síntoma de que no existe un conocimiento absoluto sobre él o como mínimo de que es muy difícil de controlar. Es por eso que muchos de los PSO adaptativos fracasan al ser comparados con el estándar. Finalmente, es la última clase de análisis la que nos ha desvelado más características clave sobre el funcionamiento del PSO en general. Esta clase de análisis que hemos denominado 'de movimiento', estudia la regla de actualización en conjunto y sobre todo nos da pistas interesantes de por qué los elementos aleatorios de la fórmula PSO son tan decisivos en su rendimiento. Se centra en la marco-varianza del PSO, que le hace ser un método bueno para funciones separables pero no tanto para las no-separables. Y hemos repasado propuestas tanto centradas en potenciar aún más esta habilidad del PSO para optimizar funciones separables como variantes que tratan de corregir este comportamiento sesgado.

Tras la review, hemos decidido hacer nuestro propio análisis del método partiendo desde cero, relacionando muchos de los conceptos en ella introducidos pero aportando muchas veces una justificación teórica y añadiendo más información por ejemplo relativa a esa varianza rotacional del PSO estándar. También en este análisis propio hemos estudiado en profundidad el rol de los elementos aleatorios para ver si en verdad son del todo necesarios o los podemos evitar de alguna forma para así ceder más control al usuario, que, recordemos siempre ha estado entre nuestros objetivos principales. Hemos conseguido entenderlos hasta tal punto que nos ha permitido realizar una distinción de sus funcionalidades y suprimirlos casi por completo hasta reducirlos únicamente a la función de determinar la dirección de los giros de los vectores social y cognitivo, desapareciendo así del escalamiento modular y del escalamiento direccional (del ángulo de los giros). Hasta donde se sabe, nunca antes una propuesta PSO había prescindido de estos elementos sin perder con ello rendimiento, lo que demuestra sin duda la validez de nuestro análisis. Hemos demostrado que la aleatoriedad es útil para cuando no se comprenden del todo los mecanismos que hacen funcionar al método con el fin de evitar mayor dependencia del criterio del usuario, pero sacrificando así potencial rendimiento. Después, hemos continuado analizando la relación entre el escalamiento modular y el direccional de los vectores de la regla de actualización y tratado de establecer un criterio para el cual el método exhibiera un mejor comportamiento. En este sentido hemos introducido el concepto de 'desvío' que nos permite alcanzar la diversidad y nivel de exploración deseables regulando los escalamientos modular y direccionales para que no tenga que ser el usuario el que se encargue. Hemos propuesto un nuevo sistema de giro para el escalamiento direccional de los vectores que se adapta, según nuestro criterio, mejor a las necesidades del PSO que otros que habíamos estudiado en la review, aportando observaciones teóricas sobre la comparación de todos ellos. Gracias a ello hemos establecido un nuevo método PSO cuyos parámetros tienen un rol muy claro y definido y que no dan lugar a confusión ni situaciones como las que se dan en el PSO estándar. También durante el proceso hemos discutido el rol de los hiperparámetros originales del PSO y hemos presentado una tesis novedosa sobre el rol de la inercia en el método y a qué se debe, contrariamente a lo que se dice, su poder e impacto positivo en el mismo. Y mención especial hemos hecho de la optimización de funciones separables, explicando de qué manera concreta es la aleatoriedad la responsable de que el PSO estándar muestre un claro favoritismo por la exploración por dimensiones. Hemos contribuido con la exposición de más características acerca de este comportamiento e incluso discutido con la presentación de una nueva propuesta PSO dedicada expresamente a la

optimización de funciones separables.

Con todo, hemos logrado comprender mucho mejor el método, entender qué es lo que hace que funcione y dónde reside la importancia de cada uno de sus hiperparámetros. Hemos detectado sus carencias y también sus puntos fuertes. Y esto nos ha llevado a construir un método PSO que aspira a tener un buen rendimiento para toda clase de funciones (de propósito general) y encima, con unos hiperparámetros con roles fáciles de interpretar. Tanto es así que hemos terminado unificándolos de cara al usuario final, por lo que podemos decir que nuestro método solamente necesita de la selección de un único hiperparámetro muy sencillo de regular (creciente con el nivel de exploración). La validez de nuestro método la hemos probado tanto teóricamente mediante un análisis de convergencia como en la práctica mediante una serie de pruebas experimentales. Por último, hemos llevado a cabo una comparativa experimental de nuestro método con otras propuestas similares y hemos corroborado que es muy prometedor. Asimismo como experimentación hemos presentado evidencias de nuestra tesis sobre el rol de la inercia en el PSO estándar, comparado nuestra propuesta de giro con las otras equivalentes y comparado también nuestra propuesta de PSO separable con el resto de las estudiadas, la cual parece buena opción para dimensiones elevadas. Durante la experimentación hemos corroborado algo que ya sabíamos y es que el verdadero potencial del PSO estándar reside en la optimización de funciones separables. Por eso no es un detalle menor haber comprobado que en la comparativa de métodos separables, el PSO está a la zaga. Esto nos invita a pensar para un futuro que quizá la mejor opción de tener un método PSO que sobresalga en la optimización de funciones separables pero que no sacrifique por ello la optimización del resto de funciones, sería combinar los dos PSO propuestos (el genérico y el separable) por ejemplo asignando un porcentaje de partículas que sigan las instrucciones de cada uno. No obstante identificar cuándo una función es o no separable no es algo muy complicado y ahorra después mucho cómputo.

Ahora, el trabajo que queda por hacer no es poco. Principalmente de todos los frentes que hemos abordado, lo más importante ha sido la introducción del nuevo PSO que recoge gran parte de las conclusiones extraídas durante el trabajo y que se muestra muy prometedor. Es así que el foco a partir de ahora debería ponerse en comprender más cosas sobre él, analizarlo más cómo se comporta en diferentes situaciones, etc. Por supuesto hay mucho margen para ampliar la experimentación sobre el método. Sería interesante probar su rendimiento para dimensiones mucho más altas, tamaños de población muy grandes, diferentes funciones, etc. También sería interesante probarlo directamente con problemas del mundo real y no tan solo con funciones de test. La comparativa se podría ampliar para incluir otros métodos PSO del estado del arte, incluso otras metaheurísticas que no sean el PSO (algoritmos genéticos, etc). Y respecto al método, también se ha quedado pendiente probar con la inclusión y/o modificación de ciertas heurísticas como la gestión activa de los límites del espacio de búsqueda, probar con una inicialización distinta para las partículas, etc. También se puede probar con distintas estructuras de enjambre (topologías) o con distintos tipos de actualización (asíncrona, etc). El método abre las puertas a la creación de nuevas variantes que sigan su paradigma. Las estrategias adaptativas pueden ganar peso con nuestro método ya que serán mucho más fáciles de diseñar ahora que el rol del(de los) hierparámetro(s) guarda relación directa con los niveles de exploración. Será interesante comparar las mismas estrategias adaptativas bajo el paradigma del PSO estándar y bajo el de nuestro método. Esto también servirá para evaluar de ma-

nera más justa las estrategias, que muchas veces exhiben un buen rendimiento no porque la estrategia en sí sea buena sino por azar. Podríamos incluso aplicar aprendizaje automático (técnicas de Deep Learning) para detectar el tipo de situaciones en las que se requiere más o menos exploración, quizá empleando Reinforcement learning. Por otro lado, podríamos extrapolar el paradigma creado a la optimización de funciones discretas y problemas de optimización combinatoria. En definitiva, en este trabajo ha sido presentada una herramienta que todavía está por explotar y de la cual cabe extraer mucho conocimiento aún.

Bibliografía

[1] Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. Computers & operations research, 13(5), 533-549.

[2] Sörensen, K., & Glover, F. (2013). Metaheuristics. Encyclopedia of operations research and management science, 62, 960-970.

[3] Holland, J. H. (1975). Adaptation in Natural and Artificial Systems. Ann Arbor, MI: University of Michigan Press.

[4] Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. science, 220(4598), 671-680.

[5] Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. IEEE computational intelligence magazine, 1(4), 28-39.

[6] Kennedy, J., & Eberhart, R. (1995, November). Particle swarm optimization. In Proceedings of ICNN'95-international conference on neural networks (Vol. 4, pp. 1942-1948). IEEE.

[7] Padhye, N., Deb, K., & Mittal, P. (2013). Boundary handling approaches in particle swarm optimization. In Proceedings of Seventh International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2012) (pp. 287-298). Springer, India.

[8] Xu, S., & Rahmat-Samii, Y. (2007). Boundary conditions in particle swarm optimization revisited. IEEE Transactions on Antennas and Propagation, 55(3), 760-765.

[9] Engelbrecht, A. (2012, June). Particle swarm optimization: Velocity initialization. In 2012 IEEE congress on evolutionary computation (pp. 1-8). IEEE.

[10] Uy, N. Q., Hoai, N. X., McKay, R. I., & Tuan, P. M. (2007, September). Initialising PSO with randomised low-discrepancy sequences: the comparative results. In 2007 IEEE Congress on Evolutionary Computation (pp. 1985-1992). IEEE.

[11] Wang, X., & Hickernell, F. J. (2000). Randomized halton sequences. Mathematical and Computer Modelling, 32(7-8), 887-899.

[12] Pant, M., Thangaraj, R., Singh, V. P., & Abraham, A. (2008, July). Particle swarm optimization using Sobol mutation. In 2008 First International Conference on Emerging Trends in Engineering and Technology (pp. 367-372). IEEE.

[13] Zielinski, K., & Laur, R. (2007). Stopping criteria for a constrained single-objective particle swarm optimization algorithm. Informatica, 31(1).

[14] Lynn, N., Ali, M. Z., & Suganthan, P. N. (2018). Population topologies for particle swarm optimization and differential evolution. Swarm and evolutionary computation, 39, 24-35.

[15] Ab Aziz, N. A., Mubin, M., Mohamad, M. S., & Ab Aziz, K. (2014). A synchronous-asynchronous particle swarm optimisation algorithm. The Scientific World Journal, 2014.

[16] Mikki, S. M., & Kishk, A. A. (2006). Quantum particle swarm optimization for electromagnetics. IEEE transactions on antennas and propagation, 54(10), 2764-2775.

[17] Yang, B., Chen, Y., & Zhao, Z. (2007, May). A hybrid evolutionary algorithm by combination of PSO and GA for unconstrained and constrained optimization problems. In 2007 IEEE International Conference on Control and Automation (pp. 166-170). IEEE.

[18] Yang, G., Chen, D., & Zhou, G. (2006, August). A new hybrid algorithm of particle swarm optimization. In International Conference on Intelligent Computing (pp. 50-60). Springer, Berlin, Heidelberg.

[19] Kennedy, J., & Eberhart, R. C. (1997, October). A discrete binary version of the particle swarm algorithm. In 1997 IEEE International conference on systems, man, and cybernetics. Computational cybernetics and simulation (Vol. 5, pp. 4104-4108). IEEE.

[20] Tan, Y., Gao, H. M., & Zeng, J. C. (2004). Particle swarm optimization for integer programming. Systems Engineering-theory & Practice, 5, 126-129.

[21] Clerc, M. (2004). Discrete particle swarm optimization, illustrated by the traveling salesman problem. In New optimization techniques in engineering (pp. 219-239). Springer, Berlin, Heidelberg.

[22] Parsopoulos, K. E., & Vrahatis, M. N. (2008). Multi-objective particles swarm optimization approaches. In Multi-objective optimization in computational intelligence: Theory and practice (pp. 20-42). IGI global.

[23] Parsopoulos, K. E., & Vrahatis, M. N. (2002). Particle swarm optimization method for constrained optimization problems. Intelligent Technologies–Theory and Application: New Trends in Intelligent Technologies, 76(1), 214-220.

[24] Lalwani, S., Sharma, H., Satapathy, S. C., Deep, K., & Bansal, J. C. (2019). A survey on parallel particle swarm optimization algorithms. Arabian Journal for Science and Engineering, 44(4), 2899-2923.

[25] Chen, C. Y., & Ye, F. (2012, May). Particle swarm optimization algorithm and its application to clustering analysis. In 2012 Proceedings of 17th Conference on Electrical Power Distribution (pp. 789-794). IEEE.

[26] Gudise, V. G., & Venayagamoorthy, G. K. (2003, April). Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. In Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No. 03EX706) (pp. 110-117). IEEE.

[27] Wachowiak, M. P., Smolíková, R., Zheng, Y., Zurada, J. M., & Elmaghraby, A. S. (2004). An approach to multimodal biomedical image registration utilizing

particle swarm optimization. IEEE Transactions on evolutionary computation, 8(3), 289-301.

[28] AlRashidi, M. R., & El-Hawary, M. E. (2008). A survey of particle swarm optimization applications in electric power systems. IEEE transactions on evolutionary computation, 13(4), 913-918.

[29] Eltamaly, A. M. (2021). A novel strategy for optimal PSO control parameters determination for PV energy systems. Sustainability, 13(2), 1008.

[30] Pedersen, M. E. H. (2010). Good parameters for particle swarm optimization. Hvass Lab., Copenhagen, Denmark, Tech. Rep. HL1001, 1551-3203.

[31] Trelea, I. C. (2003). The particle swarm optimization algorithm: convergence analysis and parameter selection. Information processing letters, 85(6), 317-325.

[32] Jiao, B., Lian, Z., & Gu, X. (2008). A dynamic inertia weight particle swarm optimization algorithm. Chaos, Solitons & Fractals, 37(3), 698-705.

[33] Shi, Y., & Eberhart, R. C. (2001, May). Fuzzy adaptive particle swarm optimization. In Proceedings of the 2001 congress on evolutionary computation (IEEE Cat. No. 01TH8546) (Vol. 1, pp. 101-106). IEEE.

[34] Reynolds, C. W. (1987, August). Flocks, herds and schools: A distributed behavioral model. In Proceedings of the 14th annual conference on Computer graphics and interactive techniques (pp. 25-34).

[35] Heppner, F., & Grenander, U. (1990). A stochastic nonlinear model for coordinated bird flocks.

[36] Eberhart, R., & Kennedy, J. (1995, October). A new optimizer using particle swarm theory. In MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science (pp. 39-43). Ieee.

[37] Shi, Y., & Eberhart, R. (1998, May). A modified particle swarm optimizer. In 1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360) (pp. 69-73). IEEE.

[38] Shi, Y., & Eberhart, R. C. (1999, July). Empirical study of particle swarm optimization. In Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406) (Vol. 3, pp. 1945-1950). IEEE.

[39] Shi, Y., & Eberhart, R. C. (1998, March). Parameter selection in particle swarm optimization. In International conference on evolutionary programming (pp. 591-600). Springer, Berlin, Heidelberg.

[40] Barrera, J., Álvarez-Bajo, O., Flores, J. J., & Coello Coello, C. A. (2016). Limiting the velocity in the particle swarm optimization algorithm. Computación y Sistemas, 20(4), 635-645.

[41] Ozcan, E., & Mohan, C. K. (1998). Analysis of a simple particle swarm optimization system. Intelligent engineering systems through artificial neural networks, 8, 253-258.

[42] Ozcan, E., & Mohan, C. K. (1999, July). Particle swarm optimization: Surfing the waves. In Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406) (Vol. 3, pp. 1939-1944). IEEE.

[43] Clerc, M., & Kennedy, J. (2002). The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE transactions on Evolutionary Computation, 6(1), 58-73.

[44] Eberhart, R. C., & Shi, Y. (2000, July). Comparing inertia weights and constriction factors in particle swarm optimization. In Proceedings of the 2000 congress on evolutionary computation. CEC00 (Cat. No. 00TH8512) (Vol. 1, pp. 84-88). IEEE.

[45] Yasuda, K., Ide, A., & Iwasaki, N. (2003, October). Adaptive particle swarm optimization. In SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme-System Security and Assurance (Cat. No. 03CH37483) (Vol. 2, pp. 1554-1559). IEEE.

[46] Van den Bergh, F., & Engelbrecht, A. P. (2006). A study of particle swarm optimization particle trajectories. Information sciences, 176(8), 937-971.

[47] Zheng, Y. L., Ma, L. H., Zhang, L. Y., & Qian, J. X. (2003, November). On the convergence analysis and parameter selection in particle swarm optimization. In Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 03EX693) (Vol. 3, pp. 1802-1807). IEEE.

[48] Zheng, Y. L., Ma, L. H., Zhang, L. Y., & Qian, J. X. (2003, December). Empirical study of particle swarm optimizer with an increasing inertia weight. In The 2003 Congress on Evolutionary Computation, 2003. CEC'03. (Vol. 1, pp. 221-226). IEEE.

[49] J. Kennedy, "Bare bones particle swarms,"Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No.03EX706), 2003, pp. 80-87, doi: 10.1109/SIS.2003.1202251.

[50] R. Poli, "Mean and Variance of the Sampling Distribution of Particle Swarm Optimizers During Stagnation,ïn IEEE Transactions on Evolutionary Computation, vol. 13, no. 4, pp. 712-721, Aug. 2009, doi: 10.1109/TEVC.2008.2011744.

[51] Jiang, M., Luo, Y. P., & Yang, S. Y. (2007). Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm. Information processing letters, 102(1), 8-16.

[52] A. M. LYAPUNOV (1992) The general problem of the stability of motion, International Journal of Control, 55:3, 531-534, DOI: 10.1080/00207179208934253

[53] Vidyasagar, M., & Society for Industrial and Applied Mathematics. (2002). Nonlinear systems analysis. Philadelphia, Pa: Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104.

[54] Kadirkamanathan, V., Selvarajah, K., & Fleming, P. J. (2006). Stability analysis of the particle dynamics in particle swarm optimizer. IEEE Transactions on Evolutionary Computation, 10(3), 245-255.

[55] Gazi, V. (2012, October). Stochastic stability analysis of the particle dynamics in the PSO algorithm. In 2012 IEEE international symposium on intelligent control (pp. 708-713). Ieee.

[56] Liu, Q. (2015). Order-2 stability analysis of particle swarm optimization. Evolutionary computation, 23(2), 187-216.

[57] Bonyadi, M. R., & Michalewicz, Z. (2015). Stability analysis of the particle swarm optimization without stagnation assumption. IEEE Transactions on Evolutionary Computation, 20(5), 814-819.

[58] Cleghorn, C. W., & Engelbrecht, A. P. (2018). Particle swarm stability: a theoretical extension using the non-stagnate distribution assumption. Swarm Intelligence, 12(1), 1-22.

[59] Cleghorn, C. W., & Engelbrecht, A. (2016, December). Particle swarm optimizer: The impact of unstable particles on performance. In 2016 IEEE symposium series on computational intelligence (SSCI) (pp. 1-7). IEEE.

[60] Van den Bergh, F., & Engelbrecht, A. P. (2002, October). A new locally convergent particle swarm optimiser. In IEEE International conference on systems, man and cybernetics (Vol. 3, pp. 6-pp). IEEE.

[61] Bergh, F.V., & Engelbrecht, A.P. (2010). A Convergence Proof for the Particle Swarm Optimiser. Fundam. Informaticae, 105, 341-374.

[62] Blackwell, T. (2011). A study of collapse in bare bones particle swarm optimization. IEEE Transactions on Evolutionary Computation, 16(3), 354-372.

[63] García-Gonzalo, E., & Fernández-Martínez, J. L. (2014). Convergence and stochastic stability analysis of particle swarm optimization variants with generic parameter distributions. Applied Mathematics and Computation, 249, 286-302.

[64] Fernandez-Martinez, J. L., & Garcia-Gonzalo, E. (2010). Stochastic stability analysis of the linear continuous and discrete PSO models. IEEE Transactions on Evolutionary Computation, 15(3), 405-423.

[65] Chatterjee, A., & Siarry, P. (2006). Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization. Computers & operations research, 33(3), 859-871.

[66] Chen, G., Huang, X., Jia, J., & Min, Z. (2006, June). Natural exponential inertia weight strategy in particle swarm optimization. In 2006 6th world congress on intelligent control and automation (Vol. 1, pp. 3672-3675). IEEE.

[67] Li, H. R., & Gao, Y. L. (2009, May). Particle swarm optimization algorithm with exponent decreasing inertia weight and stochastic mutation. In 2009 second international conference on information and computing science (Vol. 1, pp. 66-69). IEEE.

[68] Li, L., Xue, B., Niu, B., Tan, L., & Wang, J. (2009, September). A novel particle swarm optimization with non-linear inertia weight based on tangent function. In International Conference on Intelligent Computing (pp. 785-793). Springer, Berlin, Heidelberg.

[69] Ting, T. O., Shi, Y., Cheng, S., & Lee, S. (2012, June). Exponential inertia weight for particle swarm optimization. In International conference in swarm intelligence (pp. 83-90). Springer, Berlin, Heidelberg.

[70] Fan, S. K. S., & Chiu, Y. Y. (2007). A decreasing inertia weight particle swarm optimizer. Engineering Optimization, 39(2), 203-228.

[71] Gao, Y. L., An, X. H., & Liu, J. M. (2008, December). A particle swarm optimization algorithm with logarithm decreasing inertia weight and chaos mutation. In 2008 international conference on computational intelligence and security (Vol. 1, pp. 61-65). IEEE.

[72] Eberhart, R. C., & Shi, Y. (2001, May). Tracking and optimizing dynamic systems with particle swarms. In Proceedings of the 2001 congress on evolutionary computation (IEEE Cat. No. 01TH8546) (Vol. 1, pp. 94-100). IEEE.

[73] Zhang, L., Yu, H., & Hu, S. (2003, July). A new approach to improve particle swarm optimization. In Genetic and Evolutionary Computation Conference (pp. 134-139). Springer, Berlin, Heidelberg.

[74] Chuanwen, J., & Bompard, E. (2005). A hybrid method of chaotic particle swarm optimization and linear interior for reactive power optimisation. Mathematics and computers in Simulation, 68(1), 57-65.i

[75] Feng, Y., Teng, G. F., Wang, A. X., & Yao, Y. M. (2007, September). Chaotic inertia weight in particle swarm optimization. In Second International Conference on Innovative Computing, Informatio and Control (ICICIC 2007) (pp. 475-475). IEEE.

[76] Kentzoglanakis, K., & Poole, M. (2009, July). Particle swarm optimization with an oscillating inertia weight. In Proceedings of the 11th Annual conference on Genetic and evolutionary computation (pp. 1749-1750).

[77] Lei, K., Qiu, Y., & He, Y. (2006, January). A new adaptive well-chosen inertia weight strategy to automatically harmonize global and local search ability in particle swarm optimization. In 2006 1st international symposium on systems and control in aerospace and astronautics (pp. 4-pp). IEEE.

[78] Malik, R. F., Rahman, T. A., Hashim, S. Z. M., & Ngah, R. (2007). New particle swarm optimizer with sigmoid increasing inertia weight. International Journal of Computer Science and Security, 1(2), 35-44.

[79] Ememipour, J., Nejad, M.M., Ebadzadeh, M.M., & Rezanejad, J. (2009). Introduce a New Inertia Weight for Particle Swarm Optimization. 2009 Fourth International Conference on Computer Sciences and Convergence Information Technology, 1650-1653.

[80] Shi, Y., & Eberhart, R. C. (2001, May). Fuzzy adaptive particle swarm optimization. In Proceedings of the 2001 congress on evolutionary computation (IEEE Cat. No. 01TH8546) (Vol. 1, pp. 101-106). IEEE.

[81] Alfi, A., & Fateh, M. M. (2011). Intelligent identification and control using improved fuzzy particle swarm optimization. Expert Systems with Applications, 38(10), 12312-12317.

[82] Tian, D. P., & Li, N. Q. (2009, April). Fuzzy particle swarm optimization algorithm. In 2009 International Joint Conference on Artificial Intelligence (pp. 263-267). IEEE.

[83] Yadmellat, P., Salehizadeh, S. M. A., & Menhaj, M. B. (2009, June). A new fuzzy inertia weight particle swarm optimization. In 2009 International Conference on Computational Intelligence and Natural Computing (Vol. 1, pp. 507-510). IEEE.

[84] Arumugam, M.S., & Rao, M.V. (2006). On the performance of the particle swarm optimization algorithm with various inertia weight variants for computing optimal control of a class of hybrid systems. Discrete Dynamics in Nature and Society, 2006, 1-17.

[85] Zhan, Z. H., Zhang, J., Li, Y., & Chung, H. S. H. (2009). Adaptive particle swarm optimization. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 39(6), 1362-1381.

[86] Nickabadi, A., Ebadzadeh, M. M., & Safabakhsh, R. (2011). A novel particle swarm optimization algorithm with adaptive inertia weight. Applied soft computing, 11(4), 3658-3670.

[87] Kessentini, S., & Barchiesi, D. (2015). Particle swarm optimization with adaptive inertia weight. International Journal of Machine Learning and Computing, 5(5), 368.

[88] Shen, X., Chi, Z., Yang, J., & Chen, C. (2010, March). Particle swarm optimization with dynamic adaptive inertia weight. In 2010 International Conference on Challenges in Environmental Science and Computer Engineering (Vol. 1, pp. 287-290). IEEE.

[89] Adewumi, A. O., & Arasomwan, A. M. (2016). An improved particle swarm optimiser based on swarm success rate for global optimisation problems. Journal of Experimental & Theoretical Artificial Intelligence, 28(3), 441-483.

[90] Arasomwan, M. A., & Adewumi, A. O. (2013, April). On adaptive chaotic inertia weights in particle swarm optimization. In 2013 IEEE symposium on swarm intelligence (SIS) (pp. 72-79). IEEE.

[91] Xu, G. (2013). An adaptive parameter tuning of particle swarm optimization algorithm. Applied Mathematics and Computation, 219(9), 4560-4569.

[92] Chauhan, P., Deep, K., & Pant, M. (2013). Novel inertia weight strategies for particle swarm optimization. Memetic computing, 5(3), 229-251.

[93] Panigrahi, B. K., Pandi, V. R., & Das, S. (2008). Adaptive particle swarm optimization approach for static and dynamic economic load dispatch. Energy conversion and management, 49(6), 1407-1415.

[94] Tanweer, M. R., Suresh, S., & Sundararajan, N. (2015). Self regulating particle swarm optimization algorithm. Information Sciences, 294, 182-202.

[95] Harrison, K. R., Engelbrecht, A. P., & Ombuki-Berman, B. M. (2016). Inertia weight control strategies for particle swarm optimization. Swarm Intelligence, 10(4), 267-305.

[96] Rathore, A., & Sharma, H. (2017). Review on inertia weight strategies for particle swarm optimization. In Proceedings of Sixth International Conference on Soft Computing for Problem Solving (pp. 76-86). Springer, Singapore.

[97] Bansal, J. C., Singh, P. K., Saraswat, M., Verma, A., Jadon, S. S., & Abraham, A. (2011, October). Inertia weight strategies in particle swarm optimization. In 2011 Third world congress on nature and biologically inspired computing (pp. 633-640). IEEE.

[98] Suganthan, P. N. (1999, July). Particle swarm optimiser with neighbourhood operator. In Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406) (Vol. 3, pp. 1958-1962). IEEE.

[99] Ratnaweera, A., Halgamuge, S. K., & Watson, H. C. (2004). Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Transactions on evolutionary computation, 8(3), 240-255.

[100] Cui, Z., Zeng, J., & Yin, Y. (2008, November). An improved PSO with time-varying accelerator coefficients. In 2008 Eighth International Conference on Intelligent Systems Design and Applications (Vol. 2, pp. 638-643). IEEE.

[101] Chen, K., Zhou, F., Yin, L., Wang, S., Wang, Y., & Wan, F. (2018). A hybrid particle swarm optimizer with sine cosine acceleration coefficients. Information Sciences, 422, 218-241.

[102] Ziyu, T., & Dingxue, Z. (2009, July). A modified particle swarm optimization with an adaptive acceleration coefficients. In 2009 Asia-Pacific Conference on Information Processing (Vol. 2, pp. 330-332). IEEE.

[103] Chen, K., Zhou, F., Wang, Y., & Yin, L. (2018). An ameliorated particle swarm optimizer for solving numerical optimization problems. Applied Soft Computing, 73, 482-496.

[104] Jordehi, A. R. (2016). Time varying acceleration coefficients particle swarm optimisation (TVACPSO): A new optimisation algorithm for estimating parameters of PV cells and modules. Energy Conversion and Management, 129, 262-274.

[105] Kundu, R., Das, S., Mukherjee, R., & Debchoudhury, S. (2014). An improved particle swarm optimizer with difference mean based perturbation. Neurocomputing, 129, 315-333.

[106] Tian, D., Zhao, X., & Shi, Z. (2019). Chaotic particle swarm optimization with sigmoid-based acceleration coefficients for numerical function optimization. Swarm and Evolutionary Computation, 51, 100573.

[107] Jadon, S. S., Sharma, H., Bansal, J. C., & Tiwari, R. (2013). Self adaptive acceleration factor in particle swarm optimization. In Proceedings of Seventh International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2012) (pp. 325-340). Springer, India.

[108] Wu, Z., & Zhou, J. (2007, December). A self-adaptive particle swarm optimization algorithm with individual coefficients adjustment. In 2007 International Conference on Computational Intelligence and Security (CIS 2007) (pp. 133-136). IEEE.

[109] Zhan, Z. H., Xiao, J., Zhang, J., & Chen, W. N. (2007, September). Adaptive control of acceleration coefficients for particle swarm optimization based on clustering analysis. In 2007 IEEE Congress on Evolutionary Computation (pp. 3276-3282). IEEE.

[110] Liu, W., Wang, Z., Yuan, Y., Zeng, N., Hone, K., & Liu, X. (2019). A novel sigmoid-function-based adaptive weighted particle swarm optimizer. IEEE transactions on cybernetics, 51(2), 1085-1093.

[111] Mandal, S. (2017). A modified particle swarm optimization algorithm based on self-adaptive acceleration constants. International Journal of Modern Education and Computer Science, 9(8), 49.

[112] Guo, L., & Chen, X. (2009, December). A novel particle swarm optimization based on the self-adaptation strategy of acceleration coefficients. In 2009 International Conference on Computational Intelligence and Security (Vol. 1, pp. 277-281). IEEE.

[113] Ide, A., & Yasuda, K. (2005). A basic study of adaptive particle swarm optimization. Electrical Engineering in Japan, 151(3), 41-49.

[114] Bratton, D., & Kennedy, J. (2007). Defining a Standard for Particle Swarm Optimization. 2007 IEEE Swarm Intelligence Symposium, 120-127.

[115] Chen, D., & Zhao, C. (2009). Particle swarm optimization with adaptive population size and its application. Applied Soft Computing, 9(1), 39-48.

[116] Dai, H. P., Chen, D. D., & Zheng, Z. S. (2018). Effects of random values for particle swarm optimization algorithm. Algorithms, 11(2), 23.

[117] Jin, X., Liang, Y., Tian, D., & Zhuang, F. (2013). Particle swarm optimization using dimension selection methods. Applied Mathematics and Computation, 219(10), 5185-5197.

[118] Van den Bergh, F., & Engelbrecht, A. P. (2000). Cooperative learning in neural networks using particle swarm optimizers. South African Computer Journal, 2000(26), 84-90.

[119] Van den Bergh, F., & Engelbrecht, A. P. (2004). A cooperative approach to particle swarm optimization. IEEE transactions on evolutionary computation, 8(3), 225-239.

[120] Wilke, D. N., Kok, S., & Groenwold, A. A. (2007). Comparison of linear and classical velocity update rules in particle swarm optimization: notes on diversity. International Journal for Numerical Methods in Engineering, 70(8), 962-984.

[121] Paquet, U., & Engelbrecht, A. P. (2003, December). A new particle swarm optimiser for linearly constrained optimisation. In The 2003 Congress on Evolutionary Computation, 2003. CEC'03. (Vol. 1, pp. 227-233). IEEE.

[122] Wilke, D. N., Kok, S., & Groenwold, A. A. (2007). Comparison of linear and classical velocity update rules in particle swarm optimization: Notes on scale and frame invariance. International journal for numerical methods in engineering, 70(8), 985-1008.

[123] Clerc, M. (2010). Particle swarm optimization (Vol. 93). John Wiley & Sons.

[124] Duffin, K. L. & Barrett W. A. (1994). Spiders: a new user interface for rotation and visualization of n-dimensional point sets. Proceedings Visualization '94, 1994, pp. 205-211.

[125] Bonyadi, M. R., Michalewicz, Z., & Li, X. (2014). An analysis of the velocity updating rule of the particle swarm optimization algorithm. Journal of Heuristics, 20(4), 417-452.

[126] Janson S. & Middendorf M. (2007). On Trajectories of Particles in PSO. 2007 IEEE Swarm Intelligence Symposium, 2007, pp. 150-155.

[127] Salomon, R. (1996). Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. A survey of some theoretical and practical aspects of genetic algorithms. BioSystems, 39(3), 263-278.

[128] Spears, W. M., Green, D. T., & Spears, D. F. (2012). Biases in particle swarm optimization. In Innovations and developments of swarm intelligence applications (pp. 20-43). IGI Global.

[129] Hansen, N., Ros, R., Mauny, N., Schoenauer, M. & Auger, A. (2011). Impacts of Invariance in Search: When CMA-ES and PSO Face Ill-Conditioned and Non-Separable Problems. Applied Soft Computing. 11. 10.1016/j.asoc.2011.03.001.

[130] Hariya, Y., Kurihara, T., Shindo, T., & Jin'no, K. (2015, December). A study of robustness of PSO for non-separable evaluation functions. In International Symposium on Nonlinear Theory and its Applications (Vol. 1, No. 2, pp. 724-727).

[131] Clerc, M. (2012). Beyond standard particle swarm optimisation. In Innovations and Developments of Swarm Intelligence Applications (pp. 1-19). IGI Global.

[132] Zambrano-Bigiarini, M., Clerc, M., & Rojas, R. (2013, June). Standard particle swarm optimisation 2011 at cec-2013: A baseline for future pso improvements. In 2013 IEEE congress on evolutionary computation (pp. 2337-2344). IEEE.

[133] Hariya, Y., Shindo, T., & Jin'no, K. (2016, July). An improved rotationally invariant PSO: a modified standard PSO-2011. In 2016 IEEE Congress on Evolutionary Computation (CEC) (pp. 1839-1844). IEEE.

[134] Bonyadi, M. R., & Michalewicz, Z. (2014). A locally convergent rotationally invariant particle swarm optimization algorithm. Swarm intelligence, 8(3), 159-198.

[135] Hariya, Y., Shindo, T., & Jin'no, K. (2016, October). A novel particle swarm optimization algorithm for non-separable and ill-conditioned problems. In 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC) (pp. 002110-002115). IEEE.

[136] Hariya, Y., Shindo, T., & Jin'no, K. (2016). On a Rotationally Invariant of PSO. IEICE Proceedings Series, 48(B2L-F-7).

[137] Miles, R. E. (1965, December). On random rotations in R3. Biometrika, Volume 52, Issue 3-4, Pages 636–639

[138] Hussain, K., Salleh, M. N. M., Cheng, S., & Naseem, R. (2017). Common benchmark functions for metaheuristic evaluation: A review. JOIV: International Journal on Informatics Visualization, 1(4-2), 218-223.

[139] Li, L., Saldivar, A. A. F., Bai, Y., Chen, Y., Liu, Q., & Li, Y. (2019). Benchmarks for evaluating optimization algorithms and benchmarking MATLAB derivative-free optimizers for practitioners' rapid access. IEEE Access, 7, 79657-79670.

[140] Bergh, F. V. D., & Engelbrecht, A. P. (2001, July). Effects of swarm size on cooperative particle swarm optimisers. In Proceedings of the 3rd annual conference on genetic and evolutionary computation (pp. 892-899).

[141] Carlisle, A., & Dozier, G. (2001). An Off-The-Shelf PSO. Proceedings of the Workshop on Particle Swarm Optimization (Indianapolis, IN).

[142] rezaee jordehi, Ahmad & Jasni, Jasronita. (2013). Parameter selection in particle swarm optimisation: A survey. Journal of Experimental & Theoretical Artificial Intelligence. 25. 10.1080/0952813X.2013.782348.

Anexo

.1. Código Matlab del PSO propuesto

```
%Algoritmo PSO generico propuesto
function [optimal_solution] = miPSO(fitness, range, N, maxIter, n_dims, e)

    %Establecimiento de hiperparametros
    r_min = range(1);
    r_max = range(2);
    lambda=0.6+0.4*e;

    %Inicializacion aleatoria de las posiciones de las particulas
    X = r_min + (r_max - r_min).*rand(n_dims,N);
    %Inicializacion de las mejores posiciones de cada particula (pBest)
    pBest = X;
    %Inicializacion de la mejor posicion global (gBest)
    fp = cellfun(fitness,num2cell(pBest,1));
    [~,arg_g] = min(fp);
    gBest = pBest(:,arg_g);


    %Bucle principal del algoritmo
    i = 1;
    while i <= maxIter

        beta=e.*(sum((pBest-X)==0)~=n_dims);

        norm_p= sqrt(sum((pBest-X).^2,1));
        p=(pBest-X)./(norm_p+(norm_p==0));
        r1=normrnd(0,1,[n_dims,N]);
        r1=r1-sum(p.*r1,1).*p;
        norm_r1 = sqrt(sum(r1.^2,1));
        r1=r1./((norm_r1==0)+norm_r1).*norm_p;

        norm_g= sqrt(sum((gBest-X).^2,1));
        g=(gBest-X)./(norm_g+(norm_g==0));
        r2=normrnd(0,1,[n_dims,N]);
        r2=r2-sum(g.*r2,1).*g;
        norm_r2 = sqrt(sum(r2.^2,1));
        r2=r2./((norm_r2==0)+norm_r2).*norm_g;

        %Actualizar las posiciones de las particulas
        X = beta.*(pBest+lambda*r1) + (1-beta).*(gBest+lambda*r2);

        %Comprobar que las nuevas posiciones estan dentro de los limites permitidos

        %Actualizar los valores de pBest
        fx = cellfun(fitness,num2cell(X,1));
        bool = (sum((X<=r_max).*(X>=r_min))==n_dims).*(fx<fp);
        pBest = bool.*X + (1-bool).*pBest;

        %Actualizar el valor de gBest
        fp = bool.*fx + (1-bool).*fp;
        [~,arg_g] = min(fp);
```

```
        gBest = pBest(:,arg_g);

        i = i + 1;

    end

    %Determinamos el optimo encontrado
    optimal_solution = gBest;

end
```

.2. Código Matlab del PSO propuesto para funciones separables

```
%Algoritmo PSO propuesto para funciones separables
function [optimal_solution] = PSO17a(fitness, range, N, maxIter, n_dims, lambda)

    %Establecimiento de hiperparametros
    r_min = range(1);
    r_max = range(2);

    %Inicializacion aleatoria de las posiciones de las particulas
    X = r_min + (r_max - r_min).*rand(n_dims,N);
    %Inicializacion de las mejores posiciones de cada particula (pBest)
    pBest = X;
    %Inicializacion de la mejor posicion global (gBest)
    fp = cellfun(fitness,num2cell(pBest,1));
    [g,arg_g] = min(fp);
    gBest = pBest(:,arg_g);

    r=zeros(n_dims,N);

    %Bucle principal del algoritmo
    i = 1;
    while i <= maxIter

        norm= sqrt(sum((gBest-X).^2,1));
        j=sub2ind([n_dims,N],randi(n_dims,[1,N]),1:N);
        r=r*0;
        r(j)=1;
        r=r.*(-1+2*rand(1,N));
        r=r./((r==0)+abs(r));

        %Actualizar las posiciones de las particulas
        X=gBest+r.*norm*lambda;

        %Comprobar que las nuevas posiciones estan dentro de los limites
        %permitidos

        %Actualizar los valores de pBest
        fx =  cellfun(fitness,num2cell(X,1));
        fx=abs(r).*fx;
        bool = sum((X<=r_max).*(X>=r_min),1)==n_dims;
        fx=bool.*fx;
        fx(fx==0)=Inf;
        [val,ind]=min(fx,[],2);
        gBest=(val<g).*X(sub2ind([n_dims,N],(1:n_dims)',ind))+(val>=g).*gBest;
        g=fitness(gBest);

        i = i + 1;

    end

    %Determinamos el optimo encontrado
    optimal_solution = gBest;
```

```
52
53  end
```