TRABAJO FIN DE GRADO

INGENIERÍA INFORMÁTICA

Estudio de la Metaheurística "Invasive Tumor Growth Optimization" para la Resolución de Problemas de Optimización Numérica

Autor
Nieves Victoria Velásquez Díaz

Directores
Oscar Cordón García

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, Noviembre de 2017

Estudio de la Metaheurística "Invasive Tumor Growth Optimization" para la Resolución de Problemas de Optimización Numérica

Autor
Nieves Victoria Velásquez Díaz

Directores
Oscar García Cordón

Estudio de la Metaheurística "Invasive Tumor Growth Optimization" para la Resolución de Problemas de Optimización Numérica

Nieves Victoria Velásquez Díaz

Palabras clave: algoritmo, ITGO, metaheurística, problema optimización

Resumen

A partir de una metaheurística basada en el movimiento de partículas en un espacio de búsqueda determinado, se ha desarrollado un nuevo algoritmo utilizando en este principio pero adaptándolo al funcionamiento invasivo de las células tumorales.

Esta metodología se basa en que, a partir de un conjunto de células, se reparten en niveles distintos del tumor y, en función del nivel de nutrientes de cada capa, las células se irán desplazando e interactuando entre ellas con el fin de alcanzar mejores zonas con más nutrientes y evitar así la posibilidad de morir de inanición.

En primer lugar, se repartirán las distintas células en tres capas principales del tumor:

En primer lugar tendremos una primera capa más externa con una mayor concentración de nutrientes donde se encontrarán las células proliferativas centradas en expandirse dada la alta concentración de nutrientes.

Una segunda capa intermedia con un menor nivel de nutrientes donde se encuentran las células inactivas de forma que estas tienen un movimiento más limitado ya que depende de los nutrientes que van dejando tanto ellas como las células proliferativas.

Y una tercera capa en la zona más interna del tumor donde se encuentran las células moribundas. Estas apenas se desplazan dado el nivel de nutrientes tan bajo en el que se encuentran y tienen posibilidades de morir de inanición si no logran alcanzar mejores zonas.

Mediante la interacción de estas células entre ellas y su entorno se dará el crecimiento del tumor en la búsqueda de la mejor zona de nutrientes, que en nuestro caso corresponderá con la búsqueda de la mejor solución.

A o largo de este proyecto de fin de grado se procederá a explicar en mayor profundidad el mecanismo de difusión del tumor, así como se desarrollará

más a fondo esta metaheurística llegando a implementarla y se compararán los resultados obtenidos con otras metaheurísticas tanto de este tipo como de otras para poder estudiar así el rendimiento de este nuevo algoritmo frente a otros más estudiados.

Study of the Metaheuristic "Invasive Tumor Growth Optimization" for the Resolution of Numerical Optimization Problems

Nieves Victoria Velásquez Díaz

Keywords: algorithm, ITGO, metaheuristic, optimization problems

Abstract

Starting from a metaheuristic based on the movement of a specific group of particles in a certain search space, a new algorithm has been developed, adapted to the invasive growth of tumor cells.

This methodology is based on the interaction between cells distributed in three different layers of the tumor, in which each layer has a different nutrition level that establish the movement of each cell situated in it's assigned layer. Cells will interact between them and with its environment with the final purpose of reaching regions with better nutrients level ,avoiding the possibility of starving to death.

This cells are distributed in different layers of the tumor as it will be seen below:

The outer layer of the tumor, is the one in which proliferative cells are found. This type of cells have higher levels of nutrients and, because of this, are the ones with an extreme invasive behaviour.

In the intermediate layer of the tumor, the nutrient level is worse than in the outer one. Because of it, quiescent cells have a more limited movement, that depends on the nutrients left by proliferative and other quiescent cells.

Finally, in the inner layer is the one in which dying cells are located. This cells have a very limited movement because of the low nutrients level, so if they are unable to move to a better section of the tumor, they, most probably, will starve to death.

Through the interaction between this cells and its surroundings, the tumor will grow looking for a zone with better nutrient level that, in our case, will correspond to the search of the optimal solution.

During this project, we will proceed to explain in a deeper way the diffusion mechanism of the tumor, as well as the development of our metaheuristic.In

addition, we will implement it in order to study its behaviour against other metaheuristics based on the same principle as ITGO and others based on an evolutive behaviour; with the purpose of study this new method against others more studied.

Yo, **Nieves Victoria Velásquez Díaz**, alumno de la titulación Grado en ingeniería informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 74695481Z, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Nieves Victoria Velásquez Díaz

Granada a 7 de Noviembre de 2017.

D. **Oscar Cordón García**, Profesor del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Estudio de la Metaheurística "Invasive Tumor Growth Optimization" para la Resolución de Problemas de Optimización Numérica***, ha sido realizado bajo su supervisión por **Nieves Victoria Velásquez Díaz**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 7 de Noviembre de 2017.

Los directores:

Oscar Cordón García

Agradecimientos

Me gustaría dedicar estos agradecimientos a mi familia, en especial a mis padres y hermana por haberme ayudado siempre que han podido incluso cuando no entendían nada de los que les explicaba. Aunque siempre me han escuchado ayudándome, mas de una vez, a darme cuenta del error o de posibles mejoras o cambios.

Agradecerle también a mis amigos, en especial a Samuel Peregrina Morillas y a Rafel Quirós, por haber estado siempre apoyándome en los buenos y malos momentos durante estos años y por esas noches de estudio en grupo.

Índice general

Índice de figuras

Capítulo 1

Introducción y objetivos

Los problemas de optimización numérica, problemas de optimización en los que las variables consideradas son números (habitualmente reales), tienen un gran interés en la actualidad. Un ejemplo significativo de esta familia de problemas es el problema de la estimación de parámetros, en los que se persigue definir los parámetros de un modelo funcional que relacione los valores de una variable independiente con una serie de variables independientes. Este problema aparece en distintas áreas como la economía, la medicina, la biología, y la ingeniería.

La mayoría de problemas de optimización numérica son NP-completos, es decir, no puede encontrarse un algoritmo capaz de resolverlo en tiempo polinomial. Debido a su complejidad, es necesario emplear algoritmos aproximados para su resolución que encuentren soluciones de alta calidad en tiempos aceptables cuando la dimensión del problema es grande. Los métodos numéricos clásicos como el de Newton-Raphson son una buena alternativa pero presentan limitaciones relacionadas con su naturaleza de optimizadores locales. Por esta razón, el uso de metaheurísticas para su resolución ha tenido un gran desarrollo en la comunidad científica, siendo los algoritmos evolutivos una de las técnicas de resolución más extendidas.

En los últimos años, se han propuesto una gran cantidad de metaheurísticas para optimización real. El algoritmo de Invasive Tumor Growth Optimization (ITGO), basado en el principio de crecimiento de los tumores invasivos, es una propuesta reciente (2015) y muy prometedora. El objetivo del presente proyecto es hacer un estudio sobre el comportamiento de dicho algoritmo en la resolución de distintos problemas de estimación de parámetros, en comparación con algoritmos clásicos y otras metaheurísticas de optimización real. Este objetivo principal se descompone en los siguientes subobjetivos:

1) Estudio y análisis de la metaheurística Invasive Tumor Growth Opti-

mization (ITGO).

2) Selección de algunos problemas clásicos de optimización numérica, como la minimización de funciones matemáticas complejas con un número alto de variables.

3) Estudio de la aplicabilidad del algoritmo ITGO a dichos problemas, así como el de otras técnicas tales como los métodos numéricos y los algoritmos evolutivos.

4) Implementación del algoritmo ITGO y de distintos métodos alternativos de optimización real para la minimización de las funciones seleccionadas.

5) Análisis práctico del rendimiento de las técnicas implementadas en la resolución de los problemas considerados.

Capítulo 2

Introducción a las metaheurísticas

El término "Metaheurística", deriva de la composición de dos palabras griegas.

La palabra griega "heurística" se asocia con el concepto de "encontrar". Se vincula a la exclamación de "eureka" de Arquímedes al descubrir su famoso principio.
En Inteligencia Artificial, se denomina así a una serie de procedimientos y técnicas que emplean un conocimiento específico acerca de un problema para su resolución, tratando de aportar soluciones (bien exactas o aproximadas) que vienen limitadas por restricciones de tipo temporal.
Hay que tener en cuenta, que se usa el término heurístico y no el término exacto, pues el término heurístico no garantiza la optimalidad ni la factibilidad de las soluciones, simplemente se tiene cierta confianza de que la solución alcanzada tiene un alto grado de optimalidad y/o factibilidad. Es por esto que envuelve o podría aportar determinadas soluciones que es necesario rechazar.

La palabra griega "meta" significa "más allá, en un nivel más alto".
No existe una definición exacta de Metaheurísticas, pero podría decirse que son una nueva familia de algoritmos aproximados de propósito general consistentes en procedimientos iterativos que guían una heurística subordinada combinando de forma inteligente distintos conceptos para explorar y explotar adecuadamente el espacio de búsqueda.

Las propiedades fundamentales que caracterizan a las metaheurísticas son las siguientes:

- Las Metaheurísticas son estrategias que "guían" el proceso de búsqueda.

- La meta es explorar eficientemente el espacio de búsqueda para encontrar soluciones (sub)óptimas.
- Las técnicas que constituyen los algoritmos de Metaheurísticas van desde procedimientos simples de búsqueda local a complejos procesos de aprendizaje.
- Los algoritmos de Metaheurísticas son aproximados y no determinísticos.
- Incorporan mecanismos para evitar quedarse atrapados en áreas prometedoras del espacio de búsqueda.
- Los conceptos básicos de Metaheurísticas permiten una descripción de nivel abstracto.
- Las Metaheurísticas no son específicas del problema.
- Las Metaheurísticas hacen uso de conocimiento específico del dominio y / o experiencia de búsqueda (memoria) para sesgar la búsqueda.
- Actualmente, las metaheurísticas más avanzadas aprovechan el camino recorrido por la búsqueda (plasmado en algún tipo de memoria) para guiar la búsqueda.

El principal objetivo en el diseño de una metaheurística es ser eficiente y efectiva explorando el espacio de búsqueda. Este objetivo principal, engloba dos subjetivos, correspondientes a los conceptos de intensificación y diversificación.

El primer subobjetivo consiste en una **intensificación** del proceso de búsqueda, es decir, tener una característica de búsqueda en sentido local o explotación.

El segundo subobjetivo consiste en tener una apropiada **diversificación** del proceso de búsqueda, es decir, tener una característica de búsqueda global o exploración, no quedando limitada a fracciones del espacio de búsqueda.

Las versiones básicas de casi todas las metaheurísticas tienen un mecanismo de trabajo que incluye intensificación y diversificación en su proceso de búsqueda.

Los mecanismos de intensificación y diversificación se dividen en:

- Intrínseco (básico): Mecanismos asociados al comportamiento básico del algoritmo.
- Estratégicos: Técnicas y estrategias añadidas al procedimiento básico para mejorar el comportamiento global del algoritmo.

Para conseguir una metaheurística buena y efectiva, es necesario un balance correcto entre intensificación y diversificación. Esto es necesario para:

- Identificar rápidamente regiones del espacio con soluciones de buena calidad.
- No consumir mucho tiempo en regiones del espacio no prometedoras o que ya hayan sido exploradas previamente.

Hay diversas aproximaciones para la clasificación de las Metaheurísticas según sus propiedades. A continuación se presentan algunos criterios para la clasificación y se da una descripción de algunas de las Metaheurísticas básicas.

2.1. Clasificación de las Metaheurísticas.

Como se ha comentado anteriormente, las Metaheurísticas son estrategias para diseñar métodos heurísticos. Existen distintos tipos de métodos heurísticos: de relajación, constructivos, de búsqueda y evolutivos. Por tanto, una posible clasificación de las Metaheurísticas puede ir en base al método heurístico que utiliza y sería la siguiente [BMMM03]:

- Metaheurísticas de Relajación: Consiste básicamente en realizar modificaciones del modelo del problema original que posibilitan su resolución de forma más sencilla
- Metaheurísticas Constructivas: Se elige un criterio que vaya determinando cada una de las componentes de la solución y se va construyendo la misma poco a poco.
- Metaheurísticas de Búsqueda: Constituyen el núcleo principal de las Metaheurísticas en procesos de optimización. Guían las estrategias que recorren el espacio de soluciones del problema.
- Metaheurísticas Evolutivas: Conducen la evolución de las poblaciones (conjuntos de soluciones) en el espacio de búsqueda para intentar acercarse a la solución óptima.

La clasificación anterior no es única. Existen algunas Metaheurísticas no englobadas en ninguno de los tipos anteriores, como es el caso de las redes neuronales o de los sistemas de colonias de hormigas, que están inspirados en la naturaleza. Estas últimas serán en las que nos apoyemos, para desarrollar nuestro algoritmo ITGO, y para compararlo con uno similar, el PSO además de apoyarnos, también en dos algoritmos basados en la metaheurísticas evolutivas como son el algoritmo DE y JDE.

Además, hay que decir que se pueden obtener nuevas Metaheurísticas mediante la combinación de metaheurísticas de distinto tipo, como es el caso de la Metaheurística GRASP, donde se combina una fase constructiva con una fase de búsqueda.

Capítulo 3

ITGO

3.1. Descripción del problema

Con este proyecto de fin de grado se pretende estudiar un nuevo algoritmo de optimización llamado ITGO (Intrusive Tumor Growth Optimization) que se basa en el principio de crecimiento tumoral invasivo [1].

El estudio de mecanismos de crecimiento tumoral muestran que cada célula lucha por los nutrientes que hay en su microentorno para crecer y proliferar. En el algoritmo ITGO, las células tumorales se dividen en 3 categorías, células proliferativas, células inactivas y células moribundas que se distribuyen por las distintas capas del tumor como se ve a continuación.

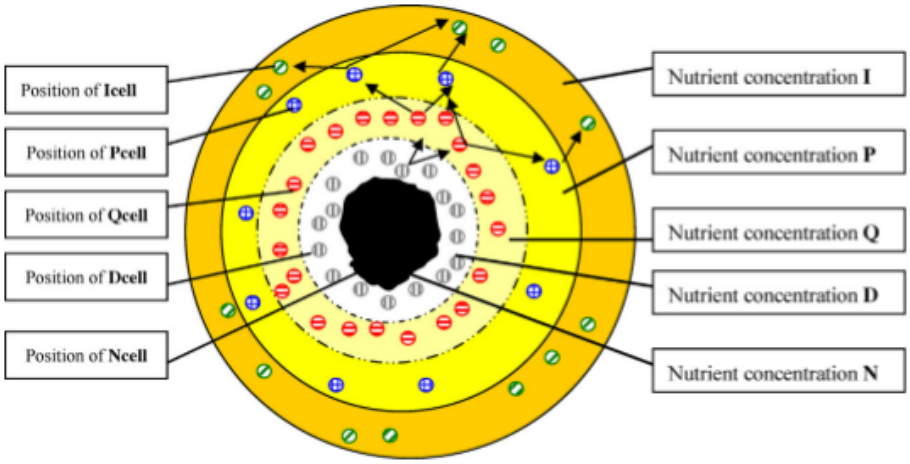

Figura 3.1: Distribución de las células en el tumor.

Como se puede observar, en función de la concentración de nutrientes de cada capa se van repartiendo las distintas células del tumor de forma que

en las capas exteriores se encuentran las **células invasivas** (ICell), que se explicarán más adelante y se encargan de suplantar células muertas, junto con las **células proliferativas** (PCell).
En la siguiente capa vemos las **células inactivas** (QCell), seguidas de las **células moribundas**(DCell) y finalmente las células muertas (NCell) que aunque están presentes en nuestro algoritmo serán suplantadas por las invasivas de manera que no quedará constancia de ellas de una iteración a la siguiente.

Con esto en mente vemos que el movimiento de las células se basa en la quimiotaxia, es decir, se mueven en guiadas por un comportamiento aleatorio y por interactuar con las otras células de distintas categorías con la finalidad de encontrar una zona con un alto contenido de nutrientes.

El comportamiento de invasión de las células del tipo proliferativas e inactivas está guiado por el vuelo de levy (o Levy flight) [2], como veremos más adelante en este capítulo en la sección 3.4; mientras que las células moribundas se guían por los otros dos tipos de células, además, cada cierto tiempo una célula moribunda sufrirá una mutación de forma que se añade diversidad en el caso en el que una célula muera, es decir, que no encuentre una zona de nutrientes válida a la que moverse.

3.2. Esquema de representación empleado

Para el desarrollo de este algoritmo, nos hemos apoyado principalmente en dos matrices llamadas “cCells” y “hCells” donde almacenaremos la posición de las células actuales y la mejor posición encontrada para cada una de ellas respectivamente, además de dos vectores de pairs llamados “nutrientes” y “fitness” donde almacenamos el valor actual de fitness (o nutrientes) de cada célula y el mejor valor de fitness hallado. Todos estos valores se irán actualizando durante la ejecución del ITGO según se vaya desplazando las células por el espacio de búsqueda.

El algoritmo nos devolverá el fitness de la mejor célula encontrada durante la ejecución.

3.3. Descripción de la función objetivo

A la hora de evaluar la posición actual de cada célula y la mejor posición encontrada, nos basaremos en un conjunto de funciones matemáticas conocidas como Funciones de test del CEC2005[3].

En este conjunto se compone de funciones de varios tipos que son:

- Funciones Unimodales (U): este tipo de funciones se caracterizan por tener un único óptimo, tanto local como global.
- Funciones Multimodales (M): este tipo de funciones se caracterizan por tener múltiples óptimos locales y además podemos subdividir este conjunto en dos diferentes, que son:
  - Funciones básicas.(MB)
  - Funciones expandidas:(ME) donde los óptimos globales se encuentran fuera del rango de inicialización.
- Funciones Híbridas Compuestas (H): compuestas mediante hibridación de 10 de las funciones mencionadas anteriormente.

En la siguiente tabla vamos a ver las distintas funciones que pertenecen a cada una de las categorías explicadas previamente de forma que tenemos:

Cuadro 3.1: Detalles CEC2005

Funcion	Nombre función	Tipo
1	Función esférica desplazada	U
2	Problema 1.2 de Schwefel	U
3	Función elíptica desplazada y altamente condicionada	U
4	Problema desplazado 1.2 de Schwefel con Ruido	U
5	Problema 2.6 de Schwefel	U
6	Función de Rosenbrock desplazada	MB
7	Función de Griewank desplazada y rotada	MB
8	Función de Ackley desplazada y rotada	MB
9	Función de Rastrigin desplazada	MB
10	Función de Rastrigin desplazada y rotada	MB
11	Función de Weierstrass desplazada y rotada	MB
12	Problema de Schwefel 2.13	MB
13	Funciones de Griewank y Rosenbrock expandidas y extendidas	ME
14	Función Scaffer expandida, extendida y rotada	ME
15	Función híbrida compuesta 1	H
16	Función híbrida compuesta 1 rotada	H
17	Función híbrida compuesta 1 con ruido	H
18	Función híbrida compuesta 2	H
19	Función híbrida compuesta 2 con una estrecha base sobre el óptimo	H
20	Función híbrida compuesta 2 con óptimos globales	H
21	Función híbrida compuesta 3	H
22	Función híbrida compuesta 3 altamente condicionada	H
23	Función híbrida compuesta 3 rotada y no continua	H
24	Función híbrida compuesta 4	H
25	Función híbrida compuesta 4 sin frontera	H

De este conjunto de funciones tomaremos un subconjunto que contendrá las funciones que se usarán para evaluar nuestro algoritmo. Este subconjunto estará compuesto por las funciones 1 a la 11, y 13 a la 15, como veremos en el análisis de resultados en el capítulo 5, por lo que no nos centraremos sólo en un tipo de función si no que veremos cómo se desenvuelve para casi todos los tipos.

3.4. Estructura del algoritmo

3.4.1. Introducción

En primer lugar se procederá a explicar la estructura principal en la que se basa el algoritmo desarrollado para este proyecto teniendo en cuenta una serie de puntos:

- Las células se dividen en 4 grupos o capas (al querer simular el hecho de que se trata de un tumor) de forma que tenemos tres conjuntos principales mencionados antes que son:
  - Las células proliferativas que se encuentran en la capa exterior, donde, al haber una mayor concentración de nutrientes son más activas.
  - Las células inactivas que se encuentran en una capa intermedia donde no abundan tanto los nutrientes por lo que se mueven más lentamente
  - Las células moribundas que están en la capa más interna, por lo que al haber una baja concentración de nutrientes apenas pueden moverse y existe el riesgo de que acaben muriendo de inanición.
  - Las células invasivas que son las que sustituyen a aquellas células moribundas tras quedarse sin nutrientes y morir como se verá con más detalle más adelante.
- Las células interactúan entre ellas y con su entorno de modo que las proliferativas se basan en su entorno en busca de mejores zonas de nutrientes, mientras que las inactivas se guiarán por otras de su mismo tipo y las de tipo proliferativas. Por otro lado, las moribundas, dada su bajo nivel de nutrientes, se guía por los otros dos tipos de células siempre que les sea posible, en caso contrario morirán.
- Para evitar que las células queden atrapadas por un óptimo local se ha añadido una funcionalidad extra donde para cada una de las células, si no han logrado mejorar su valor de fitness en un tiempo establecido (como puede ser en 3 movimientos por el espacio) darán un salto a otra zona guiadas por un random walk.

Con esto en mente, a continuación se mostrará el esquema general que va a seguir el ITGO y posteriormente estudiaremos algunas funciones más en profundidad.

Algorithm 1 Estructura del ITGO

Require: fun: valor de la función de evaluación, dim: la dimensión de la célula, tam: tamaño de la población, $iters$: número máximo de iteraciones
Ensure: $fitness$: variable con el fitness de la mejor célula encontrada

```
procedure ITGO(fun, dim, tam, iters)
    mejorSolucion
    poblacion ← poblacionInicial(dim, tam)
    while i ≤ iters do
        poblacion ← ordenamosPoblacion(tam)
        mejorSolucion ← poblacion(0)
        DividimosLaPoblacion(tam, Pcel, Icel, Mcel)
        Pcel ← crecimientoProliferativas()
        Icel ← crecimientoInactivas()
        Mcel ← crecimientoMoribundas()
        crecimientoInvasivas()
        poblacion ← Pcel, Icel, Mcel
        poblacion ← ordenamosPoblacion(tam)
        if funcObjetivo(poblacion(0)) < funcObjetivo(mejorSolucion)
then
            mejorSolucion ← poblacion(0)
        end if
        Incrementamosi
    end while
     return mejorSolucion
end procedure
```

Como podemos ver, en primer lugar generamos una población de células del tamaño y dimensión indicados, tras esto, recorremos dicha población y la ordenamos de mejor valor de fitness a peor, por lo que asignando a la variable **mejorSolucion** la primera posición del vector, estamos guardando la mejor hasta el momento.

Una vez ordenada la población, la dividiremos en tres grupos de células, **Pcel, Icel ,Mcel**, de forma que el subconjunto Pcel tendrá el 20 % de las células totales, Icel tendrá un 60 %, y Mcel tendrá el 20 % restante. Con esta división se intenta mantener un equilibrio entre la exploración y la explotación de manera que las células se van expandiendo lentamente. Esto se debe a que solo hay un 20 % destinado a ello, mientras que el conjunto de las Icel se centra más en la explotación. Por otra parte con el subconjunto restante hacemos tanto explotación y exploración como veremos más adelante.

Tras la división del conjunto total de células empiezan las cinco funcio-

nes principales del algoritmo, que son: **crecimientoProliferativas()**, **crecimientoInactivas()**, **crecimientoMoribundas()**, **crecimientoInvasivas()** con estas funciones iremos moviendo las células en el espacio de búsqueda siguiendo un patrón distinto con cada uno de los subconjuntos.

Una vez se han desplazado las células por el espacio, se agrupa de nuevo la población con los cambios hechos y se vuelve a ordenar en función del nivel de nutrientes de cada una de ellas, si la primera célula del vector es mejor que la que teníamos almacenada actualizamos la variable con la mejor encontrada y, finalmente, incrementamos en 1 el número de iteraciones.

3.4.2. Partes del ITGO

Como hemos visto previamente, las partes principales en las que podemos subdividir el algoritmo son: el movimiento de las células proliferativas, el movimiento de las células inactivas, el movimiento de las células moribundas y el de las células invasivas. Cada subconjunto se mueve siguiendo un patrón de actuación distinto que serán explicados individualmente a continuación:

Células proliferativas

Este tipo de células son las más activas dado a que tienen un mayor nivel de nutrientes, es decir mejor valor de fitness, y son las que se desplazan por el espacio de búsqueda aportando exploración al algoritmo.

Estas células tienen una habilidad invasiva fuerte, y para reflejarlo nos hemos basado el vuelo de levy. Podemos considerar el vuelo de levy como un método basado en una metodología similar a la del random walk. En este caso el tamaño de los pasos (el salto que da en el espacio de búsqueda) se basa en una distribución de levy y la dirección que toma obedece a una distribución uniforme.

A la hora de obtener el tamaño de los pasos y la dirección de los mismos para poder ver hacia dónde se va a desplazar una célula, nos basaremos en las siguientes fórmulas:

1) En primer lugar calcularemos el tamaño del paso y su dirección para ello nos basaremos en las siguientes fórmulas:

$$Levy(paso)\ paso^{-1-w}, (0 < w \leq 2) \tag{3.1}$$

$$paso = \frac{u}{|v|^{1/w}} \tag{3.2}$$

Donde u y v siguen una distribución normal del tipo $u\ N(0,1)$, y $v\ N(0,\delta_v^2)$. Para poder calcular el valor de δ_v aplicaremos la siguiente fórmula:

$$\delta_v = \frac{\Gamma(1+\varpi)sin(\Pi\varpi)/2}{\Gamma[(1+\varpi)/2]\varpi 2^{(\varpi-1)/2}}^{1/\varpi} \tag{3.3}$$

Donde ϖ es una constante entre [0.3, 1.99]

2) Una vez tenemos el valor del paso, controlaremos la longitud de este de la forma:

$$\alpha = rand(0,1)\frac{i}{iters} \tag{3.4}$$

$$newPcel = Pcel + \alpha Levy(paso) \tag{3.5}$$

Donde i indica la iteración actual del while e iters es el valor de iteraciones máximo.

De esta manera, en las primeras iteraciones del algoritmo, dará pasos cortos las primeras veces, y según vaya avanzando el algoritmo dará pasos mas largos para así evitar el caer en zonas que ya hayan sido exploradas previamente.

Con esto en mente, el método para estas células quedaría de la siguiente manera:

Algorithm 2 Crecimiento de las células Proliferativas

Require: *tamPcel*: tamaño del subconjunto Pcel, i: valor de la iteración actual, *iters*: valor de las iteraciones totales

Ensure: *celulas*: actualiza el valor de las células que pertenecen al subconjunto Pcel almacenadas en cCell y hCell

```
procedure CRECIMIENTOPROLIFERATIVAS(tamPcel, i, iters)
    newPcel
    for j do tamPcel
        for k do dim
            newPcel ← cCell[j] + αLevy(paso)
        end for
        if funcObjetivo(newPcel) < funcObjetivo(hCell[j]) then
            hCells[j] ← newPcel
            fitness[j] ← funcObjetivo(newPcel)
        else
            cCells[j] ← newPcel
            nutrientes[j] ← funcObjetivo(newPcel)
            gc ← +1
        end if
        if gc > maximo then
            randomWalk()
        end if
    end for
     return celulas
end procedure
```

Como podemos ver, en primer lugar recorremos el subconjunto de las células proliferativas y calculamos su nueva posición. Tras esto, obtenemos el valor de fitness de la nueva solución mediante la función "funcObjetivo()"; si la nueva célula mejora a la actual, la guardamos y actualizamos su valor tanto en la matriz de mejores soluciones como el vector asociado con su valor de fitness. En caso de que no haya una mejora, actualizamos su valor en la matriz de células actuales y el nuevo fitness en el vector asociado a dicha matriz (nutrientes) e incrementamos en 1 la variable "gc" donde almacenaremos las veces que no se ha obtenido una mejora para que en caso de que supere un valor máximo se de un salto en el espacio de búsqueda gracias al algoritmo "randomWalk()".

Células inactivas

Este tipo de células están en la capa intermedia por lo que no se mueven tan rápido como las de la capa superior dado al nivel de nutrientes que hay a su alrededor, por lo tanto estas células se guían siguiendo a otras de su mismo tipo y a las células de tipo proliferativas. Es debido a esto que el movimiento invasivo de estas células, se dé mediante mutación.

A la hora de ver si una célula va o no a mutar nos basaremos en el siguiente ratio:

$$Icel = newIcel, rand(0,1) < e^{(\frac{Fes}{Max_fes}-1)} \tag{3.6}$$

Como hemos comentado antes, las células de este tipo se desplazan siempre que hayan sufrido una mutación previamente. Esta mutación ocurre, como podemos ver, en la fórmula (3.6). De modo que, como depende del número de iteraciones por el que vaya el algoritmo, durante las primeras iteraciones no se dará la mutación dado que $rand(0,1)$ nunca será menor que -1, sin embargo según vaya avanzando el algoritmo estas células irán mutando con más frecuencia. A la hora de realizar el desplazamiento nos basaremos en la siguiente fórmula:

$$newIcel = \begin{cases} IcelActual + \beta \cdot step \cdot (mejorPcel + IcelActual) + \\ +\beta \cdot step \cdot ((IcelCercana1 + IcelCercana2)), rand < 0.5 \\ \\ IcelActual + \beta \cdot step \cdot (PcelActual + IcelActual) + \\ +\beta \cdot step \cdot ((IcelCercana1 + IcelCercana1)), rand > 0.5 \end{cases} \tag{3.7}$$

Como podemos ver, en la mitad de los casos seguiremos tanto a la mejor posición encontrada de una de las células proliferativas escogidas aleatoriamente, p, como a la posición de la dos células cercanas de tipo inactivas. Mientras que en la otra mitad de los casos, nos guiaremos por la posición actual de dicha célula p (no de la mejor) junto a la posición de las dos células cercanas de tipo inactivas.

Como se ha comentado anteriormente este tipo de células también se desplaza con la ayuda del vuelo de levy, por lo tanto usaremos, además, los valores de $\beta = rand(0,1)\Delta normal(0,1)$, con quien controlaremos la longitud de los pasos y el valor de *step* siendo este la longitud del paso explicada en la fórmula (3.1).

Por lo tanto el método con el que hemos desarrollado esta función es el siguiente:

Algorithm 3 Crecimiento de las células Inactivas

Require: $tamPcel$: tamaño del subconjunto Pcel, $tamIcel$: tamaño del subconjunto Icel, i: valor de la iteración actual, $iters$: valor de las iteraciones totales

Ensure: $celulas$: actualiza el valor de las células que pertenecen al subconjunto Icel almacenadas en cCell y hCell

```
procedure CRECIMIENTOINACTIVAS(tamPcel, tamIcel, i, iters)
    newPcel
    for j do tamIcel
        mejorPcel, actualPcel ← random(0, tamPCel)
        distancias ← distanciaEuclidea(Icels)
        for k do dim
            if muta then
                if rand < 0,5 then
                    newIcel ← calculoCelula(cercana1, cercana2, mejorPcel)
                else
                    newIcel ← calculoCelula(cercana1, cercana2, actualPcel)
                end if
                randomWalk()
            end if
        end for
        if gc > maximo then
            randomWalk()
        end if
    end for
     return celulas
end procedure
```

Como podemos ver, en primer lugar calculamos la distancia entre todas las células inactivas y las guardamos en un vector de apoyo.

Tras esto y para cada una de las células, en primer lugar vemos si para esa célula se va a producir la mutación. De ser así lo siguiente que haremos será escoger una célula del subconjunto de las células proliferativas y las dos más cercanas de las inactivas y junto con estos tres datos y la longitud del paso, desplazamos la célula. De una forma aleatoria decidiremos si se apoyará en la mejor célula proliferativa o en la actual. En caso de que no se

produzca la mutación no ocurrirá nada.

Finalmente y al igual que hemos hecho en el paso anterior, obtendremos el valor de fitness de la nueva célula calculada. Si es mejor actualizamos ambas matrices y ambos vectores asociados, y si no actualizamos sólo la matriz de las células actuales así como su vector relacionado e incrementamos en 1 la variable "gc" para que en caso de que lleve tiempo sin mejorar de un salto ayudándose del algoritmo "randomWalk()".

Células moribundas

Este tipo de células están en la capa interna, por lo que son las que menos se mueven dada la baja cantidad de nutrientes de la zona. Este tipo de células en caso de que no obtengan los nutrientes suficientes como para poder desplazarse pasarán al estado de "muertas" y una célula invasiva tomará su lugar como veremos en el siguiente punto. Para que se dé un desplazamiento, este tipo de células se guiarán por las células de las otras dos capas superiores.

$$newMcel = Dcel + \gamma \cdot (mejorPcel - Dcel) + \\ +\gamma \cdot ((Icel - Dcel)) \tag{3.8}$$

Como vemos, estas células se guían por otras dos, ambas escogidas de forma aleatoria. Una de las dos células seleccionadas aleatoriamente, pertenece al conjunto de las células proliferativas, en el que seleccionaremos la mejor posición de esta, mientras que la otra célula pertenece al conjunto de las células inactivas.

Por lo tanto, el método resultante será:

Algorithm 4 Crecimiento de las células Moribundas

Require: $tamPcel$: tamaño del subconjunto Pcel, $tamIcel$: tamaño del subconjunto Icel, $tamMcel$: tamaño del subconjunto Mcel, i: valor de la iteración actual, $iters$: valor de las iteraciones totales

Ensure: $celulas$: actualiza el valor de las células que pertenecen al subconjunto Mcel almacenadas en cCell y hCell

```
procedure CRECIMIENTOMORIBUNDAS(tamPcel, tamIcel, tamMcel, i, iters)
    newPcel
    for j do tamMcel
        mejorPcel ← random(0, tamPCel)
        Icel ← random(0, tamICel)
        for k do dim
            newDcel ← calculoDcel(mejorPcel, Icel)
        end for
        if gc > maximo then
            randomWalk()
        end if
    end for
     return celulas
end procedure
```

Como podemos ver, este método es más sencillo que el anterior, dado que simplemente se seleccionan las dos células comentadas previamente, y se calcula la nueva posición de las células. Finalmente y como para los casos anterior, vemos el fitness de la nueva célula obtenida, si mejora actualizamos las matrices correspondientes y los vectores y si no sólo actualizamos la matriz actual incrementando la variable "gc" igualmente para que se dé un "randomWalk()" si lleva tiempo sin mejorar.

Células invasivas

Este tipo de células se consideran hijas mutadas de las células proliferativas, y, además, pueden desplazarse por su entorno y acceder a otras capas. En este caso, una vez se genere una célula de este tipo, sustituirá a una célula que haya muerto por falta de nutrientes.

Este comportamiento está establecido por las siguientes fórmulas:

$$mejorPcel = Random(0, tamPcel) \tag{3.9}$$

$$Icel = mejorPcel + \eta \cdot (newCel - mejorPcel) \tag{3.10}$$

Donde $\eta = rand(0, 1)$. Por lo tanto el método desarrollado sería el siguiente:

Algorithm 5 Crecimiento de las células Invasivas

Require: *tamPobl*: tamaño de la población
Ensure: *celulas*: sustituye a las células muertas y actualiza los valores en las matrices cCell y hCell

```
procedure CRECIMIENTOINVASIVAS(tamPobl)
    newPcel
    for j do tamPobl·0,2
        mejorPcel ← random(0, tamPCel)
        if nutrientes < mediaNutrientes then
            nuevaCel ← generarCelula()
            for k do dim
                newIcel ← calculoIcel(mejorPcel, nuevaCel)
            end for
        end if
        if gc > maximo then
            randomWalk()
        end if
    end for
     return celulas
end procedure
```

Como vemos, en primer lugar calcularemos la media de los nutrientes de las células moribundas, todas las que queden por debajo de este valor medio serán consideradas como muertas. Tras esto, generaremos un vector de valores aleatorios y junto con este valor y la mejor posición de una de las células proliferativas seleccionada aleatoriamente, crearemos la nueva célula que sustituirá a la muerta.

Tras esto, igual que en todos los casos anteriores veremos el valor de fitness de a nueva célula, si mejora a la muerta (que suele ser lo normal) se actualizan las matrices y vectores, y si no, sólo actualiza la matriz actual.

Random Walk

Este método es el que se encarga de hacer que las células se desplacen grandes distancias siempre que se estanquen en una zona en la que lleva un tiempo sin encontrar mejora, por ejemplo que se encuentre cerca de un óptimo local. Para esto nos apoyaremos en una fórmula principal declarada como sigue:

$$cel = CelAnterior + \lambda \cdot \left(\frac{nuevaCel}{\parallel nuevaCel \parallel} \right) \tag{3.11}$$

Donde "nuevaCel" hace referencia a la generada en la fórmula (3.9).

Por lo tanto el método quedaría:

Algorithm 6 Random Walk

Require: *celula*: célula que vamos a mover, *gc*: veces que no ha mejorado
Ensure: $NuevaCelula$: nueva posición de la célula

```
procedure CRECIMIENTOINVASIVAS(tamPobl)
    nuevaCel ← generarCelula()
    for j do tamPCel
        nuevaCel ← calculoNuevaCel(nuevaCel, celula)
    end for
    cCell ← nuevaCel
    hCell ← nuevaCel
    gc ← 0
     return NuevaCelula
end procedure
```

Como podemos ver, para dar con una nueva posición en primer lugar generamos una nueva célula y tras esto aplicaremos la fórmula (3.11). Tras obtener la nueva célula, actualizaremos el valor de dicha célula en ambas matrices y en los vectores asociados y tras esto reiniciaremos el contador "gc" a 0; contador que se incrementará siempre que la célula no encuentre una mejor solución.

Capítulo 4

Descripción de los algoritmos de comparación

Para comprobar el rendimiento del ITGO lo hemos comparado en primer lugar con un algoritmo que, como el desarrollado, se basa en el movimiento de partículas en un espacio de búsqueda determinado. Por lo tanto se ha decidido escoger el PSO como principal algoritmo de comparación.

Además también se ha usado dos algoritmos de tipo evolutivo como son el DE y JDE para compararlos con el algoritmo ITGO.

Con esto presente, se procederá a explicar brevemente cada uno de los algoritmos mencionados anteriores para una mejor comprensión del proyecto.

4.1. PSO

También denominado particle swarm optimization o problema de optimización de partículas en español [4]. Este algoritmo se atribuye a los investigadores James Kennedy y Russell C. Eberhart quienes lo describieron alrededor de 1995 y se inspira en el comportamiento de las aves o los enjambres de insectos.

Este algoritmo funciona de forma que cada partícula es un animal en el espacio de búsqueda y se va desplazando por él, de forma que siempre está en movimiento; por lo que, a diferencia de otros algoritmos, sus individuos no mutan ni mueren.

Este algoritmo funciona de manera que al comenzar, a cada miembro de la población se le asigna una posición y una velocidad de forma aleatoria dentro de la dimensión del espacio de búsqueda, y tras esto, cada miembro es evaluado según una función de fitness.

Con esta evaluación en primer lugar comprobaremos si se ha mejorado el fitness de la evaluación anterior. De ser así se actualizará el mejor valor local de cada partícula; además también compararemos dicho valor con el mejor global de manera que si se mejora, lo estableceremos como el mejor global modificando así la velocidad y posición en función de este nuevo valor, como vemos a continuación:

$$v_{id} = v_{id} + c_1 \cdot rand() \cdot (pos_{id-mejor} - pos_{id}) + c_2 \cdot rand() \cdot (pos_{id-mejorGlobal} - pos_{id}) \tag{4.1}$$

$$pos_{id} = pos_{id} + v_{id} \tag{4.2}$$

Este algoritmo tiene el problema de que existe la posibilidad de que los individuos oscilen entre un óptimo local y uno global hasta que se cumplan los criterios de parada o de convergencia lo que hace que se obtengan peores tiempos que con otros, además si se da el caso de que el óptimo esté lejos del camino a seguir por el individuo, el rendimiento del algoritmo empeora dado que la convergencia se dificulta.

4.2. DE

También denominado como Evolución diferencial [5]. Este modelo fue propuesto por Storm y Price en 1998.

Este modelo fue propuesto para resolver problemas de optimización con parámetros reales. Consiste en ir evolucionando una población de vectores determinada, y para ello se basa en un operador de cruce o recombinación que se aplica una vez se produce la mutación para obtener nuevos miembros de la población a partir de los ya existentes.

Esta recombinación se conoce como mutación diferencial, y consiste en, a partir de dos individuos de la población escogidos aleatoriamente, tomar una diferencia proporcional y con esto añadírselo a un tercer miembro escogido también aleatoriamente. Tras esto, se habrá generado un nuevo miembro de la población denominado "individuo mutado" como vemos en la siguiente fórmula:

$$w_i = v_1 + \mu \cdot (v_2 - v_3) \tag{4.3}$$

La constante $\mu > 0$ conocida como la "constante de mutación" es la que establece el rango de diferenciación entre los individuos v_2 y v_3 para evitar que la búsqueda se pueda llegar a estancar.

Una vez se ha mutado, se procede a la recombinación de los individuos v_i con la finalidad de generar un individuo nuevo mezclando estos individuos con el calculado previamente w_i como podemos ver a continuación:

$$u_i(j) = \begin{cases} w_i(j) rand \leq C_\gamma \\ v_i(j) \end{cases} \tag{4.4}$$

Donde C_γ es una variable con la que predefinimos la probabilidad de las componentes de cada individuo de ser seleccionadas.

Si el individuo nuevo es mejor que el que íbamos a sustituir en un principio lo hará, formando parte de la población principal, mientras que, si por el contrario no es mejor, pasará a ser el nuevo individuo que se pretenderá mutar en la siguiente generación.

Este algoritmo tiene una serie de ventajas frente a otros algoritmos como pueden ser: el buen equilibrio entre la velocidad de convergencia y evitar que esta se produzca prematuramente, es eficiente con problemas reales y artificiales, además de ser uno de los más rápidos dentro de los algoritmos evolutivos y con un número de funciones obtiene mejores resultados que el algoritmo PSO.

4.3. JDE

Se trata de un algoritmo autoadaptado. Es una modificación del explicado anteriormente de forma que se busca mejorar el rendimiento de "DE" mediante la mejora del uso de los parámetros μ y C_γ [6].

Esta mejora ocurre al ir variando dichos valores a lo largo de la ejecución del algoritmo siguiendo la evolución de las soluciones; de esta manera se sigue un esquema autoadaptado donde estos valores se reevalúan tras cada ciclo terminado de cada generación con la finalidad de encontrar valores para μ y C_γ que den lugar a hijos con mejores soluciones y sobre estos reevaluaremos ambas variables comenzando un ciclo nuevo.

$$F_{i,G+1} = \begin{cases} F_l + rand_1 \times F_u, rand_2 < \tau_F \\ F_{i,G} \end{cases} \tag{4.5}$$

$$C_{\gamma_{i,G+1}} = \begin{cases} rand_3 \times F_u, rand_4 < \tau_{C_\gamma} \\ C_{\gamma_{i,G}} \end{cases} \tag{4.6}$$

Con los valores de F_u y F_l se pretende limitar tanto inferior como superiormente el valor de F. Los valores que se le suelen asignar a estas variables son: $F_l = 0.1$ y $F_u = 0.9$, con $\tau_F = \tau_{C_\gamma} = 0.1$ y $C_\gamma \in [0, 1]$

Con estos tres algoritmos ya explicados y junto al desarrollado en este proyecto (ITGO) en los siguientes capítulos se va a estudiar cómo se desenvuelven en situaciones similares a la hora de obtener resultados para un problema de misma dimensión.

Capítulo 5

Análisis de resultados

A continuación se procederá a estudiar una primera instancia de los datos obtenidos.

En primer lugar se ha obtenido un primer conjunto de datos evaluando las soluciones con la ayuda de las funciones del CEC2005, explicadas en la sección 3.3. Para ello se han seleccionado las funciones: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14 y 15 respectivamente con las que se evaluarán los cuatro algoritmos descritos a lo largo del proyecto; es decir el ITGO, PSO, DE y JDE.

Para obtener los datos de los distintos algoritmos se han realizado 10 ejecuciones de cada una de las funciones seleccionadas para cada uno de los algoritmos descritos, obteniendo así un conjunto de valores más robusto sobre el que poder realizar el estudio comparativo entre el rendimiento de estos contra el del ITGO.

Dado que las iteraciones tienen partes que se generan aleatoriamente, como puede ser el valor de las células de la población inicial, para evitar problemas y posibles errores se ha optado por usar una semilla fija con valor "123". Además todas las ejecuciones se han realizado con los mismos parámetros donde el valor de la dimensión de cada uno de los miembros de la población es de 30, el valor de la semilla es el fijado previamente y el número de iteraciones máximo es de 10000.

ITGO	PSO	DE	JDE	
2,838290E+04	9,954980E+03	8,955662E-09	9,206839E-09	
2,836440E+04	8,849770E+03	9,943558E-09	9,976424E-09	
2,836320E+04	7,727638E+03	8,386408E-09	8,596171E-09	
2,836640E+04	1,042791E+04	9,690594E-09	9,696545E-09	
2,836570E+04	1,238782E+04	8,874667E-09	9,834295E-09	
2,837030E+04	7,459527E+03	9,692715E-09	8,996690E-09	
2,836260E+04	1,574122E+04	9,859868E-09	9,934978E-09	
2,836570E+04	2,098579E+04	9,668987E-09	9,564698E-09	
2,836530E+04	1,351496E+04	9,349877E-09	8,320999E-09	
2,838720E+04	1,452910E+04	8,855241E-09	7,931004E-09	
2,836937E+04	1,215787E+04	9,327758E-09	9,205864E-09	MEDIA
2,838720E+04	2,098579E+04	9,943558E-09	9,976424E-09	MAX
2,836260E+04	7,459527E+03	8,386408E-09	7,931004E-09	MIN

Figura 5.1: Datos función 1

ITGO	PSO	DE	JDE	
7,004780E+04	4,211415E+04	7,672989E-09	5,146711E+01	
6,987590E+04	3,573151E+04	9,142424E-09	4,301055E+01	
7,001010E+04	3,754771E+04	9,628062E-09	3,473162E+01	
6,987590E+04	2,259711E+04	9,684856E-09	6,298393E+01	
6,997410E+04	2,024094E+04	9,035915E-09	7,825774E+01	
6,987590E+04	1,679113E+04	9,825205E-09	7,282291E+01	
6,987710E+04	2,405885E+04	9,465038E-09	5,540725E+01	
7,018320E+04	1,829554E+04	7,086396E-09	9,994828E+01	
7,014370E+04	2,571403E+04	9,642911E-09	5,056298E+01	
6,991090E+04	1,892424E+04	9,710212E-09	7,123419E+01	
6,997746E+04	2,620152E+04	9,089401E-09	6,204266E+01	MEDIA
7,018320E+04	4,211415E+04	9,825205E-09	9,994828E+01	MAX
6,987590E+04	1,679113E+04	7,086396E-09	3,473162E+01	MIN

Figura 5.2: Datos función 2

ITGO	PSO	DE	JDE	
1,694820E+09	2,727693E+07	1,309173E+06	2,830033E+07	
1,694820E+09	2,808955E+07	2,547193E+05	2,763664E+07	
1,694820E+09	9,668380E+07	9,855719E+05	2,469460E+07	
1,694820E+09	1,134022E+08	9,451400E+05	3,199433E+07	
1,694730E+09	1,598041E+08	1,097551E+06	1,169298E+07	
1,694730E+09	3,745635E+07	2,015565E+06	2,098515E+07	
1,694820E+09	3,947589E+07	1,295625E+06	2,703587E+07	
1,694820E+09	1,112962E+08	1,369033E+06	2,528370E+07	
1,694820E+09	3,795773E+07	8,588552E+05	2,204374E+07	
1,694820E+09	4,225503E+07	1,050096E+06	2,287343E+07	
1,694802E+09	6,936978E+07	1,118133E+06	2,425408E+07	**MEDIA**
1,694820E+09	1,598041E+08	2,015565E+06	3,199433E+07	**MAX**
1,694730E+09	2,727693E+07	2,547193E+05	1,169298E+07	**MIN**

Figura 5.3: Datos función 3

ITGO	PSO	DE	JDE	
8,255620E+04	4,390888E+04	8,613662E-04	1,637335E+03	
7,300560E+04	4,914043E+04	1,378041E-03	1,942909E+03	
8,887330E+04	4,201248E+04	7,268743E-03	1,661241E+03	
8,702780E+04	3,982782E+04	1,054424E-03	2,188129E+03	
8,674600E+04	4,175580E+04	8,026533E-04	3,363614E+03	
7,427250E+04	4,620068E+04	2,236115E-03	1,483039E+03	
7,751690E+04	4,398725E+04	6,376364E-04	2,349771E+03	
7,751690E+04	3,580489E+04	3,097907E-03	1,943218E+03	
7,047410E+04	2,451785E+04	1,245805E-03	1,775259E+03	
8,258180E+04	5,118153E+04	1,468327E-03	1,859609E+03	
8,005711E+04	4,183376E+04	2,005102E-03	2,020412E+03	**MEDIA**
8,887330E+04	5,118153E+04	7,268743E-03	3,363614E+03	**MAX**
7,047410E+04	2,451785E+04	6,376364E-04	1,483039E+03	**MIN**

Figura 5.4: Datos función 4

ITGO	PSO	DE	JDE	
2,660700E+04	1,229525E+04	1,061053E+01	4,162665E+03	
2,665150E+04	1,978596E+04	5,114526E+00	5,040192E+03	
2,667960E+04	1,878932E+04	1,148199E+01	4,857912E+03	
2,662760E+04	1,963263E+04	2,247957E+01	5,135505E+03	
2,664830E+04	2,003956E+04	1,601053E+02	3,670682E+03	
2,665670E+04	1,574137E+04	9,742686E+00	4,192352E+03	
2,662240E+04	1,599770E+04	1,316090E+00	3,673586E+03	
2,662240E+04	1,466884E+04	1,348472E+01	4,851620E+03	
2,669540E+04	1,300859E+04	6,235260E+00	4,825186E+03	
2,666710E+04	1,347670E+04	4,787305E+00	5,179194E+03	
2,664780E+04	1,634359E+04	2,453580E+01	4,558889E+03	MEDIA
2,669540E+04	2,003956E+04	1,601053E+02	5,179194E+03	MAX
2,660700E+04	1,229525E+04	1,316090E+00	3,670682E+03	MIN

Figura 5.5: Datos función 5

ITGO	PSO	DE	JDE	
1,449940E+10	1,220133E+09	9,490000E-09	9,206839E-09	
1,440970E+10	3,236070E+08	9,392375E-09	9,976424E-09	
1,449830E+10	9,116415E+08	1,028624E-03	8,596171E-09	
1,442440E+10	3,498692E+09	9,835446E-09	9,696545E-09	
1,448120E+10	3,410752E+08	8,435636E-09	9,834295E-09	
1,448680E+10	1,951511E+09	3,429255E-07	8,996690E-09	
1,447820E+10	1,547012E+09	9,992012E-09	9,934978E-09	
1,450460E+10	3,516353E+09	2,267122E-07	9,564698E-09	
1,440770E+10	1,625588E+09	9,895364E-09	8,320999E-09	
1,448010E+10	3,183307E+08	3,341988E-08	7,931004E-09	
1,446704E+10	1,525394E+09	1,029284E-04	9,205864E-09	MEDIA
1,450460E+10	3,516353E+09	1,028624E-03	9,976424E-09	MAX
1,440770E+10	3,183307E+08	8,435636E-09	7,931004E-09	MIN

Figura 5.6: Datos función 6

ITGO	PSO	DE	JDE	
1,269560E+03	3,385678E+02	7,396041E-03	3,065454E-01	
1,269560E+03	2,797399E+02	9,722736E-09	3,284005E-01	
1,269540E+03	1,583540E+02	7,396040E-03	1,947826E-01	
1,269560E+03	7,818842E+02	9,217230E-09	2,451631E-01	
1,269570E+03	2,253626E+02	9,857285E-03	2,421918E-01	
1,269540E+03	3,071345E+02	8,203468E-09	3,560794E-01	
1,269560E+03	2,133389E+02	9,679724E-09	2,119192E-01	
1,269560E+03	2,158740E+02	9,794860E-09	2,432079E-01	
1,269620E+03	1,325810E+02	9,984659E-09	2,528023E-01	
1,269560E+03	3,286331E+02	9,519393E-09	2,041570E-01	
1,269563E+03	2,981470E+02	2,464943E-03	2,585249E-01	**MEDIA**
1,269620E+03	7,818842E+02	9,857285E-03	3,560794E-01	**MAX**
1,269540E+03	1,325810E+02	8,203468E-09	1,947826E-01	**MIN**

Figura 5.7: Datos función 7

ITGO	PSO	DE	JDE	
2,128040E+05	2,093833E+01	2,093823E+01	2,096972E+01	
2,128660E+05	2,102857E+01	2,099490E+01	2,093202E+01	
2,136970E+05	2,094774E+01	2,093038E+01	2,098306E+01	
2,159700E+04	2,098353E+01	2,099896E+01	2,086495E+01	
2,155190E+05	2,099713E+01	2,096940E+01	2,097001E+01	
2,139200E+04	2,091593E+01	2,090840E+01	2,097438E+01	
2,139870E+05	2,103909E+01	2,088900E+01	2,086304E+01	
2,134370E+05	2,101883E+01	2,091032E+01	2,096540E+01	
2,153660E+05	2,091095E+01	2,082811E+01	2,091137E+01	
2,139200E+04	2,104955E+01	2,103409E+01	2,095371E+01	
1,562057E+05	2,098297E+01	2,094018E+01	2,093877E+01	**MEDIA**
2,155190E+05	2,104955E+01	2,103409E+01	2,098306E+01	**MAX**
2,139200E+04	2,091095E+01	2,082811E+01	2,086304E+01	**MIN**

Figura 5.8: Datos función 8

ITGO	PSO	DE	JDE	
1,575650E+05	2,465798E+02	9,657290E-09	8,346021E-09	
1,549660E+05	2,498955E+02	9,883606E-09	9,672871E-09	
1,562270E+05	3,021310E+02	9,587986E-09	9,545725E-09	
1,556720E+05	2,165274E+02	9,482374E-09	9,582177E-09	
1,574180E+05	2,721853E+02	8,955099E-09	9,991097E-09	
1,560990E+05	2,084037E+02	8,539381E-09	8,041425E-09	
1,548570E+05	2,408069E+02	9,019890E-09	9,116581E-09	
1,560770E+05	2,790528E+02	9,675680E-09	8,948336E-09	
1,560770E+05	2,829594E+02	9,871999E-09	8,942645E-09	
1,574420E+05	2,365165E+02	8,968755E-09	9,980749E-09	
1,562400E+05	2,535058E+02	9,364206E-09	9,216763E-09	MEDIA
1,575650E+05	3,021310E+02	9,883606E-09	9,991097E-09	MAX
1,548570E+05	2,084037E+02	8,539381E-09	8,041425E-09	MIN

Figura 5.9: Datos función 9

ITGO	PSO	DE	JDE	
2,383950E+05	3,904491E+02	1,072097E+02	1,308627E+02	
2,261150E+05	4,497039E+02	1,131037E+02	1,369896E+02	
2,147170E+05	3,186594E+02	1,228239E+02	9,917540E+01	
2,295590E+05	3,078369E+02	1,218728E+02	1,414468E+02	
2,389220E+05	2,202633E+02	1,207710E+02	1,414321E+02	
2,160030E+05	2,863546E+02	1,275426E+02	1,343292E+02	
2,483180E+05	2,633993E+02	1,105805E+02	1,352326E+02	
2,246630E+05	2,984171E+02	1,326319E+02	1,117168E+02	
2,449570E+05	3,046675E+02	1,265244E+02	1,447971E+02	
2,323240E+05	3,244653E+02	1,199251E+02	1,029219E+02	
2,313973E+05	3,164216E+02	1,202986E+02	1,278904E+02	MEDIA
2,483180E+05	4,497039E+02	1,326319E+02	1,447971E+02	MAX
2,147170E+05	2,202633E+02	1,072097E+02	9,917540E+01	MIN

Figura 5.10: Datos función 10

ITGO	PSO	DE	JDE	
1,449310E+05	3,823050E+01	3,091527E+01	2,839582E+01	
1,386210E+05	4,123425E+01	3,270881E+01	3,047521E+01	
1,528640E+05	4,040959E+01	3,149742E+01	2,747074E+01	
1,497630E+05	3,629839E+01	3,059170E+01	2,973581E+01	
1,506180E+05	2,753710E+01	3,165458E+01	3,039950E+01	
1,528680E+05	3,790321E+01	2,937916E+01	2,770863E+01	
1,372910E+05	3,673759E+01	2,888466E+01	2,881659E+01	
1,413420E+05	4,125745E+01	3,014323E+01	2,835101E+01	
1,422530E+05	3,601026E+01	3,170608E+01	3,108160E+01	
1,477940E+05	3,765327E+01	3,094409E+01	3,059688E+01	
1,458345E+05	3,732716E+01	3,084250E+01	2,930318E+01	**MEDIA**
1,528680E+05	4,125745E+01	3,270881E+01	3,108160E+01	**MAX**
1,372910E+05	2,753710E+01	2,888466E+01	2,747074E+01	**MIN**

Figura 5.11: Datos función 11

ITGO	PSO	DE	JDE	
1,022650E+03	1,189032E+02	2,766500E+00	2,028202E+00	
9,168000E+02	1,985603E+02	2,444907E+00	1,874087E+00	
1,351450E+03	1,071289E+02	2,532238E+00	2,056660E+00	
8,592350E+03	7,018293E+01	2,750287E+00	2,038253E+00	
8,584720E+02	1,338071E+02	2,616472E+00	1,711007E+00	
9,032030E+02	1,568779E+02	2,287157E+00	1,689691E+00	
9,030590E+02	5,014068E+01	2,701314E+00	2,099883E+00	
8,584720E+02	2,677078E+02	2,229129E+00	1,739459E+00	
1,294970E+03	2,633407E+02	2,653012E+00	2,148520E+00	
1,013610E+03	1,016112E+02	2,387269E+00	1,987304E+00	
1,771504E+03	1,468261E+02	2,536829E+00	1,937307E+00	**MEDIA**
8,592350E+03	2,677078E+02	2,766500E+00	2,148520E+00	**MAX**
8,584720E+02	5,014068E+01	2,229129E+00	1,689691E+00	**MIN**

Figura 5.12: Datos función 13

ITGO	PSO	DE	JDE	
4,915660E+01	1,357875E+01	1,314006E+01	1,327360E+01	
4,935910E+01	1,217802E+01	1,294640E+01	1,330644E+01	
5,002740E+01	1,362377E+01	1,271675E+01	1,318461E+01	
4,916180E+01	1,384995E+01	1,340086E+01	1,270529E+01	
4,917960E+01	1,331131E+01	1,307265E+01	1,327905E+01	
4,918600E+01	1,363243E+01	1,326868E+01	1,276989E+01	
4,916610E+01	1,331346E+01	1,325796E+01	1,283534E+01	
4,917710E+01	1,339152E+01	1,313402E+01	1,298547E+01	
4,929540E+01	1,384501E+01	1,318036E+01	1,308234E+01	
4,929540E+00	1,326192E+01	1,331885E+01	1,319001E+01	
4,486386E+01	1,339861E+01	1,314366E+01	1,306120E+01	MEDIA
5,002740E+01	1,384995E+01	1,340086E+01	1,330644E+01	MAX
4,929540E+00	1,217802E+01	1,271675E+01	1,270529E+01	MIN

Figura 5.13: Datos función 14

ITGO	PSO	DE	JDE	
1,417780E+03	8,609852E+02	4,000000E+02	3,000269E+02	
1,457550E+03	6,653335E+02	2,000000E+02	2,000000E+02	
1,406800E+03	6,480292E+02	4,000000E+02	2,000000E+02	
1,540230E+03	7,452112E+02	4,000000E+02	1,928687E+01	
1,368350E+03	6,575142E+02	2,000000E+02	4,802428E+01	
1,425840E+03	7,734949E+02	2,000000E+02	1,228287E+01	
1,434000E+03	7,114655E+02	4,000000E+02	9,823784E+01	
1,385490E+03	5,614219E+02	4,000000E+02	6,462884E+00	
1,519230E+03	5,984574E+02	2,000000E+02	6,315211E+01	
1,466710E+03	8,169062E+02	4,000000E+02	3,985297E+01	
1,442198E+03	7,038819E+02	3,200000E+02	9,873267E+01	MEDIA
1,540230E+03	8,609852E+02	4,000000E+02	3,000269E+02	MAX
1,368350E+03	5,614219E+02	2,000000E+02	6,462884E+00	MIN

Figura 5.14: Datos función 15

A partir de la información obtenida, vamos a proceder a estudiar los resultados de cada una de las funciones usadas para calcular el fitness del algoritmo y su margen de error, siendo este último el valor que se ve reflejado en todas las tablas.

Al estudiar las tablas de forma individual para cada una de las 14 funciones utilizadas, podemos ver un patrón en casi todas ellas, a excepción de algunas específicas, donde se cumple que, los mejores valores los obtienen las funciones DE y JDE seguido de PSO y por último ITGO.

Sin embargo, y debido a que estos datos son algo difíciles de leer para su comparación dada la cantidad de tablas, a continuación se ha realizado una tabla resumen donde se encuentran las medias calculadas de las 14 tablas previas con la finalidad de realizar un mejor estudio de los resultados obtenidos de los cuatro algoritmos con los que se ha trabajado.

	ITGO	PSO	DE	JDE
Funcion 1	2,836937E+04	1,215787E+04	9,327758E-09	9,205864E-09
Funcion 2	6,997746E+04	2,620152E+04	9,089401E-09	6,204266E+01
Funcion 3	1,694802E+09	6,936978E+07	1,118133E+06	2,425408E+07
Funcion 4	8,005711E+04	4,183376E+04	2,005102E-03	2,020412E+03
Funcion 5	2,664780E+04	1,634359E+04	2,453580E+01	4,558889E+03
Funcion 6	1,446704E+10	1,525394E+09	1,029284E-04	9,205864E-09
Funcion 7	1,269563E+03	2,981470E+02	2,464943E-03	2,585249E-01
Funcion 8	1,562057E+05	2,098297E+01	2,094018E+01	2,093877E+01
Funcion 9	1,562400E+05	2,535058E+02	9,364206E-09	9,216763E-09
Funcion 10	2,313973E+05	3,164216E+02	1,202986E+02	1,278904E+02
Funcion 11	1,458345E+05	3,732716E+01	3,084250E+01	2,930318E+01
Funcion 13	1,771504E+03	1,468261E+02	2,536829E+00	1,937307E+00
Funcion 14	4,486386E+01	1,339861E+01	1,314366E+01	1,306120E+01
Funcion 15	1,442198E+03	7,038819E+02	3,200000E+02	9,873267E+01
MEDIA	1,154482E+09	1,139187E+08	7,990466E+04	1,732929E+06
MAX	1,446704E+10	1,525394E+09	1,118133E+06	2,425408E+07
MIN	4,486386E+01	1,339861E+01	9,089401E-09	9,205864E-09

Figura 5.15: Tabla con las medias

Con la finalidad de realizar un estudio completo de los datos obtenidos, se ha decido realizar dicho estudio en varios niveles de profundidad, de forma que el un primer lugar se hablará de los datos de forma global, tras esto se bajará a un nivel de profundidad en el que la prioridad será comparar ITGO con PSO y finalmente se accederá a un nuevo nivel en el que se verá como actúa ITGO para las diferentes funciones utilizadas.

5.1. Comparativa global

Comparando los datos de ITGO con los de los otros tres algoritmos se puede observar que los mejores valores obtenidos, son $9,327758e-09$ y $9,976424e-09$ para el algoritmo DE y JDE respectivamente mientras que

para los otros dos casos se obtienen valores del tipo $5,002740e+01$ en el caso del ITGO o $1,384995e+01$ en el caso del PSO.

Aunque la diferencia entre los mejores casos es algo distante entre los algoritmos evolutivos y los basados en nubes de partículas, no lo es tanto cuando se comparan los peores valores donde obtenemos que, para los algoritmos DE y JDE los peores caso son $2,015565e+06$ y $3,199433e+07$, para PSO el peor caso es $3,516353e+09$ y para ITGO el peor caso es $1,450460e+10$.

Si nos centramos en los resultados más allá de los mejores y los peores valores, se puede comprobar a simple vista que, los algoritmos que obtienen un menor margen de error es decir, una mejor solución, para todas las funciones son los dos algoritmos de tipo evolutivo: DE y JDE. Además entre ellos se puede ver que obtienen resultados muy similares a excepción de los casos 2 y 7.

En el caso de la función 7 el óptimo está fuera del rango establecido, por lo que es normal que los márgenes de error difieran entre estos dos algoritmos dado a que uno (JDE) es una versión mejorada del otro (DE), de manera que este sí es capaz de quedarse en un rango de soluciones más cercano al óptimo que DE, mientras que PSO e ITGO van bien encaminados pero, en el caso de ITGO se estanca en un posible óptimo local y PSO da con varios resultados alrededor de $2,98147e+02$ aproximándose al óptimo pero sin llegar a obtener tan buenos resultados como DE o JDE.

Por otra parte, viendo el marco general podemos ver que, al igual que los algoritmos DE y JDE obtienen valores similares, ocurre algo parecido con los otros dos, ITGO y PSO. Ambos tienen un margen de error similar aunque el ITGO obtiene resultados algo más distantes del óptimo. Sin embargo, podemos ver que la diferencia entre dichos algoritmos no es tan abismal a excepción de casos como son los obtenidos en la función 3, 8, 9, 10 y 11.

Finalmente, si se realiza un último estudio de los resultados, se puede observar cómo para casi todos los casos los valores obtenidos por el ITGO oscilan alrededor de uno, por lo que se deduce que el algoritmo se topa con un óptimo local, mientras que para los demás casos (PSO, DE, JDE) el rango de soluciones es algo más amplio de forma que se puede decir que diversifica mejor. Esto da lugar a que se evite caer en óptimos locales y haya más probabilidad de llegar al óptimo global. Aunque no se llegue a alcanzar en ninguno de los casos.

A continuación se procederá a reflejar los datos obtenidos en una gráfica para poder observar los resultados de una manera más visual.

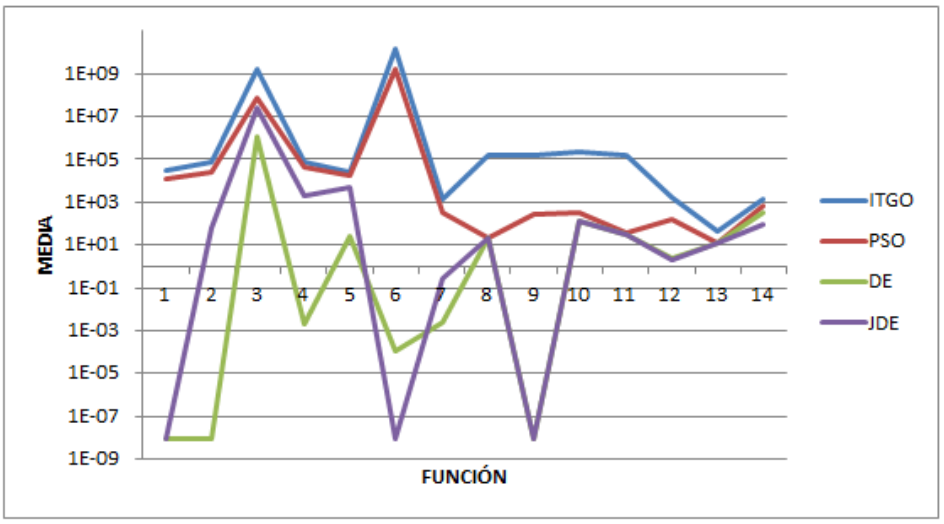

Figura 5.16: Gráfica de los valores Medios

A partir de esta gráfica se observa cómo la información obtenida a partir de las tablas no dista mucho de lo que se vé.

- El primer punto, y el que más destaca, es la diferencia de resultados entre DE y JDE contra PSO e ITGO como se ha comentado previamente, a excepción de la función 3 donde los datos son más similares. Como vemos los dos primeros tienen valores más parecidos mientras que con los dos últimos ocurre lo mismo, aunque hayan funciones en las que estos valores sean más distantes.

- También se hace más notorio el hecho de que, aunque los evolutivos obtengan mejores resultados para todas las funciones, el margen de error varía mucho más que en los otros dos casos.

    - En el caso de DE contra JDE se observa que según la función con la que se ha evaluado, se han obtenido unos valores que varían entre $9,089401e-09$ y $1,118133e+06$ de manera que se ve cómo para las funciones 1,2,4,6,7,9, en el caso de DE, y para las funciones 1,6,9 en el caso de JDE estos algoritmos son capaces de llegar a zonas cercanas al óptimo global. Pero en el resto de casos quedan bastante lejos de este valor, aunque no tanto como ocurre con PSO e ITGO.
    - Por otro lado en el caso de ITGO contra PSO se observa que los valores obtenidos son, como se ha comentado previamente, mucho más similares entre ellos y el espectro del rango de soluciones es mucho menor de lo que en un principio aparentaba. Sin embargo

se aprecia una menor oscilación entre los valores obtenidos para ITGO para estas funciones en contraposición a las anteriores, manteniéndose cerca de un mismo valor para varias funciones seguidas.

5.2. Comparativa ITGO con PSO

A continuación se va a ahondar en la comparativa entre los algoritmos PSO e ITGO para los datos obtenidos. Dado que estos son los dos algoritmos principales a comparar ya que ambos se apoyan en el mismo tipo de metaheurística basada en el desplazamientos de partículas.

Como se puede ver, aunque la diferencia con los otros dos algoritmos es relativamente grande, entre ellos se obtienen valores muchos más similares, aunque en el caso de ITGO se obtienen valores algo peores.

Si nos apoyamos en la gráfica 5.16, se puede ver que hasta la función 7 ambos algoritmos son más o menos similares, o cercanos para los valores obtenidos, aunque haya diferencias entre ellos dado que ITGO queda por encima en todos los casos.
Sin embargo a partir de esta función los resultados obtenidos para PSO e ITGO mejoran en relación a los casos anteriores, aunque entre ellos vemos que el rango de soluciones se separa quedando los resultados de PSO mucho más cerca del óptimo que ITGO.

Esta primera mejora de los resultados para las funciones multimodales se debe al funcionamiento de ambos algoritmos. Como para las funciones unimodales el óptimo local es el mismo que el global y ninguno de los dos algoritmos lo contempla, ambos realizan saltos a otras zonas con la finalidad de salir del óptimo local tras llevar varios intentos sin mejorar siendo esta la principal razón por la que salen valores tan malos. Sin embargo para las funciones multimodales al haber varios óptimos locales además del óptimo global, estas funciones reaccionan mejor dado que según se van desplazando por el espacio de búsqueda y van explorándolo se encuentran con óptimos locales que van evitando, consiguiendo así aplicar una buena exploración tanteando diversas posibles soluciones en lugar de centrarse en alejarse del óptimo local/global como ocurre para el otro tipo de funciones.

Sin embargo, se puede apreciar que, aunque se da una mejora en ambos algoritmos, entre ellos PSO es capaz de afinar mejor a la hora de aproximarse al óptimo local mientras que ITGO logra una mejora con respecto a las funciones anteriores aunque no tan notoria.

Este comportamiento puede darse debido a que al alejarse de un óptimo

local no quede del todo fuera del rango de influencia de este y no sea capaz de abandonar una zona de nutrientes para explorar una nueva, si no que se ve atraído, aunque sea levemente, por la anterior.

5.3. Estudio de los datos para ITGO

En una primera instancia, se puede ver que los valores que se han obtenido tras las ejecuciones para el ITGO, tienen un margen de error alto si se compara con los demás, siendo el máximo de $1,449940e+10$, en el caso de la función 6, mientras que para otras funciones se obtiene un margen bastante menor, siendo el mínimo $4,929540e+00$, para la función 14 de manera que se aproxima a valores de funciones más típicos de los algoritmos evolutivos. Por lo tanto se ve que el rango de soluciones de este algoritmo es bastante más amplio que el algoritmo PSO pero no tanto como los algoritmos DE y JDE como se ha comentado previamente.

Con la finalidad de ver de manera más visual estos datos, se ha realizado una gráfica con los valores medios obtenidos para cada función omitiendo las funciones 3 y 6 que, dado su alto margen de error resultaban irrelevantes para la gráfica ya que la deformaban innecesariamente, de manera que nos queda de la siguiente forma:

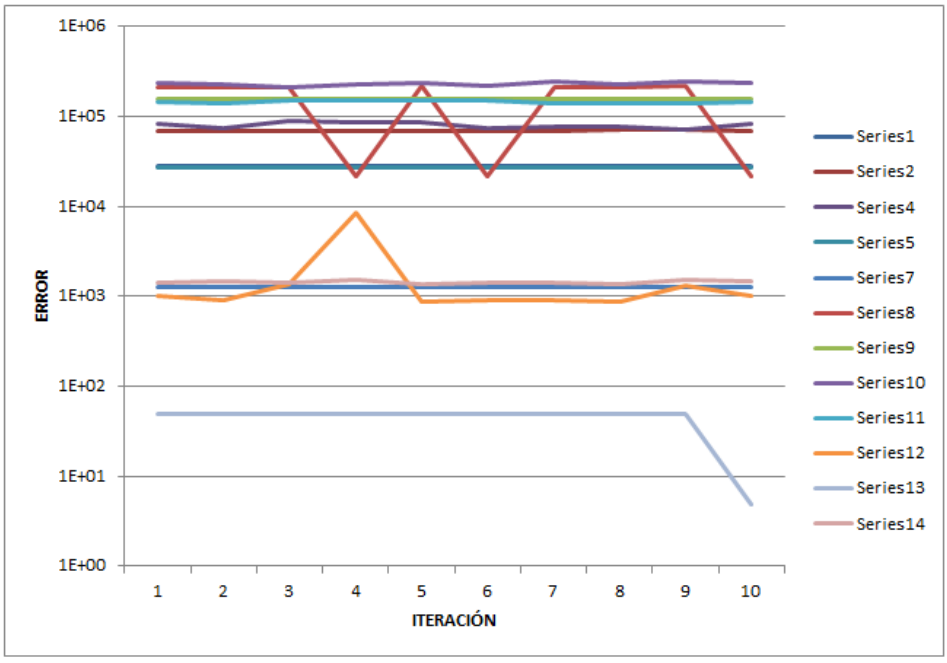

Figura 5.17: Valores medios de ITGO

Se puede observar que para la función 14 (serie 13) es para la que se obtienen el menor margen de error mientras que para la función 10 (serie 10) se obtiene el mayor margen de error, o peor resultado, además de para las funciones 3 y 6. Fijándonos en una zona más céntrica, las funciones 7, 13 y 15 (indicadas estas dos últimas en la gráfica como series 12 y 14), son las que obtienen un menor margen de error y están más próximas al valor medio obtenido.

Por otro lado si se obvian dichas funciones, (7, 10, 13, 14, 15, 3 y 6) y nos centramos en los demás casos (funciones 1, 2, 4, 5, 8, 9 y 11) se aprecia que el margen de error es, más o menos, el mismo oscilando entre valores tales como $2,1392e+04$ y $1,5756e+05$.

A partir de la gráfica detectamos dos comportamientos distintos del algoritmo:

- Un primer comportamiento en el que los valores obtenidos son valores más dispares; como ocurre para las funciones 2, 13 (serie 12) y 14 (serie 13).

- Un segundo comportamiento en el que los valores obtenidos son lineales; como ocurre con las demás funciones independientemente del rango en el que se encuentren.

Este comportamiento puede deberse a varios motivos:

- El comportamiento de las funciones 13 y 14 puede deberse a que, dado que se trata de funciones multimodales, den con varios valores de óptimos locales distintos cada vez dando lugar a que se den resultados tan disparejos ya que el algoritmo intenta salir de él y termina mientras está explorando una nueva zona del entorno.

  Sin embargo en el caso de la función 2, al ser esta de tipo unimodal, sólo hay un óptimo tanto local como global por lo que es posible que el algoritmo, por como está implementado, lo considere un óptimo local y dé un salto tras no notar mejora a una zona nueva dando lugar a los picos observados en la gráfica.

- Para el problema de las soluciones lineales puede ocurrir que, en el caso de que se trate de funciones multimodales, se encuentre con un óptimo local y no consiga dar un salto lo suficientemente grande como para alejarse de él y buscar una mejor opción.

  Mientras que en el caso de las funciones unimodales, o bien encuentre, al igual que con las multimodales, un valor que sea un óptimo local, en cuyo caso sabemos que es inventado dado a que para estas funciones el óptimo local y global son el mismo; o que no sea capaz de llegar al óptimo global en el número de iteraciones establecido y se quede siempre en el mismo punto a la hora de explorar su entorno.

En general se puede afirmar que, para este caso, se obtienen mejores resultados con las funciones híbridas y las multimodales que con las unimodales, donde se obtienen los peores resultados por lo explicado previamente de cómo son las funciones y cómo puede reaccionar el ITGO ante ellas, es decir óptimos tanto locales como globales.

5.4. Segundo conjunto de datos

Tras realizar un primer estudio de los datos obtenidos, se ha procedido a obtener un segundo conjunto donde se estudiará lo que ocurre al trabajar con una dimensión de 50 en vez de 30. Para este caso los valores para la ejecución son igual que antes, donde la semilla vale "123", el número de iteraciones máximo es de 10000 pero en este caso, el valor de la dimensión de cada uno de los miembros de la población es de 50 en lugar de 30.

Sin embargo en este caso no realizaremos el estudio para los cuatro algoritmos y las 14 funciones; si no que nos centraremos en estudiar los datos para los algoritmos ITGO y PSO y para las funciones 1, 2, 4, 7, 9 y 14.

Esto se debe a que resultaba un estudio más interesante entre PSO e ITGO dado a su cercanía entre los valores obtenidos que con respecto a los evolutivos que, como se ha visto, son los que obtienen los mejores resultados.

También se ha realizado un filtro de las funciones estudiadas debido que se quiere ver cómo reaccionan los algoritmos al tener una mayor dimensión con la que trabajar, por lo tanto los saltos en el espacio no son tan bruscos.

5.4.1. Estudio de las funciones para una dimensión de 50

En primer lugar se obtendrán los datos nuevos para estas funciones de los dos algoritmos quedando como se muestran a continuación:

ITGO	ITGO (50)	PSO	PSO(50)	
2,838290E+04	1,479360E+05	9,954980E+03	4,692964E+04	
2,836440E+04	1,480610E+05	8,849770E+03	2,993273E+04	
2,836320E+04	1,481090E+05	7,727638E+03	4,667556E+04	
2,836640E+04	1,478980E+05	1,042791E+04	4,566235E+04	
2,836570E+04	1,480800E+05	1,238782E+04	3,109409E+04	
2,837030E+04	1,478780E+05	7,459527E+03	6,616980E+04	
2,836260E+04	1,478930E+05	1,574122E+04	3,903007E+04	
2,836570E+04	1,479980E+05	2,098579E+04	5,858135E+04	
2,836530E+04	1,482150E+05	1,351496E+04	3,681196E+04	
2,838720E+04	1,480850E+05	1,452910E+04	3,663804E+04	
2,836937E+04	1,480153E+05	1,215787E+04	4,375256E+04	MEDIA
2,838720E+04	1,482150E+05	2,098579E+04	6,616980E+04	MAX
2,836260E+04	1,478780E+05	7,459527E+03	2,993273E+04	MIN

(a) Función 1

ITGO	ITGO (50)	PSO	PSO(50)	
7,004780E+04	6,576150E+06	4,211415E+04	7,733159E+04	
6,987590E+04	6,762820E+06	3,573151E+04	9,853114E+04	
7,001010E+04	6,415360E+06	3,754771E+04	7,667620E+04	
6,987590E+04	6,385680E+06	2,259711E+04	7,296670E+04	
6,997410E+04	6,486170E+06	2,024094E+04	6,462096E+04	
6,987590E+04	6,323330E+06	1,679113E+04	8,338785E+04	
6,987710E+04	6,415220E+06	2,405885E+04	6,754139E+04	
7,018320E+04	6,292020E+06	1,829554E+04	7,118025E+04	
7,014370E+04	6,942430E+06	2,571403E+04	8,116912E+04	
6,991090E+04	6,750610E+06	1,892424E+04	9,182593E+04	
6,997746E+04	6,534979E+06	2,620152E+04	7,852311E+04	MEDIA
7,018320E+04	6,942430E+06	4,211415E+04	9,853114E+04	MAX
6,987590E+04	6,292020E+06	1,679113E+04	6,462096E+04	MIN

(b) Función 2

ITGO	ITGO (50)	PSO	PSO(50)	
8,255620E+04	7,261230E+06	4,390888E+04	9,452920E+04	
7,300560E+04	9,334320E+06	4,914043E+04	1,281086E+05	
8,887330E+04	6,495920E+06	4,201248E+04	1,327084E+05	
8,702780E+04	7,046140E+06	3,982782E+04	1,103525E+05	
8,674600E+04	7,493200E+06	4,175580E+04	1,175725E+05	
7,427250E+04	6,668800E+06	4,620068E+04	1,052824E+05	
7,751690E+04	7,027740E+06	4,398725E+04	1,155130E+05	
7,751690E+04	6,759330E+06	3,580489E+04	8,429687E+04	
7,047410E+04	6,770760E+06	2,451785E+04	1,248818E+05	
8,258180E+04	7,939900E+06	5,118153E+04	7,290258E+04	
8,005711E+04	7,279734E+06	4,183376E+04	1,086148E+05	MEDIA
8,887330E+04	9,334320E+06	5,118153E+04	1,327084E+05	MAX
7,047410E+04	6,495920E+06	2,451785E+04	7,290258E+04	MIN

(c) Función 4

Figura 5.18: Funciones unimodales con dimension 50.

ITGO	ITGO (50)	PSO	PSO(50)	
1,269560E+03	1,604073E+03	3,385678E+02	9,765728E+02	
1,269560E+03	1,604133E+03	2,797399E+02	1,172464E+03	
1,269540E+03	1,604073E+03	1,583540E+02	8,762348E+02	
1,269560E+03	1,604073E+03	7,818842E+02	5,033490E+02	
1,269570E+03	1,604073E+03	2,253626E+02	6,061633E+02	
1,269540E+03	1,604053E+03	3,071345E+02	6,416474E+02	
1,269560E+03	1,604073E+03	2,133389E+02	1,064769E+03	
1,269560E+03	1,604083E+03	2,158740E+02	1,018654E+03	
1,269620E+03	1,604053E+03	1,325810E+02	7,288517E+02	
1,269560E+03	1,604073E+03	3,286331E+02	1,015793E+03	
1,269563E+03	1,604076E+03	2,981470E+02	8,604499E+02	**MEDIA**
1,269620E+03	1,604133E+03	7,818842E+02	1,172464E+03	**MAX**
1,269540E+03	1,604053E+03	1,325810E+02	5,033490E+02	**MIN**

(a) Función 7

ITGO	ITGO (50)	PSO	PSO(50)	
1,575650E+05	1,057070E+03	2,465798E+02	5,404150E+02	
1,549660E+05	1,082430E+03	2,498955E+02	5,337159E+02	
1,562270E+05	1,061170E+03	3,021310E+02	4,821570E+02	
1,556720E+05	1,025330E+03	2,165274E+02	4,733413E+02	
1,574180E+05	9,979940E+02	2,721853E+02	4,861030E+02	
1,560990E+05	1,057103E+03	2,084037E+02	5,783252E+02	
1,548570E+05	1,082463E+03	2,408069E+02	5,191098E+02	
1,560770E+05	1,061203E+03	2,790528E+02	5,214388E+02	
1,560770E+05	1,025363E+03	2,829594E+02	6,025601E+02	
1,574420E+05	1,057135E+03	2,365165E+02	4,990625E+02	
1,562400E+05	1,050726E+03	2,535058E+02	5,236229E+02	**MEDIA**
1,575650E+05	1,082463E+03	3,021310E+02	6,025601E+02	**MAX**
1,548570E+05	9,979940E+02	2,084037E+02	4,733413E+02	**MIN**

(b) Función 9

ITGO	ITGO (50)	PSO	PSO(50)	
4,915660E+01	3,876851E+01	1,357875E+01	2,328119E+01	
4,935910E+01	3,878631E+01	1,217802E+01	2,300276E+01	
5,002740E+01	3,879271E+01	1,362377E+01	2,357722E+01	
4,916180E+01	3,877281E+01	1,384995E+01	2,301311E+01	
4,917960E+01	3,878381E+01	1,331131E+01	2,342077E+01	
4,918600E+01	3,890211E+01	1,363243E+01	2,331143E+01	
4,916610E+01	3,896581E+01	1,331346E+01	2,370675E+01	
4,917710E+01	3,876331E+01	1,339152E+01	2,331632E+01	
4,929540E+01	3,896581E+01	1,384501E+01	2,357152E+01	
4,929540E+00	3,963411E+01	1,326192E+01	2,320031E+01	
4,486386E+01	3,891353E+01	1,339861E+01	2,334014E+01	**MEDIA**
5,002740E+01	3,963411E+01	1,384995E+01	2,370675E+01	**MAX**
4,929540E+00	3,876331E+01	1,217802E+01	2,300276E+01	**MIN**

(c) Función 14

Figura 5.19: Funciones multimodales con dimension 50.

Estudiando la tabla 5.18 que contiene los datos de las funciones unimodales, vemos que para este tipo de metaheurísticas siguen sin ser capaces de obtener buenos resultados llegando incluso, como se puede ver, a obtener peores valores tras aumentar el tamaño de la dimensión.

Sin embargo, si nos fijamos en la tabla 5.19 se puede ver que, estos valores no se alejan tanto como en el caso de las funciones unimodales, aunque tampoco se obtiene una mejora notable, exceptuando el caso de la función 9.

Para esta función se puede observar que, en el caso de ITGO, sí que se reduce el margen de error con respecto a los casos previos calculados con dimensión 30, alcanzando valores similares a los obtenidos para la función 7 de manera que vemos que, en este caso, los resultados al trabajar con estas funciones multimodales básicas se normalizan.
Uno de los motivos por lo que para la mayoría de los casos no se da una mejora, es que los cambios que antes podían resultar significativos, dada la menor dimensión con la que se trabajaba, ahora resultan algo más irrelevantes debido a que se necesita una mayor modificación para cada miembro de la población que de lugar a que el cambio sea importante.

Pero, al no llegar a cambiar lo suficiente, esto se refleja de manera que no se da un desplazamiento lo suficientemente grande, por lo tanto no se llega a alejar lo suficiente de la zona que está explorando y da lugar a que se queden otras zonas sin explorar y se centre más en la actual.
Esto provoca que, al quedarse en una misma zona, se haga un estudio más exhaustivo donde, para todos los casos no resulta en una mejora dado que no están en un sector del espacio con buenas soluciones; sin embargo, para el caso de la función 9 sí, dado que no salta a otra zona nueva pudiendo así obtener mejores valores al explotar una zona con casos de estudio buenos.

Obviando la comparativa entre los valores obtenidos anteriormente y los nuevos, podemos ver que el funcionamiento del ITGO y del PSO se ha mantenido aún trabajando con una mayor dimensión; es decir que para el caso de ITGO el rango de valores queda atrapado alrededor de un mismo valor en casi todos los casos, de manera similar a cómo ocurría previamente, a excepción de para la función 4 donde se dan valores algo más distintos, por lo que al aumentar la dimensión vemos que sigue próximo a un óptimo local como ocurría al trabajar con dimensión 30.

Mientras que para el PSO el rango de valores obtenido sigue siendo algo más amplio, como ocurría cuando se trabajaba con una dimensión de 30, aunque se puede ver un leve estancamiento cerca de un posible óptimo para las funciones 4 y 14.

Capítulo 6

Conclusiones

Durante este trabajo el objetivo principal ha sido el de realizar un estudio y análisis de la metaheurística Invasive Tumor Growth Optimization.

Conclusiones del algoritmo ITGO

Para ello en primer lugar se ha desarrollado el algoritmo ITGO, donde se ha comprobado que, a pesar de ser similar al PSO, ya que ambos se basan en movimientos de partículas, tienen una principal diferencia donde: en el caso del PSO todas las partículas tienen una guía principal, como ocurre con los algoritmos basados en colonias de hormigas o como ocurre en el caso de la migración de las aves. Mientras que en el caso del ITGO, las partículas con un papel principal a la hora de explorar (células proliferativas) se guían por su entorno y los nutrientes que se encuentran en este y al extenderse baso su propio criterio van influenciando a los otros dos tipos de células.

EL comportamiento de estas células da lugar a que se puedan alcanzar y explorar distintas zonas dentro del espacio de búsqueda y hacen de guía a los otros dos tipos de células dando lugar a que se produzca una mayor exploración que con PSO.

Resulta algo positivo ya que, al producirse una mayor exploración, se pueden alcanzar mejores valores mucho antes que con otro tipo de algoritmos, aunque puede suponer un gasto de recursos extra a la hora de explorar zonas que no aportan mejora alguna.

Al introducir esta nueva mecánica de subgrupos de células repartidas en varias capas, evitamos que estas se expandan a la misma velocidad y en todas direcciones, consiguiendo así obtener un ratio entre exploración y explotación en principio robusto.

Esto se logra dado que, como se ha comentado antes, con las células proliferativas se buscan nuevas zonas dando lugar a que ocurra la exploración mientras que al limitar el movimiento de las células inactivas y las moribundas mediante el nivel de nutrientes y el movimiento que realizan las demás células de su entorno, se obtiene una mejor explotación debido a que se estudia con más detenimiento su entorno y se pueden ubicar valores que, de otra forma habrían sido obviados o descartados rápidamente.

También hay que tener presente que algunas células mueren por falta de nutrientes, células moribundas, por lo que se debe tener en cuenta que si, simplemente fallecen y no ocurre nada más con ellas, la población de estas células muertas irá en aumento y, según se vayan descartando y se reordene la población principal sin ellas, se dará un comportamiento en el que se irá mermando dicha población inicial hasta que llegue un punto en el que sea insostenible.
Para evitar que esta situación pueda ocurrir, cada vez que fallece una célula y queda apartada de la población inicial se produce una mutación de una de las células proliferativas y nace una nueva célula que pasará a sustituir a la fallecida de forma que cuando se reordenen en la siguiente iteración ocupe un lugar entre las células con su mismo nivel de nutrientes. Al esta célula poder ser de cualquier tipo, es decir ir a cualquier capa, se evita el premiar a un tipo de célula desplazando las demás evitando así modificar en exceso la trayectoria que seguía cada célula durante su fase de movimiento.

En segundo lugar se ha realizado el estudio de los datos con la finalidad de comprobar el rendimiento de este algoritmo con la ayuda de las funciones del CEC2005 y mediante la comparación del mismo con los algoritmos PSO, DE, y JDE.

Conclusiones de los resultados

Tras este estudio se ha podido llegar a las siguientes conclusiones:

En primer lugar los algoritmos evolutivos, como se ha comentado antes, mejoran en varios aspectos a los basados en movimiento de partículas por lo que no ha sido de sorprender que obtuvieran mejores resultados tras los experimentos, lo que sí ha resultado ser interesante es que su rendimiento para funciones unimodales, como la 3, 4 y 5, empeora y resulta bastante más similar al obtenido por los otros dos algoritmos.

Los valores calculados no son muy distantes a los que se han podido

obtener con el algoritmo PSO, por lo que la idea del algoritmo no va desencaminada aunque no se lleguen a alcanzar el óptimo en ningún caso, por lo que vemos que admite mejora.

Si nos centramos en el algoritmo desarrollado, ITGO, estudiando los resultados obtenidos, vemos que esta primera versión del algoritmo tiende a estancarse en óptimos locales independientemente del tamaño de dimensión llegando a empeorar al aumentar dicho valor. Sin embargo también se ha podido averiguar para qué funciones su funcionamiento es mejor dentro de las funciones disponibles con las que trabajar de manera que para las funciones multimodales se puede ver que ITGO funciona en general mejor que para funciones unimodales donde se comprueba un peor rendimiento bastante lejano del óptimo.

Conclusiones finales

Como conclusión final comentar que, los datos obtenidos han sido satisfactorios, ya que se con ellos se ha podido comprobar el rendimiento del ITGO, como se ha comentado previamente, así como tener una idea de qué partes pueden modificarse, con la finalidad de añadirle pequeñas futuras mejoras, dado que tal y como está planteado se queda algo atrás con respecto a los demás y se podrá mejorar.
Algunas de las mejoras que se proponen son: el uso de un operador de cruce a la hora de mutar las células para que las que se obtengan nuevas no sean tan aleatorias y poder así controlar mejor la parte de explotación; reduciendo búsquedas en zonas no tan buenas, añadir un operador de vecino más cercano a la hora de seleccionar las dos células más próximas para el caso de las inactivas. También se podría aplicar una búsqueda local a la hora de realizar la explotación con cualquiera de las células del conjunto, intensificándola.

Con estos primeros cambios se podría mejorar ITGO de manera que fuese más robusto al mejorar tanto la exploración como la explotación.

Bibliografía

[1] ITGO. http://www.sciencedirect.com/science/article/pii/S1568494615004986.

[2] R.N. MANTEGNA. Fast, accurate algorithm for numerical simulation of levy stable stochastic process. 1994.

[3] Funciones CEC2005. http://sci2s.ugr.es/sites/default/files/files/TematicWebSites/EAMHCO/contributionsCEC05/Tech-Report-May-30-05.pdf.

[4] Algoritmo PSO. http://www.cs.us.es/~fsancho/?e=70.

[5] Algoritmo DE. http://wwwae.ciemat.es/~cardenas/docs/lessons/evoluciondiferencial.pdf.

[6] Algoritmo JDE. http://wwwae.ciemat.es/~cardenas/docs/lessons/evoluciondiferencial.pdf.

[7] J.M. MORENO y J.A. MORENO. Heurísticas en optimización. 1999.

[8] K.A. DOWSLAND y A. DÍAZ. Diseño de heurísticas y fundamentos del recocido simulado. 2003.

[9] F. GLOVER y G.A. KOCHENBERBERO. Handbook of metaheuristics. 2003.