"Optimización gravitatoria" y "Optimización por enjambre de partículas". Comportamiento en funciones no-lineales

Zapatero Moreno, María José, pzapa@ubu.es
Alegre Martínez, Jesús, jfalegre@ubu.es
Pacheco de Bonrostro, Joaquín, jpacheco@ubu.ese
Departamento de Economía Aplicada
Universidad de Burgos

RESUMEN

Proponemos dos heurísticas para obtener óptimos globales de varias funciones no lineales, algunas multimodales. Una de las heurísticas está basada en la estrategia denominada Optimización Gravitatoria; en ella se concibe el espacio de soluciones análogamente al espacio-tiempo relativista, en el que la métrica es modificada por el campo gravitatorio generado por las diferentes partículas en él inmersas. En ella, el papel de la atracción gravitatoria lo juega la función objetivo; el óptimo se encontrará en el punto donde se encuentre la mayor masa. Como esta posición se desconoce, mediremos la variación de la geometría; Igual que en relatividad general la variación de la geometría nos lleva a descubrir la mayor masa, en la heurística nos conduce al óptimo global. La segunda heurística es conocida como Optimización por Enjambre de Partículas; en ella las partículas se mueven por la inercia y la atracción de sus líderes.

***Palabras claves*:** Heurística; optimización no lineal; Optimización Gravitatoria; SGO; función multimodal; Optimización por Enjambre de Partículas; PSO; .

Área temática: Optimización.

ABSTRACT

We propose two different heuristics for obtaining global optimum of several nonlinear functions, some multimodal. One of them is based on the optimization strategy called Space Gravitational Optimization where the solution space is seen as the relativistic space-time, in which the metric is modified by the gravitational field generated by the different particles embedded in it. The role of the gravitational pull is played by the objective function; the best would be in the hypothetical point where the greatest mass lies. As this position is unknown, it's necessary to measure the change of geometry. In the same way as in general relativity the change in geometry leads us to discover the largest mass, in this heuristic leads us to the global optimum. The second heuristic is well known as Particle Swarm Optimization, in it the particles will move guided by the effect of inertia and the attraction of leading members.

Keywords: Heuristic; non-linear gravitational optimization; multimodal function; Particle Swarm Optimization; PSO; Space gravitational Optimization; S.G.O.

1. INTRODUCCIÓN

Optimización gravitatoria, SGO, es una heurística introducida recientemente por Hsiao, Chuang, Jiang and Chien (2005). En ella se concibe el espacio de soluciones de forma similar al espacio-tiempo relativista.

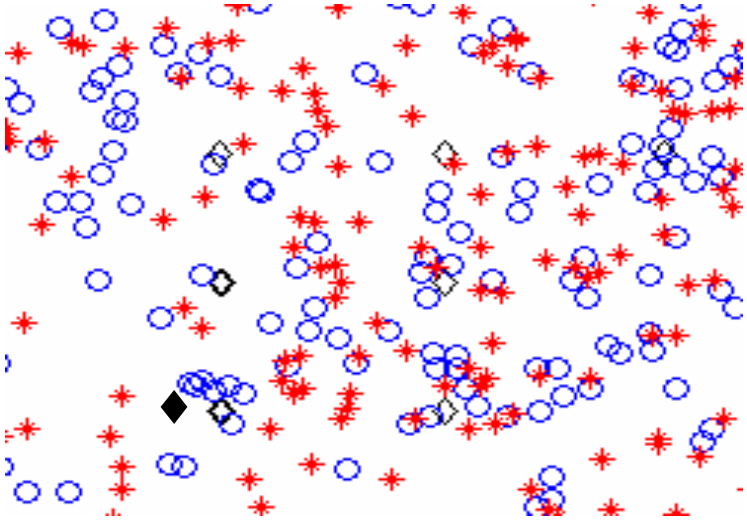
Hemos implementado esta novedosa heurística, cuya única referencia en la literatura es la mencionada en el párrafo anterior, para obtener los óptimos globales de un conjunto de funciones de optimización difícil, funciones multimodales con un número elevado o muy elevado de óptimos locales.

Según la teoría de la relatividad general el espacio se curva por efecto del campo gravitatorio. Las partículas (asteroides, agentes de búsqueda, etc…) se aceleran hacia las masas grandes por las variaciones en la geometría del espacio tiempo.

Se van a considerar en este procedimiento unos agentes de búsqueda, que vamos a denominar asteroides. Estos se están moviendo en el Universo, que será el espacio de búsqueda. Cuando un asteroide es capturado por la gravedad de una masa grande tiene dos posibilidades: salir disparado como una honda y, en ese caso, seguir buscando otra masa pesada en el universo o chocar contra su superficie. Para evitar esto último en el algoritmo se supone que el cuerpo real de las masas pesadas no existe realmente. Así cuando un asteroide se acerca al centro de una masa pesada, se incrementa considerablemente la energía cinética de la partícula por la gravedad de la masa, y el asteroide adquiere suficiente velocidad para escapar de la región del espacio que ocupa la masa pesada (óptimo local).

Un algoritmo así diseñado requiere poca memoria y poco poder computacional. La posibilidad de caer en óptimos locales es muy pequeña y los asteroides tienen muchas posibilidades de salir disparados fuera del óptimo local. Tampoco quedarán atrapados por el óptimo global, en el caso que lleguen a hasta él, sino que seguirán la búsqueda…

En el ejemplo de la imagen que se muestra a continuación se han considerado tan sólo dos asteroides, representados uno de ellos por círculos azules y otro por asteriscos rojos y observamos como los asteroides han sido atraídos hacia los diferentes óptimos locales (◊), han seguido su trayecto y han pasado cerca del óptimo global (♦).



Otro algoritmo muy relacionado con el anterior es el denominado "Optimización por Enjambre de Partículas", Particle Swarm Optimization PSO. Esta metaheurística fue introducida por James Kennedy y Russel Eberhat en 1995, para simular el movimiento de algunos seres vivos. Está inspirada en el comportamiento social de animales que se mueven al unísono como aves en bandadas, peces en bancos o insectos en enjambres.

Las partículas que forman parte de la población (enjambre) van a actuar como agentes de búsqueda en el espacio de soluciones. Cada partícula se comunica con un entorno social o grupo. Este entorno, que puede ser parte o todo el enjambre, puede variar dinámicamente. En cada partícula se guarda información de la mejor posición ocupada, es decir dónde se ha obtenido un mejor valor de la función objetivo, y del mejor valor obtenido por cualquier partícula de su entorno. La información de estas mejores posiciones guiará el comportamiento de las partículas, junto con un efecto inercial de su movimiento.

Las diferencias principales entre estas dos metaheurísticas estriban en que en Optimización Gravitatoria se tiene sólo en cuenta el efecto de la gravedad y no el efecto de la inercia y de la atracción de los miembros más relevantes que lideran el enjambre como en PSO.

Los algoritmos han sido probados con un conjunto de funciones multimodales, con un número de variables comprendido entre dos y cuatro. El trabajo se estructura de la forma siguiente: se describen ambos algoritmos, a continuación se muestran las diferentes pruebas computacionales y por último se exponen las diferentes conclusiones. Al final hay un apéndice con información sobre todas las funciones utilizadas en este trabajo.

2. DESCRIPCIÓN DE LOS ALGORITMOS

2.1 Optimización gravitatoria

El espacio de soluciones está formulado como un espacio tiempo curvado de acuerdo al concepto de Einstein. Un número de *n* asteroides o agentes de búsqueda, *i* =1..*n*, se mueve en el espacio de soluciones y va buscando el cuerpo con mayor masa (óptimo global). Denotaremos por P[i], V[i], A [i] la posición, velocidad y aceleración de cada asteroide, y por *f* a la función objetivo.

Paso I: Inmersión de los asteroides en el espacio de búsqueda:

A los asteroides inicialmente se les asigna una posición aleatoria, $P_0[i]$ *(* si el número de variables de la función es *dimension_espacio, cada componente será denotada por* $P_0[i][k]$*, k = 1..dimension_espacio*[1]*)*, dentro de su región factible, y así mismo se les asigna una velocidad aleatoria, $V_0[i]$, con valores en la región factibe, para evitar, en lo posible, que el asteroide se "*escape"* a posiciones no factibles.

De partida los asteroides tienen una aceleración nula, $A_0[i]$, y a partir de ese momento quedan sometidos únicamente a la atracción gravitatoria generada por f (ley de la gravedad de Newton). Se va a almacenar la mejor solución por la que pasa cada asteroide *i,* que inicialmente se supone *f_best*[i] = +∞.

[1] En lo que resta de trabajo asumimos que la dimensión de las magnitudes vectoriales será de *dimension_espacio.*

Los asteroides tratan de buscar los objetos de masa más pesada. Los que no queden atrapados en óptimos locales se dirigirán a la masa más pesada del Universo, el óptimo global.

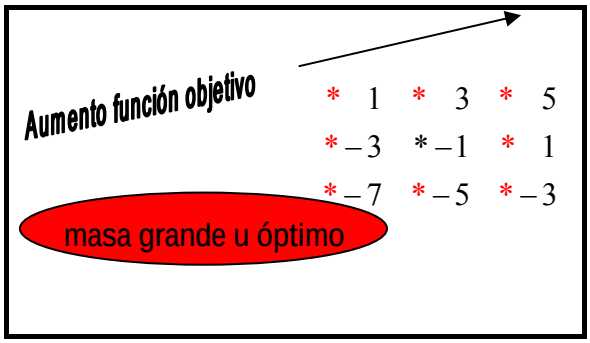
Paso II: Buscando la variación del espacio tiempo:

El espacio tiempo del universo está curvado por el campo gravitacional, y la variación de la geometría del espacio-tiempo modifica la aceleración de los asteroides. Como desconocemos la posición del óptimo (masa mayor) para determinar la trayectoria de los asteroides debemos detectar la variación en la geometría del espacio tiempo: para ello se emplea un parámetro llamado rango de detección r_d. Para calcular la aceleración de cada asteroide por variación espacio-tiempo, se emplea la siguiente fórmula que para el asteroide *i*, en su componente k es:

A[i][k] = G({f(P[i]) - f(P[i][1],.....,P[i][k] + r_d,.....)} + {f(P[i][1],.....,P[i][k] - r_d,.....) - f(P[i])}) = G{f(P[i][1],.....,P[i][k] -r_d,.....) - f(P[i][1],.....,P[i][k] + r_d,.....

donde G es un nuevo parámetro que representa la intensidad del campo gravitatorio.

En la fórmula, para calcular cada componente de la aceleración, se multiplica G por la suma de las variaciones parciales de la función objetivo en el ámbito del rango de detección para dicha componente.En el ejemplo del gráfico observamos la tendencia hacia posiciones con menor valor de la función $f(x,y)=4y+2x-7$. Los asteriscos representan diversos puntos, cuya distancia en sentido horizontal y vertical es un rango de detección (en este caso se ha considerado r_d.=1) y a su lado el valor en ellos de la función objetivo. El asterísco abajo a la izquierda estaría situado en el punto (0,0), el de encima de él en el punto (0,1), etc…



Así en este caso para el punto central la aceleración producida por la variación de la geometría sería (suponiendo que G=1) en cada componente de A_x= -2-2= -4 y A_y = -4-4 = -8. Así si la velocidad de partida fuese (10, 5) pasaría a ser (6, -3)

Paso III: Actualizando la velocidad y posición de los asteroides:

Tras cada iteración hay que recalcular la velocidad, y actualizar la posición de los asteroides a partir de las nuevas velocidades: así para el asteroide *i* su velocidad es

V[i]=V[i]+A[i], y la posición del asteroide P[i]=P[i]+V[i]

Podría ocurrir que algún asteroide se salga del espacio factible, en ese caso hemos recuperado su factibilidad, y para ello el procedimiento que hemos utilizado consiste en devolver al asteroide, de forma aleatoria, a la región factible. Simultáneamente se han ido contando las infactibilidades, y su cómputo se ha tenido en cuenta a la hora de dar valores a los parámetros, procurando que el número de infactibilidades no sea muy elevado.

Paso IV: Actualización de la mejor solución del cada asteroide:

En una matriz *posicion_best[i]* vamos almacenando la mejor posición de cada asteroide y en otra matriz, *f_best[i]*, el mejor valor de la función objetivo. Tras cada iteración si la solución obtenida por el asteroide i es mejor que la mejor conocida hasta ese momento son actualizadas *posicion_best[i]* y *f_best[i]*.

2.2 Modificaciones de Optimización gravitatoria:

De la misma forma que en el trabajo de Hsiao y otros (2005) consideramos las siguientes modificaciones del algoritmo:

1.- Efecto de los asteroides en la geometría: añadimos el efecto de los asteroides en la modificación de la geometría del espacio tiempo; éstos van a producir un campo gravitatorio que debe de ser considerado. En la fórmula que nos servía para el cálculo de la aceleración hay que añadir $\alpha.C[k]/d_i^{2,}$, donde C es el centro de masas de todos los asteroides, C[k] su componente k-ésima y d_i es la distancia entre el asteroide *i* y el centro de masas C. En cada iteración ejecutaremos un procedimiento llamado *Calcular_posicion_centro_masas_asteroide*s.

Para calcular el centro de masas utilizaremos que $C = \frac{\sum_{i=1}^{n} P[i]}{n}$.

2.- Contracción o expansión del Universo: en cada componente la actualización de la velocidad viene dada por la fórmula

velocidad[i]:= β*velocidad[i]+aceleracion[i],

donde β será un parámetro mayor que 1 si se supone que el universo se contrae (cuando todo parece aproximarse a los asteroides) y menor que 1 si se expande (cuando todo parece alejarse de los asteroide).

El pseudocódigo del algoritmo es.

Procedimiento Gravita (mejorsolucion)
Inicializar aleatoriamente todos los asteroides
Mientras t < max_iter hacer
Calcular posición centro masas asteroides;
Para cada asteroide i hacer
Calcular aceleración asteroide por variación espacio-tiempo;
Calcular velocidad tras aceleración;
Calcular posición tras variación velocidad ;
Si no factible i HacerFactible;
Obtener funcion_objetivo;
Si Valor_funcion < f_best[i] entonces actualizar posicion_best[i] y f_best[i]
t:= t+1

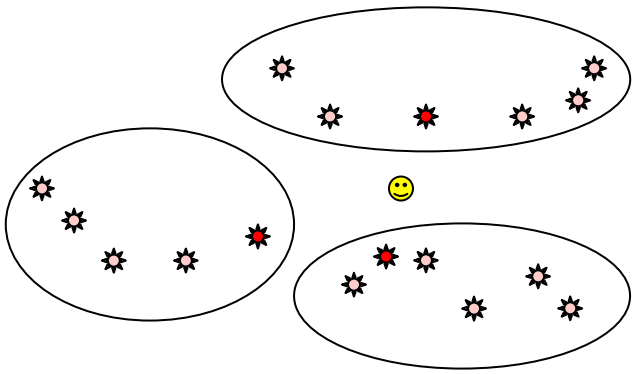
2.3 Optimización por enjambre de partículas.

El espacio de soluciones está formulado como un espacio de dimensión *dimension_espacio*. En él una población (enjambre) de partículas (insectos) de tamaño *n*, que actúan de agentes de búsqueda, se mueve en el espacio de soluciones guiadas por los miembros del enjambre que han obtenido las mejores posiciones – mejores valores de la función objetivo *f* -. El tamaño del enjambre suele oscilar entre 20 y 40 partículas. Para problemas muy difíciles se eleva a un rango de 100-200.

Cada partícula *i* se comunica con un *entorno o grupo social N(i)*, Este entorno puede ser parte o todo el enjambre, y puede variar dinámicamente. La estructura de los entornos puede tener una topología anular en la que cada partícula se relaciona con dos, una topología en forma de estrella en la que cada partícula se relaciona con todas. Otra alternativa, Kennedy y Clerc 2006, es fijar el número de partículas *k* que informan a otras que son elegidas al azar. Estas partículas se renuevan cada vez que mejora la posición grupal.

Cada partícula *i* guarda información de la mejor posición obtenida *Pbest* y del mejor valor obtenido por cualquier partícula del entorno Gbest. La información de estas mejores posiciones influye en el comportamiento de la partícula.

Así pues, cada partícula *i* del enjambre lleva asociados los siguientes vectores: P[i] posición actual, V[i] velocidad actual, Pbest[i] posición de la mejor solución encontrada, Gbest[i] mejor posición obtenida por cualquier partícula del entorno.



En el gráfico podemos ver un enjambre dentro del cual hay tres grupos o entornos sociales de partículas. Cada grupo tiene su "leader", ✸*. El óptimo se encuentra en* ☺*.*

Veamos los diferentes pasos de la versión estándar de este algoritmo.

Paso I: Inmersión aleatoria de los insectos en el espacio de búsqueda:

A las partículas inicialmente se les asigna una posición aleatoria, $P_0[i]$ *(* si el número de variables de la función es *dimension_espacio, cada componente será*

denotada por $P_0[i][k]$, k = 1..dimension_espacio[2]), dentro de su región factible, y así mismo se les asigna una velocidad aleatoria, $V_0[i]$, con valores en la región factible, para evitar, en lo posible, que la partícula se "*escape"* a posiciones no factibles.

Se va a almacenar la mejor solución por la que pasa cada partícula *i,* que inicialmente se supone *f_best*[i] = +∞. Análogamente se hace lo mismo con la mejor solución del grupo social al que pertenece *i, f_best_grupo[i]*.

Paso II: Actualizando la velocidad y posición de las partículas:

Tras cada iteración hay que recalcular la velocidad, y actualizar la posición de los insectos a partir de las nuevas velocidades. Así para la partícula *i* su velocidad es:

V[i]=*c_inercia*.V[i]+ *c_confianza1* .*rnd1*.(Pbest[i]- P[i])+ *c_confianza2*. *rnd2*.(Gbest[i]- P[i]) (*)

Donde *c_inercia* es un parámetro que representa el efecto de la inercia, controlando el efecto de la velocidad y evitando que crezca indefinidamente; *c_confianza1* y *c_confianza2* son parámetros que marcan la confianza de la partícula en sí misma y en su grupo. Los valores *rnd1* y *rnd2* son números aleatorios entre 0 y 1. Es habitual tomar *c_inercia=1* y *c_confianza1* = *c_confianza2* en el rango [0,4]. También lo es tomar estos tres parámetros de forma que su suma sea 1. En la versión estándar de Kennedy y Clerc los valores son: *c_inercia* = 1/(2+ln2) y *c_confianza1* = *c_confianza2* = 0.5 + ln2;

La posición de la partícula será: P[i]=P[i]+V[i] (**)

Podría ocurrir que algún insecto se salga del espacio factible, en ese caso un procedimiento que utilizan algunos autores es usar un parámetro llamado Vmax, y si alguna componente de la velocidad la excede, la velocidad en esa dimensión es fijada a Vmax. Nosotros el procedimiento que hemos empleado para recuperar su factibilidad consiste en devolver a la partícula, de forma aleatoria, a la región factible. Simultáneamente se han ido contando las infactibilidades, cuyo cómputo se ha tenido en cuenta a la hora de dar valores a los parámetros, procurando que el número de infactibilidades no sea muy elevado.

[2] En lo que resta de trabajo asumimos que la dimensión de las magnitudes vectoriales será de *dimension_espacio.*

Paso III: Actualización de las mejores soluciones de cada partícula: En una matriz *Pbest[i]* vamos almacenando la mejor posición de cada insecto y en otra matriz, *f_best[i]*, el mejor valor de la función objetivo. Tras cada iteración si la solución obtenida por la partícula *i* es mejor que la mejor conocida hasta ese momento son actualizadas *Pbest[i]* y *f_best[i]*. Así mismo en la matriz Gbest[*i*] se va almacenando la mejor posición del grupo social o entorno de la partícula *i*. Tras cada iteración si la solución obtenida por alguna de las partículas de $N(i)$ es mejor que la mejor conocida hasta ese momento es actualizada $Gbest[j], \forall j \in N(i)$.

El pseudocódigo del algoritmo es.

```
Procedimiento PSO (mejorsolucion)
Inicializar aleatoriamente todas los insectos
Mientras t <  max_iter hacer
        Para cada insecto i  hacer
                Calcular velocidad teniendo en cuenta la inercia, su mejor
posición y la
                  mejor posición de su entorno social (*)(**);
                Calcular posición tras variación velocidad ;
                Si no factible i  HacerFactible;
                Obtener funcion_objetivo;
                  Si Valor_funcion < f_best[i] entonces actualizar
P_best[i] y
                  f_best[i]
              Si Valor_funcion < f_best_grupo[i] entonces actualizar
G_best[j] y
            f_best_grupo[j]  ∀J ∈ N(i)
        t:= t+1
```

3. EXPERIENCIAS COMPUTACIONALES

Hemos probado la efectividad de los algoritmos con un conjunto de funciones no lineales de varias variables, entre ellas varias son multimodales (tienen diversos óptimos locales), y algunas de éstas tienen un número muy elevado de óptimos. Las funciones tienen un número de variables comprendido entre dos y cuatro. Estas funciones también han sido utilizadas, por ejemplo, como funciones test para algoritmos Scatter Search en un trabajo de Laguna y Marti (2005), y para un algoritmo genético, con carácter estocástico, en un trabajo de Tu and Lu (2004).

Las pruebas han sido realizadas en un Pentium 4, 2.8 GHz. Las implementaciones de los algoritmos de este trabajo han sido realizadas en lenguaje Pascal con el compilador Delphi 5.0., y con Matlab 6.5. Las representaciones gráficas se han hecho con Matlab.

Las funciones y sus principales características las mostramos en la tabla que viene a continuación.

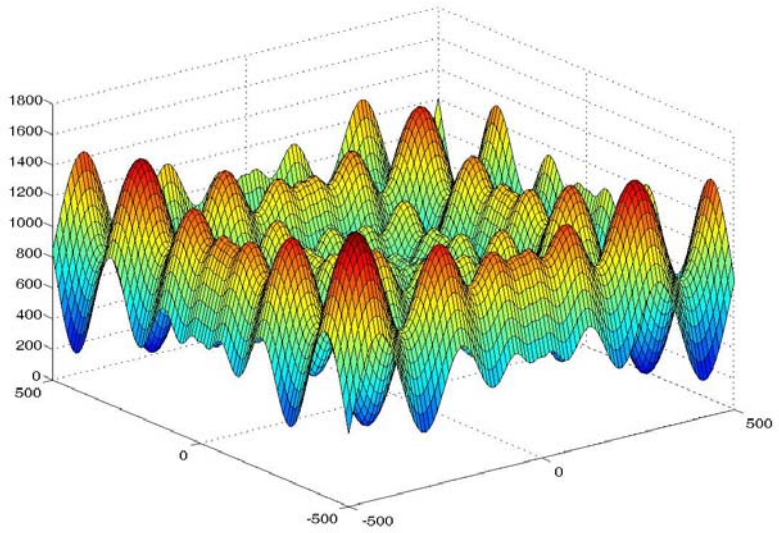
Problema	variables	nombre	x^* (óptimo)	$f(x^*)$
$g_1(x)$	2	Space_paper	(-2.8362,-2.8362)	-130.8323
$g_2(x)$	2	Branin	(9.42478, 2.475) **(!)**	0.397887
$g_3(x)$	2	Easom	(π,π)	-1
$g_4(x)$	2	Shubert	(0.0217,-0.9527)**(!)**	-186.7309
$g_5(x)$	2	Schwefel f_8	(1,…,1)	- 418.9829 · nº variables = -837.9658
$g_6(x)$	4	Rosenbrocks f_5	(1, …, 1)	0

(!) Hay múltiples soluciones óptimas

La función que denominamos $g_1(x)$, Space_paper, es una función optimizada en el trabajo de Hsiao, Chuang, Jiang Chien (2005) donde se introduce la heurística Optimización gravitatoria.

Algunas de estas funciones objetivo han sido construidas de forma que su óptimo es conocido, pero que no puede ser hallado por procedimientos generales de

búsqueda. Además los problemas no pueden ser trivialmente resueltos por procedimientos de búsqueda que no exploten la estructura especial de cada función. Varias de estas funciones son multimodales con muchos óptimos locales y que representan la clase de problemas con mayor dificultad. Un ejemplo de estas últimas es la conocida como función de Schwefel, cuya representación gráfica en el caso de dos variables podemos contemplar en la figura que viene a continuación.



El algoritmo PSO que hemos considerado es muy sencillo, en él se ha considerado un único entorno social que coincide con la totalidad del enjambre.

En el ajuste de los parámetros, sin necesidad de hacer ninguna prueba, se tiene la impresión que dependiendo de la naturaleza del problema, es decir de la función objetivo, los mejores valores para los parámetros variarán.

Aplicando el algoritmo SGO este supuesto se nos ha confirmado: al realizar las diferentes pruebas no se ha conseguido un único valor de los parámetros que vaya bien con todas las funciones. Con PSO se ha conseguido elegirlos que sean adecuados en todas las pruebas.

En los valores que hemos dado a los parámetros van a influir las diferentes magnitudes de la función objetivo, magnitud relativa de las variables, las infactibilidades cometidas, etc…. Además se han ido contando las infactibilidades, y su

cómputo se ha tenido en cuenta a la hora de dar valores a los parámetros, procurando que el número de infactibilidades no sea muy elevado.

En SGO los valores más utilizados han sido: G = 0.001, r_d = 0.001, alfa = 0.05 y beta = 1.01.

El espectro de valores utilizados ha sido:

parámetro |	valores		
rango de detección r_d	0.01	0.001	
G	0.1	0.01	0.001
alfa	0	0.05	
beta	1	1.01	0.89

En PSO los valores utilizados han sido:

parámetros |	valores	
Nº de partículas	15	130
c_inercia	0.37	0.37
c_confianza1	1.19	0.5
c_confianza2	1.19	0.5

Para cada asteroide o partícula el número de iteraciones que se ha utilizado ha sido de *max_iter =1000*.

En SGO el número de asteroides ha sido de 10-100-1000, hemos comprobado que usar más incrementa el tiempo de computación y la mejora de resultados no es sustancial.

En PSO el número de partículas utilizadas en ha sido de 15. Al constatar que con ese tamaño de enjambre (el habitual en este algoritmo) los resultados no eran muy satisfactorios lo hemos incrementado a 130.

Los resultados obtenidos se muestran en la siguiente tabla:

Prob.	variables	Óptimo global	Mejor resultado obtenido SGO		PSO	
			100 asteroides	1000 asteroides	15 insectos	100 insectos
$g_1(x)$	2	-130.8323	-130.7474	-130.8273	-128.0039	-130.8323
$g_2(x)$	2	0.397887	0.3984	0.3983	0.397887	0.397887
$g_3(x)$	2	-1	-0.788	-0.991	0	-1
$g_4(x)$	2	-186.7309	-186.61	-186.707	-186.7309	-186.7309
$g_5(x)$	2	-837.9658	-837.0950	-837.9320	-719.5274	-837.9657
$g_6(x)$	4	0	0.00096	0.00055	7.90	0.036

5. CONCLUSIONES

Entre los principales resultados extraídos podemos destacar los siguientes. Para ambos algoritmos:

- Las dificultades van apareciendo a medida que se incrementa el número de variables.
- En los ejemplos aunque estemos ante una función multimodal "difícil", como la de Schwefel con muchos óptimos locales, los algoritmos son capaces de "escapar" de ellos y llegar al óptimo global.

Para PSO:

- Con 15 partículas, sin ser muy malos los resultados, sólo se alcanza el óptimo global en dos de los casos. Sin embargo, con un número superior de partículas, 130, el óptimo global se alcanza en todos los casos menos en uno. En este caso, la función de Rosenbrocks, el algoritmo aunque no lacanza el óptimo se queda muy cerca.

Para SGO:

- Obtenemos resultados muy cercanos al óptimo global con pocos asteroides.
- En el caso con más variables se muestra esencial la modificación del algoritmo en que se considera que el universo se contrae.
- En este mismo caso la técnica SGO se muestra mucha más eficiente.

Estamos ante dos heurísticas con un comportamiento muy bueno para la optimización de funciones no lineales de pocas variables. La más novedosa de las dos, Optimización Gravitatoria, aparenta ser una técnica "prometedora" para esta finalidad. Nos queda, como tarea pendiente, estudiar su comportamiento al incrementar el número de variables de los problemas.

6. APÉNDICE

Lista de funciones empleadas:

$g_1 = x_1^4 - 16 \cdot x_1^2 + 0.5 \cdot x_1 + x_2^4 - 16 \cdot x_2^2 + 0.5 \cdot x_2$ función del Space_paper

Dominio de optimización: [-15, 15] x [-15, 15] , que denotaremos $[-15, 15]^2$

$g_2 = (x_2 - \frac{5}{4\pi^2} x_1^2 + \frac{5x_1}{\pi} - 6)^2 + 10(1 - \frac{1}{8\pi})cos(x_1) + 10$ función de Branin

Dominio de optimización: $[-15, 15]^2$

$g_3 = -cos(x_1)cos(x_2)e^{-(x_1-\pi)^2-(x_2-\pi)^2}$ función de Easom

Dominio de optimización: $[-100, 100]^2$

$g_4 = (\sum_{i=1}^{5} i\cos((i+1)x_1 + i))(\sum_{i=1}^{5} i\cos((i+1)x_2 + i))$, función de Shubert

Dominio de optimización: $[-10, 10]^2$

$g_5 = 418.9829n + \sum_{i=1}^{n} (-x_i \sin\sqrt{|x|})$ función de Schwefel

Dominio de optimización: $[-500, 500]^2$

$g_6 = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 - (x_i - 1)^2]$ función de Rosenbrocks

Dominio de optimización: $[-30, 30]^2$

7. REFERENCIAS BIBLIOGRÁFICAS

- Hsiao, Chuang, Jiang and Chien (2005) "A Novel Optimization Algorithm: Space Gravitational Optimization". Systems, Man and Cybernetics, IEEE International Conference on Volume 3, Issue , 10-12 Oct. 2005 Page(s): 2323 - 2328 Vol. 3

- Kennedy J., Eberhart R. (1995) "Particle swarm optimization". Proceedings of the 1995 IEEE International Conference on Neural Networks, IV, 1942–1948

- Laguna M. and Marti R. (2005) "Experimental Testing of Advanced Scatter Search Designs for Global Optimization of Multimodal Functions". Journal of Global Optimization Volume 33 , Issue 2 , Oct. 2005, pages: 235 – 255.

- Martínez García F. J., Moreno Pérez J.A., "Optimización por enjambre para la p-mediana continua y discreta" MAEB 2007

- Tu Z. and Lu Y. (2004). "A Robust Stochastic Genetic Algorithm (StGA) for Global Numerical Optimization". IEEE Transactions on Evolutionary Computation, vol. 8, no. 5, october 2004

- Webs

  - información sobre las funciones: http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/go.htm; http://solon.cma.univie.ac.at/glopt.html.

  - Tutoriales PSO: http://www.particleswarm.info/Standard_PSO_2006.c (Kennedy y Clerc 2006); **http://www.swarmintelligence.org/index.php (Hu, X.)**