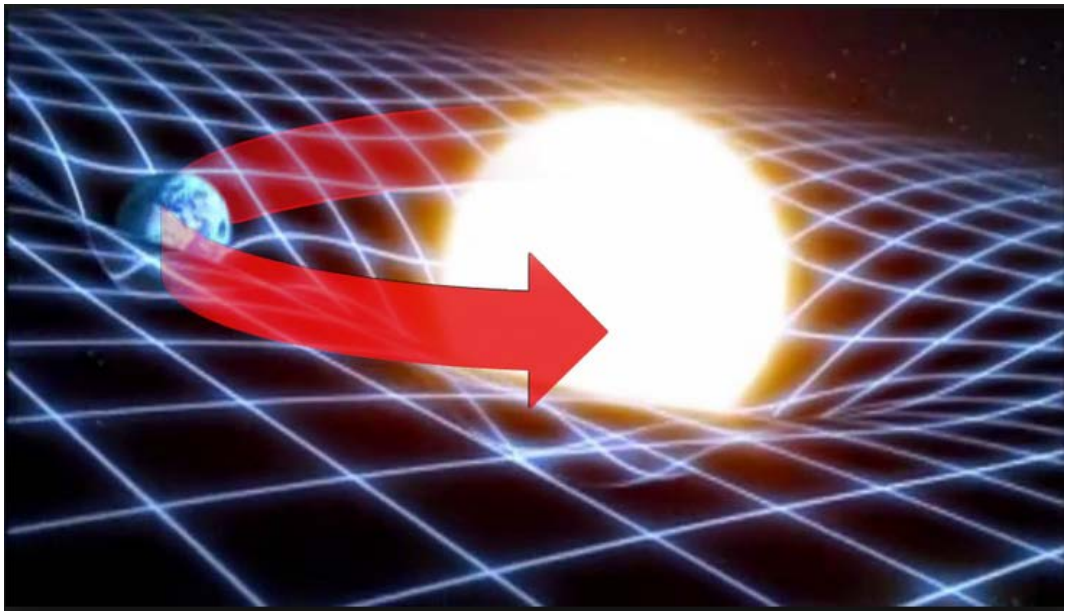
UNIVERSIDAD DE BURGOS

DEPARTAMENTO DE ECONOMÍA APLICADA

ESTUDIO E IMPLEMENTACIÓN DE OPTIMIZACIÓN GRAVITATORIA Y DESARROLLO DE DISTINTAS METAHEURÍSTICAS GENERADAS A PARTIR DE ÉL

María José Zapatero Moreno



Burgos 2015

UNIVERSIDAD DE BURGOS

DEPARTAMENTO DE ECONOMÍA APLICADA

PROGRAMA DE DOCTORADO

TÉCNICAS MODERNAS PARA LA TOMA DE DECISIONES: FUNDAMENTOS Y APLICACIONES

ESTUDIO E IMPLEMENTACIÓN DE OPTIMIZACIÓN GRAVITATORIA Y DESARROLLO DE DISTINTAS METAHEURÍSTICAS GENERADAS A PARTIR DE ÉL

María José Zapatero Moreno

Trabajo codirigido por los doctores

Dr. Jesús Francisco Alegre Martínez

Dr. Joaquín Pacheco de Bonrostro

Burgos, 2015

A la memoria de mi madre

AGRADECIMIENTOS

A mis codirectores, Jesús y Joaquín, por haberme mostrado S.G.O. Por sus imprescindibles consejos y su inestimable dirección.

A Ileana por su constante y desinteresado apoyo a lo largo de tantos años... Por su impagable ayuda con los conceptos físicos. Por mostrarme que siempre hay un camino.

A mi hermano Luis por su gran labor transformando archivos Word ilegibles en perfectas hojas Excel. A él y a Marian por su paciencia y sus importantísimos consejos en la exposición.

A Carola por su ayuda con Word y Adobe y por ejercer de madre de Guille en más de una ocasión.

A Ana y Camelia por su constante apoyo conmigo y con Guille.

A Pilar, a Toñi y a Judith por echarme una mano con Guille.

A Guillermo por haber llevado con paciencia quedarse de vez en cuando sin madre.

Al departamento de Matemática Aplicada de la Universidad de Burgos por poner a mi disposición un equipo en el que realizar las ejecuciones. A mis compañeros por sus consejos. A Santos por su ayuda logística.

A Richard Formato por poner a mi disposición sus resultados y algoritmos.

A todos sinceramente GRACIAS.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

CAPÍTULO 2. ESTADO DEL ARTE

CAPÍTULO 3. PRIMEROS RESULTADOS Y ANÁLISIS ROBUSTEZ

CAPÍTULO 4. DEPENDENCIA DE LOS PARÁMETROS

CAPÍTULO 5. HIBRIDACIÓN CON ALGORITMOS LOCALES

CAPÍTULO 6. HIBRIDACIÓN CON EL ALGORITMO SEGMENTACIÓN

CAPÍTULO 7. HIBRIDACIÓN CON ALGORITMOS DE INTENSIFICACIÓN

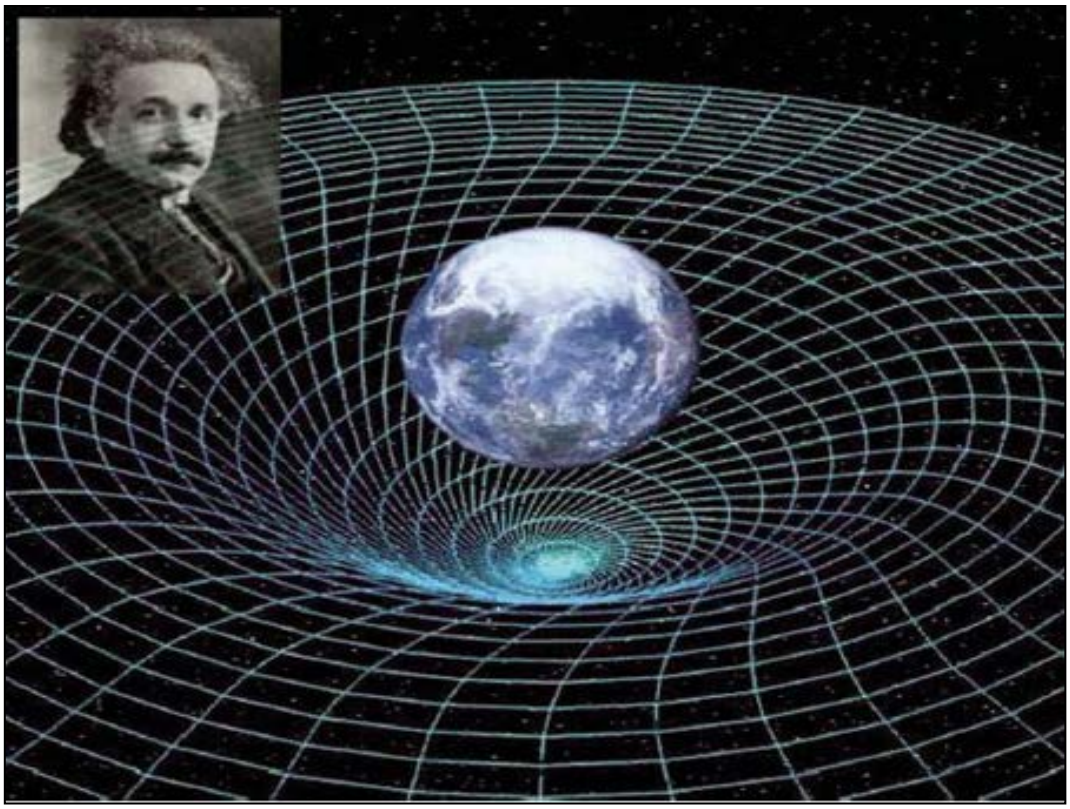
CAPÍTULO 8. PRUEBAS CON LA INSTANCIA CASSINI 2

"Si tu intención es describir la verdad,
hazlo con sencillez.
La elegancia déjasela al sastre."

Albert Einstein (1879-1955)

CAPÍTULO 1

DISEÑO DE LA INVESTIGACIÓN



1. INTRODUCCIÓN

En el año 2008 cuando estaba buscando algún proyecto sobre el que realizar mi tesina, Jesús Alegre me mostró un artículo titulado "A Novel Optimization Algorithm: Space Gravitational Optimization, S.G.O." (Hsiao y otros 2005) y me planteó desarrollar el trabajo de aptitud investigadora estudiando su motivación, implementándolo en MATLAB y comprobando su alcance.

Al ser un algoritmo nuevo y casi desconocido había muy poca literatura sobre él y parecía un campo apropiado para investigar. Además, mi formación universitaria -licenciada en Ciencias Matemáticas en la especialidad de Análisis Numérico- y mi conocimiento previo de MATLAB, por haber impartido las asignaturas de Métodos Numéricos y de Introducción al programa MATHEMATICA en Ingeniería Técnica Informática durante varios años me convertían en un buen candidato para realizar tal trabajo.

Mi experiencia en el campo del Análisis Numérico era, obviamente, amplia. Dominaba la implementación de algoritmos como Gradiente y Gradiente Conjugado y también cómo demostrar su convergencia. Por supuesto sabía que son algoritmos de descenso rápido y no me era ajena su susceptibilidad, realmente necesidad, de quedar atrapados en óptimos locales. Sin embargo mi desconocimiento de métodos heurísticos para encontrar el óptimo global de funciones continuas era total. Reconozco que en un primer momento, al entrar en contacto con algoritmos como Colonia de Hormigas, Enjambre de abejas o Búsqueda Dispersa mi estupor y mi falta de fe fueron, sin paliativos, absolutos y probablemente si el parecido entre S.G.O. y Gradiente no hubiese sido tan evidente, parámetros aparte, no habría considerado siquiera la posibilidad de perder ni 5 minutos programándolo.

Hoy en día mi opinión acerca de los algoritmos heurísticos en optimización ha cambiado completamente. No es fácil demostrar su convergencia teórica pero su convergencia práctica es definitiva en multitud de situaciones, he tenido ocasión de comprobarlo personalmente con S.G.O., son rápidos y muy eficientes y sin ellos no habría podido ser abordada la mayor parte de los problemas reales a los que se enfrenta cada día la Ciencia y la Ingeniería en campos tan diversos e importantes como el desarrollo de antenas (Formato 2010), el cálculo de órbitas de sondas y satélites espaciales, como los que podemos encontrar en la Agencia Espacial Europea, **http://www.esa.int/gsp/ACT/inf/projects/gtop/gtop.html**, el diseño de modelos logísticos en la industria del automóvil, (Alegre 2004), el campo de la optimización combinatoria, (Voudouris y Tsang 1999), en Matemática Aplicada y Computación, (Ding y otros 2012), en inteligencia artificial, (Hatamlou y otros 2011), etc. por poner algunos ejemplos. Sin duda es preferible un método efectivo que aporte una buena solución aproximada en un tiempo razonable, incluso para funciones de dimensión muy alta, aunque no sea posible acotar los errores cometidos, que tener otro que no aporte soluciones aproximadas aceptables aunque teóricamente sea posible demostrar su convergencia.

No quiero decir con ello que haya que desdeñar las demostraciones ni aceptar cualquier idea para crear un algoritmo. Creo que es importante controlar en lo posible los errores pero teniendo también en cuenta que a veces las cosas funcionan aunque matemáticamente no conozcamos con certeza el porqué.

Otra circunstancia importante que se dio para que yo me embarcara en esta aventura, es mi pasión por la Física, en especial la Mecánica Clásica y la Mecánica Cuántica. Siempre he pensado que un matemático especializado en Análisis no puede ser bueno si no conoce, al menos, los rudimentos de la disciplina que conforma su razón de ser.

Así, comenzó mi periplo por el mundo de la optimización continua y en particular el de S.G.O. Mi trabajo y mi situación personal no me dejaban mucho tiempo libre que dedicarle pero, poco a poco aquel primer estudio fue creciendo con nuevos heurísticos con los que hibridar S.G.O. y nuevos retos hasta conformar el objetivo central del presente trabajo de investigación que podría definirse más o menos como **el estudio pormenorizado del algoritmo S.G.O., sus motivaciones físicas, su estructura, su convergencia con funciones de distintas topologías y por último la forma de hibridarlo con diferentes heurísticos para mejorar su eficacia.**

Solo espero haberlo logrado.

2. OBJETIVOS ESPECÍFICOS DE LA INVESTIGACIÓN

El objetivo anteriormente citado se concreta en los siguientes puntos:

- Estudio de las motivaciones físicas de S.G.O. repasando algunos conceptos sobre campos conservativos, sistemas de ecuaciones diferenciales y su resolución numérica usando el método de Euler[1].
- Descripción exhaustiva del algoritmo utilizando herramientas de aproximación de derivadas.
- Análisis de eficacia, eficiencia y robustez del algoritmo con diversos problemas de dimensiones 2 y 4. Fijación de rangos de variación para algunos valores de S.G.O.
- Análisis pormenorizado de la influencia de los distintos parámetros en S.G.O. y determinación de rangos para todos ellos.
- Análisis de la hibridación de S.G.O. con los algoritmos de búsqueda local: Nelder-Mead[2] y Gradiente[3] realizando pruebas con una amplia serie de *benchmark functions*[4] usadas en la optimización continua con restricciones que pueden encontrarse, así como sus códigos en MATLAB, en la siguiente pagina web:

  http://wwwoptima.amp.i.kyotou.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page364.htm

- Innovación y desarrollo del algoritmo Segmentación[5] y análisis de pruebas de la hibridación S.G.O.-Segmentación con *funciones test* de dimensiones altas.
- Innovación y desarrollo del algoritmo Agujero de Gusano[6] e hibridación de S.G.O. con él y con el heurístico "Very Simple Optimization", V.S.O.[7] Análisis de los métodos hibridados realizando pruebas con varias *funciones test* de dimensiones altas.
- Diversas pruebas de los algoritmos expuestos con la función que define la trayectoria Cassini 2, obtenida de la Agencia Espacial Europea.

[1] Explicado en la sección 4.2 del capítulo 2.
[2] Detallado en la sección 6.2 del capítulo 2.
[3] Detallado en la sección 6.3 del capítulo 2.
[4] Detalladas en el apéndice 1.
[5] Detallado en la sección 2 del capítulo 6.
[6] Detallado en la sección 2 del capítulo 7.
[7] Detallado en la sección 7 del capítulo 2.

3. DISEÑO DE LA INVESTIGACIÓN

Para llevar a cabo los objetivos expuestos hemos desarrollado tres fases bien diferenciadas.

- La primera de ellas, que abarca los capítulos 2, 3 y 4, comienza con un repaso general del estado del arte en optimización continua en el que detallamos algunas heurísticas tradicionales y los algoritmos locales con los que vamos a hibridar S.G.O. Continúa con el estudio exhaustivo de las motivaciones físicas del algoritmo y sus parámetros, el desarrollo del organigrama con todas sus etapas y la implementación. También se recogen los primeros resultados y se realiza un análisis de parámetros y número de ejecuciones de S.G.O. con el fin de fijar los valores concretos que serán usados en ulteriores experimentaciones. Esta fase conformó esencialmente nuestro trabajo de aptitud investigadora. Por ello los resultados se presentan de forma diferente al resto. Hemos preferido dejarlos tal y como estaban para que el trabajo no pierda la entidad global que tuvo en su momento.

- La segunda parte abarca los capítulos 5, 6 y 7 y consiste en el desarrollo de una serie de metaheurísticos basados en la hibridación de S.G.O. con algoritmos de diversa etiología para potenciar su eficiencia. También en esta fase realizamos un buen número de experimentaciones con instancias de diferentes topologías y dimensiones. Algunos de ellos han sido diseñados íntegramente por nosotros.

- En la última fase, capítulo 8, realizamos diversas pruebas de todos los algoritmos desarrollados en un caso práctico real. Se trata de la función Cassini 2, una "caja negra" de dimensión 22 cuyo código en MATLAB, formado por múltiples subfunciones, ha sido obtenido de la página web de la Agencia Espacial Europea: **http://www.esa.int/gsp/ACT/inf/projects/gtop/gtop.html**, y define la trayectoria de la homónima sonda espacial.

- La investigación concluye con el capítulo 9, en el que exponemos las aportaciones realizadas, las conclusiones generales y algunas de las posibles líneas de investigación que quedan abiertas tras la conclusión del presente trabajo.

- Se completa con varios apéndices en los que se recogen las instancias utilizadas, los códigos fuente de los programas y parte de los resultados obtenidos. El resto se encuentran en el CD que se adjunta.

Todas las experimentaciones y todas las representaciones de las distintas funciones en dimensión 2 han sido realizadas con el programa MATLAB. Varias son las razones de tal elección.

- Los códigos MATLAB de las *instancias test* usadas para probar los algoritmos, *benchmark problems*, y de las instancias de problemas abiertos, como por ejemplo los *MGADSM problems de la* Agencia Espacial Europea, E.S.A., se encuentran con suma facilidad en internet.
- Buena parte de las aplicaciones en el campo de optimización continua se realizan en Ingeniería: antenas, trayectorias de órbitas, dispositivos electrónicos, etc. Es lógico, por tanto, implementar los algoritmos en uno de los programas de uso más extendido en Ingeniería como es MATLAB.
- Nuestro conocimiento previo de programación en MATLAB y el hecho de disponer de una licencia de Campus en la Universidad de Burgos para MATLAB-R2014 nos facilita en gran medida su uso.

4. ESTRUCTURA DEL TRABAJO DESARROLLADO

La articulación en capítulos del presente trabajo ha sido la siguiente:

- **Capítulo 1. Diseño de la investigación.** En él exponemos los inicios y la motivación del estudio. También comentamos su importancia y algunas de sus posibles aplicaciones prácticas.
- **Capítulo 2. Estado del Arte.** Comenzamos por un repaso de algunos algoritmos relevantes de optimización de funciones continuas y de los heurísticos basados en conceptos gravitatorios. Posteriormente introducimos S.G.O., su motivación física y su organigrama.
- **Capítulo 3. Primeras pruebas. Análisis de la robustez de S.G.O.** En este apartado realizamos las primeras pruebas principalmente en dimensión 2, aunque también hacemos algunas pruebas en otras dimensiones. Posteriormente llevamos a cabo dos estudios de robustez y eficacia del algoritmo: el primero respecto a la variación del parámetro *G* y el segundo respecto a la variación del número de agentes de búsqueda y del número de movimientos que realizan, probándolo repetidamente en las mismas condiciones en varias *instancias test*. Establecemos rangos de variación para los parámetros analizados.
- **Capítulo 4. Estudio de la dependencia de los parámetros.** Realizamos un estudio de los tres parámetros de S.G.O. tomando resultados intermedios y finales con el fin de establecer valores universales. Utilizamos parámetros estadísticos elaborados para medir la robustez del algoritmo respecto a la variación de todos los parámetros.
- **Capítulo 5. Hibridaciones con algoritmos locales.** En él estudiamos la mejora en la eficiencia y la eficacia de S.G.O. hibridándolo con dos algoritmos locales: Nelder-Mead y Gradiente. Escogemos una amplia lista de distintas *funciones test* con topologías y dimensiones diversas para comprobar los resultados.
- **Capítulo 6. Hibridación con el algoritmo SEGMENTACIÓN.** Proponemos y desarrollamos este nuevo heurístico basado en diseño secuencial de experimentos para fraccionar la región factible. Analizamos los resultados obtenidos al probarlo con *funciones test* de dimensiones altas.
- **Capítulo 7. Hibridación con algoritmos de intensificación.** Proponemos un nuevo heurístico llamado Agujero de Gusano e hibridamos S.G.O. con él y con V.S.O. para obtener éxito con *funciones test* de altas dimensiones.
- **Capítulo 8. Pruebas realizadas con la función de Cassini 2.** Realizamos una pequeña introducción sobre los problemas de trayectorias espaciales MGADSM, y en particular sobre el problema de la trayectoria Cassini 2. Posteriormente

exponemos los resultados obtenidos con las distintas hibridaciones planteadas a lo largo del presente trabajo.

- **Capítulo 9. Conclusiones generales y futuras líneas de investigación.** Exponemos las aportaciones obtenidas con los diferentes estudios realizados y con los diferentes metaheurísticos diseñados y proponemos nuevas hibridaciones que realizar con S.G.O y nuevas líneas de trabajo que desarrollar en el futuro.
- **Apéndices.** En este apartado recogemos las definiciones de las instancias utilizadas, así como una página web donde pueden encontrarse. También exponemos con detalle los códigos fuente en MATLAB de los heurísticos implementados y de la instancia Cassini 2. Detallamos, por último, las páginas de internet de las que hemos sacado las imágenes de las portadas de cada capítulo.
- **Referencias bibliográficas.**
- **En el CD adjunto** pueden encontrarse las hojas Excel con todos los resultados obtenidos en cada sección.

5. RELEVANCIA EN EL CAMPO DE OPTIMIZACIÓN DE FUNCIONES CONTINUAS

En la actualidad está surgiendo un buen número de heurísticos basados en concepciones físicas, y en particular en la idea de campo gravitatorio, con muy buenos resultados: Central Force Optimization, C.F.O. (Formato 2007) y sus distintas aplicaciones en el diseño de antenas (Formato 2010 ; Formato 2013), Gravitational Search Algorithm, G.S.A. (Rashedi 2009) con diversos usos en amplificadores (Kilinç 2013) o en diseños de servicios web en SOA (Palanikkumar 2012), o en la resolución de los problemas MGADSM para hallar trayectorias de naves espaciales propuestos por la Agencia Espacial Europea (Hamdy 2011). También hemos encontrado artículos en los que se modifica y mejora G.S.A. (Binjie 2013), se implementa en optimización discreta (Sudin 2013) e incluso se hibrida con P.S.O. (Mirjalili 2010) para incrementar su eficiencia o se cuestiona su fundamentación física (Gauci y otros 2012).

Sin embargo, y a pesar de que han transcurrido 10 años desde su creación, S.G.O. no parece haber alcanzado hasta el momento relevancia significativa ni tenemos constancia de que se haya planteado, salvo en nuestro caso, la idea de hibridarlo conformando así metaheurísticos más potentes[8].

Confiamos en que el presente trabajo contribuya a dar un pequeño empuje a nuestra heurística y sus correspondientes hibridaciones para llegar a tener el protagonismo que, pensamos, merece.

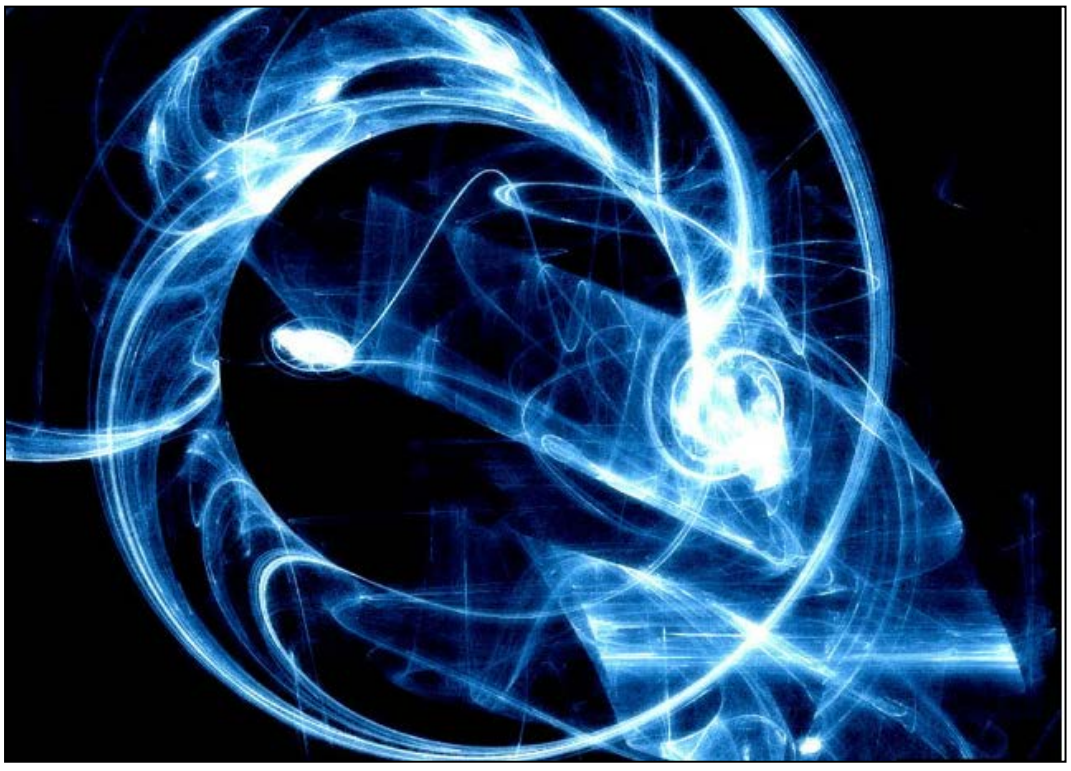
[8] Hemos realizado una búsqueda en la base de datos de la Web of Knowledge sobre S. G. O. en los últimos 5 años y no hemos encontrado ningún trabajo que lo utilice.

"Si he visto más lejos es porque estoy sentado sobre los hombros de gigantes."

Isaac Newton (1643-1723)

CAPÍTULO 2

ESTADO DEL ARTE



1. INTRODUCCIÓN

El problema matemático de encontrar el mínimo absoluto de una función continua definida en un determinado dominio ha sido abordado por diferentes técnicas que podríamos clasificar en tres grandes grupos:

- Técnicas pertenecientes al Análisis Matemático.
- Técnicas pertenecientes al Análisis Numérico.
- Técnicas heurísticas[1] desarrolladas en el ámbito de la investigación operativa con aplicaciones en distintos campos como la economía, la física, la ingeniería etc.

Las primeras aportan un conocimiento teórico más exhaustivo del problema y, en casos muy particulares y de poca complejidad, dan la solución exacta. Desafortunadamente, hay muy pocos problemas que puedan resolverse por estas técnicas.

Las segundas se caracterizan por el diseño de métodos para aproximar, de una manera eficiente, las soluciones al problema matemático planteado. Generalmente conllevan análisis de la convergencia a las soluciones (Burden, Faires 1985). La gama de problemas que estos métodos resuelven es mucho mayor que en el caso anterior. Sin embargo, cuando la complejidad computacional es muy grande, funciones con muchas variables o con muchos óptimos locales, tampoco resultan efectivos.

En las terceras, el término algoritmo heurístico, o simplemente heurístico, alude según Zanakis y Evans a "*Procedimientos simples, a menudo basados en el sentido común, que se supone ofrecerán una buena solución (aunque no necesariamente la óptima) a problemas difíciles, de un modo fácil y rápido"* (citado por Alegre 2004, página 28). En general, el objetivo de estas técnicas es encontrar una solución buena para el problema aunque no sea la óptima.

Una de las ventajas de estos métodos es que son fácilmente adaptables para solucionar modelos más complejos. Es interesante su utilización cuando no existe un método exacto de resolución o éste requiere mucho tiempo de cálculo o memoria, cuando pueden aceptarse soluciones no necesariamente óptimas, cuando los datos son poco fiables, cuando hay limitaciones de tiempo -es frecuente la necesidad de una respuesta rápida- o espacio, como paso intermedio en la aplicación de otro algoritmo, etc.

Tanto los algoritmos numéricos como los heurísticos propuestos para la optimización de funciones continuas, algunos de los cuales comentaremos en las secciones siguientes, adolecen en mayor o menor medida de un defecto que consiste en **su tendencia a caer y permanecer en óptimos locales** renunciando así a la búsqueda del óptimo global. Este hecho es evidente en los llamados algoritmos numéricos de *máximo descenso*, como por ejemplo el

[1] El término *heurístico* deriva de la palabra griega *heuriskein* que significa encontrar o descubrir.

método del Gradiente (Burden, Faires 1985), ya que están diseñados para localizar óptimos locales, y en los heurísticos locales como los conocidos Búsqueda Local (Dowsland 1993) o el Simplex de Nelder-Mead (Nelder 1965).

Aunque no es tan claro en otros heurísticos poblacionales como Enjambre de Abejas, Particle Swarm Optimization, P.S.O., (Kennedy 1995), o Colonia de Hormigas, en Ant Colony Optimization, A.C.O., (Dorigo 1991) también puede darse la tendencia anteriormente descrita, ya que la mejor solución se busca de forma conjunta compartiendo información entre todos los individuos de la colonia. Por ello, una vez que un individuo cae en un óptimo local y la solución es aceptable, atrae a otros individuos también hacia el óptimo local en vez de mantenerse buscando en otras zonas inexploradas del espacio de soluciones (Hsiao 2005). Algo similar sucede en algoritmos como Búsqueda Tabú (Glover 1989) o en algunas heurísticas inspiradas en sistemas termodinámicos simples, como Temple simulado*, S*imulated Annealing, S.A., en las que solo se permite aceptar peores soluciones bajo ciertas condiciones (Pacheco 1997). El problema se agrava de forma significativa cuando el conjunto de óptimos locales, es muy grande[2].

A diferencia de los casos expuestos, en el algoritmo que estudiaremos en el presente trabajo este problema se resuelve de forma bastante satisfactoria porque, como veremos en las secciones siguientes, la posibilidad de que un agente de búsqueda quede “atrapado” en un óptimo local, ni siquiera en el óptimo global, es mínima, lo que permite al usuario hacer una exploración más exhaustiva del espacio de soluciones (Hsiao y otros 2005) sin tener que usar técnicas de diversificación, por lo que el esfuerzo computacional y el uso de memoria son mínimos.

Nuestro objetivo a lo largo del presente capítulo será, tras realizar un somero repaso por algunas heurísticas tradicionales de optimización, estudiar en profundidad los algoritmos heurísticos que denominaremos genéricamente “**heurísticos gravitatorios”**, basados en la metáfora de asimilar la región factible a un campo gravitatorio sometido a las leyes de la física, dedicando una atención especial a un algoritmo denominado **Optimización Gravitatoria, Space Gravitational Optimization,** diseñado por **Ying-Tung Hsiao, Cheng-Long Chuang, Joe-Air Jiang y Cheng-Chih Chien (2005)** para **la optimización de funciones continuas**, que está inspirado en la teoría clásica de campos gravitatorios y en algunas nociones básicas de mecánica relativista y de astrofísica.

Para desarrollar el capítulo comenzaremos realizando un estudio general de las técnicas heurísticas, detallando algunas de ellas. A continuación expondremos minuciosamente algunas de las heurísticas gravitatorias más relevantes para nuestro trabajo, comenzando por S.G.O. y finalizaremos exponiendo los distintos algoritmos con los que hibridaremos S.G.O. a lo largo del desarrollo del presente trabajo.

[2] Situación que definiremos como multimodal.

2. METAHEURÍSTICOS

En los últimos años están cobrando gran importancia un tipo de métodos o estrategias que se han denominado metaheurísticos. Este tipo de métodos se podrían comparar con la figura del "director de orquesta", a la hora de resolver problemas de optimización[3]. Director que permita y ordene la ejecución de algoritmos heurísticos y exactos, siguiendo una partitura, un plan con el objetivo de interpretar una melodía, encontrar una solución factible, lo "mejor posible" (Alegre 2004).

De las muchas definiciones de metaheurísticos que se pueden encontrar en la literatura reciente, citaremos dos:

"... *metaheurística se refiere a una estrategia maestra que guía y modifica otras heurísticas para producir soluciones más allá de aquellas que normalmente se generan en una búsqueda de óptimos locales*." (Glover y Laguna, 1997)

"*Un meta-heurístico es un proceso generacional iterativo que guía un heurístico subordinado combinando inteligentemente diferentes conceptos para explorar y explotar los espacios de búsqueda usando estrategias de información con el fin de encontrar eficientemente soluciones casi-óptimas* ". (Osman y Kelly, 1996)

En estas definiciones queda claro que es un concepto "muy abierto" y por lo tanto sería una empresa difícil y quizá desacertada intentar una clasificación de estos métodos.

No obstante, dado que existen diferentes tipos de heurísticos, según el modo en que buscan y construyen sus soluciones, podemos establecer una posible clasificación, que puede encontrarse en Silver (1980). Aunque hay otros métodos, dependiendo del problema, las clasificaciones clásicas distinguen entre métodos constructivos y métodos de búsqueda por entornos.

- **Métodos constructivos:** consisten en ir añadiendo paulatinamente componentes individuales a la solución, hasta que se obtiene una solución factible. El más popular de estos métodos lo constituyen los algoritmos golosos o voraces (greedy), los cuales construyen paso a paso la solución seleccionando en cada paso la mejor opción según algún criterio.

- **Métodos de búsqueda por entornos:** parten de una solución factible inicial y mediante alteraciones de esa solución van pasando de forma iterativa, y mientras no se cumpla un determinado criterio de parada, a otras soluciones factibles de su entorno, soluciones vecinas, almacenando la mejor de las soluciones visitadas.

[3] Inicialmente fueron propuestos en problemas de optimización combinatoria. Hoy en día se aplican también en problemas de optimización continua con y sin restricciones.

Los algoritmos a los que vamos a dedicar nuestra atención en el presente trabajo son los métodos de búsqueda por entornos.

Las propiedades deseables en cualquier metaheurística, sea cual sea su origen, podrían ser las siguientes (Hansen y Mladenovic, 2000):

- **Sencillez**: deben basarse en principios simples y claros.
- **Coherencia:** todos los pasos aplicados a un problema particular deben derivarse de los principios de la metaheurística.
- **Eficacia:** en un problema particular deben suministrar óptimas o casi-óptimas soluciones para la mayoría de las instancias reales.
- **Eficiencia:** los tiempos de computación que emplean, en suministrar óptimas o casi-óptimas soluciones, deben de ser moderados.
- **Robustez:** deben comportarse adecuadamente con instancias variadas.
- **Amigables con el usuario:** deben estar bien definidas, ser comprensibles y fáciles de usar.
- **Innovación:** preferiblemente deben llevar a nuevas aplicaciones.

Las técnicas metaheurísticas que gozan de más éxito tradicionalmente son:

- Algoritmos genéticos (Holland 1975).
- Temple simulado, Simulated Annealing, S.A., (Kirkpatrick Y otros 1983).
- Búsqueda Tabú, Tabu Search, T.S., (Glover 1989 ; 1990).
- Algoritmos Meméticos, (Moscato 1989).
- Greedy Randomized Adaptive Search Procedure, G.R.A.S.P., (Feo 1989 y 1995).
- Búsqueda Reactiva, (Battiti 1996).
- Colonia de Hormigas, A.C.O., (Dorigo 1996).
- Enjambre de Abejas, P.S.O., (Kennedy 1995).
- Concentración Heurística, (Rosing 1997, Rosing. y Revelle 1997).
- Búsqueda Dispersa, Scatter Search, S.S., (Glover 1998 , Glover 2002, Laguna 2002).
- Búsqueda Local Guiada, Guided Local Search, G.L.S., (Voudoris 1999).
- Búsqueda en Entorno Variable, Variable Neighborhood Search, V.N.S., (Mladenovic 1995 , Mladenovic 1997).

Otros heurísticos más novedosos que, poco a poco, van cobrando cierta relevancia y conforman el objeto del presente estudio, son los heurísticos gravitatorios entre los que cabe destacar:

- Space Gravitational Optimization, S.G.O., (Hsiao 2005).
- Central Force Optimization, C.F.O., (Formato 2007).
- Gravitational Search Algotihm, G.S.A., (Rashedi 2009).

3. METAHEURÍSTICAS TRADICIONALES RELACIONADAS CON S.G.O.

Con el fin de justificar por qué vamos a exponer solo alguna de las técnicas metaheurísticas anteriormente citadas vamos a establecer una clasificación basada en la manipulación de un solo individuo o de un conjunto de individuos[4] en cada paso del algoritmo (Mercado 2013). De acuerdo con ella podemos dividir los metaheurísticos en:

- **Técnicas basadas en trayectorias:** cuando en cada etapa se tiene en cuenta a un solo individuo.
- **Técnicas basadas en población:** si se considera en cada momento a la población completa.

S.G.O. pertenece a las técnicas basadas en población al igual que A.C.O., P.S.O., S.S., C.F.O. , G.S.A. y los Algoritmos Evolutivos Genéticos y Meméticos. Expondremos con detalle todos ellos -salvo los heurísticos evolutivos porque entendemos que las ideas que los inspiran se alejan demasiado del campo de S.G.O.- comenzando por los tradicionales.

3.1. COLONIA DE HORMIGAS

La metaheurística Optimización basada en Colonia de Hormigas es un algoritmo estocástico diseñado inicialmente en el trabajo de Dorigo, Maniezzo y Colorni (1994) que ha obtenido muy buenos resultados, tanto en el campo de la optimización combinatoria como en el de la optimización continua, y se inspira en el comportamiento que rige a las hormigas de diversas especies para encontrar los caminos más cortos entre las fuentes de comida y el hormiguero (Menéndez 2009). Mientras se mueven, las hormigas van depositando en el suelo una sustancia química denominada “feromona” que deja información sobre el estado fisiológico, reproductivo y social del individuo y que las otras hormigas pueden detectar y, de esta forma, seguir su rastro.

Cuando una hormiga aislada llega a una bifurcación de caminos y no detecta feromonas elige de forma aleatoria, lo que supone que en un principio la mitad de las hormigas tomarán una senda y la otra mitad la otra. Si asumimos que las hormigas viajan a velocidad constante tenemos que el camino más corto habrá sido transitado en el mismo tiempo por mayor número de hormigas por unidad de longitud, que el más largo. Por lo tanto tendrá depositada mayor densidad de feromona lo que hará que éste sea más deseable y por tanto la mayoría elegirá transitar por él. Al mismo tiempo, la evaporación de las feromonas en los caminos menos transitados provocará que los posteriores individuos dejen de viajar por ellos.

[4] Llamado población.

Los algoritmos A.C.O. se han aplicado tradicionalmente en problemas combinatorios y de grafos y se basan en una colonia de hormigas artificiales, agentes de búsqueda, que trabajan de forma conjunta y se comunican mediante rastros de feromona artificial (Alonso 2002). Se trata de un algoritmo constructivo en el que cada agente de búsqueda va creando nuevas soluciones, nuevas trayectorias, usando dos tipos de información:

- Información heurística: depende exclusivamente del problema y no se modifica, se traduce en la probabilidad asignada a cada posible camino, $\eta_{i,j}$.
- Información de los rastros de feromona: va variando según el número de agentes que escojan el camino, $\tau_{i,j}$.

En la figura 2.1, (tomada de Alonso y otros 2002) se muestra el esquema básico de A.C.O., al llegar a cada bifurcación los individuos toman la decisión de ir por un camino u otro usando la información del algoritmo.

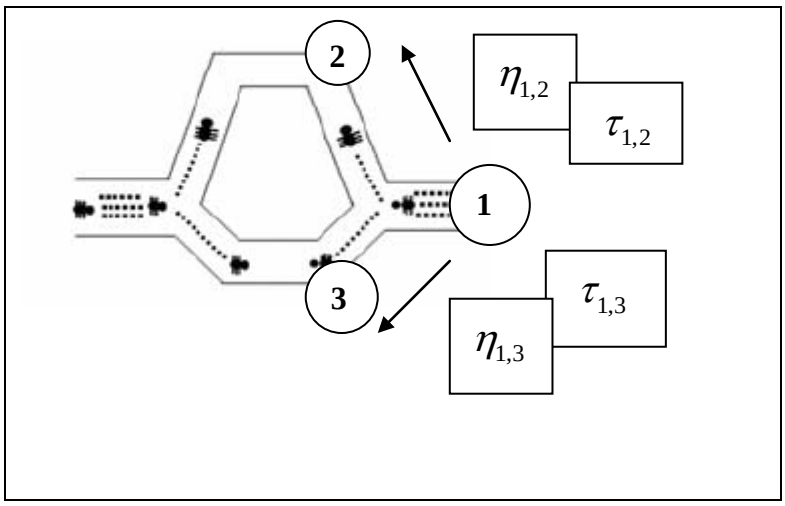


Figura 2.1. Esquema básico de A.C.O. (Alonso 2002).

Los requerimientos de memoria del algoritmo son altos, ya que cada agente debe ser capaz de reconstruir las trayectorias seguidas y la información en los nodos debe estar actualizada en todo momento.

Además es necesario penalizar rutas con parámetros de evaporación porque a veces el rastro de feromonas ocasiona que los agentes se “queden atascados” en óptimos locales y atraigan al resto de los individuos evitando que exploren nuevas zonas del espacio de soluciones.

La estructura básica de un A.C.O. es la siguiente (Alonso 2002):

```
1.-Procedimiento metaheurística A.C.O.
Inicialización de parámetros
Mientras (criterio de parada no satisfecho)
        Programación de actividades
        Generación de hormigas y actividad
        Evaporación de feromona
        Fin de programación actividades
Fin mientras
Fin procedimiento

2.-Procedimiento generación de hormigas y actividad
repetir en paralelo desde k = 1 hasta n (el número de hormigas)
        nueva hormiga
        fin repetir en paralelo
fin del procedimiento

3.- Procedimiento nueva hormiga
inicializa hormiga
L = actualiza memoria de hormiga
Mientras(estado actual ≠ estado objetivo)
        P=calcular probabilidades
        Aplicar política de decisión
        Mover al siguiente estado
        Depositar feromona en el arco visitado
        L= actualizar estado interno
Fin mientras
Liberar recursos de hormiga
Fin procedimiento
```

Figura 2.2. Esquema básico de A.C.O. (Alonso 2002).

La idea original se ha diversificado hasta rediseñar una herramienta capaz de resolver una amplia gama de problemas de optimización discreta y continua[5].

En la página web http://www.aco-metaheuristic.org/tutorials-courses.html puede encontrarse un amplio número de tutoriales sobre A.C.O.

3.2. ENJAMBRE DE ABEJAS

La metaheurística Optimización por Enjambre de Partículas, fue introducida por James Kennedy y Russel Eberhat (1995). Se inspira en el comportamiento social de individuos dentro de enjambres en la naturaleza, como abejas, bancos de peces o bandadas de aves en los que todo el enjambre se mueve con sincronía bajo el efecto de la inercia y de la atracción de los líderes más relevantes del grupo (Martínez 2007).

[5] Ver, por ejemplo, los resultados obtenidos por MIDACO-SOLVER http://www.midaco-solver.com/ en optimización continua.

En P.S.O. los miembros del enjambre se interpretan como agentes de búsqueda del espacio de soluciones. Cada individuo tiene asignados una posición y una velocidad, inicialmente aleatorias, que van variando en función de su velocidad anterior[6], de la mejor posición del individuo hasta ese momento[7] y de la mejor posición del líder de su entorno[8].

El entorno social de la partícula *i*-ésima, $N(i)$, es el conjunto de individuos que comparten información con ella. Puede, o no, ser todo el grupo y puede variarse a lo largo del desarrollo del algoritmo. En todos los casos se almacenan los valores de la función objetivo como función de adaptación, fitness.

La actualización de las velocidades y las posiciones en cada iteración se hace merced a las siguientes ecuaciones (Martínez 2007)

$$v_{j+1}(i) = c_1 v_j(i) + c_2 rnd\ (b(i) - x_j(i)) + c_3 rnd\ (g(i) - x_j(i))$$
$$1 \le i \le n \qquad \text{(número de individuos)}$$
$$1 \le j \le n\_iter \quad \text{(número de iteraciones)}$$

Figura 2.3. Ecuaciones de actualización de velocidades y posiciones en P.S.O.

donde c_1 representa la inercia del individuo, c_2 representa el grado de confianza de la partícula en sí misma y c_3 el grado de confianza de la partícula en su entorno social; $b(i)$ es la mejor posición obtenida por el individuo y $g(i)$ es la mejor posición obtenida por el entorno social del individuo. El parámetro *rnd* es un número aleatorio obtenido por la distribución uniforme $U(0,1)$ que representa el comportamiento impredecible que adoptan los miembros en el enjambre.

Una vez actualizada la velocidad se pasa a actualizar la posición de cada partícula de la siguiente forma (Martínez 2007 , Pérez 2005):

$$x_{j+1}(i) = x_j(i) + v_{j+1}(i)$$
$$1 \le i \le n \qquad \text{(número de individuos)}$$
$$1 \le j \le n\_iter \quad \text{(número de iteraciones)}$$

Figura 2.4. Ecuaciones de actualización de partículas en P.S.O.

[6] Lo que se denomina inercia.
[7] Que conforma la información del propio individuo.
[8] Que conforma la información conjunta.

He aquí el esquema de un algoritmo tipo P.S.O. (Pérez 2005)

Procedimiento metaheurística PSO
Inicialización aleatoria de las posiciones y velocidades de las partículas.
Inicialización de los parámetros c_1 , c_2 , c_3.
Mientras t < max_iter hacer
Para cada partícula i hacer
Calcular entorno de partícula;
Evaluar fitness
Actualizar mejor personal
Actualizar mejor del entorno
Calcular velocidad;
Calcular posición tras variación velocidad;
t:= t+1
fin mientras
fin procedimiento

Figura 2.5. Organigrama general de P.S.O. (Pérez 2005).

Aunque aún no hemos expuesto con detalle el algoritmo protagonista de nuestro trabajo, S.G.O., creemos interesante resaltar que P.S.O. es la heurística que más similitudes guarda con él de las vistas hasta el momento[9].

Sin embargo también son significativas las diferencias, ya que en P.S.O. la información compartida entre los individuos es fundamental para determinar las nuevas velocidades, con lo que el líder puede arrastrar al resto del enjambre a óptimos locales. En el caso de S.G.O. se usa el valor de la función objetivo en cada punto pero como aceleración propia en el punto, sin tener en cuenta al resto de los agentes, lo que conlleva que **los buenos resultados de la función objetivo nunca van a tener mayor peso específico que los malos en la construcción de las nuevas velocidades.**

En P.S.O. la información del líder es estadísticamente muy importante, en S.G.O. como veremos más adelante, no; ya que todos los asteroides tienen la misma importancia y la influencia entre ellos es muy pequeña[10].

Un tutorial sobre P.S.O. puede encontrarse en la página web http://www.swarmintelligence.org/tutorials.php

[9] Se lanzan agentes de búsqueda y se van modificando sus posiciones y velocidades.
[10] Si el parámetro α lo es.

3.3. BÚSQUEDA DISPERSA

Búsqueda dispersa fue introducido por primera vez en 1977 (Glover 1998) como un heurístico para la programación entera basado en una serie de estrategias expuestas en el congreso "Management Science and Engineering Management" celebrado en Austin, Texas, en septiembre de 1967. A pesar de ello no fue aplicado ni debatido hasta 1990, cuando fue presentado en el "EPFL Seminar on Operations Research and Artificial Intelligence Search Methods" en Lausanne (Switzerland) , (Glover 1994). Desde entonces la metodología de Scatter Search ha evolucionado mucho. Pueden encontrarse pormenorizados tutoriales en Glover (2002), Laguna (2002), Martí y otros (2011), en el que también puede encontrarse un tutorial de Path Relinking, y diversas páginas de internet como http://www.swarmintelligence.org/ (2006).

La primera descripción que se hizo pública de S.S. en Glover F. (1977) usa una sucesión de principios coordinados para generar soluciones. Las soluciones son generadas de forma determinista[11] teniendo en cuenta diversas características que debe tener una buena solución. Scatter Search orienta sus exploraciones sistemáticamente hacia un conjunto de puntos de referencia, que pueden ser por ejemplo los puntos extremos de un simplex.

Primero comienza identificando una combinación convexa, o centro ponderado de gravedad, de los puntos de referencia. Este punto central y los puntos iniciales de referencia son después utilizados para definir nuevas subregiones. Finalmente estos últimos puntos son “redondeados” para obtener las soluciones deseadas.

La versión de *Scatter Search* de 1977 se muestra en la figura 2.6. Cada uno de los puntos numerados desde 1 hasta 16 son los puntos centrales de claras sub-regiones del simplex A, B y C. El punto 8 es el centro del mismo (A, B, C). En este ejemplo A, B y C no constituyen los puntos originales de referencia (que pueden haber sido por ejemplo 6, 7 y 11 o 4, 5, 12 y 13). La elección depende generalmente de la distribución relativa entre ellos y de la región factible.

[11] No aleatoria.

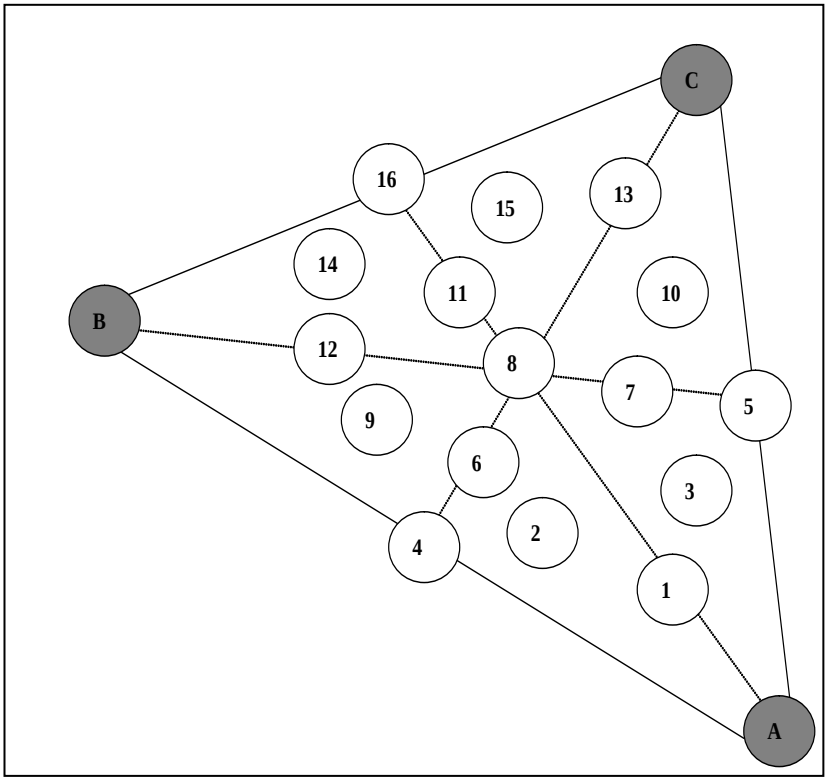


Figura 2.6. Representación de Scatter Search de 1977 (Laguna y Armentano 2002).

Del primer artículo sobre S.S., publicado por Glover en 1977, se pueden extraer los siguientes principios:

- S.S. realiza una exploración sistemática sobre una serie de buenas soluciones llamadas conjunto de referencia.
- El método se centra en combinar dos o más soluciones del conjunto de referencia. La combinación de más de dos soluciones tiene como objetivo generar centroides.
- Generar soluciones en la línea que unen dos dadas se considera una forma reducida del método.
- Al combinar se deben seleccionar pesos apropiados y no tomar valores al azar.
- Se deben realizar combinaciones “convexas” y “no convexas” de las soluciones.
- La distribución de los puntos se considera importante y deben tomarse dispersos.

Los conceptos y principios fundamentales de S.S. propuestos a comienzo de la década de los setenta conforman las estrategias para combinar soluciones así como restricciones.

De esta forma, se generaron nuevas soluciones mediante la creación de combinaciones numéricamente ponderadas de las soluciones ya existentes.

Esta técnica fue motivada por la suposición de que la información está contenida de diferentes formas en diferentes soluciones y, por ello, esta información sería explotada de un modo más efectivo de forma integrada, utilizando mecanismos de combinación de soluciones, que tratada aisladamente (Crowston 1963 ; Fisher 1963).

Cuando se combinan soluciones y se aplican mejoras sobre las mismas se obtienen mejores resultados que cuando se aplican métodos de mejora en las soluciones sin haberlas combinado previamente (Alegre 2004).

Esta idea de recombinación de soluciones muy buenas también aparece en otros algoritmos como Reencadenamiento de Trayectorias, Path Relinking. También inspira Very Simple Optimization, V.S.O., que detallaremos posteriormente.

Un posible esquema para S.S. puede ser el siguiente:

Algoritmo Scatter Search Básico
Generar un conjunto inicial de soluciones diversas
A partir de estas soluciones obtener RefSet inicial
Repetir
    Generación de subconjuntos de RefSet
    Obtener nuevas soluciones a partir de dichos subconjuntos
    Procedimiento de mejora de las soluciones.
    Actualizar RefSet con las nuevas soluciones
hasta que RefSet se estabilice (i.e. no haya nuevas soluciones)

Figura 2.7. Organigrama de S.S. (Alegre 2004).

4. LA HEURÍSTICA OPTIMIZACIÓN GRAVITATORIA , S.G.O.

En líneas generales, la heurística propuesta en S.G.O. se basa en considerar que la función objetivo puede asimilarse a una "tela de araña", analogía inspirada por la mecánica relativista, sobre la que se lanzan una serie de agentes de búsqueda que se van desplazando sobre ella, merced a las leyes de la mecánica clásica, buscando el óptimo global.

Un algoritmo así diseñado requiere poca memoria y poco poder computacional con lo que se pueden abordar problemas con equipos pequeños en tiempo razonable.

Para el estudio de dicho algoritmo, que está recogido en el artículo "A Novel Optimization Algorithm: Space Gravitacional Optimization" (Hsiao 2005), hemos seguido las siguientes etapas:

- Revisión de algunos conceptos de la mecánica clásica y la mecánica relativista.
- Análisis detallado del algoritmo "Optimización Gravitatoria".

Las representaciones gráficas de las funciones y de las simulaciones han sido hechas en su totalidad con MATLAB R2014.

4.1. CAMPOS DE FUERZAS, GENERALIDADES

En física general se dice que en una región del espacio existe un **campo de fuerzas** cuando, por el hecho de situar un cuerpo en cualquiera de sus puntos, aparece sometido a una fuerza (Catalá de Alemany 1966). En el campo gravitatorio terrestre, dicha fuerza se produce por la interacción de las masas mientras que en los campos magnéticos la fuerza se produce por interacción de las cargas.

La fuerza ejercida sobre un cuerpo en cada punto del campo es distinta y depende, tanto de la posición que ocupa el cuerpo dentro del campo como del propio cuerpo. Así podemos decir que

$$\vec{F} = A\,\vec{I}$$

Figura 2.8. Ley de la fuerza ejercida sobre un cuerpo.

donde A es un coeficiente dependiente del cuerpo que llamaremos su **magnitud activa**, la masa, la carga, etc., e $\vec{I}$ es una magnitud vectorial que depende exclusivamente del campo y que se denomina **intensidad de campo**[12]**,** que dependerá de la posición del

[12] El concepto intensidad de campo sólo depende del campo y existe (matemáticamente) sin que haya en él partícula alguna sobre la que ejercer la fuerza, sin embargo, dado que físicamente sólo hay manifestación de la fuerza si hay partícula sobre la que actúa, asumiremos a lo largo de toda la sección la idea de campo actuando sobre un cuerpo concreto.

cuerpo y de una constante asociada al campo, la constante de la gravitación universal o la constante de Coulomb, etc..

Un campo de fuerzas se representa por las llamadas **líneas de fuerza**, que son líneas virtuales trazadas en su interior y que dan una imagen gráfica de la forma del campo. Pueden definirse como aquellas líneas que en todos sus puntos son tangentes a la dirección del campo.

Supongamos que un cuerpo se traslada dentro del campo de fuerzas desde la posición 1 a la posición 2 siguiendo una trayectoria $\vec{\alpha}$ definida en un intervalo $[a,b]$.

$$\begin{aligned}\vec{\alpha}:[a,b] &\to \mathbb{R}^n \\ t &\to \vec{\alpha}(t)\end{aligned}$$

Se define **el trabajo realizado por la fuerza sobre el cuerpo a lo largo de α**[13] a la integral de línea (Apóstol 1980)

$$W = \int \vec{F}\, d\vec{\alpha} = \int_a^b \vec{F}[\vec{\alpha}(t)] \cdot \vec{\alpha}'(t)dt$$

Figura 2.9. Definición del trabajo realizado por una fuerza a lo largo de una trayectoria.

Un campo de fuerzas es **conservativo** si el trabajo realizado por la fuerza del campo sobre cualquier cuerpo depende únicamente de las posiciones inicial y final del cuerpo y no de la trayectoria, $\vec{\alpha}$, recorrida por él.

Matemáticamente, un campo de fuerzas se representa como un campo vectorial

$$\begin{aligned}\vec{F}:\mathbb{R}^n &\to \mathbb{R}^m \\ \vec{x} &\to \vec{F}(\vec{x}) = \vec{y}\end{aligned}$$

y se demuestra que **un campo vectorial continuo es conservativo si y sólo si es el gradiente de un campo escalar** (Apóstol 1980).

O lo que es lo mismo

$$\exists\ E:\mathbb{R}^n \to \mathbb{R} \quad / \quad \vec{F} = \nabla E$$

[13] Trayectoria suficientemente regular.

El campo escalar, E, recibe el nombre de **función potencial para el campo** $\vec{F}$. En el caso particular del espacio de tres dimensiones, cada lugar geométrico de los puntos que tienen la misma función potencial se denomina **superficie equipotencial.** En este caso las líneas de fuerza del campo y las **superficies equipotenciales** son siempre ortogonales.

4.2. CAMPO GRAVITATORIO EN FÍSICA NEWTONIANA

Con el descubrimiento de las leyes de la dinámica, Newton consiguió explicar de manera sistemática y unificada todos los fenómenos físicos en los que intervienen las fuerzas y el movimiento, en particular en la **segunda de ellas** expone cómo varía la velocidad de un cuerpo en función de la fuerza a la que está sometido.

$$\vec{F} = m\frac{d\vec{v}}{dt}$$

Figura 2.10. Segunda Ley de la Dinámica de Newton .

donde m es la masa del cuerpo, $\vec{F}$ es la fuerza que se ejerce sobre él y $\vec{v}$ es su velocidad.

Por otra parte, en su Ley de la Gravitación Universal establece de qué factores dependen las fuerzas que intervienen en el universo: “Cualquier cuerpo de masa m_1 en el universo atrae a cualquier otro cuerpo de masa m_2 con una fuerza que es proporcional a sus masas, con constante de proporcionalidad $G = 6.67 \cdot 10^{-11} Nm^2 Kg^{-2}$, e inversamente proporcional al cuadrado de la distancia que los separa.

$$\left|\vec{F}\right| = G\frac{m_1 \cdot m_2}{r^2}$$

Figura 2.11. Ley de la Gravitación Universal de Newton.

La dirección de la fuerza es la línea recta que une ambos cuerpos y su sentido siempre es atractivo.

Las fórmulas 2.10 y 2.11 **nos permiten simular numéricamente** la trayectoria que seguirá un planeta que orbite alrededor del sol. Veámoslo en un caso muy sencillo en dimensión 2 (Feynman 1977).

Supongamos que tenemos el sol y el planeta en un plano. En el punto $(0,0)$ ubicamos al sol y el planeta ocupa la posición (x,y). La fuerza de atracción, $\vec{F}$,puede descomponerse en sus proyecciones ortogonales como se muestra en la figura 2.12

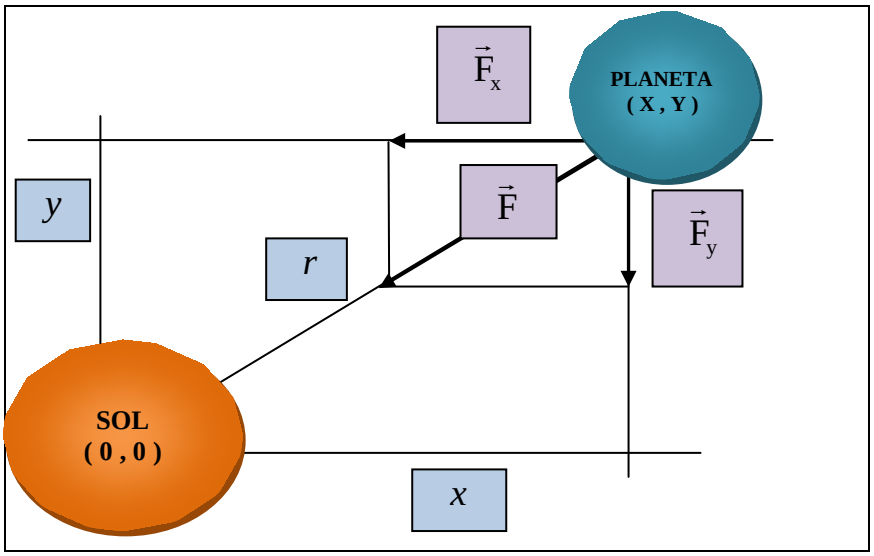


Figura 2.12. Simulación numérica de la trayectoria de un planeta alrededor del sol.

Por semejanza de triángulos, tenemos que si M es la masa del sol, m la del planeta y r la distancia que los separa

$$\frac{|\vec{F}_x|}{|\vec{F}|} = \frac{x}{r} \Rightarrow |\vec{F}_x| = \frac{|\vec{F}|\,x}{r} = \frac{GMmx}{r^3} \Rightarrow \vec{F}_x = \left(-\frac{GMmx}{r^3}, 0\right)$$
$$r = \sqrt{x^2 + y^2}$$

Figura 2.13. Descomposición del vector fuerza en sus componentes ortogonales (I).

y análogamente para la dirección vertical

$$\frac{|\vec{F}_y|}{|\vec{F}|} = \frac{y}{r} \Rightarrow |\vec{F}_y| = \frac{|\vec{F}|\,y}{r} = \frac{GMmy}{r^3} \Rightarrow \vec{F}_y = \left(0, -\frac{GMmy}{r^3}\right)$$
$$r = \sqrt{x^2 + y^2}$$

Figura 2.14. Descomposición del vector fuerza en sus componentes ortogonales (II).

Según la ecuación 2.10 considerando $\vec{v} = \left(v_x, v_y\right)$

$$\vec{F}_x = \left(m \frac{dv_x}{dt}, 0 \right) \qquad \vec{F}_y = \left(0, m \frac{dv_y}{dt} \right)$$

Figura 2.15. Descomposición del vector fuerza en sus componentes ortogonales (III).

Igualando las ecuaciones 2.13, 2.14 y 2.15 tenemos

$$a_x = \frac{dv_x}{dt} = \frac{-GMx}{r^3} \qquad a_y = \frac{dv_y}{dt} = \frac{-GMy}{r^3}$$

Figura 2.16. Ecuación para obtener la aceleración en función de la posición (I).

Es decir

$$\vec{a} = \left(\frac{-GMx}{r^3}, \frac{-GMy}{r^3} \right)$$

Figura 2.17. Ecuación para obtener la aceleración en función de la posición (II)

luego **podemos conocer la aceleración del planeta a partir de su posición**.

De este modo, **si partimos de una posición y una velocidad** del planeta conocidas en el instante t_0, $\left(x(t_0), y(t_0)\right)$ **podemos aproximar su aceleración** usando 2.17, lo que nos permite **aproximar su velocidad y posición en el instante** $t_0 + \varepsilon$ usando la fórmula 2.18, que no es más que el desarrollo de Taylor de orden 1 de las funciones velocidad y posición en función del tiempo.

$$\begin{aligned} &(v_x(t_0+\varepsilon), v_y(t_0+\varepsilon)) \approx (v_x(t_0), v_y(t_0)) + (a_x(t_0), a_y(t_0)) \cdot \varepsilon \\ &(x(t_0+\varepsilon), y(t_0+\varepsilon)) \approx (x(t_0), y(t_0)) + (v_x(t_0), v_y(t_0)) \cdot \varepsilon \end{aligned}$$

Figura 2.18. Cálculo de la posición y velocidad del planeta en el instante siguiente.

De esta forma **podemos construir numéricamente una aproximación a la trayectoria que seguirá el planeta**, la cual será, obviamente, diferente según su posición y velocidad iniciales[14].

Si observamos la ecuación 2.17 vemos que la aceleración del planeta es directamente proporcional a su posición, e inversamente proporcional al cubo de su distancia al sol. Por tanto, a medida que su trayectoria se acerca a la posición solar su aceleración se incrementa, con orden cuadrático, y es, por ello, inmediatamente expulsado.

Desde el punto de vista matemático la sencilla simulación anterior resuelve, usando el método numérico de Euler, el sistema de ecuaciones diferenciales de primer orden[15] con valores iniciales (Guzman 1975)

$$\left.\begin{aligned} x'(t) &= v_x(t) \\ y'(t) &= v_y(t) \\ x''(t) = v_x'(t) &= \frac{-GMx(t)}{\sqrt{\left(x(t)^2 + y(t)^2\right)^3}} \\ y''(t) = v_y'(t) &= \frac{-GMy(t)}{\sqrt{\left(x(t)^2 + y(t)^2\right)^3}} \end{aligned}\right\}$$

$$\left(x(t_o), y(t_0)\right) = (x_0, y_0) \qquad \left(v_x(t_0), v_y(t_0)\right) = (v_{x0}, v_{y0})$$

Figura 2.19. Sistema de ecuaciones diferenciales de primer orden.

que es autónomo[16]. La información cualitativa sobre el comportamiento general de las soluciones de sistemas autónomos de funciones suficientemente regulares puede encontrarse en muchos tratados sobre ecuaciones diferenciales, por ejemplo en Simmons (1977). Así, las posibles soluciones de 2.19, en nuestro caso las posibles trayectorias del planeta, son: o bien equilibrios del sistema, trayectorias constantes; o bien funciones periódicas, en posición y velocidad, o bien funciones no periódicas regulares, i.e. con derivada nunca nula, lo que en nuestro caso implica cómo serán las posibles trayectorias del planeta dependiendo de las condiciones iniciales.

[14] No tendremos en cuenta las fuerzas derivadas de los choques que se producirían en el caso de que el planeta cayera sobre el sol.

[15] El sistema es de primer orden en las variables $\left(x(t),\ y(t), v_x(t), v_y(t)\right)$.

[16] No depende de la variable independiente, t.

- O bien el planeta ocupa en algún momento la posición del sol con velocidad cero, en cuyo caso siempre ha estado ahí, ya que, si su posición y velocidad son nulas en algún momento necesariamente son nulas siempre.
- O bien su velocidad y posición son periódicas, es decir, el planeta se mueve a lo largo del tiempo trazando una única órbita alrededor del sol.
- O bien el vector tetradimensional formado por la posición y la velocidad en cada instante no se repite para ningún t. En este caso la velocidad y aceleración del planeta no pueden anularse al mismo tiempo, lo que implica que el planeta no puede quedar atrapado en la posición del sol.[17]

En las siguientes figuras podemos ver algunas de las posibles trayectorias de la simulación dependiendo de los valores iniciales aportados.[18] En todos los casos el sol ocupa la posición $(0,0)$.

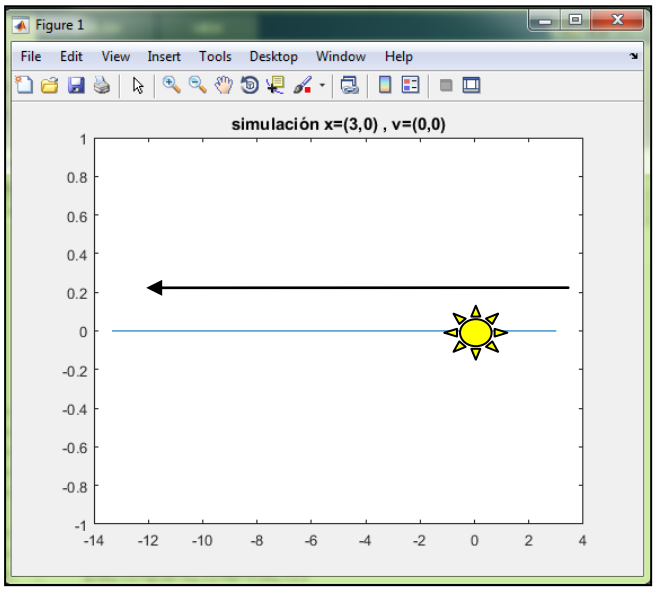


Figura 2.20. Simulación con MATLAB con valores iniciales $\vec{x}=(3,0)$, $\vec{v}=(0,0)$ el planeta atraviesa el sol y sigue su trayectoria rectilínea.

[17] Aunque matemáticamente sí puede atravesarlo.
[18] Representaciones realizadas con MATLAB.

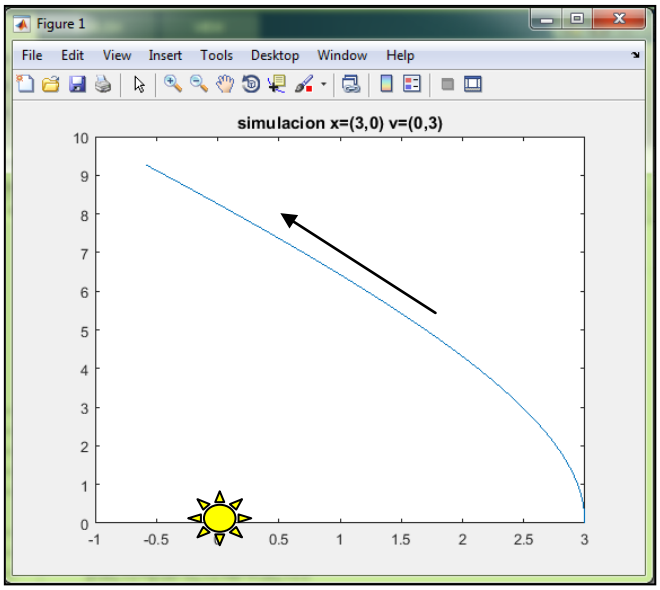


Figura 2.21. Simulación con MATLAB con valores iniciales $\vec{x} = (3,0)$, $\vec{v} = (0,3)$

el planeta curva un poco su trayectoria y luego sigue su trayectoria rectilínea.

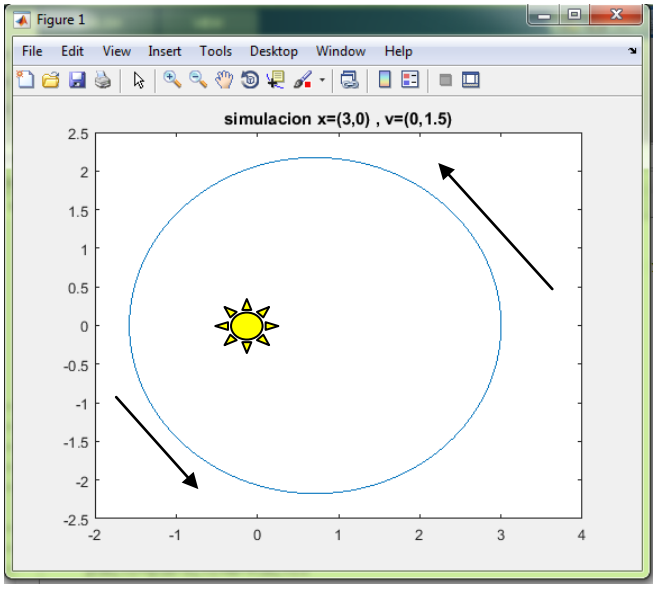


Figura 2.22. Simulación con MATLAB con valores iniciales $\vec{x} = (3,0)$, $\vec{v} = (0,1.5)$

el planeta describe una trayectoria periódica alrededor del sol.

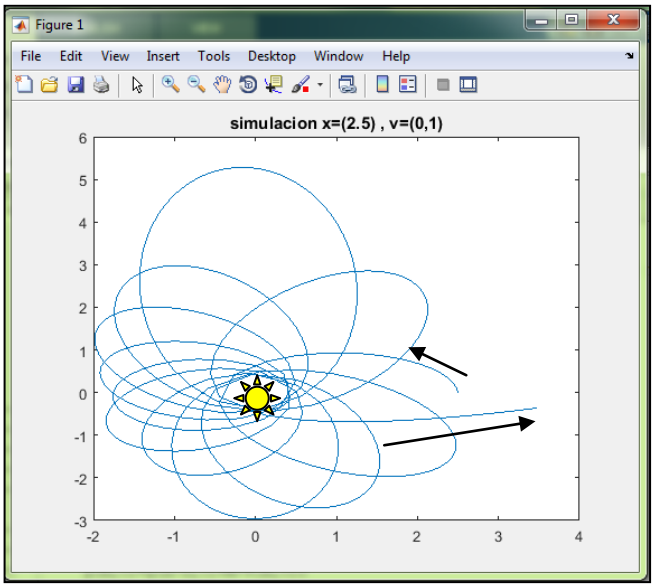


Figura 2.23. Simulación con MATLAB con valores iniciales $\vec{x} = (2.5, 0)$, $\vec{v} = (0,1)$

El planeta orbita durante un tiempo y después se escapa.

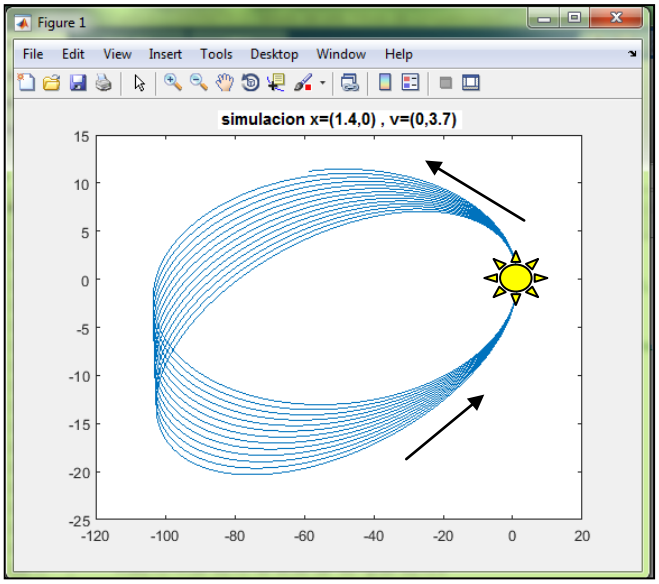


Figura 2.24. Simulación con MATLAB con valores iniciales $\vec{x} = (1.4, 0)$, $\vec{v} = (0, 3.7)$

El planeta mantiene una órbita periódica alrededor del sol.

La simulación planteada anteriormente conforma el "esqueleto" de la heurística S.G.O. donde los asteroides se asimilan al planeta y el sol se asimila a los óptimos locales o al óptimo global.

Para explicar cómo interviene la función objetivo nos interesa profundizar en el concepto de campo conservativo, lo que haremos a continuación.

El hecho de que **la aceleración del cuerpo sólo dependa de su posición en cada momento** nos lleva intuitivamente a suponer la idea de que el campo gravitatorio descrito por Newton es un **campo vectorial conservativo** y efectivamente así es. (Feynman 1977). Por tanto existirá un campo escalar, $U(x,y)$, que será su función potencial y se obtendrá integrando las componentes de $\vec{\mathrm{F}}$ respecto a x e y respectivamente. Por lo tanto, salvo constante, tenemos[19]

$$\exists U:\mathbb{R}^2\to\mathbb{R} \text{ suficientemente regular } /$$

$$-\nabla U(x,y)=-\left(\frac{\partial \mathrm{U}}{\partial \mathrm{x}},\frac{\partial \mathrm{U}}{\partial y}\right)\Big|_{(\mathrm{x,y})}=\vec{\mathrm{F}}_{\mathrm{x}}+\vec{\mathrm{F}}_{\mathrm{y}}=\left(\frac{-GMmx}{\sqrt{\left(x^2+y^2\right)^3}},\frac{-GMmy}{\sqrt{\left(x^2+y^2\right)^3}}\right)\Rightarrow$$

$$\Rightarrow U(x,y)=-\frac{GMm}{\sqrt{x^2+y^2}}+Cte=-\frac{GMm}{r}+Cte$$

Figura 2.25. Ecuación de energía potencial de un campo conservativo.

$U(x,y)$ recibe el nombre de **energía potencial** (Feynman 1977) y sólo depende de la posición que el cuerpo ocupa dentro del campo**.**

Si conocemos la energía potencial de un campo podemos realizar nuevamente la simulación numérica anterior para conocer la trayectoria que seguirá el planeta alrededor del sol ya que según la ecuación 2.16

$$a_{\mathrm{x}}=\frac{-GMx}{\sqrt{\left(x^2+y^2\right)^3}}=-\frac{1}{m}\frac{\partial U}{\partial x}\quad a_{\mathrm{y}}=\frac{-GMy}{\sqrt{\left(x^2+y^2\right)^3}}=-\frac{1}{m}\frac{\partial U}{\partial y}\Rightarrow$$

$$\Rightarrow \vec{a}=\left(-\frac{1}{m}\frac{\partial U}{\partial x},-\frac{1}{m}\frac{\partial U}{\partial y}\right)$$

Figura 2.26. Aceleración de una partícula en función del potencial de campo.

[19] El signo negativo en 2.25 no tiene sentido matemático, sino físico y se debe a que las fuerzas gravitatorias siempre son atractivas.

Luego, si partimos de una posición conocida del planeta en el instante *t*, $\left(x(t), y(t)\right)$ podemos aproximar su aceleración usando 2.25, lo que nos permitirá calcular su velocidad y posición en el instante $t+\varepsilon$ usando 2.18 como anteriormente.

La heurística S.G.O. realiza esencialmente esta simulación para calcular las trayectorias de los asteroides en un campo gravitatorio donde la energía potencial se asimila a la función objetivo.

Dado que en la heurística S.G.O. la función $U(x, y)$ cambia respecto a la considerada en la dinámica de Newton, también cambiará la expresión de la aceleración dada en 2.17 y en último término el sistema 2.19 pero, al no perder su cualidad de sistema autónomo, los resultados sobre las posibles trayectorias de los asteroides, dependiendo de sus condiciones iniciales, siguen siendo ciertas, es decir

- O bien los asteroides mantienen trayectorias constantes en puntos de equilibrio[20], que pueden ser varios según sea la función objetivo.
- O bien los asteroides mantienen trayectorias periódicas en torno a algún equilibrio.
- O bien los asteroides describen trayectorias cuyo límite respecto al tiempo es algún punto de equilibrio.
- O bien son expulsados del correspondiente punto de equilibrio tras dar, quizá, unas cuantas vueltas alrededor de él.

Intentar dar una expresión explícita de las trayectorias que los asteroides seguirán en cada caso dependiendo de cada posible función objetivo es tarea que sale fuera de los objetivos del presente trabajo. No olvidemos que S.G.O. es un heurístico inspirado en estas técnicas. Por ello, **nuestro objetivo ha sido mostrar que su inspiración tiene una base consistente desde el punto de vista físico y matemático**; pero hasta ahí. Cualquier intento de analizar y demostrar matemáticamente el comportamiento de las trayectorias que los agentes de búsqueda seguirán en función del tiempo es, desde nuestro punto de vista, claramente inabarcable.

4.3. CAMPO GRAVITATORIO EN FÍSICA RELATIVISTA

Por la formulación de la física clásica, la fuerza gravitatoria se desarrolla en un campo plano y tiene acción instantánea, i.e. velocidad infinita. Sin embargo, esta idea, problemática hasta para el propio Newton (Newton 1687, 1987), fue definitivamente rechazada con el advenimiento de la física relativista ya que Einstein postuló que existe

[20] Óptimos locales o el óptimo global.

una constante, la velocidad de la luz, que no puede ser superada por ninguna partícula o interacción (Einstein 1915). Por ello se acepta el modelo físico derivado de la física relativista que expone que **el campo no permanece plano sino que cuando se origina sufre una modificación en su geometría, se curva**. Dicha curvatura, que será mayor cuanto mayor sea la masa del cuerpo que lo origina, conlleva un trabajo que queda acumulado en el campo en forma de energía[21] que dará lugar al potencial que origina la fuerza del campo sobre cada partícula (Catalá de Alemany, 1966). Por lo tanto, en la Teoría de la Relatividad General de Einstein el campo gravitatorio no se describe a través de sus líneas de fuerza en un espacio plano sino a partir de la deformación en la curvatura del espacio que se produce como consecuencia de la presencia de las distintas masas. Un símil que se adapta perfectamente a esta idea es el del espacio visto como una "tela de araña" por la que se mueven distintas bolas que pueden representarse, en dos dimensiones, como en la figura 2.27.

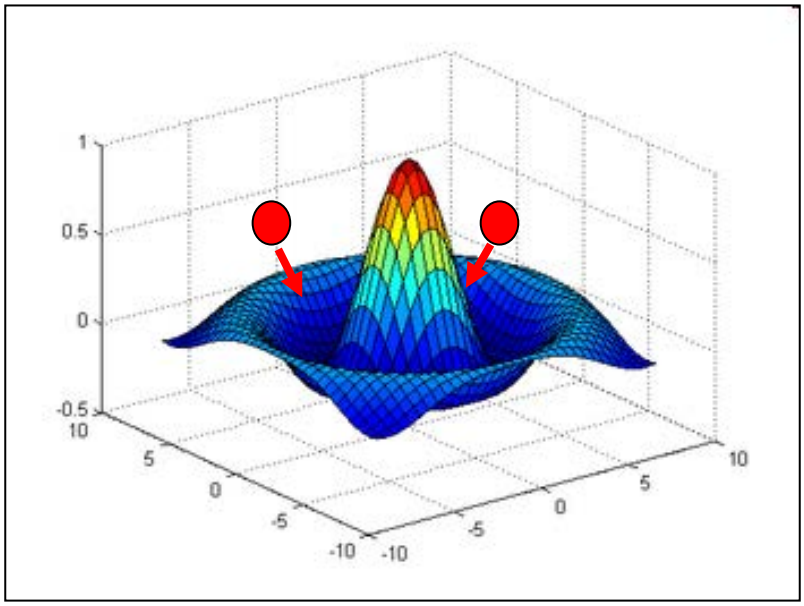


Figura 2.27. Representación de un campo gravitatorio a partir de la Relatividad General.

Además, también se considera el espacio y el tiempo como magnitudes que conforman una única superficie de cuatro dimensiones con su geometría específica[22] llamada espacio-tiempo cuyas propiedades y deformaciones dependen del estado del universo (Ludvigsen 1999).

[21] Energía de deformación.
[22] Evidentemente la geometría espacial y la temporal son diferentes.

Entre los múltiples resultados derivados de la teoría de la relatividad general de Einstein uno nos interesa especialmente por su aplicación a S.G.O. y consiste en **el modelo de universo en expansión o en contracción**. Según la dinámica relativista nuestro universo no permanece estático sino que se expande o se contrae, lo que modifica las posiciones y velocidades de los cuerpos inmersos en él, dependiendo del momento correspondiente. Los distintos resultados experimentales obtenidos en la actualidad, por ejemplo por el telescopio robótico Hubble, parecen ser consistentes con esta teoría.

4.4. ANÁLISIS DETALLADO DEL ALGORITMO "OPTIMIZACIÓN GRAVITATORIA"

4.4.1.INTRODUCCIÓN

A lo largo de las secciones anteriores hemos mostrado la idea que, a nuestro juicio, subyace bajo la heurística S.G.O., así como los fundamentos físicos en los que se inspira. En la presente sección haremos un estudio pormenorizado del algoritmo, exponiendo detalladamente todos sus pasos y los parámetros que considera.

Por simplicidad de notación, inicialmente se plantea el algoritmo para resolver el problema de hallar el óptimo global de una función real de variable vectorial de 2 dimensiones con restricciones.

El caso de n dimensiones es fácilmente extrapolable a partir del descrito ya que se trata solamente de considerar más componentes en los vectores aceleración, velocidad y posición. El gradiente de $f(\vec{x})$ siempre va a tener tantas componentes como variables tenga $f(\vec{x})$.

4.4.2.DESCRIPCIÓN GENERAL DEL ALGORITMO

El problema planteado es

$$\begin{array}{|ll|}\hline \text{Opt} & f(\vec{x}) \\ s.a. & \vec{x} \in X \subset \mathbb{R}^2 \\ \hline \end{array}$$

Figura 2.28. Problema general de optimización continua

Como ya hemos comentado, el algoritmo Optimización Gravitatoria se inspira en una simulación de varios asteroides moviéndose en un universo curvado conforme a la función objetivo, $f(\vec{x})$, en busca del cuerpo de mayor masa, el mínimo global de la función objetivo.

Tal como postula la teoría general de la relatividad de Einstein, la presencia de grandes masas, los mínimos locales y globales de $f(\vec{x})$, curva considerablemente la geometría del espacio-tiempo. Así el asteroide se acercará hacia la masa pesada bajo cuya influencia se encuentre describiendo la curva correspondiente. Existe alguna posibilidad de que el asteroide se mantenga orbitando alrededor de dicha masa pesada[23], pero, como vimos en la sección 4.2, también existe la posibilidad de que el asteroide, tras quizá varias vueltas alrededor de la masa pesada, sea expulsado; con lo que podrá seguir buscando otra masa diferente dentro del universo que lo atraiga de nuevo. **Ésta es la característica más importante del algoritmo S.G.O. ya que los agentes de búsqueda no son capturados, en general, en óptimos locales, por lo que no dejan de buscar nuevas soluciones.**

4.4.3.INICIALIZACIÓN DEL ALGORITMO

Un grupo de *n* asteroides es inicializado en *n* posiciones elegidas **al azar** dentro de la región factible, por sencillez en la implementación consideraremos que es el instante $t = 1$.[24]

$$\left[x_1(i) \, , y_1(i) \right] \quad / \quad 1 \le i \le n$$

Figura 2.29. Inicialización de posiciones.

donde $x_1(i) \, , y_1(i)$ representan las posiciones iniciales en los ejes x e y respectivamente, *i* denota el asteroide que ocupa el lugar *i*-ésimo en el grupo.

Análogamente se inicializan las velocidades y aceleraciones de cada asteroide usando las notaciones

$$\left[vx_1(i) \, , \mathrm{v}y_1(i) \right] \quad / \quad 1 \le i \le n$$

Figura 2.30. Inicialización de velocidades.

$$\left[ax_0(i) \, , \mathrm{a}y_0(i) \right] \quad / \quad 1 \le i \le n$$

Figura 2.31. Inicialización de aceleraciones.

[23] Entrada en ciclos infinitos.
[24] Matlab solo trabaja con vectores y matrices desde el índice 1.

donde las velocidades son nuevamente elegidas al azar dentro de la región factible y las aceleraciones son asignadas al valor inicial 0, es decir

$$\begin{aligned} ax_0(i) &= 0 \quad \forall i = 1,...,n \\ ay_0(i) &= 0 \quad \forall i = 1,...,n \end{aligned}$$

Figura 2.32. Aceleraciones iniciales nulas.

La razón para inicializar las aceleraciones con el subíndice $j = 0$, j indica el número de iteración, es que las aceleraciones en el momento $j = 1$ van a ser calculadas a partir de las posiciones definidas en el momento $j = 1$. Con las aceleraciones calculadas en el instante $j = 1$ calcularemos las velocidades en el instante $j = 2$ y con éstas calcularemos las posiciones en el instante $j = 3$. Las posiciones en el instante $j = 2$ se calcularán a partir de las velocidades en $j = 1$. El resto de los cálculos se hará sucesivamente.

4.4.4.DESARROLLO DEL ALGORITMO

De forma análoga a la desarrollada en la simulación de la sección 4.2, la nueva aceleración dependerá del valor de las derivadas parciales de la función objetivo en el punto que el asteroide ocupe en ese instante.

Y además, la dirección del vector aceleración del asteroide en cada instante coincidirá con la dirección del vector gradiente de la función $f(x)$, sus sentidos serán opuestos y su módulo será proporcional, con cierta constante de proporcionalidad, G, a dicho vector.

$$a(x,y) = -G\,\nabla f(x,y) = -G\left(\partial_x f(x,y), \partial_y f(x,y)\right)$$

Figura 2.33. Cálculo de la aceleración en función de la función objetivo.

Para obtener el valor del vector gradiente en cada punto podemos actuar en dos direcciones:

- Si conocemos explícitamente la función objetivo y el número de variables es pequeño, podemos hallar sus derivadas parciales y evaluarlas en el punto correspondiente.

- Si la función objetivo es una "caja negra" o el número de variables es muy grande, podemos optar por **aproximar numéricamente el gradiente en el punto**, para lo cual aproximaremos $\partial_x f(x)$ realizando un promedio entre los valores de $f(x)$ una cantidad infinitesimal antes y una cantidad infinitesimal después de la coordenada x del punto. Análogamente para aproximar $\partial_y f(x)$ realizando el correspondiente promedio al variar infinitesimalmente la coordenada y (Burden-Faires 1985). Fijaremos la cantidad infinitesimal en ambos casos, lo denominaremos radio de detección y lo denotaremos por r_d. De esta forma obtendremos las ecuaciones

$$ax_j(i) = G\left\{\frac{\left(f(x_j(i), y_j(i)) - f(x_j(i) + r_d, y_j(i))\right)}{2r_d} - \frac{\left(f(x_j(i), y_j(i)) - f(x_j(i) - r_d, y_j(i))\right)}{2r_d}\right\}$$

$$ay_j(i) = G\left\{\frac{\left(f(x_j(i), y_j(i)) - f(x_j(i), y_j(i) + r_d)\right)}{2r_d} - \frac{\left(f(x_j(i), y_j(i)) - f(x_j(i), y_j(i) - r_d)\right)}{2r_d}\right\}$$

$1 \le i \le n$ (indica el asteroide del grupo)
$1 \le j \le n\_iter$ (indica el movimiento j-esimo del asteroide)

Figura 2.34. Aproximación numérica de las aceleraciones (I).

Con lo que, tras operar queda

$$ax_j(i) = G\left\{\left(f(x_j(i) - r_d, y_j(i))\right) - \left(f(x_j(i) + r_d, y_j(i))\right)\right\}$$

$$ay_j(i) = G\left\{\left(f(x_j(i), y_j(i) - r_d)\right) - \left(f(x_j(i), y_j(i) + r_d)\right)\right\}$$

$1 \le i \le n$ (indica el asteroide del grupo)
$1 \le j \le n\_iter$ (indica el movimiento j-esimo del asteroide)

Figura 2.35. Aproximación numérica de las aceleraciones (II).

En este caso *G* no sólo va a depender de la función objetivo sino también de r_d, puesto que hemos incluido dicho cociente en el valor de la constante G. Esto no supone restricción alguna ya que es un parámetro que modificaremos para obtener los resultados más satisfactorios.

Una vez calculada la aceleración en la iteración *j*-ésima a partir de la posición en el mismo instante, ya se puede calcular la velocidad en la iteración *j+1*-ésima y la posición en la iteración *j+2*-ésima[25], de esta forma

$$vx_{j+1}(i) = vx_j(i) + ax_j(i)$$
$$vy_{j+1}(i) = vy_j(i) + ay_j(i) \qquad 1 \le i \le n\ ,\ 1 \le j \le n\_iter$$

Figura 2.36. Actualización de las velocidades.

$$x_{j+2}(i) = x_{j+1}(i) + vx_{j+1}(i)$$
$$y_{j+2}(i) = y_{j+1}(i) + vy_{j+1}(i) \qquad 1 \le i \le n\ ,\ 1 \le j \le n\_iter$$

Figura 2.37. Actualización de las posiciones.

ya que suponemos que de cada iteración a la siguiente pasa una unidad de tiempo, $t = 1$, y en ese caso la velocidad / posición en el instante siguiente coincide con la velocidad / posición en el instante actual más la aceleración / velocidad en el instante siguiente.

En el ejemplo de la figura 2.38 observamos la tendencia hacia posiciones con menor valor de la función objetivo, que en este caso es $f(x, y) = 4y + 2x - 7$. Los asteriscos representan diversos puntos[26]. A su lado recogemos el valor en cada uno de ellos de la función objetivo.

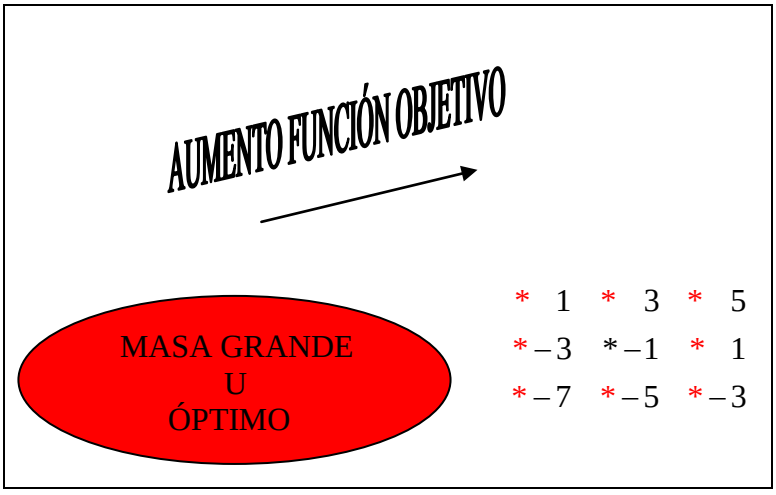


Figura 2.38. Tendencia de los asteroides hacia posiciones con menor potencial.

[25] Como ya hemos visto al inicializar los valores, la aceleración en el instante *j* se utiliza para calcular la velocidad en el instante *j*+1 y, por tanto, la posición en el instante *j+2*, no en el instante *j+1*.
[26] La malla de puntos abarca desde (0,0) en el ángulo inferior izquierdo hasta (2,2) en el superior derecho con paso 1.

Suponiendo $G=1$ y $r_d=1$, el cálculo de la aceleración en el punto central, $(1,1)$, se realizaría así: $(-3-1,-5-3)=(-4,-8)$. De este modo si la velocidad de partida fuese, por ejemplo $v_j=(10,5)$ pasaría a ser $v_{j+1}=(6,-3)$.

4.4.5.ACTUALIZACIÓN DEL ÓPTIMO Y CRITERIO DE PARADA

Tras cada actualización de posiciones comprobaremos si el correspondiente asteroide del grupo permanece o no en la región factible. Si no es así, reinicializaremos aleatoriamente su posición y velocidad tal y como hicimos al inicializar posiciones y velocidades.

Si la solución factible obtenida por el asteroide *i* es mejor que el valor óptimo obtenido hasta el momento, s^*, entonces se actualiza s* de la siguiente forma:

$$\begin{array}{|l} \textit{If} \quad f\left(x_j(i), y_j(i)\right) < s^* \\ \textit{Then} \\ s^* = f\left(x_j(i), y_j(i)\right) \\ x^* = x_j(i) \ , \ \ y^* = y_j(i) \end{array}$$

Figura 2.39. Actualización del óptimo.

El criterio de parada del algoritmo se cumplirá al completar el número total de iteraciones prefijadas *n_iter*.

4.4.6.MODIFICACIONES DEL ALGORITMO POR LA INTERACCIÓN ENTRE ASTEROIDES

Aunque la masa de cada asteroide, que suponemos igual a 1, es muy inferior a la de los óptimos, no puede obviarse que su distancia es asimismo muy pequeña. Por ello y, según los autores[27], la aceleración de cada asteroide influirá en la de los otros de la forma detallada en la figura 2.40.

[27] Hemos mantenido el algoritmo tal y como lo exponen los autores pero, a nuestro juicio, en 2.40 debería aparecer en el denominador el cubo de las distancias y no el cuadrado, merced a las consideraciones hechas en la sección 4.2, aunque también puede considerarse que forma parte de α.

$$ax_j(i) = G\left(f(x_j(i) - r_d, y_j(i)) - f(x_j(i) + r_d, y_j(i)) \right) + \frac{\alpha X_{c,j}}{r_{i,j}^2}$$

$$ay_j(i) = G\left(f(x_j(i), y_j(i) - r_d) - f(x_j(i), y_j(i) + r_d) \right) + \frac{\alpha Y_{c,j}}{r_{i,j}^2}$$

$1 \le i \le n$ (indica el asteroide del grupo)

$1 \le j \le n\_iter$ (indica el movimiento j-esimo del asteroide)

Figura 2.40. Interacción entre asteroides.

Donde r_{ij} es la distancia del asteroide *i*-ésimo al centro de gravedad del conjunto formado por los *n* asteroides en la iteración *j*-ésima, α es un parámetro que nos permitirá mejorar la eficiencia del algoritmo y $\left(X_{Cj}, Y_{Cj}\right)$ son las coordenadas de dicho centro de gravedad que pueden obtenerse de forma muy sencilla a través de las siguientes ecuaciones:

$$X_{c,j} = \frac{1}{\mathrm{n}} \sum_{\mathrm{i}=1}^{\mathrm{n}} x_j(i)$$

$$Y_{c,j} = \frac{1}{\mathrm{n}} \sum_{\mathrm{i}=1}^{\mathrm{n}} y_j(i) \qquad 1 \le i \le n \quad , \quad 1 \le j \le n\_iter$$

Figura 2.41. Cálculo de las coordenadas del centro de gravedad de los asteroides.

Físicamente, α es un parámetro que representa la influencia del efecto gravitacional entre los asteroides. Según Hsiao y otros (2005), si es muy grande se incrementaría notablemente la aceleración de cada asteroide y con ella su velocidad. Así, la probabilidad de choque entre los asteroides aumentaría notablemente.

4.4.7.MODIFICACIONES DEL ALGORITMO POR LOS MOVIMIENTOS DEL ESPACIO

Otro concepto cosmológico que puede aportarnos un nuevo parámetro que contribuya a mejorar los resultados obtenidos, es considerar el universo en contracción o en expansión para, de este modo, modificar las distancias y las velocidades de los óptimos y de los asteroides que, al formar parte del universo, se expanden o contraen con él.

Dichas modificaciones pueden hacerse añadiendo en la ecuación 2.36 un término cuyo coeficiente, β, si es menor que 1 indica que el universo está en expansión, ya que en ese caso las velocidades de los cuerpos disminuyen, si es mayor que 1 indica que está en contracción porque si el universo es más pequeño las

velocidades de los cuerpos en él inmersos aumentan y si es 1 significa que el universo está estable.

Con esta variación la ecuación 2.36 quedaría

$$vx_{j+1}(i) = \beta vx_j(i) + ax_j(i)$$
$$vy_{j+1}(i) = \beta vy_j(i) + ay_j(i) \qquad 1 \le i \le n\ ,\ 1 \le j \le n\_iter$$

Figura 2.42. Modificaciones originadas por expansión-contracción del espacio-tiempo.

Hsiao y otros (2005) exponen que cuando β es un poco mayor que 1 puede ayudar a los asteroides a salir de óptimos locales de forma significativa. No obstante tomar para β valores muy grandes revierte en pérdida de efectividad del algoritmo porque las velocidades de los asteroides son demasiado elevadas lo que provoca que se dispare el número de las infactibilidades, i.e. salidas de asteroides de la región factible.[28]

En resumen, partimos de un grupo de asteroides con velocidades y posiciones asignadas previamente, a continuación calculamos la aceleración de cada asteroide haciendo un promedio de los valores de la función objetivo en un entorno[29] de cada punto previamente elegido y posteriormente actualizamos su velocidad y su posición tal y como se desarrolla en el método numérico de Euler para la resolución de ecuaciones diferenciales de primer orden.

El pseudocódigo del algoritmo es

```
Procedimiento Gravita

Inicializar aleatoriamente todos los asteroides, f[best] = inf
Mientras t <  n_iter hacer
      Calcula 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
```

Figura 2.43. Pseudocódigo de S.G.O.

[28] No podemos avalar este comentario ya que, como se verá posteriormente, nuestros mejores resultados computacionales se han obtenido con valores de β menores que 1.

[29] Entorno de radio r_d

En la figura 2.44 podemos observar el recorrido de dos asteroides en una de las ejecuciones realizadas en dimensión 2. Uno de los asteroides está representado por círculos azules y el otro por asteriscos rojos. En la imagen podemos apreciar cómo los asteroides pasan cerca de los óptimos locales (◊) pero continúan su camino hacia el óptimo global (♦). De este modo los asteroides exploran todo el espacio de soluciones.

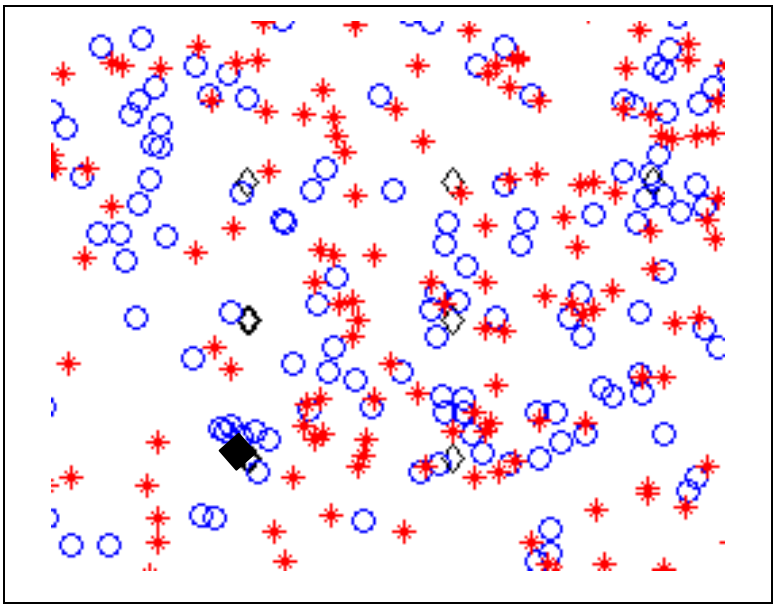


Figura 2.44. Ejecución con 2 asteroides representada con MATLAB.

5. OTRAS HEURÍSTICAS GRAVITATORIAS RECIENTES

En los últimos años se han publicado varios algoritmos de optimización basados en la analogía física con los campos gravitatorios. Algunos simulan el vuelo de un proyectil (Harkiolakis 2008), otros usan el concepto de radiación gravitacional de la teoría de la relatividad general de Einstein (Chuang y otros 2007) y otros realizan simulaciones de elementos de búsqueda, ya sean satélites-sonda o agentes, que se van trasladando por el espacio de soluciones conforme a las leyes de Newton (Formato 2007, Rashedi 2009).

Dada la gran analogía que estos dos últimos guardan con S.G.O. los denominaremos genéricamente *heurísticos gravitatorios* y los analizaremos con más detalle en los apartados siguientes.

A pesar de que los algoritmos son muy parecidos a S.G.O. hemos preferido mantener las notaciones tal y como aparecen en los artículos correspondientes y no equipararlas con las de la sección 4.

En ambos casos hemos procurado detallar las similitudes y diferencias que guardan con S.G.O.

5.1. OPTIMIZACIÓN DE FUERZA CENTRAL, CENTRAL FORCE OPTIMIZATION, C.F.O.

Central Force Optimization es un metaheurístico diseñado por **Richard Formato (2007)**[30] para resolver el problema

$$\begin{array}{ll} \text{Opt} & f(\vec{x}) \\ s.a. & \vec{x} \in \mathbb{R}_{N_d} \,/\, x_i^{\min} \le x_i \le x_i^{\max} \;\; 1 \le i \le N_d \end{array}$$

Figura 2.45. Planteamiento del problema general de optimización en C.F.O.

La base física del algoritmo C.F.O. es muy similar a la de S.G.O. y puede ilustrarse en el siguiente ejemplo: supongamos que necesitamos determinar la posición del mayor planeta del sistema solar sin un conocimiento previo de la topología del espacio. Al ser el mayor planeta, genera la mayor intensidad del campo, por lo que una aproximación a su ubicación podría obtenerse lanzando una serie de satélites-sonda que se desplazasen conforme a la ley de Newton, y que fueran aumentando sus masas a medida que se acercaran al planeta, con lo que atraerían a su vez a nuevos agentes de búsqueda. La ubicación del planeta sería obtenida sin más que registrar las posiciones ocupadas por los satélites más pesados.

[30] A lo largo de la presente sección reproduciremos algunas partes del artículo (Formato 2007).

La aceleración de un satélite debida a la atracción de un cuerpo de masa M responde a la ley de la Gravitación Universal de Newton y queda descrita a través de la ecuación 2.46 que exponemos de nuevo para el caso n-dimensional.

$$\vec{a} = -G\frac{M\vec{r}}{|\vec{r}|^3}$$

Figura 2.46. Ley de Gravitación Universal de Newton para el caso n-dimensional.

En C.F.O. la actualización de las posiciones de los satélites, que el autor denota por $\vec{R}(t)$, se lleva a cabo mediante la siguiente fórmula:

$$\vec{R}(t+\Delta t) = \vec{R}_0 + \vec{V}_0\Delta t + \frac{1}{2}\vec{a}(\Delta t)^2$$

Figura 2.47. Actualización de las posiciones de los satélites.

que es la aproximación del polinomio de Taylor de grado 2 para la función posición en función del tiempo.

Tras suficiente tiempo la mayor parte de los satélites, presumiblemente, caerán en órbitas alrededor del mayor planeta (Formato 2007). C.F.O. generaliza esta idea donde la topología del espacio viene determinada por la función objetivo, por tanto el planeta de mayor masa será el óptimo global.

Al igual que sucede en A.C.O. o en P.S.O., en el algoritmo C.F.O. la información de cada satélite, el valor de la función objetivo en su posición, influye en las aceleraciones del resto de los satélites de forma que los satélites más pesados, soluciones buenas, atraen al resto de los agentes de búsqueda hacia sí. Esta característica es la diferencia esencial entre los algoritmos C.F.O. y S.G.O. Para explicarlo con claridad necesitamos algunas definiciones.

Con $\vec{R}_j^p$ denotamos la posición del satélite *p-ésimo* en el momento *j-ésimo*.

La "masa", el valor de la función objetivo, del satélite *p-ésimo* en el instante *j-ésimo* viene definida por $M_j^p = f\left(\vec{R}_j^p\right)$.

N_p es el número total de satélites lanzados y N_d es la dimensión del espacio.

El satélite *p-ésimo* se desplaza de la posición ocupada en el instante *j-1* al instante *j* siguiendo una trayectoria que depende de su posición inicial y de la aceleración total producida por las "masas" que están a su alrededor siguiendo la siguiente ecuación:[31]

$$\vec{a}_{j-1}^{\,p} = G\sum_{\substack{k=1 \\ k\neq p}}^{N_p} U\left(M_{j-1}^{k} - M_{j-1}^{p}\right)\cdot\left(M_{j-1}^{k} - M_{j-1}^{p}\right)^{\alpha}\frac{\left(\vec{R}_{j-1}^{k} - \vec{R}_{j-1}^{p}\right)}{\left|\vec{R}_{j-1}^{k} - \vec{R}_{j-1}^{p}\right|^{\beta}}$$

Figura 2.48. Aceleración del satélite *p-ésimo* en el instante *j-ésimo.*

donde *G* es un parámetro de C.F.O. asimilado a la constante de gravitación universal,

U es la función escalón de Heaviside, $U(x) = \begin{cases} 0 & \text{si } x < 0 \\ 1 & \text{si } x \geq 0 \end{cases}$

α y β son dos parámetros de C.F.O. que en el caso de la ley de la gravitación de Newton toman los valores 1 y 3 respectivamente.

La razón de introducir la función de Heaviside en 2.48 es para evitar "masas" negativas. Al considerar las diferencias de "masas" de los distintos satélites podría ocurrir que $\left(M_{j-1}^{k} - M_{j-1}^{p}\right)$ tomara valores negativos, lo que corresponde a la influencia de "masas" más pequeñas, soluciones peores. Para evitarlo se introduce la función escalón que hará que las peores soluciones intervengan con el valor 0 en la aceleración total del satélite.

Para actualizar la posición del satélite *p-ésimo* según 2.47 necesitamos conocer su velocidad en el instante *j-1* lo que hacemos a través de la fórmula[32]

$$\vec{V}_{j-1}^{\,p} = \frac{\vec{R}_{j-1}^{p} - \vec{R}_{j-2}^{p}}{1} \qquad j \geq 2$$

Figura 2.49. Velocidad en el instante j-ésimo del satélite p-ésimo.

Así, la posición del satélite *p-ésimo* en el instante *j-ésimo* aplicando 2.47 será

$$\vec{R}_{j}^{\,p} = \vec{R}_{j-1}^{p} + \vec{V}_{j-1}^{\,p}\cdot 1 + \frac{1}{2}\vec{a}_{j-1}^{\,p}(1)^2$$

Figura 2.50. Posición del satélite *p-ésimo* en el instante *j-ésimo*.

[31] Observemos que el signo negativo de *2.48* ya está en la expresión al considerar $\left(\vec{R}_{j-1}^{k} - \vec{R}_{j-1}^{p}\right)$.

[32] En la implementación hecha por Formato se asigna el valor 0 a las velocidades para cualquier valor de *p* y *j*.

En la figura 2.51, tomada de Formato (2007), puede verse cómo el desplazamiento del satélite *p-ésimo* desde su posición en el momento *j-1-ésimo* al momento *j-ésimo* depende de la influencia sobre él de los otros agentes de búsqueda. Tendrán mayor influencia aquéllos que tengan mayor masa, i.e. estén más cerca del óptimo global.

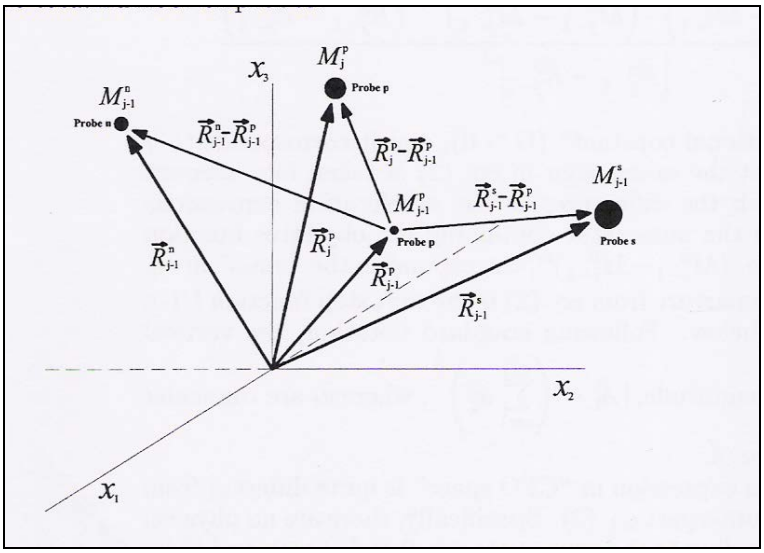


Figura 2.51. Influencia en C.F.O. de todos los satélites (tomada de Formato 2007).

A diferencia de los algoritmos A.C.O., P.S.O. o S.G.O., C.F.O., en su versión inicial, no utiliza ninguna función aleatoria ni para inicializar las posiciones de cada satélite ni para reinicializar las posiciones de los satélites que salgan de la región factible. Todos los cálculos se realizan de forma determinista. En Formato (2010 b), el propio autor introduce cierta aleatoriedad en su algoritmo.

Las posiciones iniciales se calculan a través del bucle

$$\text{For i=1 to } \mathrm{N_d} \ , n = 1 \text{ to } \frac{N_p}{N_d} : p = n + \frac{(i-1)N_p}{N_d},$$
$$R_1^p(i) = x_i^{\min} + \frac{(n-1)(x_i^{\max} - x_i^{\min})}{\frac{N_p}{N_d} - 1}$$

Figura 2.52. Bucle para calcular las posiciones iniciales en C.F.O.

El número de satélites por dimensión, $\frac{N_p}{N_d}$, es especificado por el usuario y las posiciones infactibles de los satélites se reinicializan en el punto medio entre su posición inicial y la última posición factible.

El pseudocódigo del algoritmo es (Formato 2007)

```
Procedimiento CFO

Inicializar las posiciones de todos los satélites
Inicializar todas las aceleraciones a 0.
Calcular fitness inicial

Mientras j < N_t hacer

        Para cada satélite p hacer
                Calcular la aceleración del satélite;
                Actualizar la posición del satélite;
                Si no factible p hacer factible;
                Actualizar fitness
                Si fitness (p) mejor que f_best

                        entonces actualizar posicion_best y f_best
                end si
        j=j+1
```

Figura 2.53. Pseudocódigo de C.F.O.

5.2. ALGORITMO DE BÚSQUEDA GRAVITACIONAL, GRAVITATIONAL SEARCH ALGORITHM, G.S.A.

Diseñado por **Esmat Rashedi, Hossein Nezamabadi-pour y Saeid Saryazdi (2009)[33],** la metaheurística G.S.A. también está inspirada en las leyes de Newton. Sin embargo introduce algunas diferencias teóricas en cuanto al concepto de masa y al de la constante de gravitación universal.

En primer lugar, si observamos la ecuación 2.10, correspondiente a la segunda ley de la dinámica de Newton, podemos observar la masa como el coeficiente que mide "la resistencia del cuerpo a la aceleración" producida por la fuerza y se identifica, por tanto, con la inercia del cuerpo. Por otra parte, en la ecuación 2.11, correspondiente a la ley de la gravitación universal, la masa aparece en dos formas diferentes: como agente que genera el campo gravitacional y como agente que responde pasivamente al campo generado.

Atendiendo a lo anteriormente expuesto, en física teórica se definen tres conceptos distintos de masa que originan respectivamente los siguientes tipos de masa:

- **Masa gravitacional activa,** M_a, es la medida de la fuerza del campo gravitatorio que genera un cuerpo. La intensidad del campo gravitatorio generado por un objeto de mayor masa activa es mayor que el generado por un objeto de menor masa activa.

[33] A lo largo de la presente sección expondremos muchos resultados de este artículo.

- **Masa gravitacional pasiva,** M_p, corresponde a la medida de la fuerza de la interacción de un cuerpo en un campo gravitacional creado por otro cuerpo. En el mismo campo, un cuerpo con más masa gravitacional pasiva será atraído con más fuerza que otro cuerpo con masa gravitacional pasiva menor**.**

- **Masa inercial,** M_i, es la medida de la resistencia de un cuerpo a variar su estado cuando una fuerza se aplica sobre él. Un cuerpo con mayor masa inercial cambiará su estado más lentamente que otro con menor masa inercial.

Aunque conceptualmente los tres tipos de masas son distintos, ningún experimento ha demostrado de forma clara alguna diferencia entre ellos. Además, los principios débil y fuerte de equivalencia postulados por Einstein asumen la equivalencia entre los tres conceptos de masa (Rashedi y otros 2009)[34].

En segundo lugar, se acepta que en realidad *G* no es constante sino que va decreciendo con la edad del universo siguiendo la fórmula siguiente

$$G(t) = G(t_0) \cdot \left(\frac{t_0}{t} \right)^{\beta} \quad \beta < 1$$

Figura 2.54. Variación de la "constante" *G*.

con t_0 el instante 0 del universo (Rashedi y otros 2009).

Considerando las definiciones anteriores las leyes de Newton pueden rescribirse de la siguiente forma:

Sea $\vec{\mathrm{F}}_{\mathrm{ij}}$ la fuerza ejercida por el cuerpo *i* sobre el cuerpo *j y* $\vec{a}_j$ la aceleración que experimenta el cuerpo *i*. Sean Mai , Mpj M_{ii} la masa activa del cuerpo *i*, la masa pasiva del cuerpo *j* y la masa inercial del cuerpo *i* respectivamente. Sea *r* la distancia entre los cuerpos i y j. Entonces

$$\vec{\mathrm{F}}_{\mathrm{ij}} = M_i \, \vec{a}_j$$

Figura 2.55. Fuerza ejercida por el cuerpo i sobre el cuerpo j.

[34] De hecho, los autores de G.S.A. también asumen esta equivalencia.

$$\left|\vec{F}_{ij}\right| = G(t)\frac{M_{ai}M_{pj}}{r^2}$$

Figura 2.56. Módulo de la fuerza ejercida por el cuerpo i sobre el cuerpo j.

Para resolver el problema de optimización general, el algoritmo G.S.A. propone lanzar una serie de agentes de búsqueda en el espacio de soluciones. Todos estos cuerpos tendrán asignada una masa, dependiente de la función objetivo y del resto de los agentes, y se moverán en base a unas leyes propias de G.S.A. pero inspiradas en las leyes de Newton lo que causará fuerzas de atracción entre ellos y, por ende, un movimiento global de todos los cuerpos hacia los de masa más pesada, las buenas soluciones, (Rashedi y otros 2009).

Las leyes que rigen los movimientos son las siguientes:

- **Ley de la gravedad:** Cada cuerpo atrae a otro cuerpo con una fuerza directamente proporcional al producto de sus masas activa y pasiva respectivamente e inversamente proporcional a la distancia que los separa[35].
- **Ley del movimiento:** La velocidad actual de cualquier cuerpo es igual a la suma de un escalar menor que uno multiplicado por su velocidad en el instante anterior más su aceleración en el instante anterior. Dicha aceleración es igual a la fuerza que actúa sobre el cuerpo dividida entre su masa inercial.

Veamos el desarrollo del algoritmo:

Consideremos un sistema con *N* agentes en un espacio de dimensión *n*. Para cada agente se consideran cuatro variables: su posición, su masa gravitatoria activa, su masa gravitatoria pasiva y su masa inercial[36] denotadas como

$$X_i(t) = (x_i^1(t),...,x_i^d(t),...,x_i^n(t))$$
$$M_{pi}(t), M_{ai}(t), M_{ii}(t)\ \ i = 1,...,N\ \ \ d = 1,...,n\ \ \ t = 1,...,\max - iter$$

Figura 2.57. Posición y masas de cada agente de búsqueda en G.S.A.

[35] Según los autores, los resultados computacionales son mejores si no se considera el cuadrado de la distancia.

[36] En realidad son dos, ya que los autores identifican en todo el algoritmo las masas gravitatorias activa y pasiva y la inercial de cada agente.

Para calcular las masas de cada agente en el instante t se calculan previamente $best(t)$ y $worst(t)$ como

$$best(t) = \min_{i=1,\ldots,N} f(X_i(t))$$
$$worst(t) = \max_{i=1,\ldots,N} f(X_i(t))$$

Figura 2.58. Definiciones del peor y el mejor en G.S.A.

A continuación se calculan las masas de cada agente en el instante *t*.

$$m_i(t) = \frac{f(X_i(t)) - worst(t)}{best(t) - worst(t)}$$
$$M_{ai}(t) = M_{pi}(t) = M_{ii}(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)}$$

Figura 2.59. Cálculo de las masas de cada agente en G.S.A.

Como se ve en 2.60 la asignación de masa a cada agente depende del valor de la función objetivo en todos los agentes. La información es compartida de forma que se asigna las mayores masas a las mejores soluciones.

Cada componente de la fuerza ejercida por el agente *i* sobre el agente j se define

$$\vec{F}_{ij}(t) = G(t)\frac{M_{pi}(t)\cdot M_{aj}(t)}{r_{ij} + \varepsilon}\left(x_j^d(t) - x_i^d(t)\right)$$

Figura 2.60. Definición de la fuerza ejercida por el agente *i* sobre el agente *j*.

donde G(t) no es constante sino que va disminuyendo con el tiempo según 2.54 y r_{ij} y ε son respectivamente la distancia entre los agentes *i* y *j* y un parámetro de G.S.A. al que se le asignará un valor pequeño.

Cada componente de la fuerza total ejercida sobre el agente *i* se calcula estocásticamente siguiendo la fórmula

$$\vec{F}_i^d(t) = \sum_{\substack{j=1 \\ j\neq i}}^{N} rand(j)\vec{F}_{ij}^d(t)$$

Figura 2.61. Cálculo aleatorio de las componentes de la fuerza en G.S.A.

con $rand(j)$ un número aleatorio según la distribución uniforme $U(0,1)$.

Las mayores masas tienen mayor intensidad de atracción que las menores.

La correspondiente componente de la aceleración ejercida sobre el agente *i* se calcula usando 2.55

$$a_i^d(t) = \frac{F_i^d(t)}{M_{ii}(t)}$$

Figura 2.62. Cálculo de la aceleración ejercida sobre el agente *i* en G.S.A..

Y por último se actualizan las velocidades y posiciones en base a la ley del movimiento de G.S.A.

$$v_i^d(t+1) = rand(i) \cdot v_i^d(t) + a_i^d(t)$$
$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1)$$

Figura 2.63. Actualización de velocidades y posiciones en G.S.A..

Los autores no especifican cómo resolver las situaciones de infactibilidad.

El pseudocódigo del algoritmo es (Rashedi y otros 2009)

```
Procedimiento GSA

Inicializar aleatoriamente las posiciones y velocidades de
todos los agentes

Mientras t <  max-iter

        Calcular fitness de cada agente
        Calcular G(t),best(t), worst(t) y M_i(t)
        Para cada agente i  hacer
            Calcular la fuerza ejercida sobre él;
            Calcular la aceleración;
            Actualizar la velocidad;
            Actualizar la posición;
            Si fitness (i) mejor que f_best
                entonces actualizar posicion_best y f_best
            end si
    t:=t+1
```

Figura 2.64. Pseudocódigo de G.S.A.

La mayor diferencia entre G.S.A. y S.G.O. es la asignación de masa, y por tanto de aceleración, que en el primer caso se hace compartiendo la información entre los distintos

agentes y en el segundo no. No obstante, el esquema: “lanzar asteroides, calcular aceleración, actualizar velocidades y posiciones” es igual en ambos algoritmos.

6. ALGORITMOS LOCALES PARA HIBRIDAR CON S.G.O.

6.1. INTRODUCCIÓN

En la presente sección realizaremos el estudio de los dos algoritmos locales con que vamos a hibridar S.G.O. en los próximos capítulos para conseguir un metaheurístico más potente. Comenzaremos por exponer la heurística Nelder-Mead. Posteriormente realizaremos la exposición del método del Gradiente.

6.2. EL SIMPLEX DE NELDER-MEAD

6.2.1.INTRODUCCIÓN

El simplex de Nelder-Mead, simplex no-lineal, es un algoritmo heurístico propuesto por John Nelder y Roger Mead (1965) para la optimización de funciones objetivo sin restricciones.

Dicho algoritmo, que sólo usa valores de la función, no de su derivada, se basa en conceptos geométricos. Partiendo de un conjunto de *dim+1* puntos distintos de $\mathbb{R}^{\dim}$, que conforman las *dim+1* aproximaciones iniciales al óptimo buscado, se construye su envolvente convexa en el espacio dim-dimensional, i.e el poliedro más simple, no degenerado, formado por dichos vértices; y posteriormente se modifica el poliedro de forma que en los nuevos vértices la función objetivo mejore[37]. Es, por tanto, un algoritmo de búsqueda de óptimos locales. Ejemplos de simplex serían: el segmento que une dos puntos en $\mathbb{R}$, el triángulo que une 3 puntos no alineados en $\mathbb{R}^2$, el tetraedro que une 4 puntos no coplanarios en $\mathbb{R}^3$ etc.

La deformación y variación del poliedro se hace en base a 4 operaciones básicas: **reflexión, expansión, contracción y encogimiento,** que llevan asociados 4 parámetros respectivamente que denotaremos: ρ, χ, γ y σ.

Siguiendo el algoritmo Nelder-Mead, dichos parámetros deben satisfacer

$$\rho > 0, \quad \chi > 1, \quad 0 < \gamma < 1, \quad 0 < \sigma < 1 \quad \chi > \rho$$

Figura 2.65. Condiciones que deben satisfacer los parámetros de Nelder-Mead.

[37] Disminuya.

Para nuestras experiencias computacionales hemos asumido los valores

$$\rho = 1, \quad \chi = 2, \quad \gamma = \frac{1}{2}, \quad \sigma = \frac{1}{2}$$

Figura 2.66. Parámetros elegidos para nuestras experiencias computacionales.

que son los valores usados en el algoritmo estándar (Lagarias y otros 1998).

6.2.2.DESARROLLO DEL ALGORITMO

Veamos cómo realizar la iteración *k-ésima*.

Partimos de *dim+1* puntos que forman un simplex no degerado de $\mathbb{R}^{\dim}$, en el caso *k* = 1 tendremos los *dim+1* vértices iniciales.

El objetivo de la iteración es obtener un nuevo simplex variando uno o más de los vértices que lo forman para lo que seguiremos los siguientes pasos:

- **Paso 1: ordenación**

En primer lugar ordenamos los puntos de forma creciente respecto a los valores de la función objetivo. Es decir

$$x_1^{k-1}, x_2^{k-1}, \ldots, x_{\dim}^{k-1}, x_{\dim+1}^{k-1} \quad k \geq 1$$

Figura 2.67. Ordenación de los vértices del simplex.

cumpliendo que

$$f\left(x_1^{k-1}\right) \leq f\left(x_2^{k-1}\right) \leq \ldots \leq f\left(x_{\dim}^{k-1}\right) \leq f\left(x_{\dim+1}^{k-1}\right)$$

Figura 2.68. Criterio de ordenación de los vértices del simplex.

Puesto que nuestro problema es minimizar la función objetivo entendemos que el primer punto es el mejor, el último el peor, etc.

- **Paso 2: reflexión**

Hallamos el punto de reflexión que será el punto simétrico, ponderado por ρ, del peor respecto al centro de gravedad que forman los dim mejores, esto es:

Dado[38] $\boxed{\overline{x} = \frac{1}{\dim}\sum_{i=1}^{\dim} x_i^{k-1}}$

Entonces

$$\boxed{x_{refle} = \overline{x} + \rho\left(\overline{x} - x_{\dim+1}\right)}$$

Figura 2.69. Construcción del punto de reflexión

Evaluamos la función objetivo $f(x_{refle})$

- **Caso I: el punto de reflexión ha mejorado al mejor**

$$\boxed{f(x_{refle}) < f(x_1)}$$

  - **Paso 3: expansión**

  Calculamos el punto de expansión del centro de gravedad respecto al reflejado para ver si podemos mejorar aún más

$$\boxed{x_{\text{expan}} = \overline{x} + \chi\left(x_{refle} - \overline{x}\right)}$$

Figura 2.70. Construcción del punto de expansión

De nuevo evaluamos la función objetivo $f(x_{\text{expan}})$

Si hemos mejorado, i.e. $f(x_{\text{expan}}) < f(x_{refle})$ aceptamos el punto de expansión y sacamos al peor punto para formar el nuevo simplex. **Fin de la iteración.**

Si no hemos mejorado, i.e. $f(x_{\text{expan}}) \geq f(x_{refle})$ aceptamos el punto reflejado y sacamos al peor punto para formar el nuevo simplex. **Fin de la iteración.**

- **Caso II: el punto de reflexión no ha mejorado al mejor pero sí ha mejorado a alguno de los n mejores.**

$$\boxed{f(x_1) \leq f(x_{refle}) < f(x_{\dim})}$$

[38] Para no entorpecer la notación omitiremos a partir de este momento el superíndice *k*.

Aceptamos el punto reflejado y sacamos al peor punto para formar el nuevo simplex. **Fin de la iteración**

- **Caso III: el punto de reflexión no ha mejorado a ninguno de los n mejores**

$$f(x_{refle}) \geq f(x_{\dim})$$

  - **Paso 4: contracción**

La contracción se hace hacia el punto reflejado o hacia el peor punto dependiendo de cuál sea mejor de los dos

- **Hacia fuera:** si el punto reflejado es mejor que el peor i.e. $f(x_{refle}) < f(x_{\dim+1})$

$$x_{con-f} = \overline{x} + \gamma\left(x_{refle} - \overline{x}\right)$$

Figura 2.71. Construcción del punto contraído hacia afuera.

Evaluamos la función objetivo en el punto contraído

$$f(x_{con-f})$$

Si hemos mejorado, i.e. , $f(x_{con-f}) < f(x_{refle})$ aceptamos el punto contraído y sacamos al peor para formar el nuevo simplex. **Fin de la iteración.**

Si no hemos mejorado, i.e. $f(x_{con-f}) \geq f(x_{refle})$ vamos al **paso 5.**

- **Hacia dentro:** si el punto reflejado iguala o empeora el peor i.e. $f(x_{refle}) \geq f(x_{\dim+1})$

$$x_{con-d} = \overline{x} - \gamma\left(\overline{x} - x_{\dim+1}\right)$$

Figura 2.72. Construcción del punto contraído hacia dentro.

Evaluamos la función objetivo en el punto contraído

$$f(x_{con-d})$$

Si hemos mejorado i.e. $f(x_{con-d}) < f(x_{\dim+1})$ aceptamos el punto contraído y sacamos al peor para formar el nuevo simplex **Fin de la iteración.**

Si no hemos mejorado i.e. $f(x_{con-d}) \geq f(x_{\dim+1})$ vamos al **Paso 5.**

- **Paso 5: encogimiento**

Tomamos n puntos nuevos de la forma

$$v_i = x_1 + \sigma(x_i - x_1) \quad i = 2,...,\dim+1$$

El nuevo simplex tiene como vértices $x_1, v_2, ..., v_{\dim+1}$. **Fin de la iteración.**

El algoritmo se mantiene hasta que se haya completado un número prefijado de iteraciones, valnel.

En la figura 2.73, obtenida de Lagarias (1998), tenemos cinco gráficos de movimientos del simplex. En el primero, vemos cómo ha variado tras una reflexión, en el segundo tras una reflexión-expansión, en el tercero y cuarto tras una contracción, hacia fuera y hacia dentro respectivamente, y en el último tras un encogimiento. En todos los casos el simplex inicial aparece con líneas punteadas.

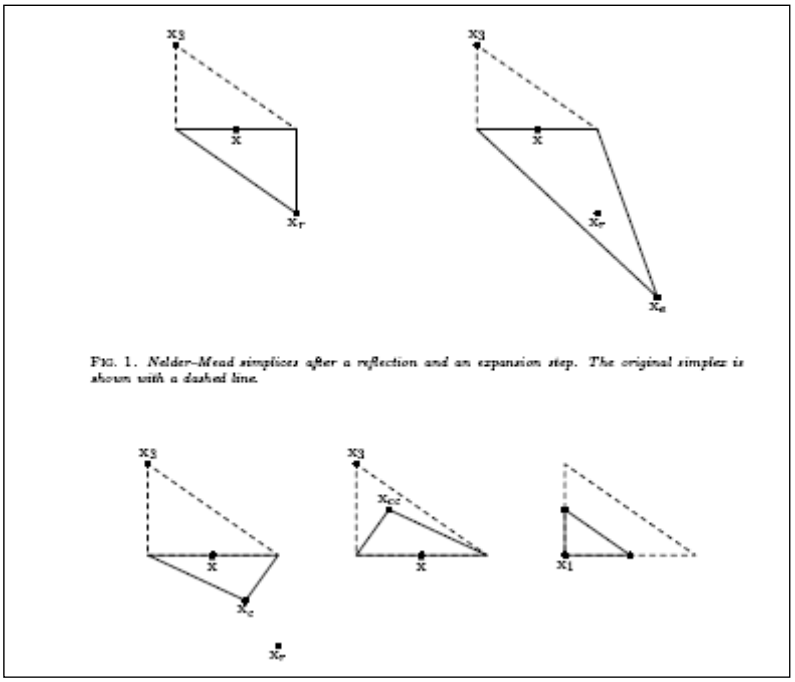


Figura 2.73. Variaciones en el simplex tras todos los posibles movimientos

Presentamos un organigrama del método de Nelder-Mead en la figura 2.74.

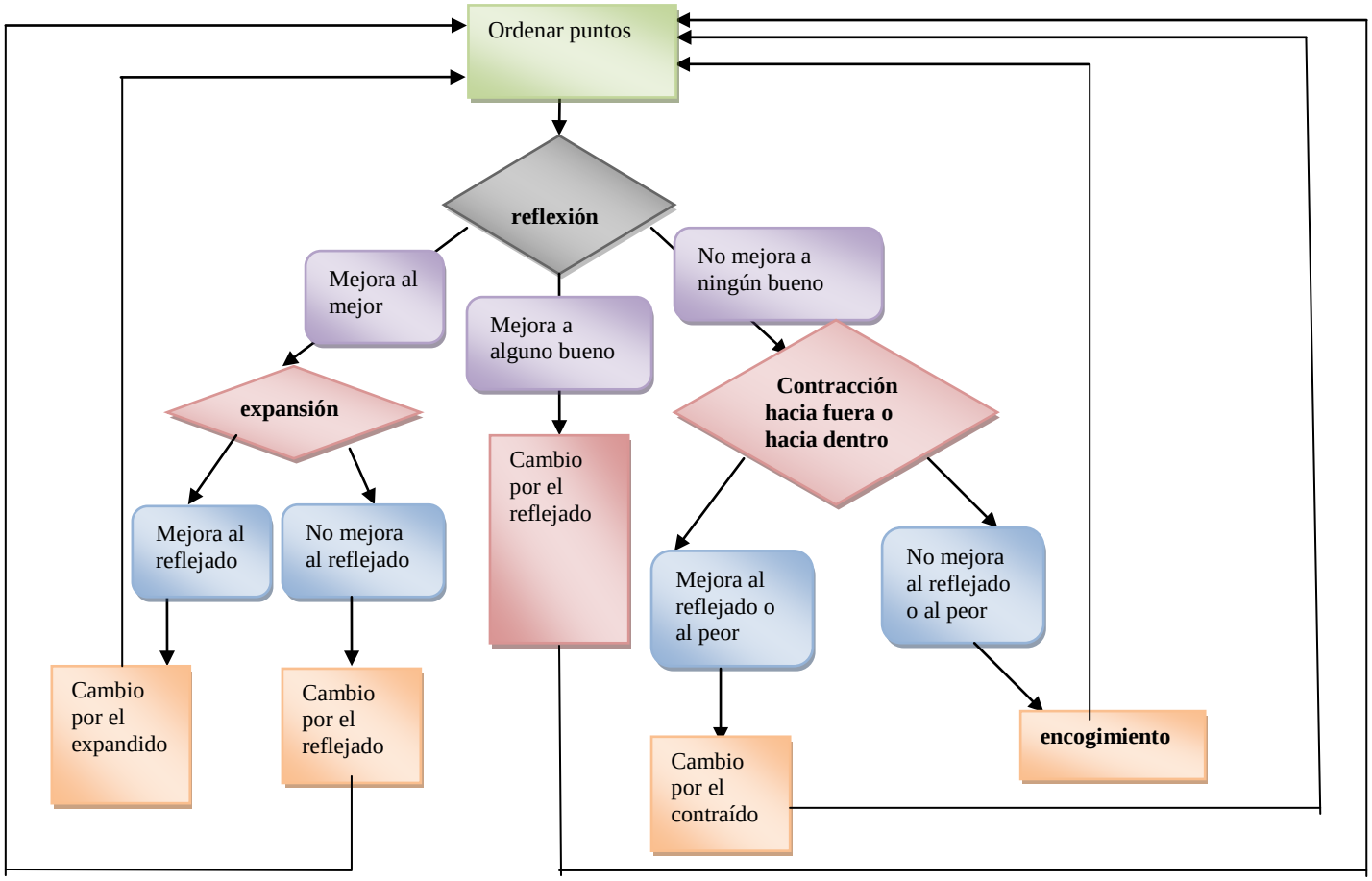


Figura 2.74. Organigrama de Nelder-Mead.

6.3. EL MÉTODO DEL GRADIENTE

6.3.1.INTRODUCCIÓN

El símil de “la tela de araña” usado para explicar intuitivamente la curvatura del espacio-tiempo en S.G.O. también puede usarse a nivel de la superficie terrestre para explicar el movimiento de una pelota colocada en la ladera de una montaña.

Efectivamente, si soltamos una pelota en el aire caerá siguiendo un movimiento uniformemente acelerado en dirección vertical hacia abajo hasta llegar a la superficie de la tierra, pero si la soltamos sobre una montaña, o en una gran tela de araña, ¿cuál es la dirección que sigue? Sabemos que en este caso la pelota tiende a perder la mayor altura posible en el mínimo tiempo, en realidad la energía potencial, ya que en este caso viene determinada por la altura, es decir, sigue la dirección de máximo descenso sobre la superficie en que se encuentre.

Dicha dirección es la marcada por el vector gradiente del campo escalar cuya gráfica corresponde a la superficie y su sentido es el contrario.

Es decir, la aceleración que actúa sobre el objeto depositado sobre una superficie es proporcional, con sentido contrario, al gradiente del campo escalar que define la superficie sobre la que se encuentra.

Dicha idea, análoga a la que sustenta la simulación S.G.O., es el pilar fundamental del grupo de métodos numéricos de optimización llamados **de máximo descenso** como el método del Gradiente (Burden-Faires 1985), el método del Gradiente Conjugado (Hernández 2006 , Kindelán 2007) o el Método de Fletcher-Reeves (Kindelán 2007). Sin embargo, a diferencia de S.G.O., aquellos están diseñados para localizar con gran precisión óptimos locales, por lo tanto, su efectividad es alta[39] pero su trabajo se desarrolla a nivel de entornos.

Por ello pensamos que se presentan como el complemento ideal para hibridarlos con el algoritmo "Optimización gravitatoria" creando un metaheurístico que potencie su efectividad. Es lo que haremos con uno de ellos, el método del gradiente.

El problema planteado es encontrar el mínimo absoluto sin restricciones de un campo escalar que supondremos suficientemente regular

$$\begin{array}{rcl} f:\mathbb{R}^n & \rightarrow & \mathbb{R} \\ \bar{x} & \rightarrow & f(\bar{x}) \end{array}$$

Figura 2.75. Problema general de optimización a nivel de entornos.

Al tratarse de un problema de optimización sin restricciones, el óptimo global, en realidad, se encontrará en un óptimo local, ya que, obviamente, no puede encontrarse en la frontera. Luego, el problema se traduce finalmente en localizar un óptimo local de $f(x)$.

Si la función es suficientemente regular, el punto buscado, x^*, va a verificar $\nabla f\left(x^*\right)=\vec{0}$ pero, en general, la resolución del correspondiente sistema no lineal de ecuaciones es muy costoso. Algunos algoritmos numéricos como el de *Newton* (Burden-Faires 1985, Hernández 2006), plantean directamente encontrar la solución aproximada del sistema partiendo de una aproximación inicial, que se va modificando a

[39] Todos tienen convergencia al menos lineal.
La convergencia de un algoritmo es lineal cuando $\exists \lambda \in (0,1) / \left|x_{i+1}-x^*\right| \leq \lambda \left|x_i-x^*\right|$, x^* representa la posición del óptimo buscado.

lo largo de una serie de etapas, dando lugar a una sucesión de iterantes que converge a la solución del sistema. Además, sus resultados son excelentes, ya que su orden de convergencia es cuadrático[40]. Sin embargo, presenta el inconveniente de que el iterante inicial ha de estar suficientemente próximo a la solución, i.e. convergencia local. El método que expondremos a continuación, en cambio, utiliza la idea del máximo descenso comentada anteriormente y su convergencia está garantizada para cualquier iterante inicial, i.e. convergencia global (Hernández 2006).

6.3.2.EL MÉTODO DEL GRADIENTE

Dado $\overline{x_0} \in \mathbb{R}^n$ cumpliendo $\nabla f\left(\overline{x_0}\right) \neq \vec{0}$ y dado $\vec{d} \in \mathbb{R}^n$, se demuestra que la derivada direccional de *f* en la dirección de $\vec{d}$ puede calcularse como

$$\boxed{Df(\vec{x}_0, \vec{d}) = \nabla f(\vec{x}_0).\vec{d}}$$

Figura 2.76. Cálculo alternativo de la derivada direccional.

Si tomamos el vector $\vec{d}$ unitario, tenemos que la dirección de máximo descenso del campo escalar se obtendrá cuando el producto escalar tome el valor mínimo. Es decir, cuando ambos vectores formen un ángulo de π radianes, luego la dirección de máximo descenso es la marcada por el gradiente en el punto y su sentido es el contrario.

Para obtener el máximo descenso del campo escalar sólo necesitamos obtener el módulo del vector, es decir, la distancia que tenemos que recorrer en la dirección opuesta al gradiente para obtener la máxima disminución de la función objetivo. Dicha magnitud recibe el nombre de **parámetro de descenso** y para obtenerla hay que resolver el siguiente problema unidimensional:

Dados $\overline{x_0}$ como anteriormente y $\vec{d}$ la dirección de máximo descenso en el punto, minimizar la función escalar

$$\boxed{\begin{aligned} &h_{x_0,d} : \mathbb{R}^+ \to \mathbb{R} \\ &\qquad \alpha \to h_{x_0,d}(\alpha) = f(\vec{x}_0 + \alpha\vec{d}) - f(x_0) \end{aligned}}$$

Figura 2.77. Problema unidimensional de optimización planteado en Gradiente.

[40] La convergencia de un algoritmo es cuadrática cuando $\exists \lambda \in (0,1) / \left|x_{i+1} - x^*\right| \leq \lambda \left|x_i - x^*\right|^2$, x^* representa la posición del óptimo.

Lo cual se verifica si

$$h'_{x_0,d}(\alpha) = 0 \Leftrightarrow \nabla f(\vec{x}_0 + \alpha\vec{d}).\vec{d} = 0$$

Figura 2.78. Condición suficiente de solución del problema unidimensional.

El problema unidimensional anterior requiere la resolución de una ecuación que puede hacerse de forma exacta o aproximada. En el método del gradiente su solución es exacta. También hay variantes del método que utilizan un paso, α, fijo y así evitan resolver la ecuación escalar. Esta variante es la que hibridaremos con S.G.O.

En la figura 2.79 se da una representación de dos iteraciones del método del gradiente sobre las curvas de nivel de cierta función $f(x)$ (tomada de Kindelán 2007). En ella se observa que dos líneas de descenso son siempre perpendiculares.

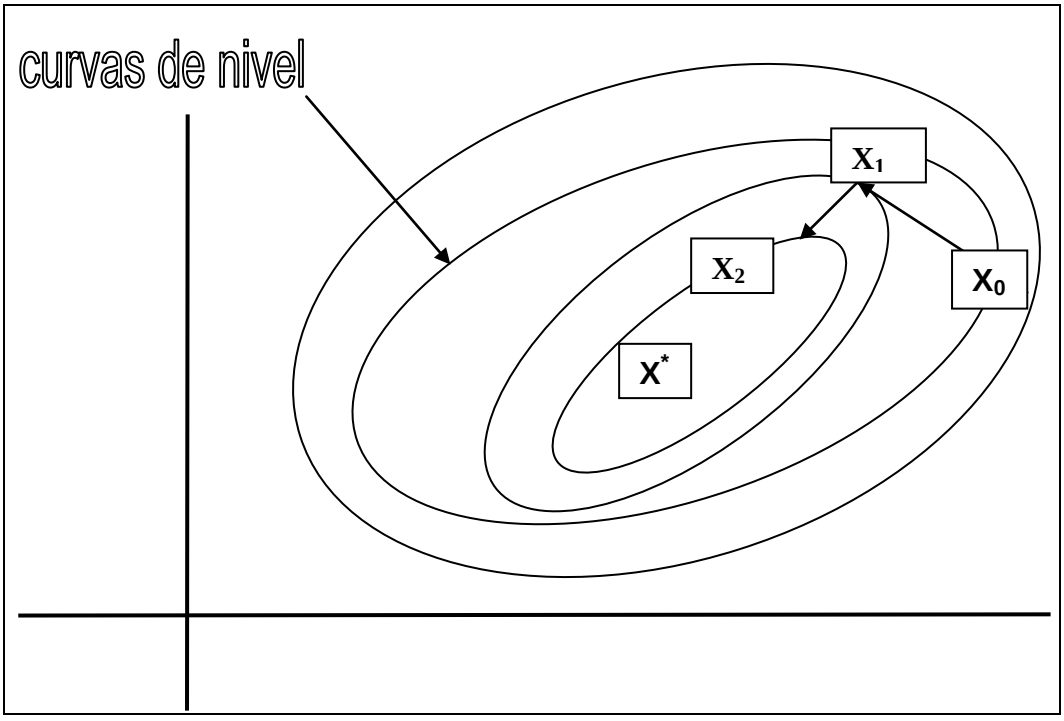


Figura 2.79. Representación del método del Gradiente.

Presentamos en la figura 2.80 el pseudocódigo del método del gradiente (Hernández 2006).

Procedimiento Gradiente (máximo descenso)

Inicializar punto x_0 *, K=0*

Etapa 1: Calcular $\nabla f\left(\overline{x_K}\right)$

Si $\left\|\nabla f\left(\overline{x_K}\right)\right\| = 0 \Rightarrow FIN$

Si no pasar a Etapa 2

Etapa 2: $\vec{d}_k = -\nabla f\left(\overline{x_k}\right)$

Resolver el problema unidimensional (α_k)

$$\min_{\alpha>0} h_{x_k,d_k}(\alpha) = f\left(\overline{x_k} + \alpha\vec{d}_k\right) - f(\vec{x}_k)$$

$x_{k+1} = x_k + \alpha_k$

$k = k+1$

volver a Etapa 1 o acabar cuando k = max_iter

(o bien $\left\|x_{k+1} - x_k\right\| < tol$)

Figura 2.80. Pseudocódigo del método del Gradiente.

7. OTRAS HEURÍSTICAS QUE HIBRIDAR CON S.G.O: VERY SIMPLE OPTIMIZATION, V.S.O.

7.1. INTRODUCCIÓN

La heurística poblacional V.S.O. es un nuevo algoritmo determinístico diseñado por Formato (2013)[41] basado en la idea geométrica que podríamos definir como: "búsqueda de mejores soluciones en las rectas que unen las soluciones buenas".

La idea en la que está inspirado también fundamenta otros algoritmos como Path Relinkin, Encadenamiento de trayectorias, (Alegre 2004) y es, en cierto modo, similar a la de la heurística Nelder-Mead pero, a diferencia de ésta y aquélla, la implementación de V.S.O. es muy simple[42] ya que en vez de construir y modificar el simplex *n+1-dimensional* o encadenar trayectorias entre dos puntos, V.S.O. construye las rectas que unen los agentes de búsqueda con el mejor, a modo de fitness, y considera los puntos medios de cada segmento a la espera de mejorar los resultados.

Inicialmente el algoritmo fue diseñado para la optimización de problemas de antenas pero su versatilidad hace que sea fácilmente modificado y adaptado a cualquier problema de optimización (Formato 2013).

7.2. DESARROLLO DEL ALGORITMO

Partimos del problema

$$\begin{array}{ll} \text{Opt} & f(\vec{x}) \\ s.a. & \vec{x} \in \mathbb{R}^{N_d} / x_i^{\min} \le x_i \le x_i^{\max} \;\; 1 \le i \le N_d \end{array}$$

Figura 2.81. Definición del problema general de optimización en V.S.O.

Para realizar la iteración *j-ésima* partimos de un conjunto formado por *p* puntos de la región factible y una matriz en la que están recogidas sus coordenadas.

$\vec{R}_{j-1}^{\,p} = \sum_{k=1}^{N_d} x_k^{p,j-1} \vec{e}_k$ con $\vec{e}_k$ el correspondiente vector de la base canónica.

Sea $\vec{R}_{j-1}^{*}$ el vector que contiene las coordenadas del mejor valor de la iteración anterior, i.e. *j-1-ésima.* Consideramos los segmentos de rectas que unen todos los elementos del conjunto con el mejor.

[41] A lo largo de esta sección tomaremos muchos elementos de este artículo.
[42] De ahí su nombre.

$$\vec{R}_j^p = \vec{R}_{j-1}^p + \rho\left(\vec{R}_{j-1}^* - \vec{R}_{j-1}^p\right) \qquad 0 \le \rho \le 1$$

Figura 2.82. Construcción de las rectas en V.S.O..

El autor escoge entonces el punto medio de los segmentos tomando $\rho = 0.5$ y actualiza la matriz con las nuevas coordenadas y el vector de las coordenadas del mejor, que puede o no haber variado.

También cimenta la justificación para la elección de ρ en los resultados empíricos obtenidos, pero especifica que no ha sido investigado y abre la posibilidad a la elección de nuevos valores (Formato 2013).

En la figura 2.83 (tomada de Formato 2013) se representa V.S.O.

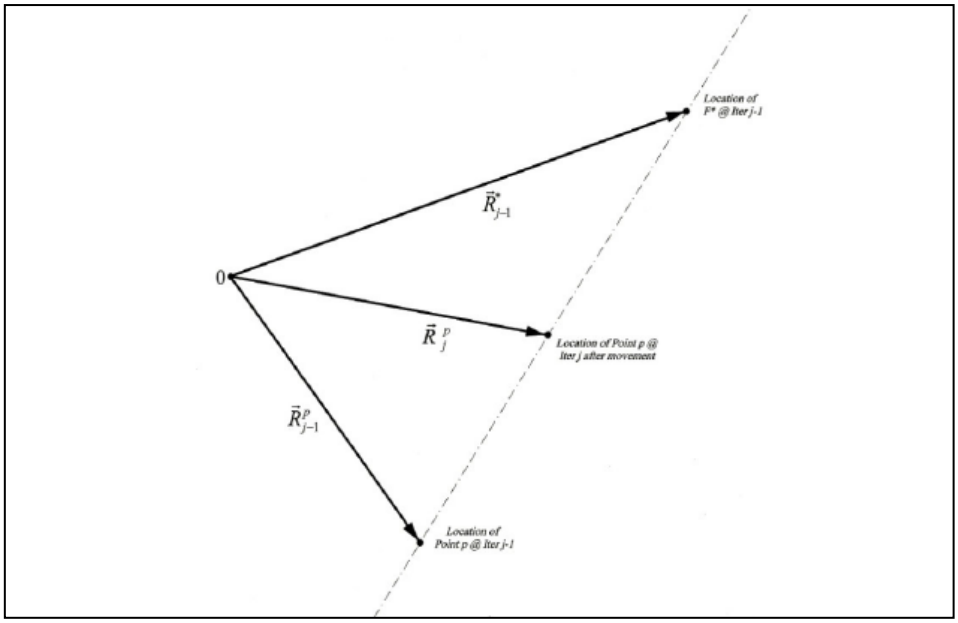


Figura 2.83. Representación de V.S.O.

Inicializa el conjunto tomando los puntos en las intersecciones de la diagonal principal de la región factible con líneas paralelas a los ejes coordenados equiespaciadas con distancia γ , $0 \le \gamma \le 1$ un parámetro escogido uniformemente atendiendo a los requerimientos del problema planteado. Esto es

$$\vec{D} = \vec{X}_{\min} + \gamma\left(\vec{X}_{\max} - \vec{X}_{\min}\right)$$

Figura 2.84. Inicialización de *p* en V.S.O.

No detalla el criterio de parada seguido.

El pseudocódigo de V.S.O. es

Procedimiento V.S.O.

I-*Inicialización:*
j=0
Algoritmo de inicialización
Encontrar el mejor del grupo $\vec{R}_0^*$
II- Hacer hasta (criterio de parada)
j=j+1
construir las rectas que unen los puntos con el mejor
considerar los punto medios de las rectas
hallar el mejor del grupo $\vec{R}_j^*$.

Figura 2.85. Pseudocódigo de V.S.O.

Hemos realizado algunos cambios en estos pasos con el fin de hibridarlo con S.G.O. Llegado el momento los expondremos con detalle.

8. CONCLUSIONES

En el presente capítulo hemos realizado un exhaustivo estudio de S.G.O., tanto a nivel de sus motivaciones físicas como de su implementación, y hemos comprobado que su inspiración física y matemática está teóricamente bien sustentada.

Ha llegado el momento de comprobar empíricamente el alcance de nuestra heurística, tanto en solitario como hibridado con algunas de las heurísticas expuestas, así como algunos innovadores algoritmos que iremos detallando.

Este será nuestro objetivo a lo largo del resto de los capítulos de la presente tesis.

"¿Te lo dijo la computadora central de la ciudad? ¡R2D2, sabes bien que no debes confiar en una computadora extraña!"

C3PO Star Wars episodio V (Irving Kershner 1980)

CAPÍTULO 3

ANÁLISIS DE LA ROBUSTEZ DE S.G.O.

1. INTRODUCCIÓN

El algoritmo S.G.O. tiene 2 componentes bien diferenciadas. Por un lado está su **componente aleatoria**, ya que las posiciones y velocidades de los agentes de búsqueda se inicializan al azar tanto al principio como tras la salida de la región factible; sin embargo también tiene una fuerte **componente determinista** ya que la posición que un asteroide ocupará en el instante siguiente se calcula unívocamente, salvo errores de redondeo, a partir de su posición, velocidad y aceleración en la posición actual.[1]

Su cualidad aleatoria nos obliga a reflexionar acerca de la **robustez** del algoritmo, es decir, ¿la repetición de la ejecución del algoritmo varias veces manteniendo fijos tanto los parámetros como el número de llamadas a la función objetivo conllevará resultados e infactibilidades[2] similares o dispares?

Para responder a esta cuestión realizaremos una primera fase de pruebas con una muestra de instancias de dimensiones 2 y 4 y recopilaremos algunos parámetros estadísticos que nos ayuden a responderla.

La muestra elegida pertenece a un grupo de funciones que han sido utilizadas como *benchmark functions* en diversos trabajos con diferentes algoritmos (Laguna y Martí 2005 ; Tu y Lu 2004 ; Formato 2010 b). Un estudio detallado de algunas de ellas se hará en el apartado siguiente y también puede encontrarse en Yao y otros (1999) y en las páginas web http://www-optima.amp.i.kyotou.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page364.htm ; http://solon.cma.univie.ac.at/glopt.html .

Muchas son multimodales y, por tanto de difícil optimización (Zapatero 2009).

También recogemos la expresión explícita de cada una, junto al dominio de optimización, en el apéndice 1.

[1] Suponiendo fijos los parámetros α, β y G.

[2] Recordemos que denominamos infactibilidades a las salidas de los asteroides de la región factible.

2. PRIMERA FASE DE PRUEBAS CON S.G.O.

Aunque intuimos que la topología de la función y la región factible juegan un papel fundamental en el éxito del heurístico S.G.O., no tenemos constancia empírica. Además existen pocos trabajos que nos indiquen cuáles son los valores apropiados para ejecutar S.G.O. Por ello, en principio, elegiremos los valores de los parámetros -α, β y G -, del número de asteroides -*n-,* y del número máximo de movimientos de cada agente -*n_iter*- un poco "a ciegas". El parámetro que tiene, a nuestro entender, mayor influencia en los resultados es *G*. Así, hemos comenzando variándolo en un rango de 6 valores, específicos en cada caso, a la vez que mantenemos fijos el resto de valores.

La inicialización de las posiciones y de las velocidades de los asteroides, tanto en la primera fase como tras la salida de un asteroide de la región factible, se ha hecho aleatoriamente dentro del dominio de cada instancia[3].

En todos los ejemplos de dimensión 2 hemos usado $n = 1000$ asteroides con $n\_iter = 1000$ movimientos. La razón de tal elección ha sido que, como se verá más adelante, hemos obtenido valores muy buenos[4] en un tiempo computacional pequeño. Si el problema no requiere tanta precisión podrían rebajarse estas cantidades. De hecho, a modo de ejemplo, hemos ejecutado el programa con la primera función, Space-Paper, para los valores $n = 100$ y $n\_iter = 1000$ y hemos comprobado que los resultados siguen siendo bastante aceptables. No obstante, la pérdida de precisión puede ser más acusada dependiendo de la instancia considerada.

En todas las ejecuciones hemos elegido $\alpha = 0.005$, ya que, como se comprobará posteriormente, las influencias entre los asteroides no han sido significativas en ninguna de las funciones estudiadas en el presente apartado.

Como rango de detección, r_d, hemos usado en todos los casos el valor 10^{-6} pues consideramos que las aproximaciones a las derivadas parciales han de hacerse en un intervalo suficientemente pequeño para que sean aceptables, evitando en lo posible los errores de redondeo.

El valor del parámetro β ha variado en función de la dimensión del problema. En cada caso especificaremos sus valores concretos.

Una vez fijados todos los parámetros hemos repetido el algoritmo 50 veces de las que hemos extraído 10 resultados que se recogen en las distintas filas dentro de una misma

[3] Dominio definido por el problema de optimización planteado.
[4] Errores menores que 10^{-6}.

columna en cada tabla. Pueden verse todos los valores en la carpeta "PRIMEROS RESULTADOS" del CD adjunto

Para realizar las correspondientes experiencias computacionales hemos implementado el algoritmo en distintos equipos, entre ellos un Pentium 4, 2.6 GHz y un Intel(R) Core(TM) i5-4460 a 3.2 GHz, usando para ello los programas MATLAB 6.5, MATLAB R2006, MATLAB R2014 y MATLABR2015a. Los tiempos de ejecución han sido muy dispares, por ello no los hemos registrado, pero en todos los casos mínimos. También hemos desarrollado la implementación de los algoritmos en PASCAL usando como compilador DELPHI 5. Los tiempos de computación en este caso han sido inferiores aunque los resultados han permanecido similares.

Además de considerar el óptimo y el punto donde se alcanza, hemos tenido en cuenta el número de infactibilidades. En esta fase de la experimentación no hemos registrado tiempos de ejecución ni número de llamadas a la función objetivo aunque éstas últimas, como veremos posteriormente, solo dependen de n y n_iter y pueden calcularse exactamente.

Hemos expuesto todos los datos en cada tabla usando tres colores distintos para indicar los tres datos diferentes y hemos mantenido el siguiente orden:

Negro: valor del óptimo.

Azul: punto donde se alcanza (coordenada x e y)

Rojo: número de infactibilidades.

Por ejemplo, el dato de la figura 3.1 muestra que el valor del óptimo es -130.826375, que está ubicado en (-2.833522,-2.822867) y que el número de infactibilidades ha sido 15 sobre 1.000.000 de posibles salidas (el producto del número de asteroides por el número de movimientos de cada asteroide) lo que supone un 0%.

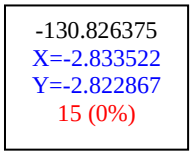


Figura 3.1. Exposición de los resultados

Dentro de cada columna, es decir, fijados todos los parámetros, el valor absoluto del número de infactibilidades no permanece constante en cada repetición del algoritmo pero sí su porcentaje, lo que avala la robustez del algoritmo, por ello sólo hemos indicado los porcentajes al principio de cada columna.

Al partir de *funciones test,* el valor del óptimo global ha sido conocido de antemano aunque en ocasiones con solo 4 dígitos significativos. Cuando hemos dispuesto de datos con suficientes dígitos de aproximación hemos marcado en negrita los **resultados con errores menores que 10^{-6} y entenderemos que hemos alcanzado el óptimo**. Si la aproximación de la que partíamos era peor hemos marcado en negrita **los resultados que han mejorado el dato de la *función test*.**

La primera función que exponemos no está en la lista anteriormente citada. Ha sido extraída directamente del artículo de Hsiao y otros (2005) en el que se introduce S.G.O.

2.1. LA FUNCIÓN SPACE-PAPER

$$f(x_1, x_2) = x_1^4 - 16 \cdot x_1^2 + 0.5x_1 + x_2^4 - 16x_2^2 + 0.5x_2$$

Dominio de optimización: $[-50,50] \times [-50,50]$

Ésta es una de las funciones que Tung Hsiao y otros (2005) usan en su artículo como función test para probar S.G.O. En la siguiente figura vemos su representación gráfica.

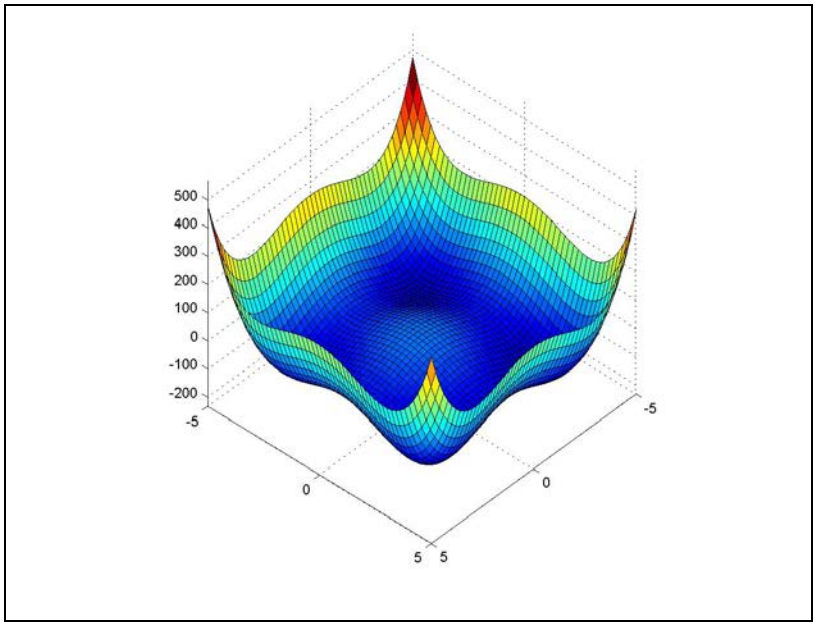


Figura 3.2. Representación con MATLAB de la función SPACE_PAPER .

Por técnicas exactas pertenecientes al campo del Análisis Matemático se encuentran los 9 extremos relativos que la función tiene. El mínimo absoluto es -130.832323 y se encuentra en el punto $(-2.836207, -2.836207)$.

Hemos hecho una serie de experimentaciones dejando los valores de α, β, n y n_iter fijos y variando *G* en el rango *G = 1 , 10 , 100 , 1000 , 10000 y 100000.*

Tal y como se aprecia en las tablas siguientes los mejores resultados se han obtenido para $\beta = 0.99$ y $G = 10^3$ / $G = 10^4$ **en los que hemos alcanzado el óptimo**. No obstante, hemos querido recoger los valores para β = 1.01 porque los autores parecen obtener mejores resultados en este caso.

En ninguno de los casos expuestos los asteroides han quedado atrapados en óptimos locales.

En los casos $G = 10^4$ y $G = 10^5$ las infactibilidades son demasiado altas por los valores que toma el gradiente fuera de la zona donde se encuentra el óptimo, sin embargo los resultados son muy buenos.

Tabla 3.3[5]

$\alpha = 0.005$, *β = 0.99 , G = 1 , 10 , 100 , 1000 , 10000 y 100000 , n=1000 , n_iter=1000*

Nº ejec.	*G = 1*	*G = 10*	*G = 100*	*G = 1000*	*G = 10000*	*G = 100000*
1	-130.826375 X=-2.833523 Y=-2.822867 15 (0%)	-130.830206 X=-2.838074 Y=-2.844077 7 (0%)	-130.832295 X= -2.837066 Y=-2.835872 1372 (0.14%)	-130.832320 X=-2.836262 Y=-2.835913 20963 (2.1%)	-130.832322 X=-2.836109 Y=-2.836305 369364(36.9%)	-130.813790 X=-2.829583 Y=-2.813078 989187 (98.9%)
2	-130.810007 X=-2.858914 Y=-2.823066 27	-130.828892 X=-2.837085 Y=-2.825915 6	-130.832264 X=-2.837545 Y=-2.836403 1401	-130.832322 X=-2.836160 Y=-2.836130 20607	-130.832322 X=-2.836374 Y=-2.836238 373929	-130.794045 X=-2.811998 Y=-2.811498 989273
3	-130.829621 X=-2.829915 Y=-2.842849 21	-130.830874 X=-2.842880 Y=-2.835687 6	-130.832315 X=-2.836641 Y=-2.836428 1288	-130.832322 X=-2.836138 Y=-2.836266 20230	-130.832321 X=-2.836133 Y=-2.836377 364635	-130.771766 X=-2.853438 Y=-2,875660 989327
4	-130.816766 X=-2.814512 Y=-2.832327 17	-130.831802 X=-2.838276 Y=-2.839650 6	-130.832317 X=-2.836410 Y=-2.835840 1257	-130.832316 X=-2.835914 Y=-2.836534 19809	-130.832321 X=-2.836403 Y=-2.836062 367450	-130.746890 X=-2.784286 Y=-2.837209 989325
5	-130.826659 X=-2.823811 Y=-2.831457 21	-130.831965 X=-2.833552 Y=-2.838213 4	-130.832317 X=-2.835799 Y=-2.836296 1313	**-130.832323** **X=-2.836178** **Y=-2.836224** **21190**	-130.832322 X=-2.836197 Y=-2.836036 374491	-130.808965 X=-2.833254 Y=-2.809337 989485
6	-130.830063 X=-2.840205 Y=-2.828849 18	-130.831256 X=-2.837487 Y=-2.830596 1	-130.832313 X=-2.836427 Y=-2.835709 1225	-130.832321 X=-2.836384 Y=-2.836321 22818	-130.832322 X=-2.836310 Y=-2.836249 374203	-130.6412655 X=-2.798307 Y=-2.768282 989282
7	-130.828884 X=-2.845699 Y=-2.840229 20	-130.832168 X=-2.836989 Y=-2.834160 12	-130.832314 X=-2.835842 Y=-2.835826 1266	-130.832313 X=-2.836739 Y=-2.836069 21393	-130.832322 X=-2.836345 Y=-2.836293 367897	-130.676162 X=-2.882658 Y=-2.887199 989447
8	-130.822042 X=-2.821254 Y=-2.845999 18	-130.831669 X=-2.833277 Y=-2.832790 4	-130.832279 X=-2.836906 Y=-2.835282 1309	-130.832321 X=-2.836028 Y=-2.836256 21654	**-130.832323** **X=-2.836297** **Y=-2.836186** **376113**	-130.802160 X=-2.808183 Y=-2.848718 989328
9	-130.813598 X=-2.812601 Y=-2.830940 19	-130.831480 X=-2.840377 Y=-2.839155 7	-130.832309 X=-2.835602 Y=-2.836431 1250	**-130.832323** **X=-2.836143** **Y=-2.836219** **20830**	-130.832322 X=-2.836106 Y=-2.836088 374591	-130.774584 X=-2.794571 Y=-2.827179 989366
10	-130.823741 X=-2.827445 Y=-2.822410 23	-130.832165 X=-2.837448 Y=-2.838038 8	-130.832305 X=-2.836141 Y=-2.835474 1265	-130.832314 X=-2.835707 Y=-2.836101 21056	**-130.832323** **X=-2.836184** **Y=-2.836185** **381621**	-130.818253 X=-2.816904 Y=-2.828074 989306

Figura 3.3. Tabla de resultados SPACE-PAPER (I)

Los tiempos de las ejecuciones han sido similares, ya que, como veremos posteriormente con detalle, solo dependen de *n* y *n_iter.* Han rondado los 49 segundos.

[5] Hemos resaltado en negrita los resultados con error menor que 10^{-6}.

A modo de ejemplo presentamos uno de los resultados obtenidos para $n=100$ en las mismas condiciones de la tabla 3.3. Aunque no son tan buenos como para $n=1000$ observamos que son similares a los de la tabla 3.6 obtenida para $\beta=1.01$ $n=1000$. El resto de los parámetros ha permanecido igual.

Tabla 3.4

$\alpha=0.005$, *β = 0.99 , G = 1 , 10 , 100 , 1000 , 10000 y 100000 , n=100 , n_iter=1000*

Nº ejec	*G=1*	*G=10*	*G=100*	*G=1.000*	*G=10.000*	*G=100.000*
1	-130.456630 X=-2.940858 Y=-2.853058 1 (0%)	-130.807515 X=-2.822270 Y=-2.812111 0 (0%)	-130.831063 X=-2.830092 Y=-2.834892 132 (0%)	-130.832043 X=-2.839078 Y=-2.835555 1859 (0.2%)	-130.832225 X=-2.836976 Y=-2.837766 45611 (0.5%)	-130.299849 X=-2.785574 Y=-2.715383 98976 (9.9%)

Figura 3.4. Tabla de resultados SPACE-PAPER (II).

También a modo de ejemplo presentamos uno de los resultados obtenidos para $n\_iter=100$ en las mismas condiciones de la tabla 3.3. Son similares a los registrados en la tabla 3.6, obtenida para $\beta=1.01$ $n=1000$.

Tabla 3.5

$\alpha=0.005$, *β = 0.99 , G = 1 , 10 , 100 , 1.000 , 10.000 y 100.000 , n=1000 , n_iter=100*

Nº ejec	*G=1*	*G=10*	*G=100*	*G=1.000*	*G=10.000*	*G=100.000*
1	-130.8316756 X=-2.84065558 Y=-2.8357208 400119 (40%)	-130.8273385 X=-2.82444481 Y=-2.84028978 11032 (11%)	-130.8281428 X=-2.82645572 Y=-2.83030397 6679 (6.7%)	-130.8158961 X=-2.83285392 Y=-2.81380587 74222 (74.2%)	-130.8288507 X=-2.84390438 Y=-2.84314222 95622 (95.6%)	-130.8262819 X=-2.82765 Y=-2.84687467 98210 (98.2%)

Figura 3.5. Tabla de resultados SPACE-PAPER (III).

Como hemos comentado anteriormente, en la tabla 3.6 recogemos los datos que hemos obtenido en las mismas condiciones que la tabla 3.3 con *β = 1.01*. Los resultados en nuestro caso han resultado peores que los recogidos en la tabla 3.3. El efecto de "espacio en contracción" no parece ser eficiente para esta instancia.

Tabla 3.6

$\alpha = 0.005$, *β = 1.01 , G = 1 , 10 , 100 , 1000, 10000 y 100000 , n=1000 , n_iter=1000*

Nº ejec.	*G = 1*	*G = 10*	*G = 100*	*G = 1000*	*G = 10000*	*G = 100000*
1	-130.812211 X=-2.843420 Y=-2.812207 8032 (0.8%)	-130.735554 X=-2.877925 Y=-2.871110 7 052 (0.7%)	-130.827232 X= -2.848564 Y=-2.838312 13698 (1.4%)	-130.832202 X=-2.836794 Y=-2.834365 126437 (12.6%)	-130.832277 X=-2.835480 Y=-2.835273 744829 (74.5%)	-130.771545 X=-2.836436 Y=-2.792470 9899143 (99%)
2	-130.830235 X=-2.843436 Y=-2.832694 7932	-130.807307 X=-2.857507 Y=-2.853991 6999	-130.828919 X=-2.839844 Y=-2.826586 13886	-130.830906 X=-2.830248 Y=-2.833300 130526	-130.831181 X=-2.830684 Y=-2.838426 744116	-130.680831 X=-2.881403 Y=-2.886942 989169
3	-130.687071 X=-2.768494 Y=-2.841230 7823	-130.787214 X=-2.813503 Y=-2.806272 7074	-130.832003 X=-2.836298 Y=-2.839353 14001	-130.831333 X=-2.831304 Y=-2.833626 128845	-130.832063 X=-2.835758 Y=-2.833441 746789	-130.673667 X=-2.843618 Y=-2,765601 989439
4	-130.759867 X=-2.788995 Y=-2.828886 7723	-130.790882 X=-2.800959 Y=-2.828628 7147	-130.825632 X=-2.841189 Y=-2.849685 13855	-130.832256 X=-2.837639 Y=-2.836355 127589	-130.820557 X=-2.837853 Y=-2.817118 744489	-130.807016 X=-2.851106 Y=-2.859800 989417
5	-130.788684 X=-2.802913 Y=-2.820133 7829	-130.798136 X=-2.848222 Y=-2.805802 7137	-130.824414 X=-2.840135 Y=-2.851322 14094	-130.832270 X=-2.837352 Y=-2.836762 124523	-130.832116 X=-2.833676 Y=-2.836115 745333	-130.812895 X=-2.858865 Y=-2.845394 989464
6	-130.814618 X=-2.852491 Y=-2.852953 7956	-130.831443 X=-2.840031 Y=-2.832653 7084	-130.812101 X=-2.825250 Y=-2.858639 13711	-130.832256 X=-2.837639 Y=-2.836355 127589	-130.830550 X=-2.828853 Y=-2.837202 745993	-130.761364 X=-2.878161 Y=-2.815809 989241
7	-130.830514 X=-2.834211 Y=-2.828983 7969	-130.822093 X=-2.826711 Y=-2.851240 7105	-130.802250 X=-2.855292 Y=-2.812329 14102	-130.826596 X=-2.833089 Y=-2.823225 129121	-130.830230 X=-2.844201 Y=-2.837088 749510	-130.7907370 X=-2.820195 Y=-2.868183 989184
8	-130.819235 X=-2.853702 Y=-2.846073 7946	-130.773831 X=-2.814461 Y=-2.8726291 7032	-130.830114 X=-2.835568 Y=-2.827946 13976	-130.831436 X=-2.841252 Y=-2.837614 131259	-130.831234 X=-2.836443 Y=-2.830399 746270	-130.697017 X=-2.803519 Y=-2.779633 989267
9	-130.822638 X=-2.850726 Y=-2.826795 7914	-130.815036 X=-2.855843 Y=-2.848329 7083	-130.830873 X=-2.842609 Y=-2.838171 13945	-130.832025 X=-2.833486 Y=-2.834860 127379	-130.831491 X=-2.831181 Y=-2.835456 746630	-130.814210 X=-2.820963 Y=-2.817977 989391
10	-130.588192 X=-2.873610 Y=-2.756679 7902	-130.806255 X=-2.62379 Y=-2.846988 7124	-130.823345 X=-2.843696 Y=-2.821268 13640	-130.830869 X=-2.830587 Y=-2.832527 126840	-130.828410 X=-2.825208 Y=-2.835343 747397	-130.802462 X=-2.854766 Y=-2.860165 989304

Figura 3.6. Tabla de resultados SPACE-PAPER (IV).

A pesar de que los resultados recogidos en la tabla 3.6 no son tan buenos como los anteriores, sí se mantienen en algunos casos 4 dígitos de aproximación.

Los tiempos de las ejecuciones han rondado los 49 segundos, obviamente como en el caso de la tabla 3.3.

2.2. LA FUNCIÓN DE BRANIN EN DIMENSIÓN 2

$$f(x_1,x_2)=\left(x_2-\frac{5}{4\pi^2}x_1^{\,2}+\frac{5x_1}{\pi}-6\right)^2+10\left(1-\frac{1}{8\pi}\right)cos(x_1)+10$$

Dominio de optimización: $[-5,15]\times[-5,15]$

Una característica importante de esta función es que el gradiente es muy grande cerca de la región en la que están los distintos puntos donde se alcanza el óptimo, ya que es multimodal.

Su representación gráfica es

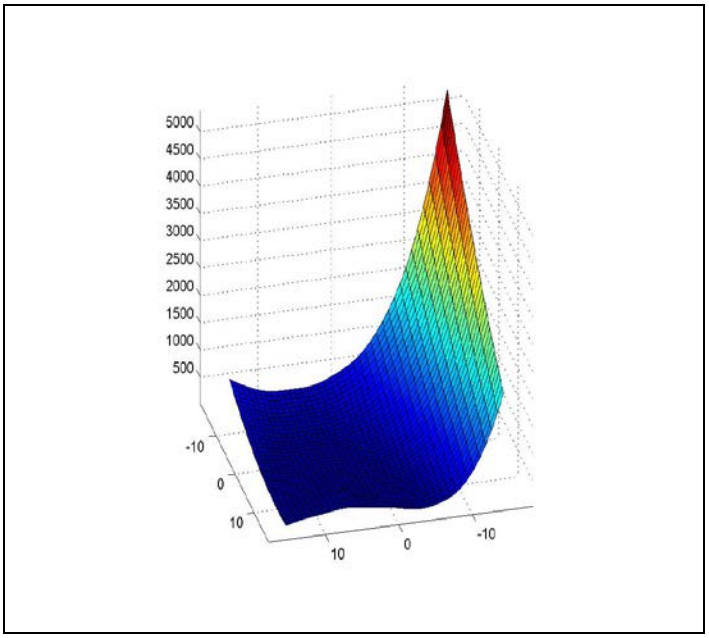


Figura 3.7. Representación con MATLAB de la función BRANIN.

Según Yao y otros (1999) el óptimo global es 0.397887 y uno de los puntos en los que se alcanza es $(9.42478, 2.475)$.

En funciones de este tipo[6] reducir la región de factibilidad, y con ella la zona donde el gradiente es elevado, ayuda mucho a controlar las infactibilidades y obtener mejores resultados con menos iteraciones. El espectacular incremento de infactibilidades de $G=10^5$ a $G=5\cdot10^5$ se explica precisamente por este hecho.

Nuestro objetivo ha consistido en comprobar si el programa podía alcanzar el óptimo con una precisión mayor que 10^{-6} y obtener distintas posiciones donde se encuentra. **Ambos retos han sido logrados.**

[6] También le sucede a la función de Rosenbrock.

De nuevo podemos observar que los asteroides no han quedado atrapados en óptimos locales.

Sólo exponemos los datos para $\beta = 0.99$ porque con $\beta = 1.01$ hemos obtenido peores resultados.

Tabla 3.8[7]

$\alpha = 0.005$, *β = 0.99 y G = 10 , 100 , 1000 , 10000 , 100000 y 500000 , n=1000 , n_iter=1000*

Nº ejec.	*G = 10*	*G = 100*	*G = 1000*	*G = 10000*	*G = 100000*	*G = 500000*
1	0.397946 X=9.421461 Y=2.249799 56541 (5.6%)	0.397975 X= 9.423305 Y=2.257595 48742 (4.9%)	0.397889 X=9.4254899 Y=2.250758 25363 (2.5%)	0.397888 X=-3.141067 Y=12.248752 3757 (0.37%)	**0.397887** **X=3.141576** **Y=2.249993** **7529 (0.8%)**	0.397902 X=9.423437 Y=2.246382 662354 (66.2%)
2	0.398025 X=3.141811 Y=-2.261557 56343	0.397969 X=9.421012 Y=2.243304 49415	**0.397887** **X=-3.141932** **Y=12.250682** **25926**	**0.397887** **X=3.141737** **Y=2.250013** **3769**	**0.397887** **X=9.424727** **Y=2.249896** **7089**	0.397917 X=9.423796 Y=2.244179 662015
3	0.398183 X=3.134104 Y=2.261076 56797	0.398108 X=3.144761 Y=2.234359 48569	0.397975 X=-3.138266 Y=12.236109 25358	**0.397887** **X=9.424646** **Y=2.249820** **3840**	**0.397887** **X=9.424769** **Y=2.244986** **6707**	0.398234 X=-3.147231 Y=12.277425 662431
4	0.398004 X=3.136797 Y=2.251249 56755	0.398031 X=-3.136710 Y=12.232962 48693	0.397894 X=-3.141239 Y=12.251764 25006	**0.397887** **X=-3.141852** **Y=12.250414** **3877**	**0.397887** **X=3.141587** **Y=2.250013** **6833**	0.397896 X=-3.140265 Y=12.246225 663530
5	0.397899 X=3.140082 Y=2.251975 57091	0.398124 X=9.431291 Y=2.249411 49260	0.397892 X=3.142522 Y=2.248430 25712	**0.397887** **X=3.141755** **Y=2.249755** **3801**	**0.397887** **X=3.141620** **Y=2.250007** **6862**	0.397990 X=9.423167 Y=2.258261 664035
6	0.397918 X=3.139639 Y=2.255002 56682	0.397953 X=3.144514 Y=2.252660 48488	0.397898 X=3.140101 Y=2.251377 25650	**0.397887** **X=-3.141577** **Y=12.250057** **3992**	**0.397887** **X=9.424812** **Y=2.250045** **6665**	0.398361 X=9.431627 Y=2.271220 663146
7	0.397947 X=-3.139394 Y=12.250750 56515	0.397934 X=3.138694 Y=2.249731 48822	0.397893 X=-3.142691 Y=12.253196 24687	**0.397887** **X=9.424824** **Y=2.250382** **3791**	**0.397887** **X=3.141556** **Y=2.250059** **6876**	0.398110 X=3.136482 Y=2.263979 662871
8	0.397993 X=9.428627 Y=2.247212 57104	0.397909 X=9.424154 Y=2.253942 48776	0.397975 X=9.421594 Y=2.241179 25642	**0.387887** **X=-3.141584** **Y=12.250151** **3867**	**0.397887** **X=3.141603** **Y=2.249938** **7037**	0.398065 X=3.135533 Y=2.253364 662586
9	0.397923 X=3.139757 Y=2.247085 55940	0.397899 X=-3.140396 Y=12.244814 49150	0.397889 X=-3.140926 Y=12.248499 25755	**0.397887** **X=-3.141515** **Y=12.249834** **3696**	**0397887** **X=9.424769** **Y=2.249869** **6707**	0.397893 X=-3.140420 Y=12.247218 661966
10	0.397996 X=9.421336 Y=2.240095 56715	0.398030 X=9.430224 Y=2.254955 47804	0.397903 X=9.425058 Y=2.254183 25530	**0.397887** **X=3.141483** **Y=2.250483** **4036**	**0.397887** **X=3.141555** **Y=2.250110** **7316**	0.397938 X=3.140923 Y=2.243540 662814

Figura 3.8. Tabla de resultados BRANIN.

En esta ocasión los tiempos de ejecución han permanecido alrededor de 45 segundos.

[7] Hemos resaltado en negrita los resultados con error menor que 10^{-6}.

2.3. LA FUNCIÓN DE SCHWEFEL EN DIMENSIÓN 2

$$f(\overline{x}) = \sum_{i=1}^{n} (-x_i \sin\sqrt{|\overline{x}|})$$

Dominio de optimización: $[-500,500]\times[-500,500]$.

Esta es otra función multimodal pero, su gradiente no aumenta tanto como en el caso anterior. A continuación damos la representación de la "montaña rusa".

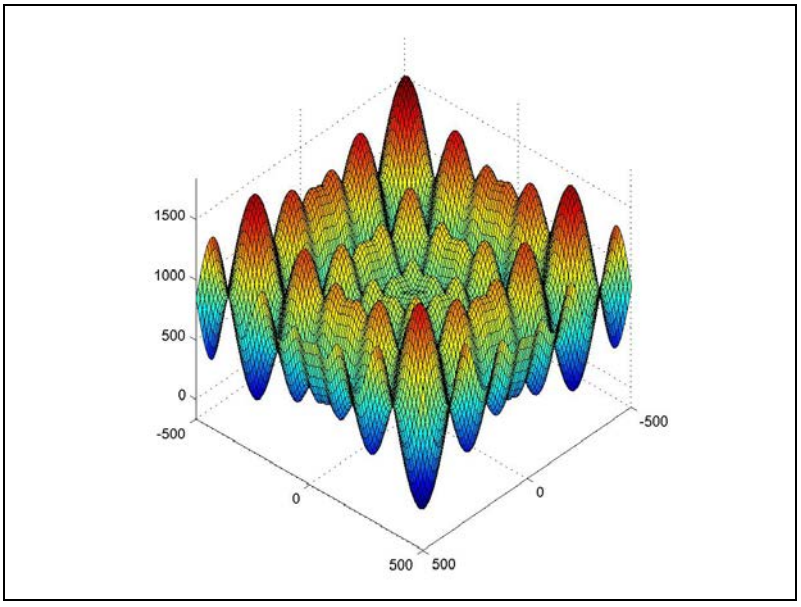


Figura 3.9. Representación con MATLAB de la función SCHWEFEL.

Según Yao y otros (1999) el óptimo global es[8] -837.9657.

Nuestro objetivo en este caso es ver si el algoritmo es capaz de saltar sobre los óptimos locales sin quedarse atrapado en ellos para localizar el óptimo global y como se aprecia en la tablas, **en muchos casos lo ha conseguido**.

Por la topología de la curva, los asteroides son constantemente acelerados y decelerados, así las infactibilidades se mantienen controladas a lo largo de todos los valores de G.

De nuevo observamos que los asteroides no han sido capturados en óptimos locales.

En la tabla 3.10 presentamos algunos de los resultados obtenidos.

[8] Dado que solo conocemos 4 decimales de aproximación del óptimo, hemos mejorado los resultados en muchas ocasiones.

Tabla 3.10[9]

$\alpha = 0.005$, *β = 0.99 y G = 100 , 1.000 , 10.000 , 100.000 , 10^6 y $5 \cdot 10^6$ n=1000 , n_iter=1000*

Nº ejec.	*G = 100*	*G = 1.000*	*G = 10.000*	*G = 100.000*	*G = 10^6*	*G = 5*10^6*
1	-837.964583 X= 421.053926 Y=420.921989 323 (0.0%)	**-837.965775** **X=420.980876** **Y=420.966722** **217 (0.0%)**	**-837.965763** **X=420.960350** **Y=420.973031** **161 (0.0%)**	**-837.965768** **X=420.965512** **Y=420.974699** **196 (0.0%)**	**-837.965771** **X=420.966086** **Y=420.964304** **313 (0.0%)**	**-837.965774** **X=420.968323** **Y=420.967416** **12569 (12.6%)**
2	-837.892922 X=421.699521 Y=420.760938 279	-837.965616 X=420.989026 Y=420.939717 216	**-837.965769** **X=420.962172** **Y=420.968018** **162**	**-837.965773** **X=420.971952** **Y=420.968397** **207**	**-837.965771** **X=420.972175** **Y=420.967537** **302**	**-837.965774** **X=420.967000** **Y=420.968085** **12512**
3	-837.951621 X=421.218534 Y=420.745664 290	**-837.965771** **X=420.965989** **Y=420.965101** **230**	**-837.965772** **X=420.972298** **Y=420.969092** **164**	**-837.965774** **X=420.968058** **Y=420.968997** **186**	**-837.965774** **X=420.968895** **Y=420.970354** **272**	**-837.965766** **X=420.966195** **Y=420.961207** **13780**
4	-837.806866 X=421.225878 Y=422.060895 345	**-837.965756** **X=420.972854** **Y=420.957522** **237**	**-837.965769** **X=420.962653** **Y=420.969371** **181**	**-837.965765** **X=420.962305** **Y=420.963481** **187**	**-837.965774** **X=420.970597** **Y=420.967904** **295**	**-837.965774** **X=420.968947** **Y=420.968925** **13743**
5	-837.939027 X=421.273700 Y=420.623809 314	-837.965449 X=420.918243 Y=420.973659 240	**-837.965773** **X=420.967305** **Y=420.971699** **178**	**-837.965774** **X=420.968939** **Y=420.967459** **225**	**-837.965774** **X=420.968597** **Y=420.969378** **304**	-837.965757 X=420.969231 Y=420.980519 12779
6	-837.920429 X=421.408463 Y=420.561311 338	-837.965065 X=421.043363 Y=420.961643 223	**-837.965772** **X=420.972171** **Y=420.970307** **163**	**-837.965774** **X=420.968321** **Y=420.967191** **240**	**-837.965770** **X=420.974653** **Y=420.969082** **294**	**-837.965773** **X=420.970328** **Y=420.967108** **13200**
7	-837.918017 X=421.580375 Y=420.903178 339	-837.965562 X=420.930868 Y=420.953034 248	**-837.965769** **X=420.964585** **Y=420.964305** **146**	**-837.965767** **X=420.966364** **Y=420.961926** **217**	**-837.965773** **X=420.966200** **Y=420.971035** **283**	**-837.965774** **X=420.968530** **Y=420.970520** **12515**
8	-837.959370 X=421.154794 Y=420.841705 301	**-837.965772** **X=420.965360** **Y=420.970395** **242**	**-837.965773** **X=420.966194** **Y=420.970869** **163**	**-837.965773** **X=420.967457** **Y=420.970842** **232**	**-837.965774** **X=420.970548** **Y=420.968365** **271**	**-837.965772** **X=420.965242** **Y=420.966943** **13070**
9	-837.783432 X=420.057892 Y=421.753320 297	-837.965345 X=421.026656 Y=420.975711 221	**-837.965771** **X=420.966083** **Y=420.964937** **194**	**-837.965772** **X=420.968827** **Y=420.964730** **180**	**-837.965764** **X=420.959680** **Y=420.967924** **254**	**-837.965770** **X=420.964565** **Y=420.965286** **12299**
10	-837.884303 X=421.575081 Y=421.495892 325	-837.965654 X=420.946362 Y=420.989918 242	**-837.965773** **X=420.970641** **Y=420.970786** **174**	**-837.965774** **X=420.968321** **Y=420.967191** **240**	**-837.965771** **X=420.968072** **Y=420.973403** **303**	**-837.965773** **X=420.969244** **Y=420.971139** **12729**

Figura 3.10. Tabla de resultados SCHWEFEL 2.

En esta ocasión los tiempos de cada ejecución han sido algo inferiores a 55 segundos. El número de llamadas a la función objetivo es fijo porque solo depende de *n* y *n_iter,* como veremos con detalle más adelante, pero los tiempos lógicamente dependen de la definición de cada función.

[9] Hemos resaltado en negrita los resultados mejores que el de partida.

2.4. LA FUNCIÓN DE EASOM EN DIMENSIÓN 2

$$f(x_1, x_2) = -cos(x_1)cos(x_2)e^{-(x_1-\pi)^2-(x_2-\pi)^2}$$

Dominio de optimización: $[-100,100]\times[-100,100]$

Esta función tiene un único óptimo global mientras que en el resto del dominio es bastante plana, por efecto de la exponencial negativa. Es una función de mal comportamiento para S.G.O. ya que, el gradiente en casi toda la región factible es prácticamente nulo, lo que provoca aceleraciones muy pequeñas. Además, al no tener un líder que dirija a los asteroides, la entrada al óptimo es aleatoria. Aquí tenemos su representación gráfica.

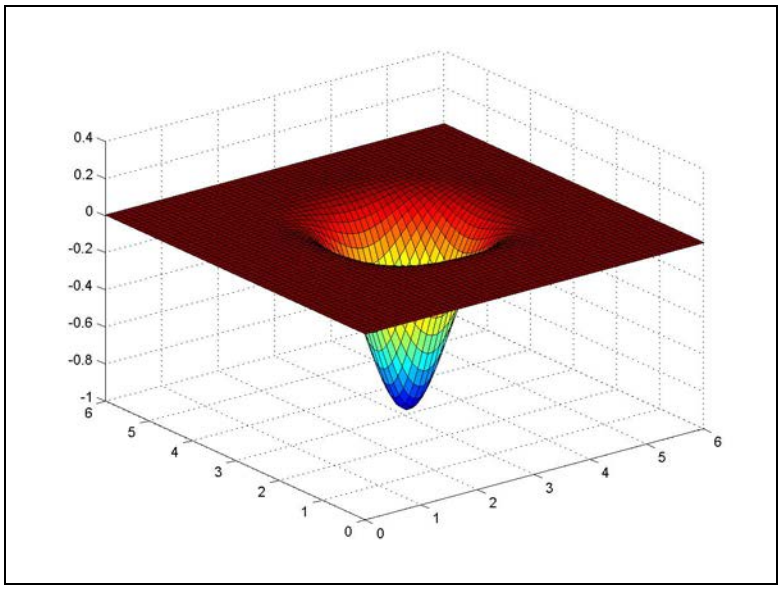


Figura 3.11. Representación con MATLAB de la función EASOM.

Según Yao y otros (1999) el óptimo global es -1 y se alcanza en $(-\pi,\pi)$**.** Nuestro objetivo en este caso es comprobar si, a pesar de la poca velocidad que se imprime a los asteroides, alguno es capaz de explorar la región del óptimo con error menor que 10^{-6}. **Los resultados han sido positivos, aunque sólo para *G* = *100000***, no obstante los resultados de *G* = *10000* también son muy buenos.

Observemos que en este caso el porcentaje de infactibilidades permanece bastante estable al variar *G*. Los gradientes son prácticamente nulos en casi todo punto, por eso sólo valores de *G* muy grandes conseguirían aumentar significativamente las salidas de asteroides de la zona factible.

Tabla 3.12[10]

$\alpha = 0.005$, *β = 0.99 y G = 10 , 100 , 1.000 , 10.000, 100.000 y* 10^6 *, n=1000 , n_iter=1000*

Nº ejec.	*G = 10*	*G = 100*	*G = 1.000*	*G = 10.000*	*G = 100.000*	*G =* 10^6
1	-0.992245 X= 3.199543 Y=3.098799 1714 (0.2%)	-0.999192 X= 3.161124 Y=3.129049 1776 (0.2%)	-0.999945 X= 3.137963 Y=3.136734 1841(0.2%)	-0.999998 X= 3.140359 Y=3.141628 1770 (0.2%)	**-1.000000** **X= 3.141613** **Y=3.141684** **1831 (0.2%)**	-0.999100 X= 3.119368 Y=3.131292 1782 (0.2%)
2	-0.998529 X=3.124156 Y=3.167621 1805	-0.995335 X=3.106232 Y=3.098387 1813	-0.999963 X=3.139999 Y=3.136929 1823	-0.999998 X=3.140871 Y=3.142458 1772	**-1.000000** **X=3.141912** **Y=3.141984** **1721**	-0.970109 X=3.031831 Y=3.231991 1779
3	-0.993712 X=3.197439 Y=3.174541 1796	-0.997549 X=3.181989 Y=3.139602 1849	-0.999125 X=3.159801 Y=3.125719 1822	-0.999997 X=3.141866 Y=3.140330 1733	**-1.000000** **X=3.141637** **Y=3.141829** **1789**	-0.996386 X=3.174803 Y=3.105386 1846
4	-0.992604 X=3.129258 Y=3.210840 1761	-0.997403 X=3.101047 Y=3.151045 1732	-0.999912 X=3.134140 Y=3.139732 1774	-0.999999 X=3.142029 Y=3.141067 1789	**-1.000000** **X=3.141521** **Y=3.142046** **1792**	-0.979289 X=3.255071 Y=3.174238 1748
5	-0.996935 X=3.110317 Y=3.174273 1654	-0.993369 X=3.139110 Y=3.075048 1728	-0.999988 X=3.142193 Y=3.138804 1775	-0.999997 X=3.142818 Y=3.140793 1796	-0.999998 X=3.142750 Y=3.141659 1751	-0.989288 X=3.060960 Y=3.167593 1726
6	-0.993745 X=3.162339 Y=3.202848 1686	-0.999350 X=3.161036 Y=3.149056 1772	-0.999788 X=3.141127 Y=3.153468 1741	-0.999997 X=3.141336 Y=3.140193 1780	-0.999998 X=3.142593 Y=3.141337 1754	-0.994397 X=3.158679 Y=3.200355 1822
7	-0.999796 X=3.131046 Y=3.136587 1839	-0.997200 X=3.172950 Y=3.171358 1758	-0.998881 X=3.163891 Y=3.125797 1793	-0.999998 X=3.141505 Y=3.140486 1871	-0.999997 X=3.142139 Y=3.142805 1801	-0.994803 X=3.156233 Y=3.084504 1807
8	-0.993077 X=3.101203 Y=3.196357 1689	-0.982655 X=3.235356 Y=3.195149 1769	-0.998321 X=3.121933 Y=3.114502 1736	-0.999989 X=3.144146 Y=3.140819 1696	-0.999996 X=3.140695 Y=3.143025 1751	-0.985732 X=3.134531 Y=3.239191 1779
9	-0.987991 X=3.052487 Y=3.152142 1912	-0.977691 X=3.032093 Y=3.196749 1768	-0.990496 X=3.121380 Y=3.218767 1788	-0.999972 X=3.145725 Y=3.140335 1750	-0.999994 X=3.140703 Y=3.139878 1819	-0.991623 X=3.068785 Y=3.124108 1794
10	-0.989026 X=3.136996 Y=3.227220 1770	-0.977781 X=3.019494 Y=3.133897 1767	-0.993288 X=3.155427 Y=3.076038 1827	-0.999988 X=3.144331 Y=3.142464 1816	-0.999998 X=3.142044 Y=3.140565 1728	-0.988596 X=3.226061 Y=3.119045 1757

Figura 3.12. Tabla de resultados EASOM 2.

Con el valor *G=100000* hemos obtenido los mejores resultados, aunque los valores registrados para *G=10000* también han sido muy aceptables.

Los tiempos de ejecución para esta función oscilan alrededor de 45 segundos.

[10] Hemos resaltado en negrita los resultados con error menor que 10^{-6}.

2.5. LA FUNCIÓN DE SHUBERT EN DIMENSIÓN 2

$$f(x_1, x_2) = \left(\sum_{i=1}^{5} i \cos\left((i+1)x_1 + i\right) \right) \left(\sum_{i=1}^{5} i \cos\left((i+1)x_2 + i\right) \right)$$

Dominio de optimización: $[-10,10] \times [-10,10]$.

Esta función también pertenece a las llamadas multimodales, Observemos que los papeles de x e y son intercambiables, i.e. es simétrica respecto al plano $x = y$, por tanto si en $\left(x^*, y^*\right)$ hay localizado un óptimo, también lo habrá en $\left(y^*, x^*\right)$. Su representación es

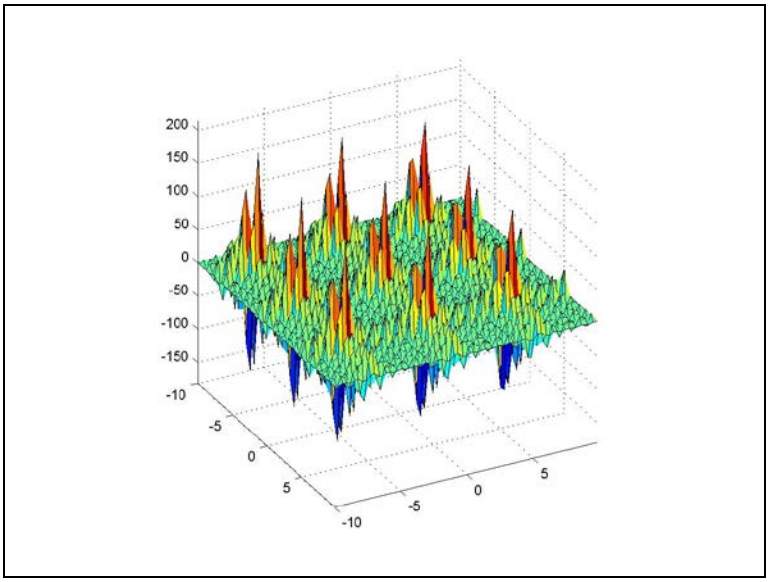


Figura 3.13. Representación con MATLAB de la función SHUBERT.

Según Yao y otros (1999) el óptimo vale -186.7309. **Partimos solo de 4 dígitos significativos.**

Nuestro objetivo será localizar el óptimo en más de un punto. Notemos que el algoritmo sí ha sido capaz de detectar diferentes puntos donde se alcanza el óptimo.

En este caso la topología de la curva ha motivado que los resultados mejores se hayan obtenido para valores de *G* muy inferiores al resto de los ejemplos y que valores altos de *G* hayan originado infactibilidades mayores que en casos anteriores.

Observemos que **si redondeáramos a 4 decimales el número de datos resaltados en negrita, con errores menores de 10^{-6} sería mucho mayor, lo que nos hace pensar que nuestro algoritmo ha encontrado el óptimo con mayor exactitud.**

Los asteroides no han sido capturados en ningún óptimo local.

Tabla 3.14[11]

$\alpha = 0.005$, *β = 0.99 y G = 1, 10, 100 , 1.000 , 10.000 y 100.000 , n=1000 , n_iter=1000*

Nº ejec.	*G = 1*	*G = 10*	*G = 100*	*G = 1000*	*G = 10000*	*G = 100000*
1	-186.721942 X= 5.482570 Y=-7.710255 67328 (6.7%)	-186.730886 X= -7.083442 Y=4.858131 66062 (6.6%)	-186.730895 X= -0.800383 Y=4.858102 48470 (4.8%)	-186.720190 X= -7.081821 Y=4.859437 44067 (4.4%)	-186.730787 X= -7.708226 Y=-0.800537 265469 (26.5%)	-186.726308 X= -1.423849 Y=-0.799725 782260 (78.2%)
2	-186.726682 X=-7.082692 Y=4.859145 67879	-186.730892 X=5.482930 Y=-1.425182 66132	**-186.730903** **X=5.482887** **Y=-1.425084** **43843**	-186.725871 X=-7.084693 Y=4.858964 44685	-186.730699 X=-1.425361 Y=-0.800514 265975	-186.728728 X=4.859006 Y=5.482675 781586
3	-186.718398 X=-1.422813 Y=-7.083468 67723	-186.730893 X=5.482860 Y=-7.708394 65761	-186.730871 X=-7.083441 Y=-1.425018 44434	-186.717791 X=-7.085518 Y=-1.423791 43267	-186.724236 X=-0.800856 Y=4.859666 265729	-186.726836 X=-0.800150 Y=-1.426440 782265
4	-186.718667 X=-7.710194 Y=-0.798974 68226	-186.730873 X=5.482940 Y=-1.425226 65110	**-186.730908** **X=4.858039** **Y=-0.800322** **46757**	-186.724315 X=5.482360 Y=4.856447 44502	-186.719834 X=4.857543 Y=-0.802496 265097	-186.719287 X=-1.427246 Y=5.482135 781770
5	-186.725441 X=-1.423859 Y=-7.082626 68265	-186.730887 X=-7.083437 Y=4.857988 66306	-186.730868 X=-0.800372 Y=4.857935 46570	-186.719476 X=5.484848 Y=-1.424045 45201	-186.714859 X=4.856277 Y=-0.802302 266103	-186.727963 X=5.483644 Y=4.857228 782118
6	-186.722788 X=-7.709743 Y=-7.082273 66893	-186.730722 X=5.482812 Y=4.858335 65901	**-186.730904** **X=4.858093** **Y=-7.083485** **46014**	-186.727645 X=5.483308 Y=4.856955 44464	-186.724065 X=-0.801969 Y=-7.707714 265305	-186.729403 X=4.858076 Y=5.482039 782216
7	-186.730271 X=4.857552 Y=5.483004 67516	**-186.730904** **X=4.858098** **Y=-0.800305** **65520**	-186.730893 X=-7.708319 Y=-0.800403 44849	-186.709401 X=-0.797275 Y=4.858716 43194	-186.723560 X=-0.798502 Y=-7.708451 266089	-186.726200 X=5.484323 Y=-1.425119 782370
8	-186.718644 X=-7.081662 Y=4.859484 68070	-186.730889 X=-0.800232 Y=-7.708343 65576	**-186.730904** **X=4.858016** **Y=5.482870** **45011**	-186.719062 X=-0.800668 Y=-7.706085 46652	-186.729753 X=-1.425539 Y=-0.800908 266248	-186.729901 X=-0.801180 Y=-1.425307 782004
9	-186.721805 X=5.484891 Y=4.858169 67093	-186.730685 X=-7.083564 Y=-1.424824 66588	-186.730869 X=-1.425181 Y=-7.083384 47843	-186.713892 X=-7.710364 Y=-7.081698 44355	-186.719935 X=-7.083340 Y=4.855892 265641	-186.711192 X=-1.423651 Y=5.480291 782059
10	-186.728384 X=4.859087 Y=-7.083659 67240	-186.730817 X=4.857863 Y=-7.083461 66692	-186.730851 X=-7.708465 Y=-7.083468 46185	-186.719552 X=-1.424948 Y=-0.798061 43457	-186.715826 X=-7.084860 Y=-7.706139 266265	-186.723243 X=-1.425846 Y=-0.802031 781668

Figura 3.14. Tabla de resultados SHUBERT 2.

Los tiempos de ejecución para esta instancia han rondado los 80 segundos. Se explica el aumento por el mayor número de operaciones realizado en esta ocasión.

[11] Hemos resaltado en negrita los resultados que mejoran el de partida.

2.6. LA FUNCIÓN DE ROSENBROCK EN DIMENSIÓN 2

$$f(\overline{x}) = \sum_{i=1}^{n-1} \left(100(x_{i+1} - x_i^{\,2})^2 + (x_i - 1)^2 \right)$$

Dominio de optimización: $[-10,10] \times [-10,10]$.

Al igual que le sucede a la función de BRANIN, esta función tiene gradientes muy elevados fuera de la zona del óptimo pero muy bajos cerca del óptimo. He aquí su representación.

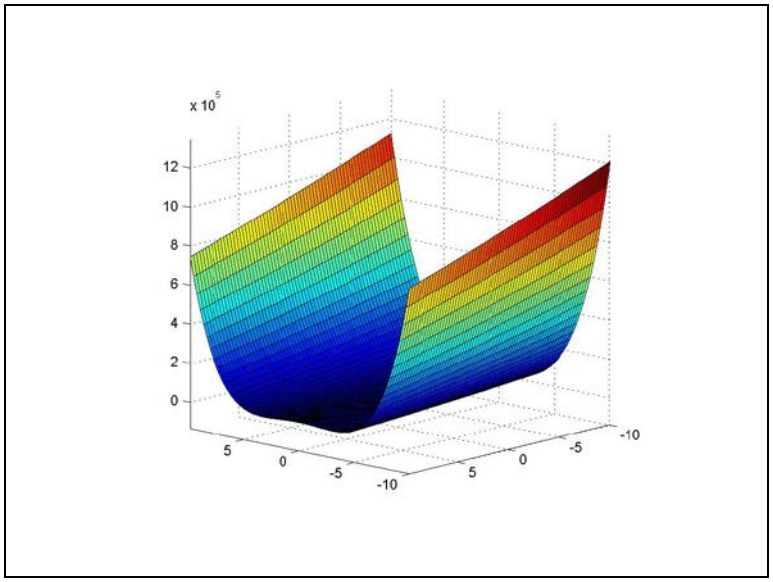


Figura 3.15. Representación con MATLAB de la función ROSENBROCK.

Según Yao y otros (1999) el óptimo vale **0** y se alcanza en $(1,1)$**.**

Nuestro objetivo será comprobar si se alcanzan óptimos menores que una millonésima sin reducir la región factible. **El resultado ha sido afirmativo** aunque parece que para *G = 1* ha sido por "azar favorable".

Aunque los mejores resultados se han obtenido para *G = 1000* observemos que el valor de G = 100 ha dado todos los resultados inferiores a una cienmilésima.

La explicación para el gran porcentaje de infactibilidades en *G = 1000 y G = 10000* se encuentra, como en casos anteriores, en la topología de la curva.

Tabla 3.16[12]

$\alpha = 0.005$, *β = 0.99 y G = 0.1, 1, 10, 100 , 1.000 y 10.000 , n=1000 , n_iter=1000*

Nº ejec.	*G = 0.1*	*G = 1*	*G = 10*	*G = 100*	*G = 1.000*	*G = 10.000*
1	1.7904 E-6 X= 1.001249 Y=1.002547 41480 (4.1%)	6.1698 E-5 X= 0.997428 Y=0.994121 22760 (2.3%)	7.3706 E-6 X= 1.001473 Y=1.003177 28886 (2.9%)	6.0742 E-6 X= 0.997555 Y=0.995149 39118 (4.0%)	1.6374 E-6 X= 0.999374 Y=0.998638 558841 (56.0%)	1.0388 E-5 X= 1.002811 Y=1.005788 956125(95.6%)
2	5.2232 E-5 X=0.994173 Y=0.987953 41269	8.6813 E-5 X=0.993862 Y=0.988464 22474	3.6387 E-5 X=0.998554 Y=0.996525 28553	1.0597 E-6 X=1.000182 Y=1.000466 39316	7.4218 E-6 X=0.997309 Y=0.994669 564575	7.8150 E-6 X=0.997397 Y=0.994904 955931
3	3.6939 E-5 X=1.003880 Y=1.007309 4633	8.3086 E-4 X=1.027023 Y=1.053775 22759	1.5758 E-5 X=0.998004 Y=0.995670 28708	2.9633 E-6 X=1.001713 Y=1.003412 39252	5.0977 E-6 X=0.998278 Y=0.996705 561162	2.1895 E-4 X=0.985651 Y=0.971147 956259
4	7.6852 E-5 X=0.992239 Y=0.984947 40399	7.8498 E-5 X=0.991488 Y=0.983295 22381	2.0097 E-5 X=1.000688 Y=1.001820 28665	8.4644 E-6 X=0.997528 Y=0.994909 39552	3.0835 E-6 X=0.998434 Y=0.996950 557490	3.0754 E-4 X=0.986458 Y=0.971986 956392
5	7.9479 E-5 X=0.991924 Y=0.984291 41365	4.5750 E-6 X=1.000697 Y=1.001193 22417	2.6041 E-5 X=1.005099 Y=1.010244 28307	5.3651 E-6 X=0.999165 Y=0.998116 39053	**2.1602 E-7** **X=1.000052** **Y=1.000059** **560039**	2.0530 E-4 X=0.988097 Y=0.975539 955847
6	1.4969 E-4 X=0.999475 Y=1.000172 4654	1.2581 E-5 X=0.999795 Y=0.999945 22468	3.5622 E-5 X=0.994473 Y=0.988752 28282	1.2384 E-6 X=0.999011 Y=0.997972 39103	3.0835 E-6 X=0.998434 Y=0.996950 557490	1.6667 E-4 X=0.999373 Y=0.997457 956422
7	1.6763 E-4 X=1.008388 Y=1.015861 41161	**9.7077 E-7** **X=1.000644** **Y=1.001213** **22557**	2.8761 E-5 X=1.004976 Y=1.010178 28457	4.3807 E-6 X=0.998721 Y=0.997609 38877	5.0977 E-6 X=0.998278 Y=0.996705 561162	1.8851 E-4 X=0.995829 Y=0.992985 956579
8	7.4281 E-5 X=1.008618 Y=1.017304 40600	1.3128 E-5 X=1.003620 Y=1.007239 22820	2.3394 E-5 X=0.998715 Y=0.996966 28492	5.7279 E-6 X=1.001507 Y=1.002832 38694	**9.1946 E-7** **X=0.999090** **Y=0.998151** **560565**	2.7267 E-4 X=1.010035 Y=1.021482 956105
9	8.3841 E-5 X=0.991013 Y=0.981932 41891	2.4434 E-4 X=0.984411 Y=0.969180 22416	1.1739 E-5 X=0.998933 Y=0.997543 28228	8.5358 E-6 X=1.000194 Y=1.000680 38298	3.9788 E-6 X=0.999332 Y=0.998477 562745	1.5179 E-4 X=0.994313 Y=0.989752 956475
10	9.7634 E-5 X=1.000421 Y=0.999855 41647	9.2585 E-6 X=1.000112 Y=0.999921 22729	1.3718 E-5 X=1.000573 Y=1.001513 28474	3.4836 E-6 X=0.998655 Y=0.997441 38474	1.4750E-6 X=0.998790 Y=0.997593 559540	4.5444 E-4 X=0.986396 Y=0.974619 955999

Figura 3.16. Tabla de resultados ROSENBROCK 2.

A la vista de la tabla 3.16 podemos comprobar los buenos resultados de S.G.O. con esta instancia. Para *G=100* Y *G=1000* todos los resultados son muy aceptables.

Los tiempos de ejecución para esta instancia han rondado los 45 segundos. Permanecen similares a los obtenidos para otras instancias en dimensión 2.

[12] Hemos resaltado en negrita los resultados con error menor que 10^{-6}.

2.7. LA FUNCIÓN DE GRIEWANK EN DIMENSIÓN 2

$$f(x_1, x_2) = \frac{x_1^2 + x_2^2}{4000} - \cos(x_1)\cos\left(\frac{x_2}{\sqrt{2}}\right) + 1$$

Dominio de optimización: $[-600, 600] \times [-600, 600]$.

Estamos ante otra función multimodal con gradiente no muy grande. Veamos su representación

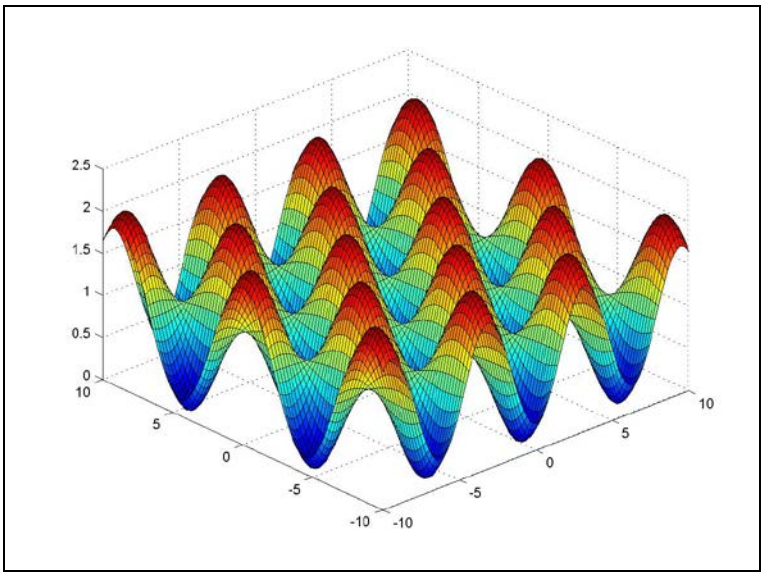


Figura 3.17. Representación con MATLAB de la función GRIEWANK.

Según Yao y otros (1999) el óptimo vale 0.

Nuestro objetivo será obtener óptimos menores que 10^{-6} **lo que conseguimos para valores de *G* más altos que en casos anteriores**. Las infactibilidades han permanecido muy controladas, por las aceleraciones y deceleraciones provocadas por la topología de la función objetivo.

Por analogía con la función de Schwefel hemos elegido los mismos valores para *G* no obstante, las infactibilidades en este caso son menores.

Para *G = 100* y *G = 1000* observamos que algún asteroide sí ha sido atrapado en óptimos locales lo que explicamos por la poca aceleración imprimida a cada uno de ellos, cosa que no ha sucedido para ningún valor mayor del parámetro.

Con $G = 10^7$ las infactibilidades aumentan considerablemente, no hemos creído conveniente incluirlos en la tabla.

Tabla 3.18[13]

$\alpha = 0.005$, *β = 0.99 y G = 100 , 1.000 , 10.000 , 100.000, 10^6 y 5*10^6 , n=1000 , n_iter=1000*

Nº ejec.	*G = 100*	*G = 1.000*	*G = 10.000*	*G = 100.000*	*G = 10^6*	*G = 5*10^6*
1	7.2099 E-3 X=-0.029878 Y=0.164527 237 (0.0%)	6.0182 E-3 X=-0.000941 Y=-0.155149 173 (0.0%)	1.1460 E-3 X=-0.022349 Y=0.059853 122 (0.0%)	**5.5468 E-7** **X= 0.001040** **Y=-0.000237** **30 (0.0%)**	2.7366 E-6 X=0.002297 Y=0.000623 444 (0.0%)	6.4738 E-4 X=-0.020303 Y=0.041994 8323 (0.8%)
2	7.7579 E-3 X=0.123479 Y=-0.023757 209	7.0031 E-3 X=0.025605 Y=-0.163439 185	3.0505 E-3 X=-0.051585 Y=-0.082961 113	2.8627 E-6 X=0.002373 Y=-0.000423 27	**1.7875 E-7** **X=0.000403** **Y=-0.000625** **405**	1.8765 E-3 X=0.016937 Y=0.083235 8219
3	4.8302 E-3 X=0.094766 Y=-0.037002 237	6.6202 E-3 X=-0.052880 Y=-0.144614 201	1.6408 E-4 X=0.005723 Y=0.024295 123	6.3174 E-6 X=-0.001894 Y=0.004251 24	1.7233 E-4 X=-0.018453 Y=0.002822 438	2.6198 E-3 X=0.057744 Y=-0.061742 8353
4	7.8477 E-3 X=3.114471 Y=4.416084 213	7.5511 E-3 X=3.125339 Y=-4.424707 191	2.3988 E-3 X=0.068538 Y=-0.014127 117	1.5906 E-6 X=-0.001401 Y=0.001559 23	**7.5441 E-7** **X=0.000896** **Y=0.001187** **462**	4.1989 E-4 X=-0.027554 Y=-0.012662 8162
5	1.6013 E-3 X=-0.048305 Y=-0.041682 218	9.6435 E-3 X=3.111759 Y=4.352476 196	3.5164 E-5 X=-0.007927 Y=-0.003862 125	**6.7249 E-7** **X=0.000059** **Y=0.001637** **31**	2.1371 E-5 X=0.006535 Y=0.000153 475	5.5585 E-4 X=0.010798 Y=0.044591 8402
6	5.1526 E-3 X=0.013131 Y=-0.142351 248	8.5765 E-3 X=0.023924 Y=-0.182162 202	1.8153 E-3 X=0.027385 Y=-0.075885 120	**8.8901 E-7** **X=0.000721** **Y=0.001585** **26**	1.9550 E-6 X=-0.001389 Y=-0.001989 430	9.6314 E-4 X=-0.006129 Y=0.061435 8340
7	5.3571 E-3 X=0.067395 Y=-0.111200 248	8.2907 E-3 X=3.109913 Y=-4.396441 177	1.8157 E-3 X=0.037168 Y=-0.067069 109	**8.1530 E-7** **X=0.001064** **Y=0.000998** **22**	8.0603 E-6 X=0.002219 Y=-0.004729 431	1.2137 E-3 X=0.044935 Y=-0.028550 8454
8	1.9475 E-3 X=0.020337 Y=0.083420 236	3.5505 E-3 X=-0.076877 Y=-0.048854 191	3.4219 E-4 X=-0.017464 Y=0.027530 110	2.7309 E-6 X=0.000348 Y=-0.003267 22	1.8596 E-5 X=0.005408 Y=-0.003980 437	2.3002 E-3 X=-0.065297 Y=0.025945 8324
9	8.5052 E-3 X=-3.150721 Y=-4.373602 220	5.8526 E-3 X=-0.097426 Y=0.066707 204	1.3171 E-3 X=0.029293 Y=0.059582 120	**1.4865 E-7** **X=-0.000208** **Y=-0.000713** **29**	2.3573 E-5 X=0.002391 Y=0.009098 413	4.0478 E-3 X=-0.061197 Y=0.093328 8374
10	5.9474 E-3 X=-0.108386 Y=0.017531 235	2.3390 E-3 X=0.014868 Y=0.094388 196	7.2840 E-4 X=0.037970 Y=-0.005397 97	1.5305 E-6 X=-0.001271 Y=-0.001700 26	2.6385 E-5 X=0.006149 Y=0.005464 427	5.2904 E-3 X=-0.100203 Y=-0.033033 8247

Figura 3.18. Tabla de resultados GRIEWANK 2.

A pesar de la difícil topología de esta función, los resultados han sido excelentes.

Los tiempos de las ejecuciones han oscilado entre 45 y 47 segundos.

[13] Hemos resaltado en negrita los resultados con error menor que 10^{-6}.

2.8. FUNCIONES DE DIMENSIÓN 4

A continuación expondremos algunos resultados computacionales obtenidos para dos *funciones test* definidas en dimensión 4. Nos ha parecido apropiado establecer un estudio paralelo entre dimensiones 2 y 4, por ello la primera de ellas, la función de ROSENBROCK, también aparece en la sección anterior; sin embargo también hemos querido dotar de entidad independiente el caso tetradimensional, por ello la segunda, la función de RASTRIGIN, sólo aparece en esta sección.

Como veremos, en todos los casos se ha llegado al óptimo previsto, aunque con un error sensiblemente superior al de los casos anteriores.

La inicialización aleatoria de las posiciones y velocidades se ha llevado a cabo como en el caso anterior.

En esta ocasión hemos realizado 30 ejecuciones con cada elección de parámetros porque el tiempo de cada experimentación ha aumentado respecto a los empleados en dimensión 2. Todas están en el CD adjunto. En cada caso hemos expuesto solo 5 resultados.

A la vista de que $n = 1000$ asteroides con $n\_iter = 1000$ no nos aportaba resultados tan satisfactorios como en el caso bidimensional, hemos repetido la ejecución aumentando el número de asteroides a $n = 4000$, sin embargo los resultados no han mejorado tanto como esperábamos.

En todas las experiencias hemos elegido $\alpha = 0.005$, ya que los resultados con otros valores no han sido mejores.

Como rango de detección, r_d, hemos usado en todos los casos el valor 10^{-6} por idéntico argumento al de dimensión 2.

El parámetro β ha variado según el problema. En cada caso expondremos su valor.

Mantenemos la misma notación que en las secciones anteriores respecto a las tablas.

2.9. LA FUNCIÓN DE ROSENBROCK EN DIMENSIÓN 4

$$f(\overline{x}) = \sum_{i=1}^{n-1} [100\left(x_{i+1} - x_i^{\,2}\right)^2 + (x_i - 1)^2]$$

Dominio de optimización: $x_i \in [-10,10] \;\; i = 1,...,4$

Análogamente a lo que sucede en dimensión 2, el óptimo vale 0 y se alcanza en $(1,1,1,1)$ (Yao y otros 1999)**.** Nuestro objetivo será obtener el valor con la mayor precisión posible.

Dado que en dimensión 2 obtuvimos los mejores valores para *G = 100* , *G = 1000* y *G = 10000* esperábamos que se repitiera en dimensión 4, sin embargo, los resultados obtenidos para el valor 10 han sido mucho mejores que para el valor 10000.

Manteniendo los parámetros como en la sección 2.6 los errores no eran inferiores a 10^{-2} y las infactibilidades permanecían más altas, por ello hemos disminuido el valor de β.

Al dar a *β* el valor[14] 0.5 hemos obtenido **precisión hasta 10^{-3}.**

Observemos que los asteroides no han quedado atrapados en óptimos locales.

En la figura 3.19 presentamos una tabla que recoge los resultados obtenidos.

[14] Universo en expansión.

Tabla 3.19[15]

$\alpha = 0.005$, $\beta = 0.5$, $G = 10, 100$ y 1.000, $n = 1000$, $n\_iter = 1000$

Nº ejec.	*G = 10*	*G = 100*	*G = 1000*
1	1.493020 E-2 X = 1.012839 ; Y = 1.030265 Z = 1.053597 ; T = 1.115416 5624 (0.5%)	**3.724195 E-4** **X = 0.999726 ; Y = 1.000062** **Z = 1.000251 ; T = 1.002328** **261966 (26.2%)**	**9.292293 E-4** **X = 0.993841 ; Y = 0.987168** **Z = 0.975022 ; T = 0.949993** **992328 (99.2%)**
2	5.690113 E-2 X = 0.996027 ; Y = 0.982374 Z = 0.956051 ; T = 0.933301 5473	**3.047605 E-4** **X = 1.000506 ; Y = 1.001063** **Z = 1.002411 ; T = 1.006528** **264871**	1.920882 E-3 X = 0.991121 ; Y = 0.981268 Z = 0.962855 ; T = 0.927021 991360
3	4.306695 E-2 X = 1.016830 ; Y = 1.048674 Z = 1.099316 ; T = 1.217896 5416	**4.524562 E-4** **X = 1.000296 ; Y = 0.999967** **Z = 1.000010 ; T = 1.002052** **264327**	2.070745 E-3 X = 0.990606 ; Y = 0.980453 Z = 0.961388 ; T = 0.923660 992668
4	4.713738 E-2 X = 1.011778 ; Y = 1.015148 Z = 1.022881 ; T = 1.064480 5555	**4.365824 E-4** **X = 1.000520 ; Y = 1.001580** **Z = 1.003870 ; T = 1.009598** **271650**	2.779150 E-3 X = 0.989260 ; Y = 0.978111 Z = 0.958091 ; T = 0.916497 992171
5	4.587724 E-2 X = 1.015011 ; Y = 1.021363 Z = 1.036048 ; T = 1.090977 5342	**3.973426 E-4** **X = 1.000094 ; Y = 1.000718** **Z = 1.001119 ; T = 1.004130** **263595**	2.459886 E-3 X = 0.990093 ; Y = 0.980452 Z = 0.963327 ; T = 0.926533 991268

Figura 3.19. Tabla de resultados ROSENBROCK 4 (I).

En el caso tetradimensional, manteniendo fijos los valores de los parámetros, los tiempos de ejecución han ascendido a un valor medio de 84 segundos. Como ya hemos comentado en varias ocasiones, haremos un estudio detallado de tiempos y números de llamadas a la función objetivo posteriormente.

Buscando cotas de error parecidas al caso bidimensional hemos aumentado el valor del número de asteroides hasta $n = 4000$, pero, como veremos en la siguiente tabla, no hemos conseguido disminuir el orden de los errores, aunque sí hay mayor número de "datos buenos" en el total de los experimentos realizados. En ambos casos las infactibilidades han permanecido similares.

[15] Hemos resaltado en negrita los resultados con error menor que 10^{-3}.

Tabla 3.20[16]

$\alpha = 0.005$, $\beta = 0.5$, $G = 10, 100$ y 1.000, $n = 4.000$, $n\_iter = 1000$

Nº ejec.	**G = 10**	**G = 100**	**G = 1.000**
1	1.774646 E-2 X = 0.986807 ; Y = 0.976701 Z = 0.943976 ; T = 0.896661 21458 (0.5%)	**1.959001 E-4** **X = 0.999660 ; Y = 0.999605** **Z = 0.999808 ; T = 1.000848** **1095541 (27.4%)**	1.938552 E-3 X = 0.990550 ; Y = 0.981247 Z = 0.963127 ; T = 0.926476 3972322 (99.3%)
2	2.150253 E-2 X = 1.014752 ; Y = 1.028529 Z = 1.046622 ; T = 1.102832 21389	**1.950390 E-4** **X = 0.999622 ; Y = 0.999527** **Z = 0.999649; T = 1.000528** **1137672**	2.050906 E-3 X = 0.990323 ; Y = 0.980652 Z = 0.960933 ; T = 0.923444 3972772
3	2.727067 E-2 X = 0.991487 ; Y = 0.979146 Z = 0.946382 ; T = 0.904083 21076	**1.953830 E-4** **X = 0.999656 ; Y = 0.999601** **Z = 0.999803 ; T = 1.000833** **1094544**	2.084063 E-3 X = 0.990512 ; Y = 0.980562 Z = 0.960467 ; T = 0.922816 3962734
4	2.834308 E-2 X = 0.989622 ; Y = 0.993775 Z = 0.987353 ; T = 0.983679 20933	**1.964666 E-4** **X = 0.999615 ; Y = 0.999519** **Z = 0.999639 ; T = 1.000510** **1115338**	2.154045 E-3 X = 0.989628 ; Y = 0.979648 Z = 0.960467 ; T = 0.922816 3968247
5	2.847728 E-2 X = 1.020436 ; Y = 1.032546 Z = 1.076512 ; T = 1.164138 22137	**1.967790 E-4** **X = 0.999618 ; Y = 0.999523** **Z = 0.999647 ; T = 1.000527** **1102538**	2.175518 E-3 X = 0.991110 ; Y = 0.980773 Z = 0.963039 ; T = 0.927589 3962734

Figura 3.20. Tabla de resultados ROSENBROCK 4 (II).

Los resultados no son tan buenos como los obtenidos en dimensión 2 pero siguen siendo muy aceptables, tanto para *G=100* como para *G=1000*.

El aumento de *n* al valor 4000 ha sido el responsable del correspondiente incremento de los tiempos de ejecución hasta los 334 segundos por término medio.

En general, pasar de dimensión 2 a dimensión 4, manteniendo fijos los valores de los parámetros, ha originado una pérdida de precisión en los resultados obtenidos desde errores menores que 10^{-6} a 10^{-3} y el aumento, aproximadamente al doble, de los tiempos de ejecución.

[16] Hemos resaltado en negrita los resultados con error menor que 10^{-3}.

2.10. LA FUNCIÓN DE RASTRIGIN EN DIMENSIÓN 4

$$f(\overline{x}) = 10n + \sum_{i=1}^{n} \left(x_i^{\,2} - 10\cos(2\pi x_i) \right)$$

Dominio de optimización: $x_i \in [-5.12, 5.12] \;\; i = 1, ..., n$

Se trata de una función multimodal en la que los gradientes van aumentando a medida que nos acercamos a la frontera. Veamos su representación en dos dimensiones

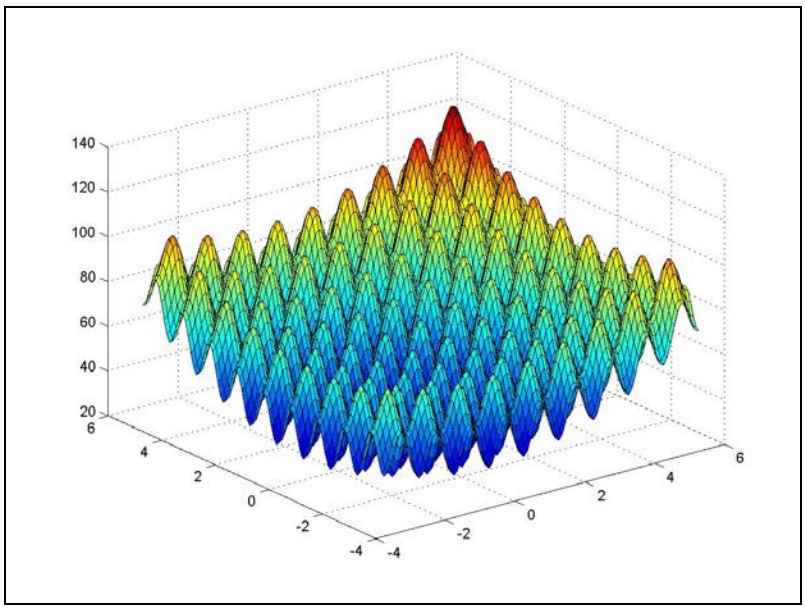


Figura 3.21. Representación con MATLAB de la función RASTRIGIN en dimensión 2.

Según Yao y otros (1999) el óptimo vale **0** y se alcanza en $(0,0,0,0)$**.** Nuestro objetivo será alcanzar este valor con el menor error posible, en el mejor caso ha sido 10^{-3}.

Hemos mantenido β en el valor 0.99. Los valores *G = 100, G = 1000 y G = 10000* han dado los mejores resultados.

En ningún caso los asteroides han sido atrapados en óptimos locales.

A continuación exponemos algunos resultados obtenidos para $n = 1000$ y $n = 4000$ asteroides. Como se observa en las tablas, para $n = 4000$ hemos obtenido el óptimo en más ocasiones pero no con menor error significativo. En ambos casos el porcentaje de infactibilidades ha permanecido constante.

Los tiempos de ejecución son similares a los obtenidos en el apartado anterior.

Tabla 3.22[17]

$\alpha = 0.005$, β = 0.99 , *G = 100 , 1000 y 10000 , n = 1000* , $n\_iter = 1000$

G = 100	G = 1000	G = 10000
3.149083 E-1 X = 0.007906 ; Y = 0.006387 Z = 0.002505 ; T = -0.038535 278846 (27.9%)	**5.134259 E-4** **X = 0.000931 ; Y = 0.000564** **Z = -0.001157 ; T = 0.000254** **228397 (22.8%)**	4.380862 E-1 X = 0.016127 ; Y = -0.015629 Z = 0.011730 ; T = -0.039684 445602 (44.6%)
9.801462 E-1 X =-0.002156 ; Y = -0.061555 Z = -0.034535 ; T = 0.002364 277533	**9.768845 E-4** **X = 0.000435 ; Y = 0.002169** **Z = 0.000114 ; T = -0.000135** **234095**	1.078852 X = -0.012217 ; Y = -0.014802 Z = 0.003598 ; T = -1.001424 446202
6.683291 E-1 X = 0.002050 ; Y = 0.054280 Z = 0.020144 ; T = -0.006428 276052	3.593326 E-3 X = 0.000675 ; Y = 0.003878 Z = 0.001502 ; T = -0.000599 230196	1.217385 X = 0.017933 ; Y = -0.054234 Z = 0.014713 ; T = -0.052053 445931
7.979612 E-1 X = -0.006389 ; Y = -0.018174 Z = -0.023098 ; T = -0.056135 278116	5.211864 E-3 X = -0.001651 ; Y = 0.004023 Z = 0.002685 ; T = -0.000386 228854	1.142693 X = 0.980949 ; Y = -0.004985 Z = -0.022242 ; T = -0.005483 446785
8.271900 E-1 X = 0.016524 ; Y = 0.054715 Z = 0.001899 ; T = 0.030519 275831	2.299216 E-3 X = -0.000794 ; Y = -0.000739 Z = 0.003003 ; T = 0.001180 231761	1.229697 X = -0.020148 ; Y = 0.000575 Z = -0.980800; T = -0.024053 446055

Figura 3.22. Tabla de resultados RASTRIGIN 4 (I).

Tabla 3.23[18]

$\alpha = 0.005$, β = 0.99 *y G = 100 , 1000 y 10000 , n = 4000* , $n\_iter = 1000$

G = 100	G = 1000	G = 10000
4.159741 E-2 X= 0.009951 ; Y = -0.002597 Z = 0.000549 ; T = 0.010182 1117150 (27.9%)	**3.960606 E-4** **X = -0.001477 ; Y = 0.001803** **Z = 0.002016 ; T = -0.000475** **920086 (23.0%)**	4.209558 E-1 X = -0.013135 ; Y = -0.005087 Z = 0.013792 ; T = -0.041754 1784010 (44.6%)
5.487117 E-1 X = -0.026223 ; Y = -0.007892 Z = -0.030704 ; T = -0.032883 1117587	**9.196756 E-4** **X = -0.000376 ; Y = -0.000681** **Z = -0.002002 ; T = 0.000154** **938090**	3.891758 E-1 X = -0.024119 ; Y = 0.017457 Z = -0.032204 ; T = 0.006559 1784143
4.296691 E-1 X = 0.024313 ; Y = 0.029133 Z = -0.020251 ; T = -0.017894 1118496	**5.824940 E-4** **X = 0.000639 ; Y = 0.000891** **Z = -0.000220 ; T = 0.001298** **929618**	7.325628 E-1 X = 0.021603 ; Y = 0.021062 Z = 0.045684 ; T = 0.026692 1786343
2.274273 E-1 X = -0.027525 ; Y = 0.017338 Z = -0.007349 ; T = -0.006024 1122602	**5.609437 E-4** **X = 0.001165 ; Y = 0.000079** **Z = 0.000102 ; T = -0.001205** **925535**	8.782121 E-1 X = 0.023587 ; Y = 0.061509 Z = -0.006204 ; T = 0.009802 1784278
7.833185 E-1 X = 0.038995 ; Y = 0.015865 Z = -0.046789 ; T = 0.003194 1117961	**7.874577 E-4** **X = -0.000833 ; Y = 0.001717** **Z = -0.000336 ; T = -0.000462** **935549**	4.215182 E-1 X = -0.005932 ; Y = 0.037562 Z = 0.002904 ; T = 0.026032 1786572

Figura 3.23. Tabla de resultados RASTRIGIN 4 (II).

[17] y [18] Hemos resaltado en negrita los resultados con error menor que 10^{-3}.

3. LA ROBUSTEZ DEL ALGORITMO Y LA VARIACIÓN DE G

Tras la primera fase de pruebas contamos con una muestra adecuada para estudiar el comportamiento de S.G.O. en funciones de dimensión 2.

- La eficacia del método ha sido empíricamente demostrada, ya que en todas las ocasiones **hemos obtenido errores menores que** 10^{-6}**.**
- La eficiencia del algoritmo ha sido muy positiva. Los tiempos de ejecución han sido mínimos.
- La robustez del algoritmo ha sido parcialmente comprobada, en tanto en cuanto el porcentaje de infactibilidades se ha mantenido estable en las repeticiones de los experimentos con los valores fijos para los parámetros.

Sin embargo, parece lógico pensar que un porcentaje de infactibilidades similar no implica necesariamente resultados similares. Por ello, dado que tenemos registrados valores para rangos de variación de *G,* nos parece interesante realizar un *estudio de promedios y desviaciones típicas de óptimos alcanzados en función de G* para estudiar de forma más exhaustiva la robustez de S.G.O. al variar dicho parámetro. Tengamos en cuenta, además, que parte de su aleatoriedad aparece cuando un asteroide sale de la región factible, lo que sucede principalmente[19] por exceso de su aceleración que es proporcional a la constante de gravitación del campo,

La otra componente aleatoria de S.G.O. aparece al determinar la primera posición de los asteroides, en la que intervienen directamente los valores de *n y n_iter.* Posponemos el estudio de su influencia en los resultados hasta el apartado siguiente.

Para realizar el análisis de la robustez de S.G.O. en función de *G* hemos considerado las muestras de datos expuestas en la sección anterior con los valores para *n*, *n_iter*, α y β tal y como se han fijado en cada caso. Hemos considerado el mejor óptimo obtenido, el promedio y la desviación típica en cada uno de los 6 valores de *G* propuestos para cada instancia. Si el comportamiento del algoritmo en la fase salida de asteroides no es robusto tendremos que las mayores desviaciones se darán para los mayores valores de *G.*

Hay también otra cuestión que nos preocupa y es *la posibilidad de fijar un valor, o al menos un rango de valores universales para G*, *que aporten eficacia a S.G.O. independientemente de la instancia considerada.* Con ese fin **hemos extraído en cada caso una terna de valores consecutivos concretos de *G* en los que se han obtenido los mejores datos.**

Veamos los resultados obtenidos.

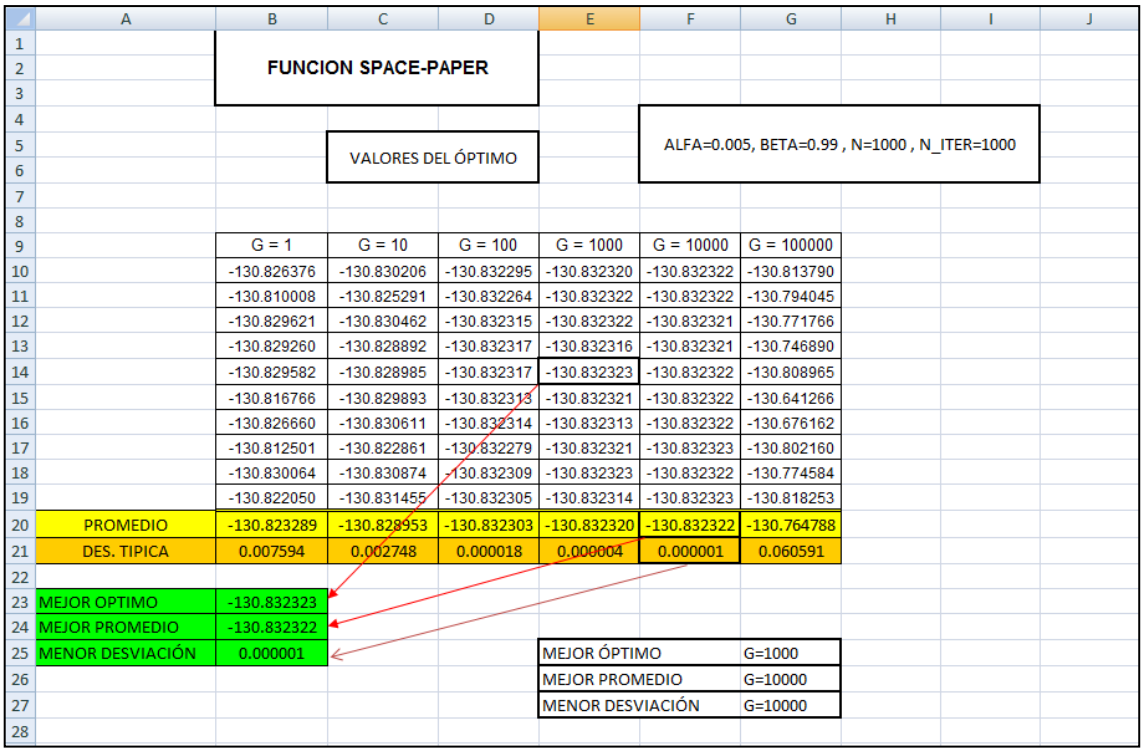
[19] Ya que con los valores $\alpha = 0.005$ y $\beta = 0.99$ varían poco las velocidades y aceleraciones.

3.1. FUNCIÓN SPACE-PAPER

Como se observa en la hoja de Excel 3.24, los menores valores de desviación típica han coincidido con los del mejor promedio y se han dado en *G=10000*, en los que había un 36,9% de infactibilidades. Cuando damos a *G* el valor adecuado para que S.G.O. encuentre resultados buenos, todos los resultados lo son.

El mejor óptimo se ha registrado para *G=1000*. En este caso la desviación ha sido también muy pequeña, 0.000004, no obstante, los resultados han sido muy cercanos a los de *G=10000* .

Como era previsible la mayor desviación se ha registrado para *G=100000,* donde las infactibilidades han llegado al 98%, pero su valor es muy pequeño, 0.060591, lo que nos da clara idea de la robustez del algoritmo para esta instancia.



FUNCION SPACE-PAPER

VALORES DEL ÓPTIMO

ALFA=0.005, BETA=0.99 , N=1000 , N_ITER=1000

	G = 1	G = 10	G = 100	G = 1000	G = 10000	G = 100000
	-130.826376	-130.830206	-130.832295	-130.832320	-130.832322	-130.813790
	-130.810008	-130.825291	-130.832264	-130.832322	-130.832322	-130.794045
	-130.829621	-130.830462	-130.832315	-130.832322	-130.832321	-130.771766
	-130.829260	-130.828892	-130.832317	-130.832316	-130.832321	-130.746890
	-130.829582	-130.828985	-130.832317	-130.832323	-130.832322	-130.808965
	-130.816766	-130.829893	-130.832313	-130.832321	-130.832322	-130.641266
	-130.826660	-130.830611	-130.832314	-130.832313	-130.832322	-130.676162
	-130.812501	-130.822861	-130.832279	-130.832321	-130.832323	-130.802160
	-130.830064	-130.830874	-130.832309	-130.832323	-130.832322	-130.774584
	-130.822050	-130.831455	-130.832305	-130.832314	-130.832323	-130.818253
PROMEDIO	-130.823289	-130.828953	-130.832303	-130.832320	-130.832322	-130.764788
DES. TIPICA	0.007594	0.002748	0.000018	0.000004	0.000001	0.060591

MEJOR OPTIMO	-130.832323
MEJOR PROMEDIO	-130.832322
MENOR DESVIACIÓN	0.000001

MEJOR ÓPTIMO	G=1000
MEJOR PROMEDIO	G=10000
MENOR DESVIACIÓN	G=10000

Figura 3.24. Hoja Excel de Variación de G con SPACE PAPER 2.

Mejores valores de *G* para esta instancia: *G=100 , G=1.000 y G=10.000*

3.2. FUNCIÓN DE BRANIN 2

En esta ocasión, el mejor óptimo y el mejor promedio se han dado para el mismo valor de *G=10000*, en cambio la menor desviación, 0, se ha registrado para *G=100000*. A la vista de la figura 3.25 los resultados son muy similares. En ambos casos, las infactibilidades no han llegado al 1%.

La mayor desviación se han registrado curiosamente para *G=10000.* Su valor, apenas 3 milésimas, indica que *e*n este caso el comportamiento de S.G.O. ha resultado robusto en todos los casos.

FUNCION BRANIN 2

VALORES DEL ÓPTIMO

ALFA=0.005, BETA=0.99 , N=1000 , N_ITER=1000

	G = 10	G = 100	G = 1000	G = 10000	G = 100000	G = 500000
	0.397946	0.397975	0.397889	0.397888	0.397887	0.397902
	0.398025	0.397969	0.397887	0.397887	0.397887	0.397917
	0.398183	0.398108	0.397975	0.397887	0.397887	0.398234
	0.398004	0.398031	0.397894	0.397887	0.397887	0.397896
	0.397899	0.398124	0.397892	0.397887	0.397887	0.39799
	0.397918	0.397953	0.397898	0.397887	0.397887	0.398361
	0.397947	0.397934	0.397893	0.397887	0.397887	0.39811
	0.397993	0.397909	0.397975	0.387887	0.397887	0.398065
	0.397923	0.397899	0.397889	0.397887	0.397887	0.397893
	0.397996	0.39803	0.397903	0.397887	0.397887	0.397938
PROMEDIO	0.397983	0.397993	0.397910	0.396887	0.397887	0.398031
DES. TIPICA	0.000082	0.000078	0.000035	0.003162	5.8514E-17	0.000162

MEJOR OPTIMO	0.387887
MEJOR PROMEDIO	0.396887
MENOR DESVIACIÓN	5.85139E-17

MEJOR ÓPTIMO	G=10.000
MEJOR PROMEDIO	G=10.000
MENOR DESVIACIÓN	G=100.000

Figura 3.25. Hoja de Excel de variación de G con BRANIN 2.

Mejores valores de *G* para esta instancia: *G=1.000* , *G=10.000* y *G=100.000*

3.3. FUNCIÓN DE SCHWEFEL 2

Hemos registrado el mejor óptimo para *G=1000* y el mejor promedio y la menor desviación para *G=1000000* lo que nos da idea de que el mejor valor hallado ha sido casual aunque en todos los casos son muy similares. No obstante, las desviaciones en estos casos han permanecido inferiores s 3 diezmilésimas lo que nos ratifica en demostrar la robustez del algoritmo también para esta instancia. En ambos casos las infactibilidades han supuesto el 0%.

La mayor desviación ha sido poco mayor de 6 centésimas y se ha registrado para *G=100* con infactibilidades del 0%. La robustez de S.G.O. en este caso está probada.

A la hora de elegir nuestro intervalo de mejores *G's* nos encontramos ante la disyuntiva de considerar o no el valor de *G=1000* donde hemos obtenido el mejor óptimo. Hemos optado por considerar los 3 valores en los que se registran las menores desviaciones desechando dicho valor.

FUNCION SCHWEFEL 2

VALORES DEL ÓPTIMO

ALFA=0.005, BETA=0.99 , N=1000 , N_ITER=1000

	G = 100	G = 1000	G = 10000	G = 100000	G = 1000000	G = 5000000
	-837.964583	-837.965775	-837.965763	-837.965768	-837.965771	-837.965774
	-837.892922	-837.965616	-837.965769	-837.965773	-837.965771	-837.965774
	-837.951621	-837.965771	-837.965772	-837.965774	-837.965774	-837.965766
	-837.806866	-837.965756	-837.965769	-837.965765	-837.965774	-837.965774
	-837.939027	-837.965449	-837.965773	-837.965774	-837.965774	-837.965757
	-837.920429	-837.965065	-837.965772	-837.965774	-837.96577	-837.965773
	-837.918017	-837.965562	-837.965769	-837.965767	-837.965773	-837.965774
	-837.95937	-837.965772	-837.965773	-837.965773	-837.965774	-837.965772
	-837.783432	-837.965345	-837.965771	-837.965772	-837.965764	-837.96577
	-837.884303	-837.965654	-837.965773	-837.965774	-837.965771	-837.965773
PROMEDIO	-837.902057	-837.965577	-837.965770	-837.965771	-837.965772	-837.965771
DES. TIPICA	0.062487	0.000233	0.000003	0.000003	0.000003	0.000005

MEJOR OPTIMO	-837.965775
MEJOR PROMEDIO	-837.965772
MENOR DESVIACIÓN	0.000003

MEJOR ÓPTIMO	G=1.000
MEJOR PROMEDIO	G=1.000.000
MENOR DESVIACIÓN	G=1.000.000

Figura 3.26. Hoja de Excel de variación de *G* con SCHWEFEL 2.

Mejores valores de *G* para esta instancia: *G=10.000* , *G=100.000* y *G=1.000.000*

3.4. FUNCIÓN DE EASOM 2

En esta ocasión los 3 valores, mejor óptimo, mejor promedio y menor desviación, se han obtenido para el mismo valor *G*=100000, con 0.2 % de infactibilidades registradas. Ni mayores ni menores valores de *G* han originado más infactibilidades.

Las desviaciones para *G=100 y* G=1*000000* han sido las más altas, poco más de 8 milésimas. El algoritmo se mantiene muy estable, especialmente en los valores de *G* para los que se obtienen los mejores resultados, así su robustez está comprobada también para esta instancia.

Para la elección de nuestra terna de valores para *G* hemos optado por los mejores óptimos registrados.

FUNCION EASOM 2

VALORES DEL ÓPTIMO

ALFA=0.005, BETA=0.99 , N=1000 , N_ITER=1000

	G = 10	G = 100	G = 1000	G = 10000	G = 100000	G = 10^6
	-0.992245	-0.999192	-0.999945	-0.999998	-1	-0.9991
	-0.998529	-0.995335	-0.999963	-0.999998	-1	-0.970109
	-0.993712	-0.997549	-0.999125	-0.999997	-1	-0.996386
	-0.992604	-0.997403	-0.999912	-0.999999	-1	-0.979289
	-0.996935	-0.993369	-0.999988	-0.999997	-0.999998	-0.989288
	-0.993745	-0.99935	-0.999788	-0.999997	-0.999998	-0.994397
	-0.999796	-0.9972	-0.998881	-0.999998	-0.999997	-0.994803
	-0.993077	-0.982655	-0.998321	-0.999989	-0.999996	-0.985732
	-0.987991	-0.977691	-0.990496	-0.999972	-0.999994	-0.991623
	-0.989026	-0.977781	-0.993288	-0.999988	-0.999998	-0.988596
PROMEDIO	-0.993766	-0.991753	-0.997971	-0.999993	-0.999998	-0.988932
DES. TIPICA	0.003787	0.008815	0.003318	0.000008	0.000002	0.008745

MEJOR OPTIMO	-1.000000
MEJOR PROMEDIO	-0.999998
MENOR DESVIACIÓN	0.000002

MEJOR ÓPTIMO	G=100.000
MEJOR PROMEDIO	G=100.000
MENOR DESVIACIÓN	G=100.000

Figura 3.27. Hoja de Excel de variación de *G* con EASOM 2.

Mejores valores de *G* para esta instancia: *G=1.000* , *G=10.000* y *G=100.000*

3.5. FUNCIÓN DE SHUBERT 2

Como en el caso de la función de EASOM, los 3 valores que analizamos se han dado para el mismo valor de *G=100*, con un porcentaje de infactibilidades cercano al 5%.

En este caso las desviaciones más altas se han registrado en los valores mayores de G pero han sido menores de 6 milésimas con lo que una vez más hemos comprobado la robustez de S.G.O. Para *G=100000* las infactibilidades han rondado el 78% pero el algoritmo ha permanecido estable.

Hemos tomado la elección de mejores valores de *G* con los criterios seguidos en el caso de la función de EASOM 2.

FUNCION SHUBERT 2

VALORES DEL ÓPTIMO

ALFA=0.005, BETA=0.99 , N=1000 , N_ITER=1000

	G = 1	G = 10	G = 100	G = 1000	G = 10000	G = 100000
	-186.721942	-186.730886	-186.730895	-186.720190	-186.730787	-186.726308
	-186.726682	-186.730892	-186.730903	-186.725871	-186.730699	-186.728728
	-186.718398	-186.730893	-186.730871	-186.717791	-186.724236	-186.726836
	-186.718667	-186.730873	-186.730908	-186.724315	-186.719834	-186.719287
	-186.725441	-186.730887	-186.730868	-186.719476	-186.714859	-186.727963
	-186.722788	-186.730722	-186.730904	-186.727645	-186.724065	-186.729403
	-186.730271	-186.730904	-186.730893	-186.709401	-186.723560	-186.726200
	-186.718644	-186.730889	-186.730904	-186.719062	-186.729753	-186.729901
	-186.721805	-186.730685	-186.730869	-186.713892	-186.719935	-186.711192
	-186.728384	-186.730817	-186.730851	-186.719552	-186.715826	-186.723243
PROMEDIO	-186.723302	-186.730845	-186.730887	-186.719720	-186.723355	-186.724906
DES. TIPICA	0.004242	0.000079	0.000020	0.005443	0.005816	0.005763

MEJOR OPTIMO	-186.730908
MEJOR PROMEDIO	-186.730887
MENOR DESVIACIÓN	0.000020

MEJOR ÓPTIMO	G=100
MEJOR PROMEDIO	G=100
MENOR DESVIACIÓN	G=100

Figura 3.28. Hoja de Excel de variación de *G* con SHUBERT 2.

Mejores valores de *G* para esta instancia: *G=1* , *G=10* y *G=100.*

3.6. FUNCIÓN DE ROSENBROCK 2

El mejor óptimo, el mejor promedio y la menor desviación se han registrado para *G=1000* con 56% de infactibilidades. Obviamente las salidas de la región factible no quitan estabilidad al algoritmo en esta instancia.

Las mayores desviaciones se han alcanzado para *G=1* y han sido algo superiores a 2 diezmilésimas. Para *G=10* , *100* y *1000* las desviaciones han sido prácticamente nulas.

En la hoja de Excel 3.29 podemos comprobar que S.G.O. se presenta también muy robusto en este caso.

Los criterios de elección de los valores de *G* son similares a los anteriores.

FUNCION ROSENBROCK 2

VALORES DEL ÓPTIMO

ALFA=0.005, BETA=0.99 , N=1000 , N_ITER=1000

	G = 0.1	G = 1	G = 10	G = 100	G = 1000	G = 10000
	0.000002	0.000062	0.000007	0.000006	0.000002	0.000010
	0.000052	0.000087	0.000036	0.000001	0.000007	0.000008
	0.000037	0.000831	0.000016	0.000003	0.000005	0.000219
	0.000077	0.000078	0.000020	0.000008	0.000003	0.000308
	0.000079	0.000005	0.000026	0.000005	0.000000	0.000205
	0.000150	0.000013	0.000036	0.000001	0.000003	0.000167
	0.000168	0.000001	0.000029	0.000004	0.000005	0.000189
	0.000074	0.000013	0.000023	0.000006	0.000001	0.000273
	0.000084	0.000244	0.000012	0.000009	0.000004	0.000152
	0.000098	0.000009	0.000014	0.000003	0.000001	0.000454
PROMEDIO	0.000082	0.000134	0.000022	0.000005	0.000003	0.000198
DES. TIPICA	0.000049	0.000256	0.000010	0.000003	0.000002	0.000133

MEJOR OPTIMO	0.000000
MEJOR PROMEDIO	0.000003
MENOR DESVIACIÓN	0.000002

MEJOR ÓPTIMO	G=1.000
MEJOR PROMEDIO	G=1.000
MENOR DESVIACIÓN	G=1.000

Figura 3.29. Hoja Excel de variación de *G* con ROSENBROCK 2.

Mejores valores de *G* para esta instancia: *G=10* , *G=100* y *G=1.000.*

3.7. FUNCIÓN DE GRIEWANK 2

En este caso, hemos obtenido el mejor promedio y la menor desviación para *G=100000* y el mejor óptimo para *G=1000000* con un porcentaje de infactibilidades en ambos casos del 0%.

Las infactibilidades no han llegado en ningún caso al 1%. En este caso la robustez del algoritmo, que queda ampliamente comprobada, no se ve afectada por las reinicializaciones de posiciones y velocidades de los agentes de búsqueda.

Los mayores valores de desviaciones se han alcanzado para los menores valores de *G*.

FUNCION GRIEWANK 2

VALORES DEL ÓPTIMO

ALFA=0.005, BETA=0.99 , N=1000 , N_ITER=1000

	G = 100	G = 1000	G = 10000	G = 100000	G = 10^6	G = $5*10^6$
	0.007210	0.006018	0.001146	0.000001	0.000003	0.000647
	0.007758	0.007003	0.003051	0.000003	0.000000	0.001877
	0.004830	0.006620	0.000164	0.000006	0.000172	0.002620
	0.007848	0.007551	0.002399	0.000002	0.000001	0.000420
	0.001601	0.009644	0.000035	0.000001	0.000021	0.000556
	0.005153	0.008577	0.001815	0.000001	0.000002	0.000963
	0.005357	0.008291	0.001816	0.000001	0.000008	0.001214
	0.001948	0.003551	0.000342	0.000003	0.000019	0.002300
	0.008505	0.005853	0.001317	0.000000	0.000024	0.004048
	0.005947	0.002339	0.000728	0.000002	0.000026	0.005290
PROMEDIO	0.005616	0.006545	0.001281	0.000002	0.000028	0.001993
DES. TIPICA	0.002381	0.002246	0.000998	0.000002	0.000052	0.001620

MEJOR OPTIMO	0.000000
MEJOR PROMEDIO	0.000002
MENOR DESVIACIÓN	0.000002

MEJOR ÓPTIMO	G=1.000.000
MEJOR PROMEDIO	G=100.000
MENOR DESVIACIÓN	G=100.000

Figura 3.30. Hoja Excel de variación de *G* con GRIEWANK 2.

Mejores valores de *G* para esta instancia: *G=10.000* , *G=100.000* y *G=1.000.000.*

3.8. FUNCIÓN DE ROSENBROCK 4

En este caso hemos tenido en cuenta los 5 resultados obtenidos con los valores $\alpha = 0.005$, $\beta = 0.5$, $G = 10$, 100 y 1.000, $n = 4000$, $n\_iter = 1000$.

Al igual que en el caso bidimensional hemos obtenido el mejor promedio, el mejor óptimo y la menor desviación típica para el mismo valor de *G*, pero en esta ocasión ha sido para *G=100* en vez de *G=1000.* El porcentaje de infactibilidades ha sido cercano al 27%. Al tomar $\beta = 0.5$ las velocidades se han reducido a la mitad y con ellas las infactibilidades y, por ello, ha sido necesario aumentar *G* para obtener el óptimo.

La mayor desviación ha sido inferior a 5 milésimas. Se ha alcanzado para el menor valor de *G.*

Aunque hemos alcanzado el óptimo con menor precisión que en dimensión 2, la robustez del algoritmo ha quedado igualmente comprobada, como puede apreciarse en la figura 3.31.

	A	B	C	D	E	F	G	H	I	J
1										
2		FUNCION ROSENBROCK 4								
3										
4						ALFA=0.005, BETA=0.99 , N=4000 , N_ITER=1000				
5			VALORES DEL ÓPTIMO							
6										
7										
8										
9		G = 10	G = 100	G = 1000						
10		0.017746	0.000196	0.001939						
11		0.021503	0.000195	0.002051						
12		0.027271	0.000195	0.002084						
13		0.028343	0.000196	0.002154						
14		0.028477	0.000197	0.002176						
15	PROMEDIO	0.024668	0.000196	0.002081						
16	DES. TIPICA	0.004815	0.000001	0.000094						
17										
18	MEJOR OPTIMO	0.000195				MEJOR ÓPTIMO	G=100			
19	MEJOR PROMEDIO	0.000196				MEJOR PROMEDIO	G=100			
20	MENOR DESVIACIÓN	0.000001				MENOR DESVIACIÓN	G=100			
21										

Figura 3.31. Hoja Excel de variación de *G* con ROSENBROCK 4.

Mejores valores de *G* para esta instancia: G=*10*, *G=100* y *G=1.000*

3.9. FUNCIÓN DE RASTRIGIN 4

Hemos considerado los 5 resultados logrados para los valores $\alpha = 0.005$, β = 0.99 , *G = 100 , 1000 y 10000 , n = 1000* , $n\_iter = 1000$

También en esta ocasión hemos obtenido los 3 mejores valores para el mismo valor de *G=1000.*

En esta ocasión las desviaciones típicas han sido considerablemente superiores a las del resto de las instancias consideradas, pero siguen siendo muy inferiores a los promedios obtenidos, lo que sigue garantizando la robustez del algoritmo incluso para una instancia con una topología tan complicada como la que nos ocupa.

	A	B	C	D	E	F	G	H	I	J
1										
2		FUNCION RASTRIGIN 4								
3										
4						ALFA=0.005, BETA=0.99 , N=1000 , N_ITER=1000				
5			VALORES DEL ÓPTIMO							
6										
7										
8										
9		*G = 100*	*G = 1000*	*G = 10000*						
10		0.314908	0.000513	0.438086						
11		0.980146	0.000977	1.078852						
12		0.668329	0.003593	1.217385						
13		0.797961	0.005212	1.142693						
14		0.827190	0.002299	1.229697						
15	PROMEDIO	0.717707	0.002519	1.021343						
16	DES. TIPICA	0.250991	0.001927	0.331687						
17										
18	MEJOR OPTIMO	0.000513				MEJOR ÓPTIMO	G=1.000			
19	MEJOR PROMEDIO	0.002519				MEJOR PROMEDIO	G=1.000			
20	MENOR DESVIACIÓN	0.001927				MENOR DESVIACIÓN	G=1.000			
21										
22										

Figura 3.32. Hoja Excel de variación de *G* con RASTRIGIN 4.

Mejores valores de *G* para esta instancia: *G=100, G=1.000 y G=10.000*

3.10. CONCLUSIONES

A la vista de los resultados expuestos podemos concluir

- Las desviaciones típicas han permanecido en todos los casos en valores muy bajos, lo que significa que S.G.O. *es un algoritmo robusto respecto a la variación de G.* Su componente aleatoria en las salidas de los asteroides no resta estabilidad en los resultados.
- En dimensión 2 se ha mostrado además como un algoritmo eficaz, ya que los resultados han sido muy buenos.
- Dentro de cada instancia, las menores desviaciones se han obtenido para los valores de *G* en los que se encontraban los mejores resultados, es decir, cuando S.G.O. obtiene una vez un buen resultado no es aislado, sino que obtiene muchas veces buenos resultados.
- Tal y como era de esperar, la terna de mejores valores para *G* ha variado en cada instancia, pues depende, obviamente, de la topología de la función.

A la vista de las conclusiones hemos alcanzado el primero de los objetivos planteados en la presente sección, *comprobar la robustez de S.G.O.*, pero no así el segundo: *fijar un rango de valores universales para G.* Sin embargo creemos imprescindible determinarlo de cara a posteriores experimentaciones. Por ello hemos realizado un estudio estadístico de los valores de *G* en los que hemos obtenido el mejor óptimo, el mejor promedio y la menor desviación en cada caso, recogido en la hoja de Excel de la figura 3.33, y hemos elegido para *G* la terna consecutiva de mejores resultados. Esta es *G = 1.000 , 10.000* y *100.000*.

En posteriores experimentaciones, sea cual sea la instancia, mantendremos los valores para *G* en este rango.

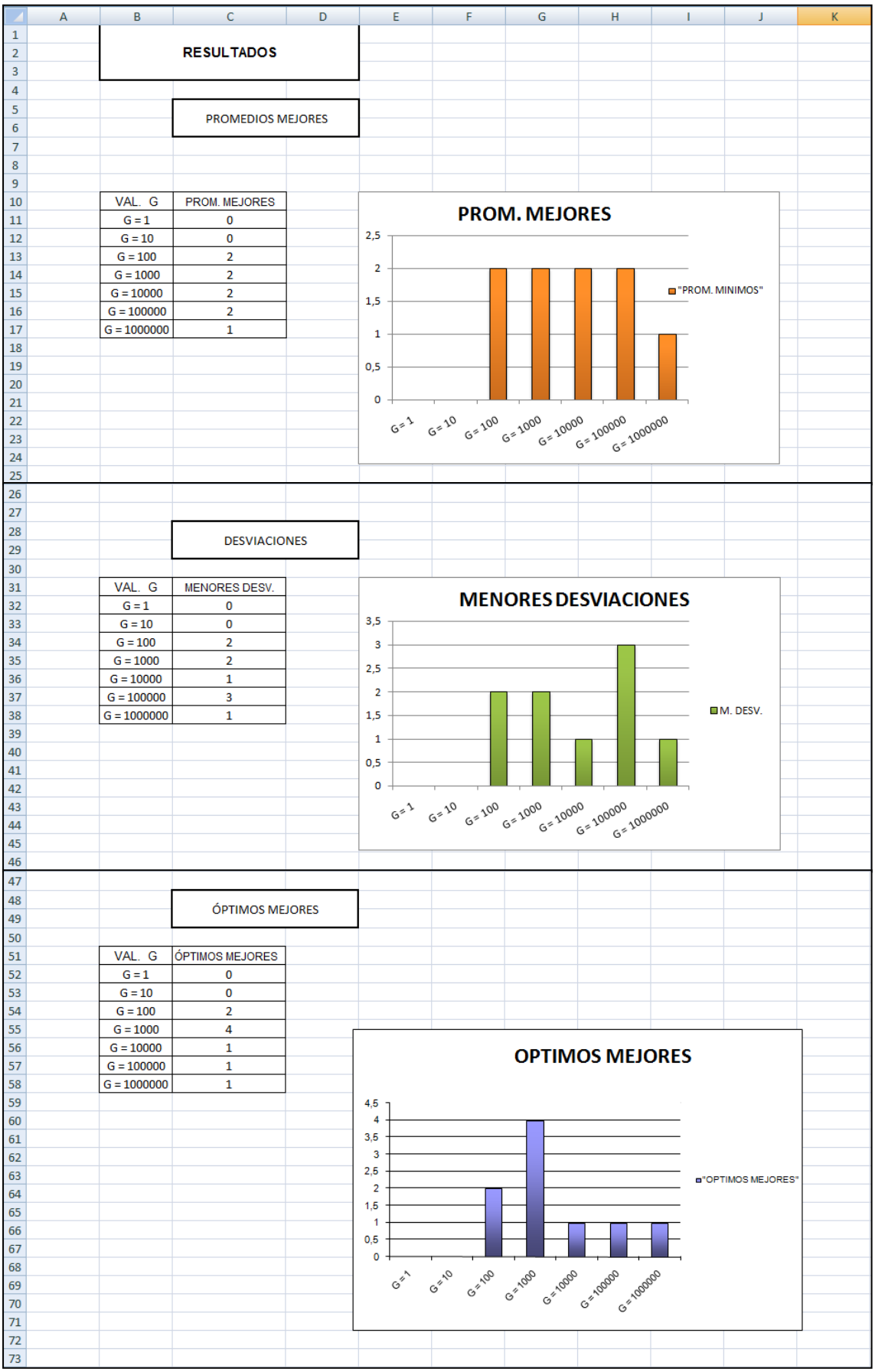


Figura 3.33. Hoja Excel de mejores valores generales para *G*.

4. LA ROBUSTEZ DEL ALGORITMO CON ASTEROIDES Y MOVIMIENTOS

Como ya sabemos, parte de la componente aleatoria de S.G.O. proviene de la inicialización de las posiciones y velocidades de los asteroides, lo que exclusivamente depende de *n*, y *n_iter*. *Uno de nuestros objetivo a lo largo de esta sección será intentar investigar cómo influye cada uno de ellos en la robustez de S.G.O.*

Por otra parte, nos interesa empezar a considerar el número de llamadas a la función objetivo[20] y la eficiencia que tendrá S.G.O. cuando lo enfrentemos a instancias de dimensiones superiores a 2. Dado que nuestra intención es encontrar valores universales, i.e. independientes de las instancias, para los parámetros $\alpha, \beta y\, G$ solo quedará al albedrío del usuario variar los valores de *n* y *n_iter.* La pregunta obvia que nos formulamos es *¿cómo influye la variación de n y n_iter en la eficiencia y la eficacia de S.G.O.?*

La eficiencia está intrínsecamente relacionada con el número de llamadas a la función objetivo. Para estudiar la eficacia haremos un estudio similar al realizado en la sección anterior respecto a la variación de *G*. Ambos conformarán nuestro segundo objetivo en esta sección.

El número total de llamadas a la función objetivo al ejecutar el algoritmo pueden calcularse de la siguiente forma:

$$n \cdot n\_iter^{(1)} + 2\dim n \cdot n\_iter^{(2)} = n \cdot n\_iter\left(2\dim+1\right)$$

(1) Para calcular el óptimo en cada iteración.

(2) Para calcular las aceleraciones que permitirán calcular las nuevas posiciones.

Por lo tanto, el número de evaluaciones de la función objetivo puede variarse únicamente a través del producto $n \cdot n\_iter$.

Así, haremos un estudio inicial sobre una muestra, de forma que varíen ambos factores en potencias de 10 pero manteniendo su producto constante e igual a $1.000.000$, de esta forma analizaremos la influencia de cada uno de ellos en los resultados e intentaremos determinar los valores más apropiados para ambos.

Como muestra hemos elegido 5 de las 9 funciones de la sección anterior. Hemos procurado que abarquen diferentes topologías. Todas ellas tienen dimensión 2. En cada caso hemos tomado los parámetros $\alpha, \beta y\, G$ concretos con los que hemos obtenido los mejores resultados.

[20] Ya hemos comentado que no consideramos los tiempos de ejecución una eficaz medida de la eficiencia de un algoritmo.

Las instancias elegidas son:

- SCHWEFEL 2
- BRANIN 2
- EASOM 2
- SHUBERT 2
- ROSENBROCK 2

Realizaremos 10 ejecuciones con cada pareja de valores de *n* y *n_iter* y cada función.

Hemos desechado directamente las parejas extremas: $n = 1 \Rightarrow n\_iter = 10^6$ y $n = 10^6 \Rightarrow n\_iter = 1$. En el primer caso porque con un sólo asteroide no tiene sentido calcular el centro de gravedad del conjunto de asteroides y el programa funciona mal. En el segundo caso porque lanzar un millón de asteroides y no dejar que se muevan no representa de ninguna forma S.G.O.

En esta ocasión no expondremos las abscisas donde hemos encontrado los valores de los óptimos, aunque las hemos recogido todas en las correspondientes hojas Excel en el CD adjunto. A modo de ejemplo exponemos la hoja de Excel de resultados de variación *n, n_iter* con la función de ROSENBROCK 2 en la figura 3.34. También hemos incluido en esta hoja unos gráficos realizados con MATLAB para ver la variación de tiempos y de valores óptimos a lo largo de las 10 ejecuciones llevadas a cabo.

Tal y como hemos hecho en la sección anterior, detallaremos las hojas de Excel con los mejores promedios, los mejores resultados y las menores desviaciones en cada instancia. También incluimos un estudio sobre los tiempos de ejecución[21], ya que no solo se emplea tiempo en evaluar la función objetivo. Completamos las hojas de Excel con algunos gráficos para apreciar los resultados más fácilmente.

Posteriormente resumiremos todos los datos con una serie de gráficos de barras recogidos en la figura 3.40.

[21] Hemos tenido la precaución de realizar todas las ejecuciones de la misma función con el mismo equipo.

PRUEBAS N-NITER (10 EJEC.)

FUNCIÓN ROSENBROCK 2

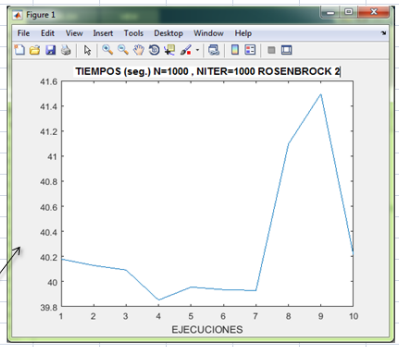
ALFA	0.005
BETA	0.99
G	1.000

N=10, N_ITER=100.000

EJECUCIÓN	XOPTIMO	YOPTIMO	FOPTIMO	TIEMPO (seg.)	INFACTIBILIDADES
1	0.979428	0.960585	0.000594	44.4	800516 (80%)
2	1.009030	1.015979	0.000549	43.8	800764
3	1.019564	1.042591	0.001332	43.8	800534
4	1.006492	1.012323	0.000092	44.3	800445
5	1.000316	0.995477	0.002658	44.2	799323
6	0.994500	0.991026	0.000429	43.9	799058
7	1.023410	1.047523	0.000550	43.9	799413
8	0.983924	0.970248	0.000717	43.8	799674
9	1.002248	1.006035	0.000241	43.7	800122
10	0.990772	0.979285	0.000634	43.8	800071

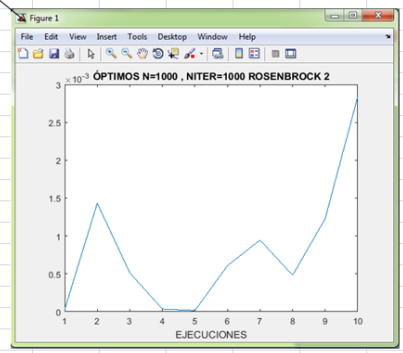
N=100, N_ITER=10.000

EJECUCIÓN	XOPTIMO	YOPTIMO	FOPTIMO	TIEMPO (seg.)	INFACTIBILIDADES
1	0.997426	0.993211	0.000278	40.3	800533 (80%)
2	0.992516	0.986449	0.000241	40.4	799523
3	1.022907	1.050495	0.002252	40.3	800140
4	1.037003	1.077291	0.001736	40.4	799567
5	0.980481	0.960840	0.000406	40.4	800179
6	0.999605	1.000536	0.000176	40.3	800660
7	0.963129	0.927191	0.001378	40.4	800644
8	0.982740	0.963548	0.000796	40.4	800166
9	0.963458	0.927479	0.001395	40.4	799216
10	0.973912	0.945817	0.001403	40.1	799417



N=1.000, N_ITER=1.000

EJECUCIÓN	XOPTIMO	YOPTIMO	FOPTIMO	TIEMPO (seg.)	INFACTIBILIDADES
1	0.998916	0.997302	0.000029	40.2	798247 (79,8%)
2	1.018292	1.040233	0.001433	40.1	798545
3	1.009229	1.020606	0.000511	40.1	798715
4	0.995467	0.990558	0.000036	39.9	800007
5	1.003407	1.007022	0.000015	40.0	798868
6	0.980125	0.962108	0.000609	39.9	799662
7	1.024939	1.048700	0.000946	39.9	799733
8	1.021804	1.043854	0.000481	41.1	799108
9	1.026519	1.051450	0.001228	41.5	800017
10	1.052995	1.108255	0.002838	40.2	799668



N=10.000, N_ITER=100

EJECUCIÓN	XOPTIMO	YOPTIMO	FOPTIMO	TIEMPO	INFACTIBILIDADES
1	0.993487	0.983887	0.001021	39.8	793028 (79,4%)
2	0.945218	0.895078	0.003270	39.8	794165
3	0.985743	0.970089	0.000460	39.7	793730
4	1.015926	1.033857	0.000560	39.7	794300
5	1.015106	1.032465	0.000639	39.7	794156
6	1.026446	1.055920	0.001242	39.8	793342
7	1.032472	1.065206	0.001117	39.7	794021
8	0.983033	0.964627	0.000586	39.7	793178
9	1.016391	1.032051	0.000369	39.8	792531
10	1.001220	1.001179	0.000161	39.7	793875

N=100.000, N_ITER=10

EJECUCIÓN	XOPTIMO	YOPTIMO	FOPTIMO	TIEMPO (seg.)	INFACTIBILIDADES
1	0.991239	0.981237	0.000251	39.8	726841 (72'6%)
2	0.979817	0.956330	0.001786	39.6	726343
3	1.019178	1.037846	0.000445	39.6	727595
4	0.998757	0.999160	0.000272	39.7	727391
5	0.988083	0.978007	0.000431	39.7	727264
6	1.011636	1.024402	0.000234	39.6	726971
7	1.026116	1.051669	0.000837	39.7	727439
8	0.992502	0.987280	0.000549	39.7	727449
9	0.989876	0.980449	0.000138	39.7	727388
10	0.998070	0.993047	0.000964	39.7	725864

Figura 3.34. Hoja Excel de resultados de variación *n* , *n_iter* de ROSENBROCK 2.

4.1. RESULTADOS *n, n_iter* DE LA FUNCIÓN DE SCHWEFEL 2

En la figura 3.35 podemos apreciar que las diferencias de tiempos de ejecución rondan los 7 segundos, a nuestro juicio poco significativo.

El mejor promedio y la menor desviación se han alcanzado para $n=100 \Rightarrow n\_iter=10^4$, sin embargo el mejor óptimo se ha obtenido para $n=10 \Rightarrow n\_iter=10^5$ con una desviación típica cercana a 100. Como en ocasiones anteriores, el mejor óptimo se ha encontrado de forma casual, pero las diferencias entre el mejor óptimo y el mejor promedio son poco significativas. El algoritmo se ha mostrado robusto ya que los valores de promedios de óptimos hallados a partir de $n=100 \Rightarrow n\_iter=10^4$ han sido muy similares como se observa en la figura 3.35

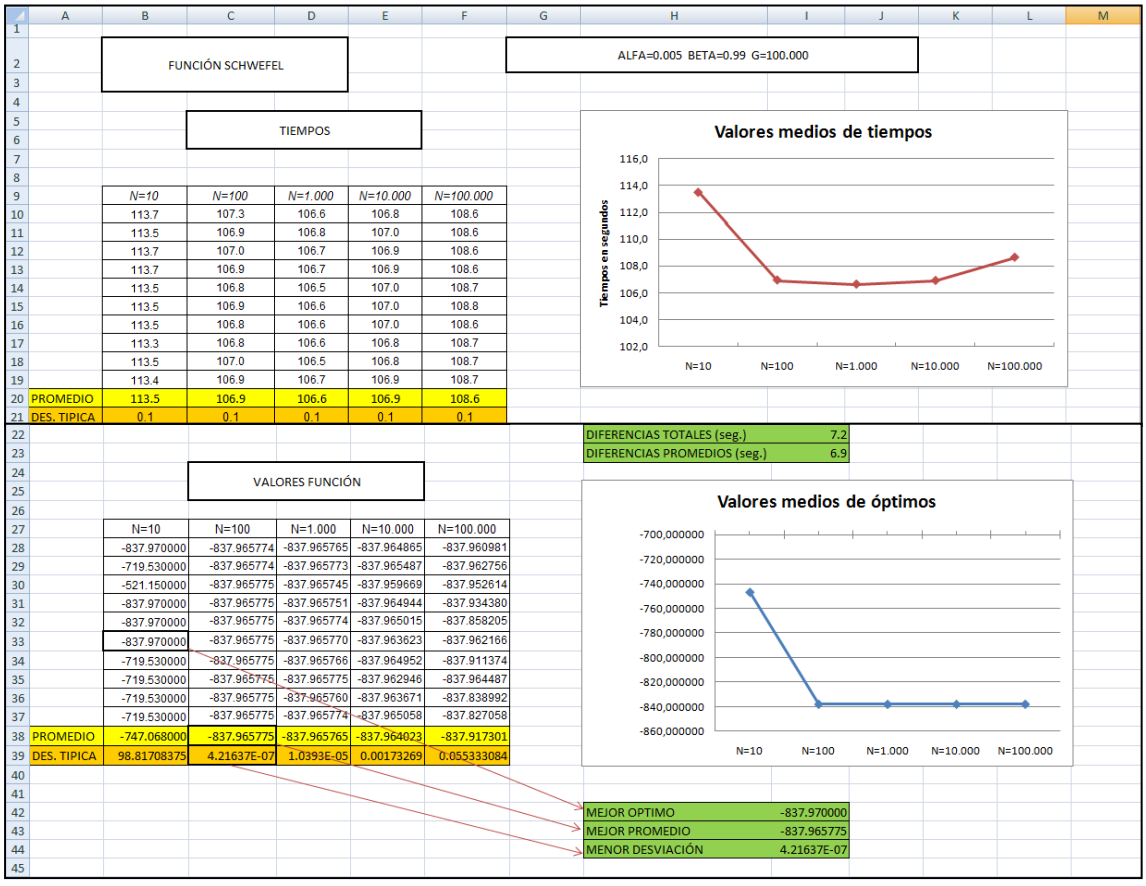
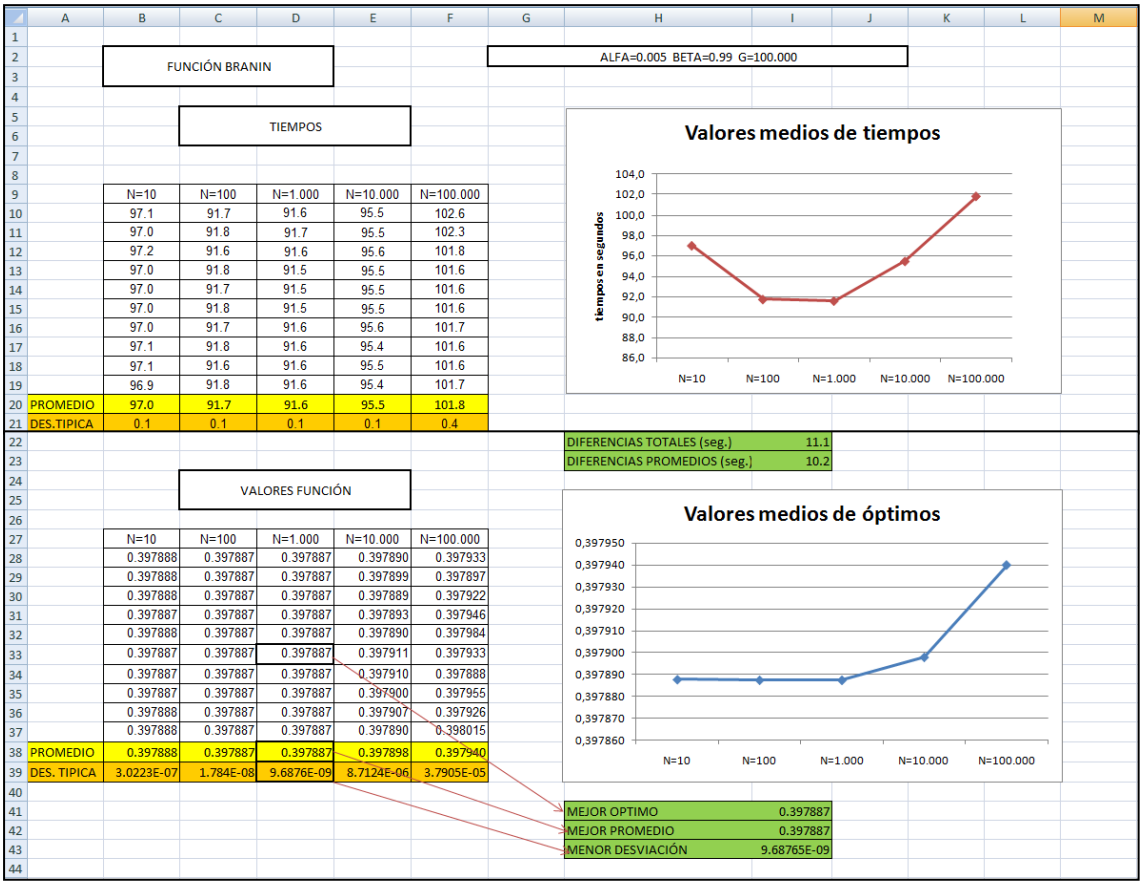


Figura 3.35. Hoja Excel con resultados *n , n_iter* de SCHWEFEL 2.

4.2. RESULTADOS *n, n_iter* DE LA FUNCIÓN DE BRANIN 2

En esta ocasión, el mejor óptimo, el mejor promedio y la menor desviación se han dado en la misma columna: $n = 1000 \Rightarrow n\_iter = 10^3$ en la que, además, se han obtenido los menores tiempos.

Los peores valores de promedios y de desviaciones típicas se han obtenido para $n = 10^5 \Rightarrow n\_iter = 10$ coincidiendo con el mayor tiempo de ejecución. Claramente en esta instancia es preferible dejar que los asteroides se muevan. No obstante, todas las desviaciones típicas son prácticamente nulas, lo que corrobora la robustez de S.G.O. para esta instancia.



	N=10	N=100	N=1.000	N=10.000	N=100.000
	97.1	91.7	91.6	95.5	102.6
	97.0	91.8	91.7	95.5	102.3
	97.2	91.6	91.6	95.6	101.8
	97.0	91.8	91.5	95.5	101.6
	97.0	91.7	91.5	95.5	101.6
	97.0	91.8	91.5	95.5	101.6
	97.0	91.7	91.6	95.6	101.7
	97.1	91.8	91.6	95.4	101.6
	97.1	91.6	91.6	95.5	101.6
	96.9	91.8	91.6	95.4	101.7
PROMEDIO	97.0	91.7	91.6	95.5	101.8
DES.TIPICA	0.1	0.1	0.1	0.1	0.4

	N=10	N=100	N=1.000	N=10.000	N=100.000
	0.397888	0.397887	0.397887	0.397890	0.397933
	0.397888	0.397887	0.397887	0.397899	0.397897
	0.397888	0.397887	0.397887	0.397889	0.397922
	0.397887	0.397887	0.397887	0.397893	0.397946
	0.397888	0.397887	0.397887	0.397890	0.397984
	0.397887	0.397887	0.397887	0.397911	0.397933
	0.397887	0.397887	0.397887	0.397910	0.397888
	0.397887	0.397887	0.397887	0.397900	0.397955
	0.397888	0.397887	0.397887	0.397907	0.397926
	0.397888	0.397887	0.397887	0.397890	0.398015
PROMEDIO	0.397888	0.397887	0.397887	0.397898	0.397940
DES. TIPICA	3.0223E-07	1.784E-08	9.6876E-09	8.7124E-06	3.7905E-05

Figura 3.36. Hoja Excel con resultados *n* , *n_iter* de BRANIN 2.

4.3. RESULTADOS *n, n_iter* DE LA FUNCIÓN DE EASOM 2

Esta instancia presenta valores de promedios mejores en los valores intermedios que en los extremos. Los mejores valores de óptimo, promedios y desviaciones se han alcanzado para $n=100 \Rightarrow n\_iter=10^4$. Todas las desviaciones típicas registradas han sido inferiores a 2 centésimas. Nuevamente hemos confirmado la robustez de S.G.O. para esta instancia.

Los mayores tiempos se han registrado para $n=10 \Rightarrow n\_iter=10^5$ a la vez que el peor promedio y el peor óptimo.

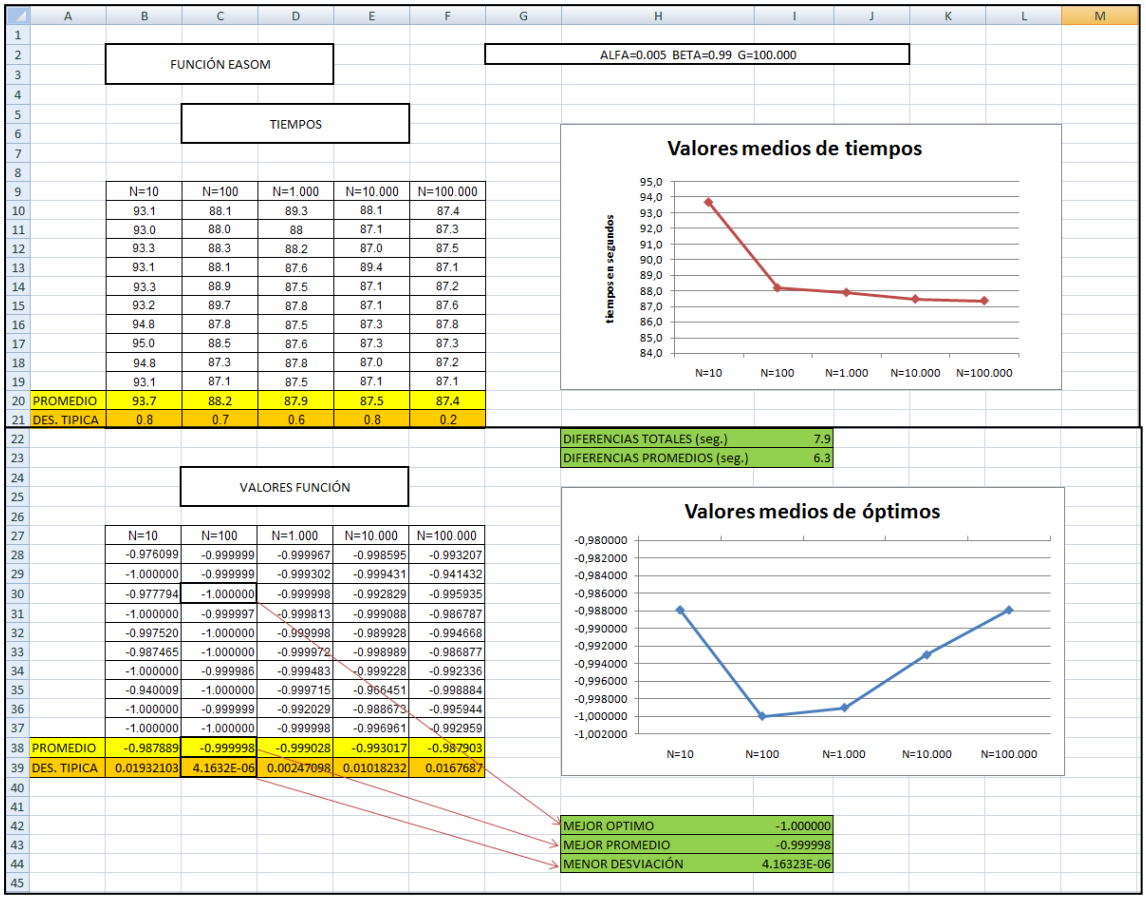


Figura 3.37. Hoja Excel de resultados *n* , *n_iter* de EASOM 2.

4.4. RESULTADOS *n, n_iter* DE LA FUNCIÓN DE SHUBERT 2

Los mejores valores de óptimos, promedios y desviaciones se han alcanzado para $n = 100 \Rightarrow n\_iter = 10^4$. No obstante, todos los promedios han permanecido en valores similares.

La mayor desviación ha sido algo superior a 6 centésimas lo que confirma de nuevo la robustez del algoritmo.

Las diferencias de tiempos se mantienen muy estables en las cuatro primeras columnas, no así en la última que disminuye en 40 segundos. Al ser una función que conlleva un alto número de operaciones el cálculo de las aceleraciones tiene un peso significativo.

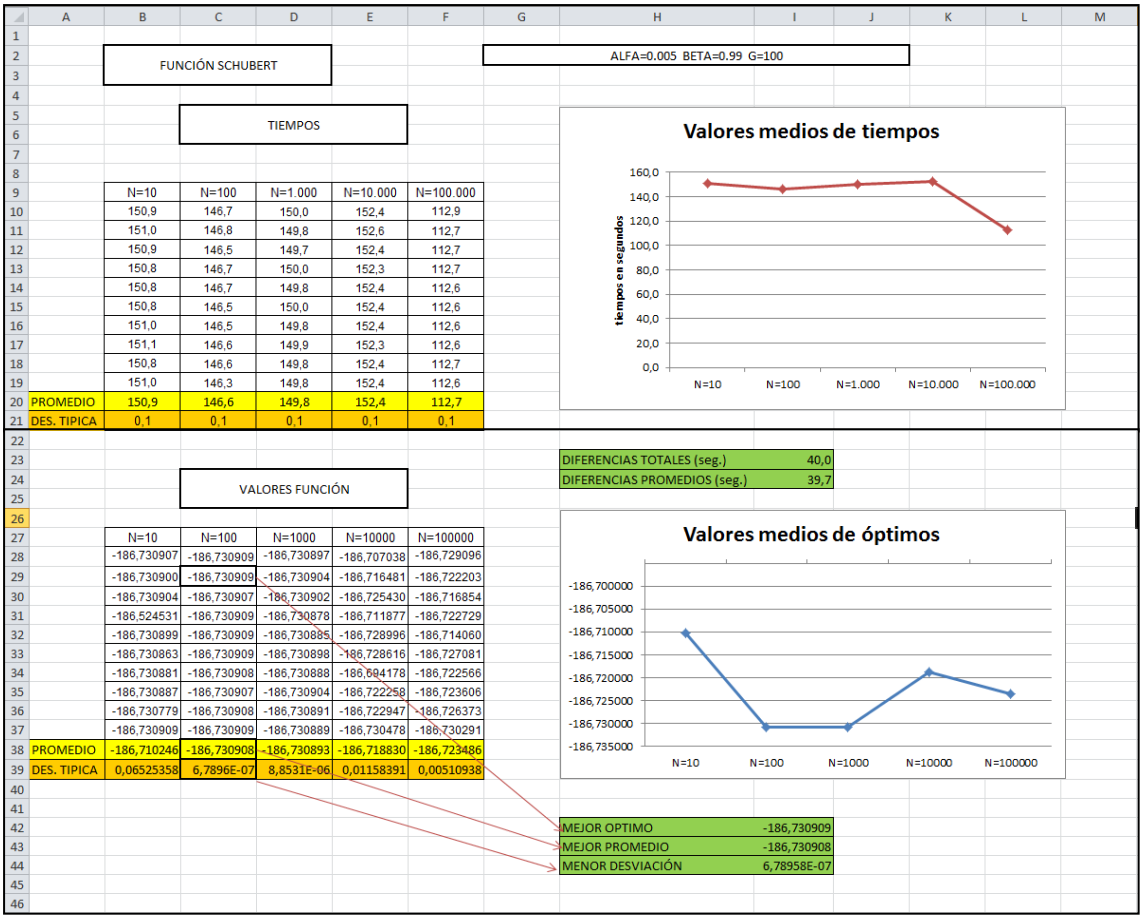


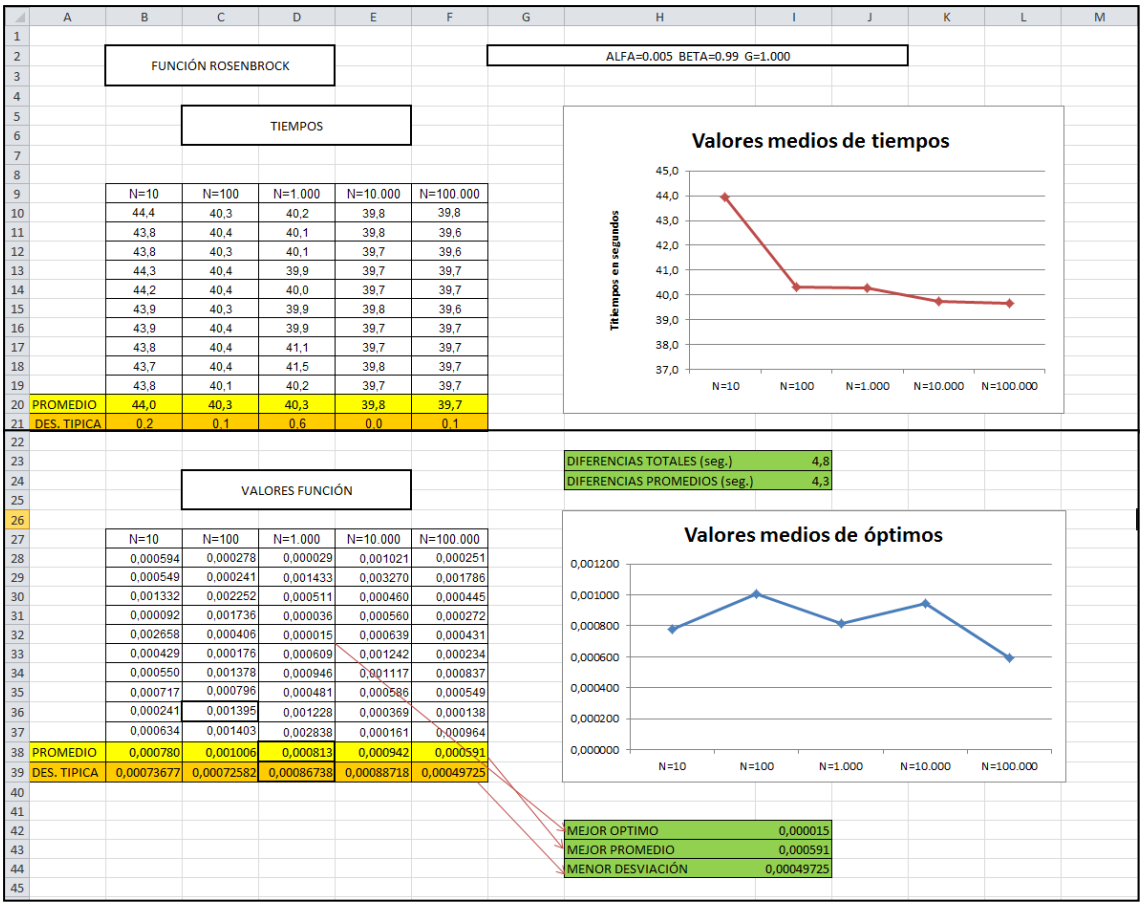
Figura 3.38. Hoja Excel de resultados *n* , *n_iter* de SHUBERT 2.

4.5. RESULTADOS *n, n_iter* DE LA FUNCIÓN DE ROSENBROCK 2

El mejor valor para el óptimo se ha alcanzado en $n = 1.000 \Rightarrow n\_iter = 10^2$, sin embargo los mejores resultados de promedio y desviación se han dado en $n = 100.000 \Rightarrow n\_iter = 10$. Es la única instancia donde hemos obtenido los resultados más estables en valores para *n* tan extremos.

No obstante, la mayor desviación es algo inferior a 9 diezmilésimas lo que avala claramente la robustez del algoritmo en esta ocasión.

La máxima diferencia de tiempos es de 5 segundos, muy poco significativa.



	N=10	N=100	N=1.000	N=10.000	N=100.000
	44,4	40,3	40,2	39,8	39,8
	43,8	40,4	40,1	39,8	39,6
	43,8	40,3	40,1	39,7	39,6
	44,3	40,4	39,9	39,7	39,7
	44,2	40,4	40,0	39,7	39,7
	43,9	40,3	39,9	39,8	39,6
	43,9	40,4	39,9	39,7	39,7
	43,8	40,4	41,1	39,7	39,7
	43,7	40,4	41,5	39,8	39,7
	43,8	40,1	40,2	39,7	39,7
PROMEDIO	44,0	40,3	40,3	39,8	39,7
DES. TIPICA	0,2	0,1	0,6	0,0	0,1

	N=10	N=100	N=1.000	N=10.000	N=100.000
	0,000594	0,000278	0,000029	0,001021	0,000251
	0,000549	0,000241	0,001433	0,003270	0,001786
	0,001332	0,002252	0,000511	0,000460	0,000445
	0,000092	0,001736	0,000036	0,000560	0,000272
	0,002658	0,000406	0,000015	0,000639	0,000431
	0,000429	0,000176	0,000609	0,001242	0,000234
	0,000550	0,001378	0,000946	0,001117	0,000837
	0,000717	0,000796	0,000481	0,000586	0,000549
	0,000241	0,001395	0,001228	0,000369	0,000138
	0,000634	0,001403	0,002838	0,000161	0,000964
PROMEDIO	0,000780	0,001006	0,000813	0,000942	0,000591
DES. TIPICA	0,00073677	0,00072582	0,00086738	0,00088718	0,00049725

Figura 3.39. Hoja Excel de resultados *n* , *n_iter* de ROSENBROCK 2.

4.6. CONCLUSIONES

Hemos recogido los valores de *n* y *n_iter* en los que se han alcanzado los mejores resultados, tanto de promedios de tiempos como de promedios y desviaciones típicas de valores de la función, en la hoja Excel de la figura 3.40. Las diferencias de tiempos de ejecución no han sido significativas en ningún caso por lo que no tendremos en cuenta sus desviaciones típicas.

A la vista de los resultados obtenidos en la muestra seleccionada podemos obtener varias conclusiones:

- En la mayor parte de los casos analizados las desviaciones típicas de óptimos han sido muy pequeñas lo que demuestra la robustez del algoritmo respecto a la variación de los valores *n*, *n_iter*.
- Los mejores promedios en cuanto a tiempos de ejecución dependen de cada función, ya que, si ésta conlleva gran número de operaciones, los tiempos empleados en evaluar las nuevas posiciones de los asteroides incrementan el tiempo total de ejecución. Por ello el algoritmo se ha mostrado más eficiente en el caso $n = 100.000 \Rightarrow n\_iter = 10$.
- Los mejores resultados en cuanto a promedios de valores de la función también dependen de cada instancia. A diferencia del caso anterior, el valor más repetido en cuanto a eficacia de S.G.O. ha sido $n = 100 \Rightarrow n\_iter = 10^4$.
- Las menores desviaciones típicas, que conllevan los comportamientos más estables del algoritmo, han resultado en los mismos valores que los mejores promedios. Lo que implica, en última instancia, la robustez de S.G.O.

Es importante reseñar que, a la vista de los resultados, para que el algoritmo desarrolle toda su potencia es necesario lanzar una mínima cantidad de asteroides, i.e., una **masa crítica de agentes de búsqueda** que, a su vez, necesitan un tiempo suficiente para localizar el óptimo. En este caso esa masa crítica corresponde al rango de valores entre $n = 100$ *y* $n = 1.000$ asteroides que se muevan entre $n\_iter = 10^4$ y $n\_iter = 10^3$ unidades de tiempo. En la muestra analizada, lanzar menos asteroides representa una masa crítica demasiado pequeña para explorar la región factible y lanzar más asteroides, permitiendo menos movimientos, provoca “aglomeraciones” que dificultan la búsqueda del óptimo.

Puesto que, a nuestro juicio, la eficacia prima sobre la eficiencia y además el rango de variación de los tiempos no difiere en gran medida en las funciones consideradas, nos decantamos por usar los valores $n = 100$ o $n = 1000$.

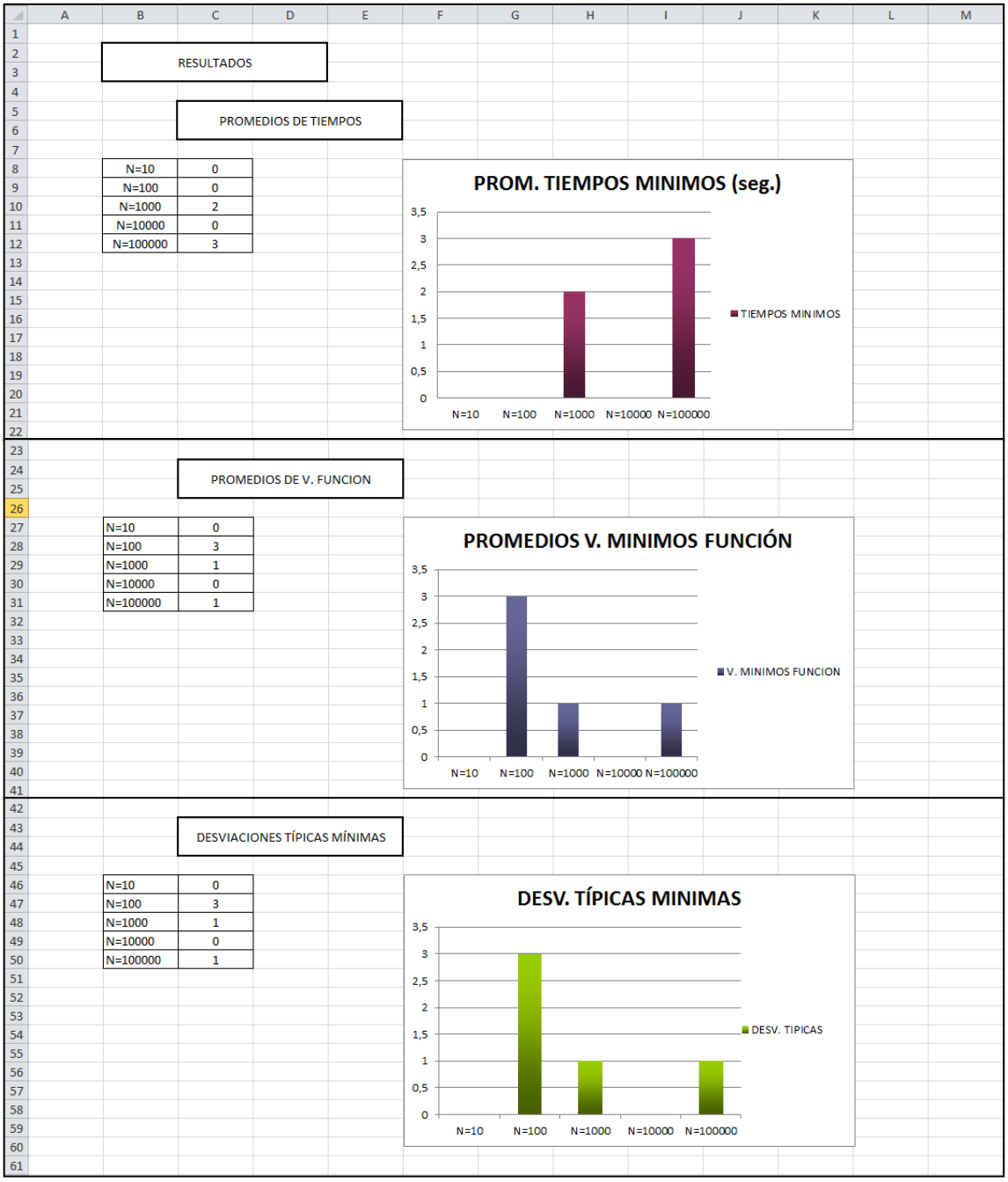


Figura 3.40. Hoja de Excel de resultados generales *n* , *n_iter*.

5. CONCLUSIONES

En el presente capítulo hemos realizado una primera tanda de pruebas con S.G.O., hemos comprobado su eficiencia y eficacia en dimensiones bajas y hemos analizado su robustez respecto a la variación de los valores *G*, *n* y *n_iter*. Todo ello nos permite tomar decisiones respecto a los valores que van a tomar en posteriores ejecuciones. Pero no olvidemos que S.G.O. dispone de otros 2 parámetros -α y β- respecto de los cuales no tenemos prácticamente ninguna información y que es necesario estudiar.

Ésta será nuestra tarea a lo largo del capítulo siguiente, en la que elegiremos una serie de funciones de distintas dimensiones para observar los resultados obtenidos al variar los 3 parámetros -α, β y G- a la vez. La robustez del algoritmo, comprobada a lo largo de las secciones precedentes, nos permitirá repetir un número pequeño de veces cada uno de los experimentos. También tendremos fijado el rango de variación de *G* que será $G = 10^3, 10^4$ y 10^5.

Como consecuencia del estudio realizado en el apartado 4, elegiremos en todos los casos la pareja $n = 100 \Rightarrow n\_iter = 10^4$, pero tomaremos mediciones intermedias de tiempos y valores de la función cada 1000 iteraciones para observar la evolución del algoritmo.

A partir del siguiente capítulo no tendremos en cuenta las infactibilidades producidas, porque consideramos que no nos han aportado conocimiento efectivo sobre ningún aspecto del algoritmo.

"Hay solo dos clases de lenguajes de programación: aquellos de los que la gente está permanentemente quejándose y aquellos que nadie usa."

Bjarne Stroustrup (creador del lenguaje C++) (1950-)

CAPÍTULO 4

DEPENDENCIA DE LOS PARÁMETROS

1. INTRODUCCIÓN

Tras los primeros análisis de S.G.O. en dimensiones 2 y 4 hemos conseguido fijar sendos rangos de variación para el parámetro *G* y los valores de *n* y *n_iter* en los que el algoritmo se ha mostrado más eficaz. Pero no olvidemos que disponemos de otros dos parámetros α , β cuyos rangos sería interesante analizar y, en lo posible, establecer.

En las mediciones llevadas a cabo hasta el momento, los valores para ambos han sido elegidos de forma que su influencia fuera pequeña. Por ello sería interesante evaluar los resultados en experimentos en que su influencia sea mayor, y así ver cómo podemos sacar el máximo provecho de ellos.

Por otra parte, los parámetros estadísticos utilizados para estudiar la robustez y eficacia de S.G.O. en dimensión 2 nos parecen apropiados pero insuficientes. Creemos necesario analizarlas utilizando herramientas estadísticas más elaboradas que introduciremos en el presente capítulo.

Para llevar a cabo la evaluación y las mejoras consideradas en los párrafos precedentes, realizaremos a lo largo del presente capítulo distintas ejecuciones con S.G.O. variando sus tres parámetros principales, α , β y G , en un rango de 3 posibles valores cada uno, y registraremos y analizaremos estadísticamente los resultados obtenidos.

Tomaremos también datos intermedios de tiempos[1] y valores de la función objetivo para comprobar la evolución del algoritmo, es decir, para buscar en qué momento, definido como punto de estabilización -P.E.-, el grupo de asteroides no ha conseguido mejorar el óptimo. De esta forma tendremos también un estudio sobre su eficiencia.

Con el fin de realizar las pertinentes ejecuciones, 5 con cada instancia y cada trío de elección de parámetros, elegiremos una muestra de 6 *funciones test* con diferentes topologías y de dimensiones que varían entre 3 y 30. Hemos fijado el óptimo en todos los casos en el valor 0, para así facilitar el cálculo de los errores[2]. Sus expresiones analíticas se recogen en el apéndice 1.

Los valores de *n* y *n_iter* se han mantenido en los rangos de variación óptimos obtenidos del capítulo anterior.

[1] Ya que los datos de tiempos han sido muy diferentes dependiendo del equipo con el que hemos trabajado preferimos contar el nº de llamadas a la función objetivo.
[2] Si el óptimo inicial no es 0 trasladaremos la función en el eje de la variable dependiente usando el mejor dato que tenemos.

2. MOTIVACIÓN EN LA ELECCIÓN DE LAS FUNCIONES Y PARÁMETROS

Hemos elegido las instancias en base a cuatro criterios:

- Explorar dimensiones mayores que en las primeras mediciones.
- Explorar diferentes topologías.
- Escoger solo una función de cada dimensión.
- Hemos elegido la misma función, de Jong, para realizar las pruebas en dimensiones 3 y 30. Así comprobaremos cómo afecta el aumento de la dimensión del espacio de oportunidades en la fiabilidad del algoritmo.

En cuanto al rango de variación de los parámetros, 3 en cada caso, han sido los siguientes:

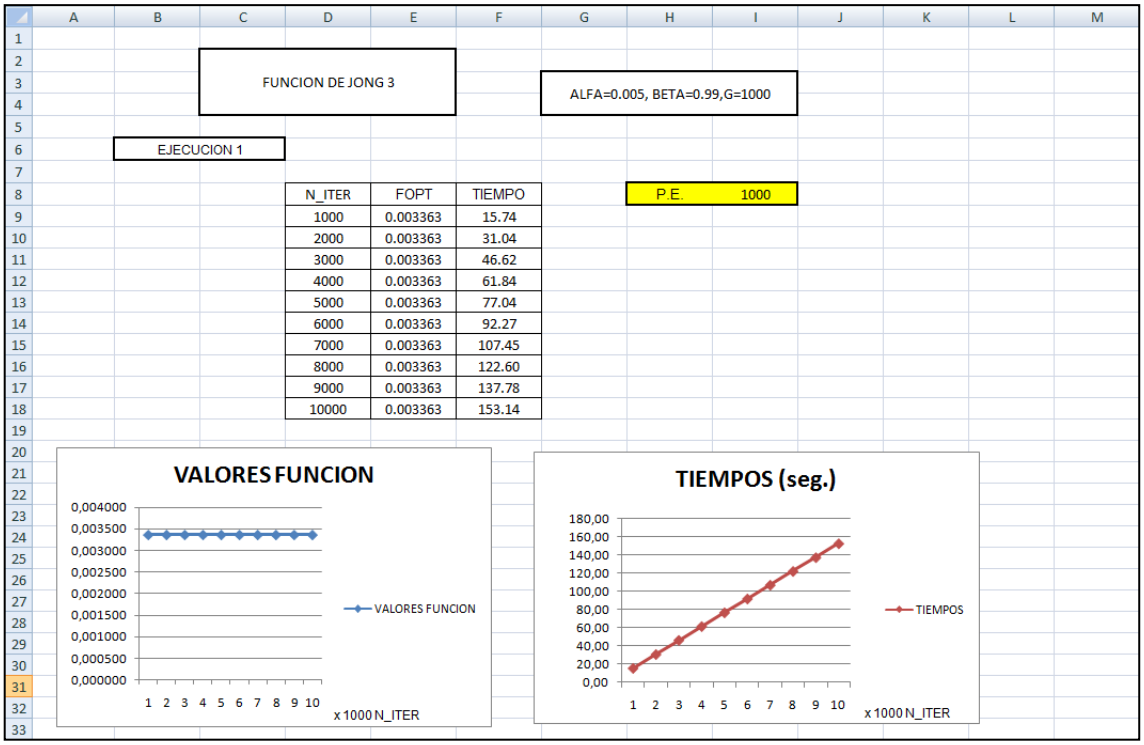
- *G* **variará en el rango 10^3, 10^4, 10^5,** determinado por los resultados obtenidos en el capítulo anterior.
- α **variará en el rango 0.005, 0.5, 1**. Su motivación física es la atracción que se produce entre asteroides, por tanto no puede ser negativa. En las mediciones realizadas hasta el momento le hemos dado valores cercanos a 0, por ello consideraremos valores mayores para ponderar más las interacciones entre los agentes de búsqueda.
- β **variará en el rango 0.4, 0.99, 1.5**. Su analogía física es la contracción o expansión del espacio. Hemos considerado oportuno tomar un valor de cada uno de los casos.

3. DESARROLLO DE LAS EJECUCIONES

Por último, expondremos cómo se han realizado las ejecuciones.

- Para cada función hemos hecho 5 ejecuciones con la misma terna de parámetros, lo que arroja un total de $27 \cdot 5 = 135$ ejecuciones por función. En la figura 4.1. presentamos un ejemplo de una de las 135 hojas de Excel generadas. Los gráficos de los tiempos solo se presentan en la primera de las 5 ejecuciones, es decir, solo 27 veces, ya que en todos los casos la variación de tiempo es lineal e independiente de los valores de los parámetros; no así los gráficos de los valores de la función que sí se han realizado para cada una de las 135 ejecuciones.

- En cada una de las 5 ejecuciones hemos lanzado *n = 100* asteroides que se han movido un total de *n_iter = 10000* iteraciones y hemos tomado los resultados intermedios cada 1.000 iteraciones, es decir, disponemos de 10 datos intermedios de tiempos y valores de la función objetivo. También hemos registrado el punto de estabilización del algoritmo, *P.E.*, que es el número intermedio de iteraciones a partir del cual no se ha mejorado el óptimo, figura 4.1.



N_ITER	FOPT	TIEMPO
1000	0.003363	15.74
2000	0.003363	31.04
3000	0.003363	46.62
4000	0.003363	61.84
5000	0.003363	77.04
6000	0.003363	92.27
7000	0.003363	107.45
8000	0.003363	122.60
9000	0.003363	137.78
10000	0.003363	153.14

Figura 4.1. Ejemplo de hoja Excel generada con una ejecución concreta con una terna de valores para la función de JONG 3.

- Una vez realizadas las 5 ejecuciones con cada terna concreta de parámetros, hemos creado una nueva hoja de Excel, PROMEDIOS, en la que se recogen los promedios y las desviaciones típicas de las 5 ejecuciones realizadas, en cada valor intermedio de *n_iter*. También hemos guardado el mejor óptimo obtenido en las 5 ejecuciones, mejor óptimo parcial, $mop_{\alpha\beta G}$, y el mejor promedio de óptimos parcial, $mpp_{\alpha\beta G}$, que obviamente será el promedio correspondiente a *n_iter=10.000*. También hemos hallado el punto de estabilización máximo*, P.E.M*, que es el máximo de los *P.E.* de las 5 ejecuciones. En la figura 4.2 exponemos un ejemplo de una hoja de Excel llamada PROMEDIOS.

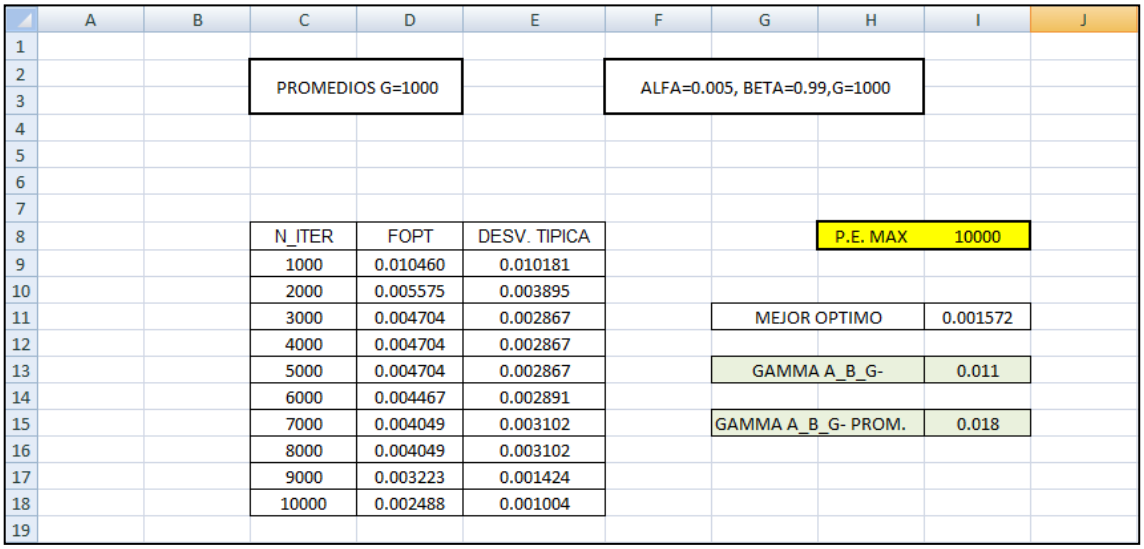
- Tras las 135 ejecuciones hemos guardado el mejor óptimo total, mot, y el peor óptimo total, pot, de las 135 ejecuciones.

- Con los datos totales hemos vuelto a cada una de las 27 hojas de Excel PROMEDIOS y hemos calculado los parámetros

$$\gamma_{\alpha\beta G} = \frac{mop_{\alpha\beta G} - mot}{pot - mot} \qquad \text{y} \qquad \gamma p_{\alpha\beta G} = \frac{mpp_{\alpha\beta G} - mot}{pot - mot}$$

que nos van a aportar una forma más certera de **cuantificar la robustez del algoritmo en función de los parámetros:** valores de ambos próximos a 0 indican que el funcionamiento del algoritmo es robusto y eficaz para la correspondiente terna de parámetros, mientras que si dichos valores se aproximan a 1 significa que el comportamiento del algoritmo para dicha terna es poco válido.

Hemos incluido dichos datos en la correspondiente hoja de Excel PROMEDIOS de las 5 ejecuciones tal y como puede observarse en el ejemplo recogido en la figura 4.2. perteneciente a la función de JONG de dimensión 3.



PROMEDIOS G=1000		
N_ITER	FOPT	DESV. TIPICA
1000	0.010460	0.010181
2000	0.005575	0.003895
3000	0.004704	0.002867
4000	0.004704	0.002867
5000	0.004704	0.002867
6000	0.004467	0.002891
7000	0.004049	0.003102
8000	0.004049	0.003102
9000	0.003223	0.001424
10000	0.002488	0.001004

ALFA=0.005, BETA=0.99,G=1000	
P.E. MAX	10000
MEJOR OPTIMO	0.001572
GAMMA A_B_G-	0.011
GAMMA A_B_G- PROM.	0.018

Figura 4.2. Ejemplo de hoja Excel PROMEDIOS de las 5 ejecuciones con una terna concreta para la función de JONG 3.

Para almacenar y manipular la ingente cantidad de datos generados hemos optado por crear 9 archivos en Excel llamados genéricamente "FUNCION_A(*)_B(**)"[3]. En cada uno de ellos hemos congelado los valores de α y β y hemos variado los valores de *G* en sus 3 posibilidades[4].

Cada archivo consta de 19 hojas: 15 con las ejecuciones para cada *G*, 3 con los correspondientes promedios y una última, RESULTADOS ARCHIVO, con los mejores y peores resultados obtenidos dentro del archivo, es decir, con los valores α y β fijos y los valores de *G* tomando sus 3 posibles valores. *E.F.O*. es el número de evaluaciones de la función objetivo, valor que, como hemos comentado ya en ocasiones precedentes, se mantiene fijo para cada instancia y además nos parece más significativo que el tiempo para medir la eficiencia de un algoritmo.

Mostramos un ejemplo concreto de una hoja RESULTADOS ARCHIVO en la figura 4.3. perteneciente a la función de JONG, de dimensión 3.

[3] (*) toma los valores : "+", "espacio en blanco" o " ++" dependiendo de los 3 valores del parámetro α y (**) toma los valores:" -", "espacio en blanco" o "+" dependiendo de los 3 valores del parámetro β.

[4] Para los 3 posibles valores de *G* también hemos usado la notación *G-, G* o *G+.*

	A	B	C	D	E	F	G	H	I	J
1										
2			FUNCION DE JONG 3							
3							ALFA=0.005, BETA=0.99, TODOS G			
4										
5										
6		E.F.O.								
7		700000								
8		1400000								
9		2100000		MEJOR OPTIMO	1.47093E-09	G=1E+05 , EJ-5, P.E.=3000 , E.F.O=2.100.000				
10		2800000		MEJOR PROMEDIO	2.13842E-09	G=1E+05 , P.E.M.=10.000 , E.F.O=7.000.000				
11		3500000								
12		4200000		PEOR OPTIMO	0.006818391	G=1E+04 , EJ-3 , P.E.=10000 , E.F.O=7.000.000				
13		4900000		PEOR PROMEDIO	0.002668049	G=1E+04 , P.E.M.=10000 , E.F.O=7.000.000				
14		5600000								
15		6300000								
16		7000000								
17										

Figura 4.3. Ejemplo de hoja Excel RESULTADOS ARCHIVO para la función de JONG 3.

Para recabar todos los datos obtenidos en cada instancia hemos creado un archivo en Excel llamado genéricamente "RESULTADOS FUNCIÓN" que consta de dos hojas: en la primera, RESULTADOS GENERALES, se detallan los resultados mejor y peor alcanzados, tanto absolutos como en promedios, así como los parámetros para los que se han obtenido. Mostramos un ejemplo perteneciente a la función de JONG en dimensión 3, en la figura 4.4. Con la letra D denotamos los valores de las desviaciones típicas. También detallamos la diferencia entre el mejor y el peor óptimos recogidos, lo que corresponde al rango de la muestra obtenida.

	A	B	C	D	E	F	G	H	I	J	K	L
1												
2			FUNCION DE JONG 3									
3							RESULTADOS GENERALES					
4												
5												
6		E.F.O.										
7		700000										
8		1400000										
9		2100000		MEJOR OPTIMO	1.3354E-14	ALFA=0.005 , BETA=0.4 , G=1E+05 , P.E.=3.000 , E.F.O.=2.100.000						
10		2800000		MEJOR PROMEDIO	7.9631E-13	ALFA=0.005 , BETA=0.4 , G=1E+05 , P.E.M.=10.000 , E.F.O.=7.000.000 D=1.161103E-12						
11		3500000										
12		4200000		PEOR OPTIMO	0.1405761	ALFA=0.005 , BETA=0.4 , G=1E+03 , P.E.=1.000 , E.F.O.=700.000						
13		4900000		PEOR PROMEDIO	0.04649993	ALFA=0.005 , BETA=0.4 , G=1E+03 , P.E.M.=1.000 , E.F.O.=700.000 D=0.054032						
14		5600000										
15		6300000										
16		7000000		FMAX-FMIN	0.1405761							
17												

Figura 4.4. Ejemplo de hoja Excel de resultados generales para la función de JONG 3.

En la segunda hoja de Excel, GAMMAS GENERALES, recabamos los 27 valores obtenidos, uno para cada terna de los parámetros, para las correspondientes instancias de $\gamma_{\alpha\beta G}$ y $\gamma p_{\alpha\beta G}$. También introducimos un gráfico de barras realizado con los valores de $\gamma_{\alpha\beta G}$ con el fin de analizar los resultados obtenidos. En la figura 4.5 mostramos un ejemplo perteneciente a la función de JONG en dimensión 3.

TERNA		GAMMAS ABSOLUTOS	GAMMAS PROMEDIOS
1	A_B_G-	0.011	0.018
2	A_B_G	0.003	0.019
3	A_B_G+	0.000	0.000
4	A_B-_G-	0.014	0.331
5	A_B-_G	0.002	0.013
6	A_B-_G+	0.000	0.000
7	A_B+_G-	0.003	0.011
8	A_B+_G	0.000	0.013
9	A_B+_G+	0.003	0.013
10	A+_B_G-	0.010	0.023
11	A+_B_G	0.001	0.006
12	A+_B_G+	0.001	0.003
13	A+_B-_G-	0.003	0.013
14	A+_B-_G	0.005	0.016
15	A+_B-_G+	0.099	0.221
16	A+_B+_G-	0.001	0.008
17	A+_B+_G	0.009	0.017
18	A+_B+_G+	0.002	0.007
19	A++_B_G-	0.013	0.022
20	A++_B_G	0.008	0.012
21	A++_B_G+	0.000	0.003
22	A++_B-_G-	0.003	0.019
23	A++_B-_G	0.009	0.017
24	A++_B-_G+	0.004	0.011
25	A++_B+_G-	0.000	0.013
26	A++_B+_G	0.008	0.018
27	A++_B+_G+	0.003	0.012

GAMMA ABSOLUTO MAX.	0.099
GAMMA PROM. MAX.	0.331

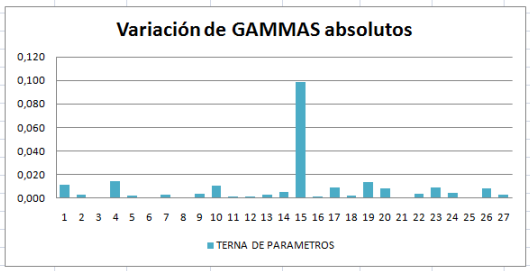


Figura 4.5. Ejemplo de hoja Excel de gammas generales para la función de JONG .

A continuación exponemos los resultados obtenidos para cada función y alguna de las hojas de Excel realizadas. Todos los archivos de Excel pueden encontrarse en el CD anexo a este trabajo.

4. RESULTADOS

4.1. FUNCIÓN DE JONG EN DIMENSIÓN 3

Para poner los ejemplos sobre el desarrollo de las ejecuciones hemos usado como figuras los resultados de esta instancia, por ello hemos preferido no repetirlas.

Como se ve en la hoja de resultados generales de la figura 4.4, se ha alcanzado el óptimo de forma muy satisfactoria, lo que pone de manifiesto que para funciones con esta topología y dimensión el algoritmo responde perfectamente.

Las desviaciones típicas han sido en los 27 casos menores que 0.06, siendo más pequeñas en los mejores óptimos. Concluimos que, cuando los parámetros son los adecuados, el algoritmo presenta resultados obviamente mejores pero también más próximos entre sí. Así, podemos usar las desviaciones típicas como parámetro de control para saber si estamos o no cerca del óptimo.

Además los valores del parámetro $\gamma_{\alpha\beta G}$ se han mantenido en todos los casos menores que 10^{-1} y los valores de $\gamma p_{\alpha\beta G}$ sólo han superado una décima en el caso $\alpha = 0.5$, $\beta = 0.4$, $G = 10^5$ en el que ha alcanzado el valor 0.22.

Podemos deducir la robustez del algoritmo para esta instancia y para cualquier terna de parámetros.

Respecto al punto máximo de estabilidad, en la mayor parte de los casos ronda el valor 10000, salvo en un caso que coincide con la excepción anterior y es 1000. Este hecho nos hace suponer que los asteroides no se quedan atrapados sino que exploran todo el espacio y llegan sin problemas al óptimo global. No obstante, aumentar *n_iter* no ha sido necesario, dado que hemos encontrado el óptimo con errores menores que 10^{-13}.

En definitiva, creemos que en este caso S.G.O. ha superado la prueba de robustez y eficacia satisfactoriamente con cualquier elección de las efectuadas.

4.2. FUNCIÓN DE PERM EN DIMENSIÓN 4

A la vista de la hoja de RESULTADOS PERM, figura 4.6, podemos observar que hemos obtenido el óptimo con un error menor que $5 \cdot 10^{-2}$. La respuesta del algoritmo sigue siendo buena para esta función.

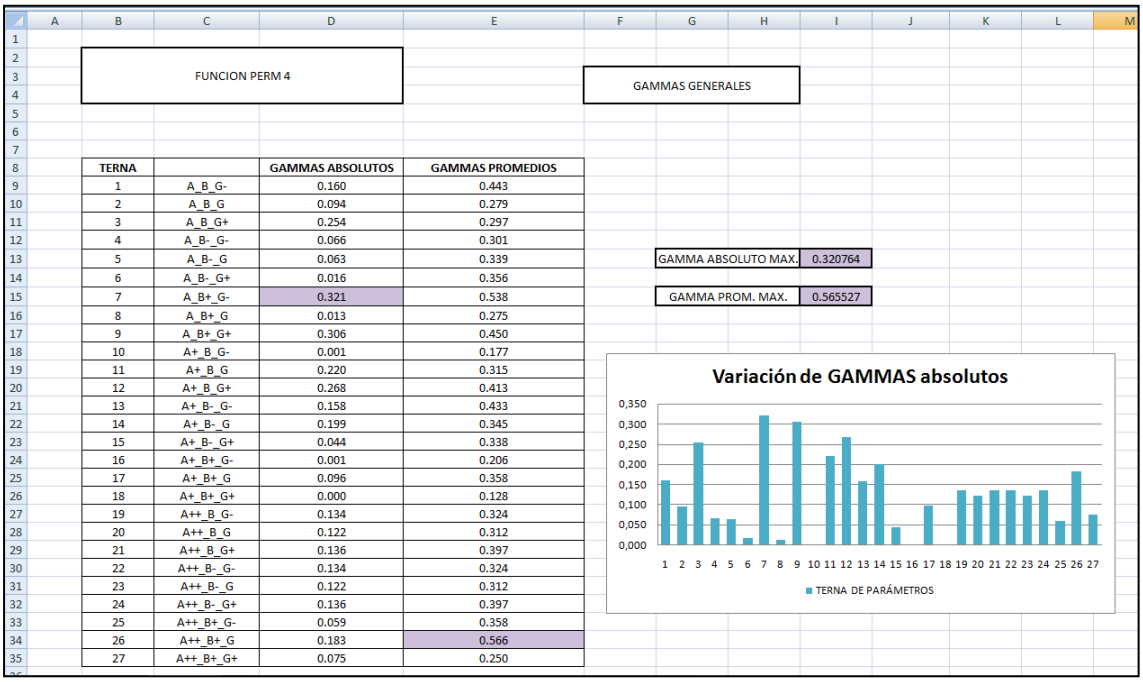
	A	B	C	D	E	F	G	H	I	J	K	L
1												
2			FUNCION PERM 4									
3							RESULTADOS GENERALES					
4												
5												
6		E.F.O.										
7		900000										
8		1800000										
9		2700000		MEJOR OPTIMO	0.043232	ALFA=0.5 , BETA=1.5 , G=1E+05 , P.E.=1.000 , E.F.O.=900.000						
10		3600000		MEJOR PROMEDIO	0.192334	ALFA=0.5 , BETA=1.5 , G=1E+05 , P.E.M.=9.000 , E.F.O.=8.100.000 D=0.227198						
11		4500000										
12		5400000		PEOR OPTIMO	1.206209	ALFA=0.005 , BETA=0.4 , G=1E+05 , P.E.=3.000 , E.F.O.=2.700.000						
13		6300000		PEOR PROMEDIO	0.700927	ALFA=1 , BETA=1.5 , G=1E+04 , P.E.M.=8.000 , E.F.O.=7.200.000 D=0.406956						
14		7200000										
15		8100000										
16		9000000		FMAX-FMIN	1.162977							
17												

Figura 4.6. Hoja Excel de resultados de la función PERM 4.

Las desviaciones típicas han permanecido en los 27 casos menores que 0.4, lo que nos indica que no ha habido resultados excesivamente dispersos.

En todos los casos el *P.E.M*. supera el valor 6000 y en la mayor parte ronda el valor 9000 con lo que podemos concluir que los agentes de búsqueda no han quedado atrapados en las primeras 10000 iteraciones.

Como se aprecia en la figura 4.7, los valores de $\gamma_{\alpha\beta G}$ han permanecido por debajo de 0.3 y los de $\gamma p_{\alpha\beta G}$ inferiores a 0.5. Dichos valores avalan la robustez de S.G.O. para esta instancia en todas las ternas de parámetros.



FUNCION PERM 4

GAMMAS GENERALES

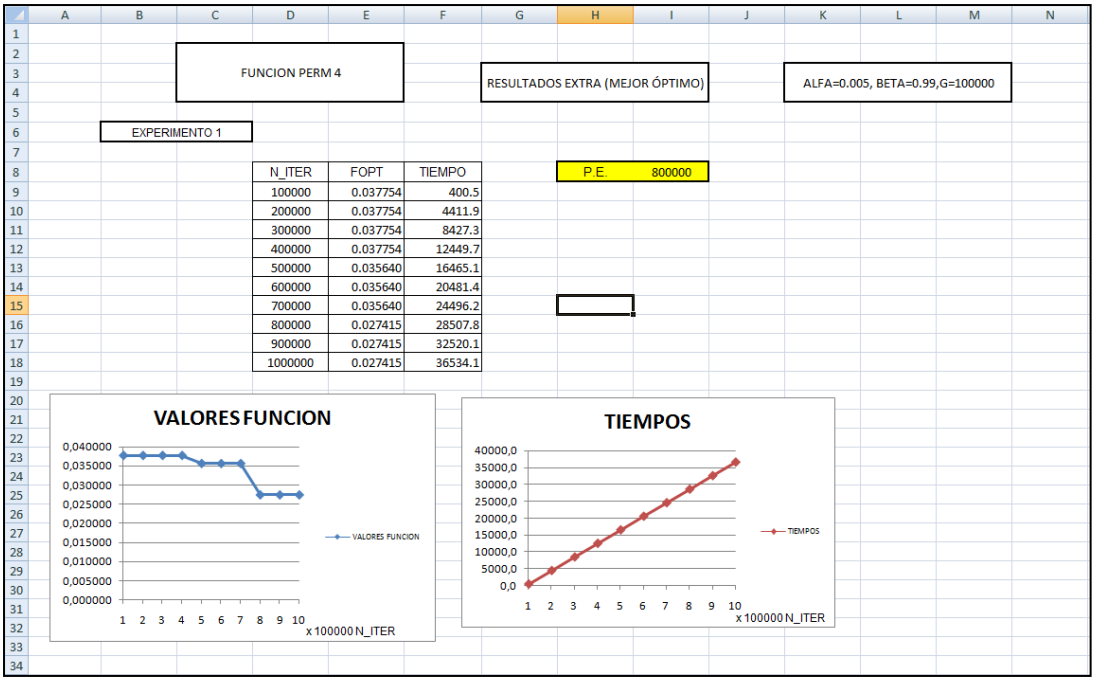
TERNA		GAMMAS ABSOLUTOS	GAMMAS PROMEDIOS
1	A_B_G-	0.160	0.443
2	A_B_G	0.094	0.279
3	A_B_G+	0.254	0.297
4	A_B-_G-	0.066	0.301
5	A_B-_G	0.063	0.339
6	A_B-_G+	0.016	0.356
7	A_B+_G-	0.321	0.538
8	A_B+_G	0.013	0.275
9	A_B+_G+	0.306	0.450
10	A+_B_G-	0.001	0.177
11	A+_B_G	0.220	0.315
12	A+_B_G+	0.268	0.413
13	A+_B-_G-	0.158	0.433
14	A+_B-_G	0.199	0.345
15	A+_B-_G+	0.044	0.338
16	A+_B+_G-	0.001	0.206
17	A+_B+_G	0.096	0.358
18	A+_B+_G+	0.000	0.128
19	A++_B_G-	0.134	0.324
20	A++_B_G	0.122	0.312
21	A++_B_G+	0.136	0.397
22	A++_B-_G-	0.134	0.324
23	A++_B-_G	0.122	0.312
24	A++_B-_G+	0.136	0.397
25	A++_B+_G-	0.059	0.358
26	A++_B+_G	0.183	0.566
27	A++_B+_G+	0.075	0.250

GAMMA ABSOLUTO MAX. 0.320764

GAMMA PROM. MAX. 0.565527

Figura 4.7. Hoja Excel de gammas generales para la función PERM 4.

Dado que los resultados no han sido tan espectaculares como en el caso anterior, nos hemos planteado incrementar *n_iter* en los dos casos en los que hemos encontrado el mejor óptimo y el mejor promedio, para ver hasta qué punto podemos mejorar los resultados obtenidos dejando más movimientos a los agentes de búsqueda. Los dos experimentos realizados, consistentes en incrementar *n_iter* hasta 10^6 tomando valores parciales cada 10^5 iteraciones, se recogen en el archivo PERM_R_EXTRA, figura 4.8. El mejor óptimo obtenido es 0.027 por lo que entendemos que el esfuerzo que supone aumentar *n_iter* no compensa los resultados obtenidos.



FUNCION PERM 4

RESULTADOS EXTRA (MEJOR ÓPTIMO)

ALFA=0.005, BETA=0.99,G=100000

EXPERIMENTO 1

N_ITER	FOPT	TIEMPO
100000	0.037754	400.5
200000	0.037754	4411.9
300000	0.037754	8427.3
400000	0.037754	12449.7
500000	0.035640	16465.1
600000	0.035640	20481.4
700000	0.035640	24496.2
800000	0.027415	28507.8
900000	0.027415	32520.1
1000000	0.027415	36534.1

P.E. 800000

Figura 4.8. Hoja Excel de resultados PERM_RESULTADOS EXTRA.

4.3. FUNCIÓN DE SCHWEFEL EN DIMENSIÓN 6

Ya en pruebas anteriores con esta función en dimensión 2 hemos podido comprobar que en ocasiones encuentra el mejor óptimo por azar. Lógicamente, aumentar la dimensión agrava el problema lo que puede observarse en la figura 4.9. El mejor óptimo es 0.00086 (¡excelente!), sin embargo el mejor promedio es 45.98 lo que da una información más fiable sobre el funcionamiento en este caso de S.G.O.

	A	B	C	D	E	F	G	H	I	J	K	L
1												
2				FUNCION SCHWEFEL 6								
3							RESULTADOS GENERALES					
4												
5												
6		E.F.O.										
7		1300000										
8		2600000										
9		3900000		MEJOR OPTIMO	0.000862	ALFA=0.005 , BETA=0.99 , G=1E+04 , P.E.=2.000 , E.F.O.=2.600.000						
10		5200000		MEJOR PROMEDIO	45.984527	ALFA=1 , BETA=0.99 , G=1E+04 , P.E.M.=9.000 , E.F.O.=11.700.000 D=35.039808						
11		6500000										
12		7800000		PEOR OPTIMO	909.048473	ALFA=0.5 , BETA=0.4 , G=1E+03 , P.E.=9.000 , E.F.O.=11.700.000						
13		9100000		PEOR PROMEDIO	707.559568	ALFA=1, BETA=0.4 , G=1E+03 , P.E.M.=10.000 , E.F.O.=13.000.000 D=62.579256						
14		10400000										
15		11700000										
16		13000000		FMAX-FMIN	909.047611							
17												

Figura 4.9. Hoja Excel de resultados de la función de SCHWEFEL 6.

La diferencia entre el mejor y el peor óptimos es de 909.05. Debemos concluir que el "excelente resultado" se debe al azar, aunque hemos obtenido valores similares en más de una ocasión.

Las desviaciones típicas, altas incluso en el caso del mejor promedio, 35, nos indican que el comportamiento de S.G.O. no es completamente estable en esta ocasión.

El *P.E.*M., toma todos los valores posibles entre 1000 y 10000 en algún caso, lo que nos indica que los asteroides han quedado atrapados, sin embargo en el 50% de las 27 mediciones se ha superado el valor 8000 por lo que entendemos que el algoritmo se ha defendido razonablemente y ha mejorado los resultados hasta el final.

Según vemos en la figura 4.10, el parámetro $\gamma_{\alpha\beta G}$ ha superado el valor 0.4 solo en el 20% de las ocasiones aunque en un caso ha llegado a 0.66. El parámetro $\gamma p_{\alpha\beta G}$ ha superado el valor 0.4 en el 40% de los casos y ha llegado al valor 0.77. Es innegable que con esta función y en dimensión alta el algoritmo ha perdido robustez, pero creemos que, en general, sigue mostrándose suficientemente robusto.

FUNCION SCHWEFEL 6

GAMMAS GENERALES

TERNA		GAMMAS ABSOLUTOS	GAMMAS PROMEDIOS
1	A_B_G-	0.369	0.448
2	A_B_G	0.000	0.326
3	A_B_G+	0.261	0.452
4	A_B-_G-	0.130	0.282
5	A_B-_G	0.130	0.403
6	A_B-_G+	0.130	0.317
7	A_B+_G-	0.433	0.487
8	A_B+_G	0.334	0.447
9	A_B+_G+	0.087	0.156
10	A+_B_G-	0.424	0.544
11	A+_B_G	0.025	0.111
12	A+_B_G+	0.131	0.367
13	A+_B-_G-	0.545	0.719
14	A+_B-_G	0.274	0.336
15	A+_B-_G+	0.298	0.403
16	A+_B+_G-	0.268	0.482
17	A+_B+_G	0.342	0.385
18	A+_B+_G+	0.123	0.170
19	A++_B_G-	0.505	0.601
20	A++_B_G	0.012	0.051
21	A++_B_G+	0.371	0.424
22	A++_B-_G-	0.659	0.778
23	A++_B-_G	0.369	0.496
24	A++_B-_G+	0.001	0.383
25	A++_B+_G-	0.403	0.486
26	A++_B+_G	0.127	0.422
27	A++_B+_G+	0.092	0.132

GAMMA ABSOLUTO MAX.	0.659
GAMMA PROM. MAX.	0.778

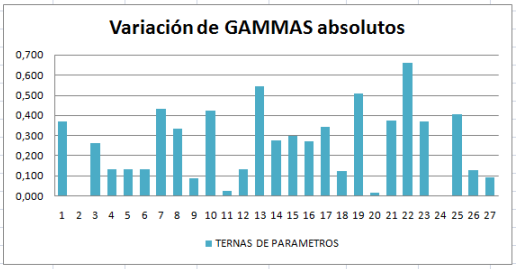
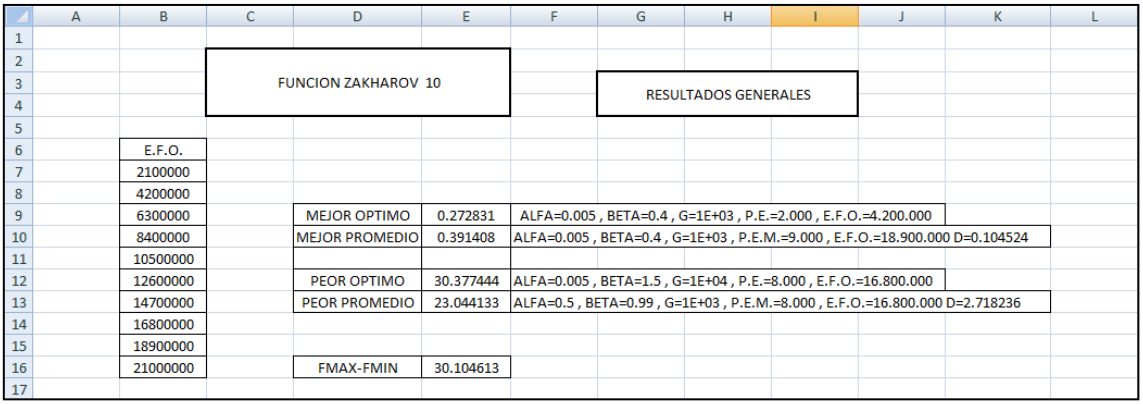


Figura 4.10. Hoja Excel de gammas generales para la función de SCHWEFEL 6.

Su particular topología de “montaña rusa” que va aumentando su gradiente a medida que se acerca a la frontera hace de esta función un difícil objetivo para S.G.O. En posteriores capítulos veremos que hibridando S.G.O. con otros algoritmos seremos capaces de encontrar el óptimo, incluso en mayores dimensiones, de forma mucho más satisfactoria.

4.4. FUNCIÓN DE ZAKHAROV EN DIMENSIÓN 10

Trabajar en dimensión 10 y superiores ya es un reto para cualquier heurístico y también lo es para nuestro algoritmo, en cuanto una de las 10 componentes de la abscisa se dispersa un poco, el valor de la función se aleja mucho del óptimo. Por eso, como se observa en la figura 4.11, obtener 0.27 como mejor óptimo encontrado y 0.39 como promedio de mejores alcanzado, nos parece un resultado bastante aceptable. Veámoslo con mayor detalle.



	A	B	C	D	E	F	G	H	I	J	K	L
1												
2				FUNCION ZAKHAROV 10								
3							RESULTADOS GENERALES					
4												
5												
6		E.F.O.										
7		2100000										
8		4200000										
9		6300000		MEJOR OPTIMO	0.272831	ALFA=0.005 , BETA=0.4 , G=1E+03 , P.E.=2.000 , E.F.O.=4.200.000						
10		8400000		MEJOR PROMEDIO	0.391408	ALFA=0.005 , BETA=0.4 , G=1E+03 , P.E.M.=9.000 , E.F.O.=18.900.000 D=0.104524						
11		10500000										
12		12600000		PEOR OPTIMO	30.377444	ALFA=0.005 , BETA=1.5 , G=1E+04 , P.E.=8.000 , E.F.O.=16.800.000						
13		14700000		PEOR PROMEDIO	23.044133	ALFA=0.5 , BETA=0.99 , G=1E+03 , P.E.M.=8.000 , E.F.O.=16.800.000 D=2.718236						
14		16800000										
15		18900000										
16		21000000		FMAX-FMIN	30.104613							
17												

Figura 4.11. Hoja Excel de resultados de la función de ZAKHAROV 10.

Los valores de las desviaciones típicas han permanecido menores que 4 en el 63% de los casos y en el mejor promedio ha sido de una décima. Como en casos anteriores, cuando S.G.O. encuentra valores buenos, encuentra muchos valores buenos.

Los valores de *P.E.M.* han sido mayores o iguales a 8000 en el 88% de los casos. Los agentes de búsqueda han explorado el espacio de soluciones sin quedar atrapados.

En la figura 4.12 observamos que $\gamma_{\alpha\beta G}$ ha tomado valores menores que 0.5 en el 85% de los casos. Su mayor valor ha sido 0.6 solo alcanzado en un caso $\alpha = 0.5$, $\beta = 0.99$, $G = 10^3$. Los valores de $\gamma p_{\alpha\beta G}$ menores o iguales que 0.5 abarcan el 37%, el mayor valor es 0.76 y se alcanza en el mismo caso que anteriormente. A nuestro juicio estos datos avalan sin duda la robustez del algoritmo para las ternas observadas.

FUNCION ZAKHAROV 10

GAMMAS GENERALES

TERNA		GAMMAS ABSOLUTOS	GAMMAS PROMEDIOS
1	A_B_G-	0.338	0.644
2	A_B_G	0.444	0.645
3	A_B_G+	0.123	0.368
4	A_B-_G-	0.000	0.004
5	A_B-_G	0.476	0.666
6	A_B-_G+	0.078	0.199
7	A_B+_G-	0.441	0.621
8	A_B+_G	0.438	0.695
9	A_B+_G+	0.397	0.585
10	A+_B_G-	0.607	0.756
11	A+_B_G	0.320	0.577
12	A+_B_G+	0.102	0.403
13	A+_B-_G-	0.476	0.639
14	A+_B-_G	0.485	0.613
15	A+_B-_G+	0.231	0.258
16	A+_B+_G-	0.320	0.577
17	A+_B+_G	0.453	0.642
18	A+_B+_G+	0.443	0.609
19	A++_B_G-	0.556	0.690
20	A++_B_G	0.441	0.613
21	A++_B_G+	0.106	0.364
22	A++_B-_G-	0.558	0.707
23	A++_B-_G	0.485	0.572
24	A++_B-_G+	0.098	0.255
25	A++_B+_G-	0.581	0.656
26	A++_B+_G	0.320	0.608
27	A++_B+_G+	0.264	0.541

GAMMA ABSOLUTO MAX.	0.607
GAMMA PROM. MAX.	0.756

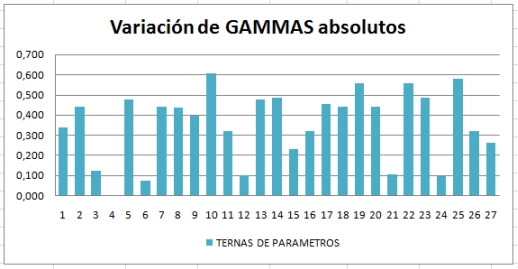


Figura 4.12. Hoja Excel de gammas generales para la función de ZAKHAROV 10.

Podemos concluir que S.G.O también ha superado la prueba de eficacia y robustez con esta instancia.

4.5. FUNCIÓN DE ROSENBROCK EN DIMENSIÓN 20

En la figura 4.13 observamos que el algoritmo no ha conseguido encontrar el óptimo, quizá porque los asteroides no han desarrollado la velocidad suficiente. Además, hemos obtenido varias ejecuciones con datos repetidos que atribuimos a un fallo en la implementación de la aleatoriedad del algoritmo, en realidad pseudoaleatoriedad, en la elección de las posiciones y velocidades iniciales o tras la salida del asteroide de la zona de factibilidad. Este hecho solo ha sucedido con esta función y esta dimensión y no podemos atribuirlo solo al tamaño de la dimensión porque, como veremos en el apartado 3.6, no ha sucedido lo mismo en el caso de dimensión 30.

Analicemos el resto de los datos.

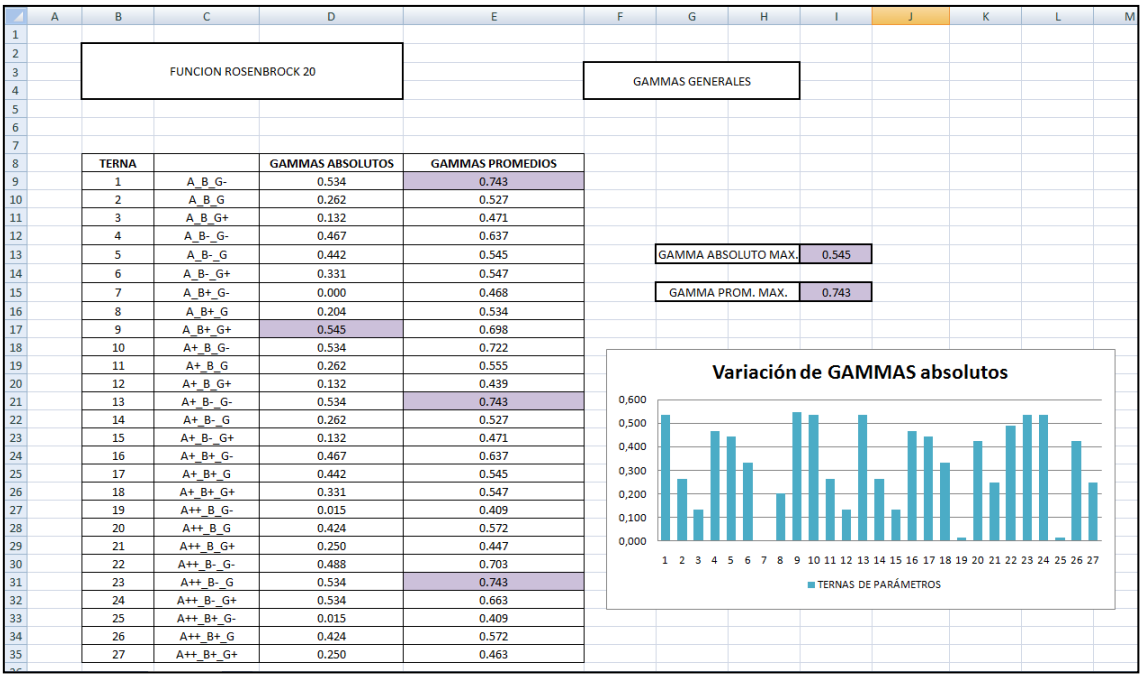
	A	B	C	D	E	F	G	H	I	J	K	L
1												
2												
3				FUNCION ROSENBROCK 20			RESULTADOS GENERALES					
4												
5												
6		E.F.O.										
7		4100000										
8		8200000										
9		12300000		MEJOR OPTIMO	69653.4225	ALFA=0.005 , BETA=1.5 , G=1.000 , P.E.=10.000 , E.F.O.=41.000.000						
10		16400000		MEJOR PROMEDIO	160832.924	ALFA=1 , BETA=0.99 , G=1.000 , P.E.M.=9.000 , E.F.O.=11.700.000 D=66079.54						
11		20500000										
12		24600000		PEOR OPTIMO	292657.815	ALFA=0.5 , BETA=0.4 , G=1.000 , P.E.=1.000 , E.F.O.=4.100.000						
13		28700000		PEOR PROMEDIO	235332.339	ALFA=1, BETA=0.4 , G=10.000 , P.E.M.=10.000 , E.F.O.=41.000.000 D=49746.41496						
14		32800000										
15		36900000										
16		41000000		FMAX-FMIN	223004.393							
17												

Figura 4.13. Hoja Excel de resultados de la función de de ROSENBROCK 20.

Las desviaciones típicas son mucho más altas que en el resto de los casos analizados, lo que corrobora el mal funcionamiento del algoritmo para esta instancia.

Sólo en el caso $\alpha = 0.005$, $\beta = 1.5$, $G = 10^4$ el *P.E.M.* ha sido inferior a 8000, pensamos que los asteroides han estado todo el tiempo explorando el espacio de soluciones pero no han tenido la suficiente velocidad para llegar al óptimo razonablemente.

Sin embargo, los valores de $\gamma_{\alpha\beta G}$ se mantienen menores o iguales a 0.53 en los 27 casos y los de $\gamma p_{\alpha\beta G}$ en el 50% de ellos, según se observa en la figura 4.14. El algoritmo mantiene su robustez aunque en esta ocasión haya quedado muy lejos del óptimo global.



FUNCION ROSENBROCK 20

GAMMAS GENERALES

TERNA		GAMMAS ABSOLUTOS	GAMMAS PROMEDIOS
1	A_B_G-	0.534	0.743
2	A_B_G	0.262	0.527
3	A_B_G+	0.132	0.471
4	A_B-_G-	0.467	0.637
5	A_B-_G	0.442	0.545
6	A_B-_G+	0.331	0.547
7	A_B+_G-	0.000	0.468
8	A_B+_G	0.204	0.534
9	A_B+_G+	0.545	0.698
10	A+_B_G-	0.534	0.722
11	A+_B_G	0.262	0.555
12	A+_B_G+	0.132	0.439
13	A+_B-_G-	0.534	0.743
14	A+_B-_G	0.262	0.527
15	A+_B-_G+	0.132	0.471
16	A+_B+_G-	0.467	0.637
17	A+_B+_G	0.442	0.545
18	A+_B+_G+	0.331	0.547
19	A++_B_G-	0.015	0.409
20	A++_B_G	0.424	0.572
21	A++_B_G+	0.250	0.447
22	A++_B-_G-	0.488	0.703
23	A++_B-_G	0.534	0.743
24	A++_B-_G+	0.534	0.663
25	A++_B+_G-	0.015	0.409
26	A++_B+_G	0.424	0.572
27	A++_B+_G+	0.250	0.463

GAMMA ABSOLUTO MAX.	0.545
GAMMA PROM. MAX.	0.743

Figura 4.14. Hoja Excel de gammas generales para la función de ROSENBROCK 20.

El mal comportamiento, inusual por otra parte, de S.G.O. con esta instancia solo puede atribuirse a su peculiar topología que, al tener gradientes muy bajos cerca del óptimo, hace que se ralentice en gran medida el movimiento de los agentes de búsqueda. Intuimos que los cálculos para esta función, también en menores dimensiones, provocan más errores que en el resto de las funciones analizadas.

En posteriores capítulos veremos que las hibridaciones de S.G.O. sí han sido capaces de encontrar el óptimo sin problemas.

4.6. FUNCIÓN DE JONG EN DIMENSIÓN 30

Es indudable que la eficacia de S.G.O. está intrínsecamente relacionada con la topología de la función, pero también con la dimensión y el tamaño de la región factible. De hecho, hemos podido comprobarlo con las funciones de los apartados 4.3 y 4.5 ya que en el capítulo 3 tenemos los resultados obtenidos en ambos casos para dimensión 2 y son mucho mejores a los obtenidos en el presente capítulo, secciones 2.3 y 2.6 respectivamente. Nos planteamos si la pérdida de eficacia del algoritmo al aumentar la dimensión será generalizada para todas las funciones de cualquier topología o si, por el contrario, no afectará a todas las funciones por igual. Esa es la razón por la que nos hemos decantado por la función del apartado 4.1 pero en dimensión 30.

La función de JONG tiene una topología muy asequible para nuestro algoritmo y los resultados que hemos obtenido con ella han sido siempre excelentes. Veamos qué ocurre al multiplicar la dimensión del espacio de soluciones por 10. En la siguiente figura presentamos los resultados generales de JONG 30.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1													
2				FUNCION DE JONG 30			RESULTADOS GENERALES						
3													
4													
5													
6		E.F.O.											
7		6100000											
8		12200000											
9		18300000		MEJOR OPTIMO	1.6759E-14	ALFA=0.005 , BETA=0.4 , G=1E+05 , P.E.=9.000 , E.F.O.=54.900.000							
10		24400000		MEJOR PROMEDIO	1.4586E-13	ALFA=0.005 , BETA=0.4 , G=1E+05 , P.E.M.=10.000 , E.F.O.=6.000.000 D=1.647758E-13							
11		30500000											
12		36600000		PEOR OPTIMO	81.708457	ALFA=1 , BETA=0.4 , G=1E+04 , P.E.=5.000 , E.F.O.=30.500.000							
13		42700000		PEOR PROMEDIO	72.407024	ALFA=0.5, BETA=0.4 , G=1E+03 , P.E.M.=10.000 , E.F.O.=61.000.000 D=3.360183							
14		48800000											
15		54900000											
16		61000000		FMAX-FMIN	81.708457								
17													

Figura 4.15. Hoja Excel de resultados de la función de JONG 30.

Como puede observarse, los resultados en los casos del mejor óptimo y mejor promedio siguen siendo del mismo orden que los obtenidos para el caso de dimensión 3. No así el peor resultado -que se ha multiplicado por casi 600- y el peor promedio -que se ha multiplicado casi por 2000-. Ambos se registran para los valores altos del parámetro α, lo que ha sido habitual en las mediciones llevadas a cabo en el presente capítulo. Parece que considerar las interacciones entre asteroides perjudica la robustez de S.G.O.

En este caso el aumento de dimensión no ha supuesto ningún hándicap en la consecución del óptimo.

En cuanto a las desviaciones típicas, se han mantenido excelentes salvo en el peor promedio. La dimensión no ha afectado a la robustez de S.G.O.

Los *P.E.M.* son similares a los obtenidos en dimensión 3.

En la figura 4.16 podemos observar que los valores del parámetro $\gamma_{\alpha\beta G}$ en el caso de dimensión 30 han variado notablemente respecto a los correspondientes de dimensión 3. En este caso encontramos que sólo el 55% es menor de 0.5 y llega a alcanzarse el valor 0.84 en una ocasión $\alpha = 0.5$, $\beta = 0.4$, $G = 10^3$. Los valores de $\gamma p_{\alpha\beta G}$ superan el valor 0.5 en un 50% y llega a alcanzarse el valor 0.89. No obstante, los valores en las ternas 4, 5 y 6 son mínimas, es decir, para algunas ternas de parámetros el algoritmo se mantiene robusto mientras que para otras elecciones su comportamiento no es tan efectivo.

FUNCION DE JONG 30

GAMMAS GENERALES

TERNA		GAMMAS ABSOLUTOS	GAMMAS PROMEDIOS
1	A_B_G-	0.606	0.725
2	A_B_G	0.295	0.340
3	A_B_G+	0.219	0.230
4	A_B-_G-	0.030	0.032
5	A_B-_G	0.000	0.000
6	A_B-_G+	0.000	0.000
7	A_B+_G-	0.443	0.644
8	A_B+_G	0.563	0.618
9	A_B+_G+	0.234	0.257
10	A+_B_G-	0.635	0.701
11	A+_B_G	0.223	0.252
12	A+_B_G+	0.137	0.144
13	A+_B-_G-	0.837	0.886
14	A+_B-_G	0.636	0.761
15	A+_B-_G+	0.000	0.000
16	A+_B+_G-	0.745	0.784
17	A+_B+_G	0.575	0.642
18	A+_B+_G+	0.237	0.256
19	A++_B_G-	0.644	0.726
20	A++_B_G	0.366	0.407
21	A++_B_G+	0.069	0.085
22	A++_B-_G-	0.577	0.762
23	A++_B-_G	0.746	0.859
24	A++_B-_G+	0.008	0.017
25	A++_B+_G-	0.656	0.727
26	A++_B+_G	0.572	0.618
27	A++_B+_G+	0.269	0.274

GAMMA ABSOLUTO MAX.	0.837
GAMMA PROM. MAX.	0.886

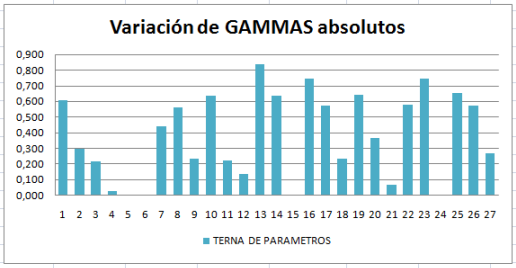


Figura 4.16. Hoja Excel de gammas generales para la función JONG 30.

En definitiva, aunque a la vista de los resultados S.G.O. sigue siendo un buen heurístico, los parámetros de control nos indican que su funcionamiento no es tan consistente al aumentar la dimensión. Lo cual es lógico, ya que aumentar el número de abscisas comporta, cada vez, mayor dificultad.

5. DETERMINACIÓN DE VALORES GENERALES PARA LOS PARÁMETROS

En la siguiente hoja Excel, figura 4.17, se muestran los parámetros, los puntos de estabilización, los promedios y las desviaciones típicas de los casos en los que se han alcanzado los mejores y peores resultados para todas las instancias. Nuestra intención es extraer de los datos un patrón fijo de "parámetros buenos" que poder usar en cualquier problema de optimización de forma universal.

FUNCIÓN	PARÁM. MEJORES TOTALES					PARÁM. MEJORES PROMEDIOS					
	OPTIMO	ALFA	BETA	G	P.E.	OPTIMO	ALFA	BETA	G	P.E.M.	D.T.
DE JOUNG(3)	1.3354E-14	0.005	0.4	1.E+05	3000	7.96306E-13	0.005	0.4	1.E+05	10000	1.16118E-12
PERM (4)	0.04323231	0.5	1.5	1.E+05	1000	0.192334052	0.5	1.5	1.E+05	9000	2.27198E-01
SCHWEFEL (6)	0.000862	0.005	0.99	1.E+04	2000	45.98452717	1	0.99	1.E+04	9000	35.03980827
ZAKHAROV(10)	0.27283053	0.005	0.4	1.E+03	2000	0.391407872	0.005	0.4	1.E+03	9000	0.10452398
ROSENBROCK(20)	69653.42	0.005	1.5	1.E+05	10000	160832.92	1	0.99	1.E+03	9000	49746.41
JOUNG30(30)	1.6759E-14	0.005	0.4	1.E+05	9000	1.4586E-13	0.005	0.4	1.E+05	10000	1.6478E-05

FUNCIÓN	PARÁM. PEORES TOTALES					PARÁM. PEORES PROMEDIOS					
	OPTIMO	ALFA	BETA	G	P.E.	OPTIMO	ALFA	BETA	G	P.E.M.	D.T.
DE JOUNG(3)	0.140576	0.005	0.4	1.E+03	1000	0.046500	0.005	0.4	1.E+03	1000	0.054032
PERM (4)	1.206209	0.005	0.4	1.E+05	3000	0.700927	1	1.5	1.E+04	8000	0.406956
SCHWEFEL (6)	909.048473	0.5	0.4	1.E+03	9000	707.5595677	1	0.4	1.E+03	10000	62.57925648
ZAKHAROV(10)	30.3774438	0.005	1.5	1.E+04	8000	23.04413298	0.5	0.99	1.E+03	8000	2.71823577
ROSENBROCK(20)	292657.82	0.5	0.4	1.E+03	1000	235332.34	1	0.4	1.E+04	10000	66079.54
JOUNG30(30)	81.7084568	1	0.4	1.E+04	5000	72.40702419	0.5	0.4	1.E+03	10000	3.360183

Figura 4.17. Hoja Excel de RESULTADOS GENERALES.

Como puede apreciarse, en algunas ocasiones, de JONG por ejemplo, se ha obtenido el mejor y el peor resultado con los mismos valores para α , β y sólo ha variado el valor de G; Además, todos los posibles valores de los parámetros aparecen tanto para los resultados mejores como para los peores. En general, los parámetros de los mejores resultados coinciden con los de los mejores promedios, pero no siempre es así, en el caso de los peores resultados y promedios las diferencias son aún mayores.

El parámetro α no toma el valor 1 en los mejores resultados -aunque sí en promedio-, pero tampoco toma dicho valor en los peores -aunque sí en promedio-. El parámetro β no toma el valor 0.99 en los resultados malos -aunque sí en promedio-, sin embargo es el menos frecuente en los resultados buenos -tanto en los mejores como en promedio-. G toma todos sus posibles valores tanto en los resultados buenos como en los malos…

No parece que pueda obtenerse una combinación infalible de valores universales para los parámetros aunque sí pueden extraerse algunas conclusiones

- $\alpha = 0.005$ es el valor más frecuente en los mejores resultados, 4 de 6, y en los mejores promedios, 3 de 6. Como ya hemos comentado, parece que S.G.O. funciona mejor si no se tienen en cuenta las interacciones entre asteroides.

- $\beta = 0.4$ es el valor más frecuente en los mejores resultados y mejores promedios, 3 de 6 en ambos casos.

- $G = 10^5$ es el valor más frecuente en los mejores resultados, 4 de 6, y en los mejores promedios, 3 de 6.

Las modas anteriores no son definitivas pero sí son estrictamente superiores.

Además, si analizamos los valores de $\gamma_{\alpha\beta G}$ para esta elección concreta de parámetros vemos que en todos los casos el comportamiento de S.G.O. ha sido robusto. Figura 4.18.

FUNCIÓN	$\gamma_{\alpha=0.005,\beta=0.4,G=10.0000}$ **(TERNA nº 6)**
JONG 3	0
PERM 4	0.016
SCHWEFEL 6	0.130
ZAKHAROV 10	0.078
ROSENBROCK 20	0.331
JONG 30	0

Figura 4.18. Tabla de valores $\gamma_{\alpha\beta G}$ para la terna elegida.

En conclusión

Una buena combinación parece ser $\alpha = 0.005$, $\beta = 0.4$, $G = 10^5$

aunque los resultados no avalan de forma rotunda esta elección.

Respecto a los puntos de estabilización, no parece fácil extraer un patrón respecto al momento en el que el algoritmo va a estar cerca del óptimo. Sí se observa que todos los promedios conllevan puntos altos de máxima estabilización, el algoritmo no ha caído por azar en el óptimo sino que lo ha ido buscando activamente. No creemos conveniente reducir *n_iter* .

Sí parece significativo el comportamiento de las desviaciones típicas, ya que en todos los casos son menores en los casos en que hemos obtenido los mejores resultados. Como ya observamos en dimensión 2, a medida que el algoritmo se acerca al óptimo los resultados obtenidos sufren menos variaciones. Puede ser útil considerarlas para saber si las mediciones tomadas con una función desconocida son aceptables o no.

6. CONCLUSIONES

Como hemos comprobado en los estudios realizados en este capítulo, el algoritmo S.G.O. se presenta como robusto y eficaz, incluso en altas dimensiones, para resolver problemas de optimización continua, pero pierde eficacia con el aumento de dimensión y además se muestra **sensible a la elección de los parámetros**. Por ello no es posible encontrar a priori valores adecuados, ni siquiera rangos adecuados, para todas las instancias.

Además, hemos observado que, dada una terna de parámetros concreta, el algoritmo alcanza su mejor resultado con 10000 iteraciones y aumentar dicho número no mejora sensiblemente la solución obtenida, lo que nos lleva a pensar que en algunas ocasiones "**necesita una ayuda"** para obtener resultados más satisfactorios.

En algunas de las ternas de parámetros estudiadas con instancias de dimensiones altas, los parámetros de control, γ , han sido superiores a 0.5, lo que indica que S.G.O. no tendrá un buen comportamiento con esas elecciones. Sin embargo, con la terna elegida, $\alpha = 0.005$, $\beta = 0.4$, $G = 10^5$, el algoritmo se ha mostrado robusto y consistente incluso en dimensiones elevadas. Pensamos que, con estos valores podremos obtener muy buenos resultados.

Los puntos máximos de estabilización de las distintas ejecuciones se han producido en un alto porcentaje en torno a 9000, lo que indica que los agentes de búsqueda no han quedado atrapados sino que han permanecido activos buscando el óptimo global.

Por sí solo S.G.O. **puede resolver una amplia gama de problemas de optimización satisfactoriamente** pero, a medida que incrementamos la dimensión de las funciones es necesario llevar a cabo de forma previa y pormenorizada **un estudio de los parámetros adecuados,** lo que incrementa el tiempo y obviamente rebate en gran medida su mayor ventaja que es su eficiencia.

Creemos que estos problemas pueden solventarse hibridando S.G.O. con algún otro algoritmo local, heurístico o no, que ayude a los asteroides a llegar al óptimo y, de esta forma, generar un metaheurístico que en tiempo razonable y sin costosos estudios previos sobre los parámetros consiga alcanzar el óptimo global.

Este es nuestro objetivo para el siguiente capítulo.

"Cuando estás pateando bien, la única pregunta debe ser por qué parte del hoyo va a entrar la bola , no si va a entrar."

Jack Nicklaus (1940-)

CAPÍTULO 5

HIBRIDACIÓN CON ALGORITMOS LOCALES

1. INTRODUCCIÓN

En el capítulo anterior hemos visto que S.G.O. es extremadamente dependiente de los parámetros y exige un estudio previo que resulta costoso en términos de tiempo. Además, a medida que aumentamos la dimensión en algunas instancias, parece que el algoritmo se queda atrapado "en el green" del campo gravitatorio definido por la función objetivo y necesita un empuje que le ayude a llegar "al hoyo".

En el presente capítulo vamos a intentar resolver estas deficiencias hibridándolo separadamente con dos algoritmos de búsqueda local: el heurístico Nelder-Mead y el algoritmo Gradiente. Ya expusimos ambos con detalle en las secciones 6.2 y 6.3 del capítulo 2.

Las razones de tal elección son las siguientes:

- Tanto Nelder-Mead como Gradiente son algoritmos rápidos y el número de llamadas a la función objetivo es considerablemente inferior al de S.G.O., con lo que los tiempos de ejecución de los métodos simple e hibridado van a ser muy similares.

- Nelder-Mead es un heurístico de búsqueda local de probada eficacia y amplia relevancia en la literatura de optimización (Lagarias J. y otros 1998).

- Gradiente es un algoritmo de búsqueda local de rápida convergencia y uso generalizado en el campo del Análisis numérico (Burden-Faires 1985).

Ambos algoritmos están diseñados para localizar óptimos locales, sin embargo existen diferencias entre ellos:

- Gradiente solo parte de un punto y lo guía hasta su mínimo valor posible. La única información que utiliza es el valor de la función objetivo en un entorno del punto. En este sentido es más directo que Nelder-Mead. Sin embargo, tiene mucha facilidad de caer en óptimos locales.

- Nelder-Mead parte de un *dim+1* simplex, por tanto la información que usa se basa en el valor de la función objetivo en un conjunto de *dim+1* puntos y además tiene cierta componente aleatoria definida a través de sus parámetros. Es menos directo que Gradiente y tiene mayores posibilidades que él de escapar de un óptimo local.

- Nelder-Mead solo precisa la continuidad[1] de la función para ser efectivo, mientras que Gradiente necesita que la función tenga derivadas parciales continuas[2]. El

[1] Continuidad en el dominio en que estemos implementándolo.
[2] Es decir, que sea C^1 en el dominio en que estemos implementándolo, aunque puede no serlo en el mínimo.

campo de acción de Nelder-Mead es, por tanto, mayor que el del algoritmo Gradiente.

Por ello, hemos decidido realizar ambas hibridaciones, aunque no de forma repetida, es decir, comenzaremos probando la eficacia de la hibridación S.G.O.-Nelder-Mead y en los casos en que ésta no satisfaga nuestras expectativas, probaremos la hibridación de S.G.O. con Gradiente.

Los criterios de parada para implementar los algoritmos locales serán: realizar un número fijo de iteraciones, *valnel* en Nelder-Mead y un número máximo de iteraciones, *nvecesgrad*, en gradiente, subordinado a que el algoritmo acabe en el momento en que no haya realizado ninguna mejora. Por ello, las evaluaciones de la función objetivo ya no pueden calcularse a priori y variarán para cada instancia.

- Probaremos los heurísticos con todas las instancias presentadas en el artículo "Experimental testing of Advanced Scatter Search Designs for Global Optimization of Multimodal Functions" (Laguna y Martí 2005). Ya que forman un amplia gama de 40 *benchmark functions* de topología muy diversa y dimensiones que varían entre 2 y 30 y además, constituyen los *problemas-modelo* mayoritariamente elegidos para probar la eficacia de los algoritmos de optimización continua. Entendemos que es un perfecto control de calidad para nuestras hibridaciones. Pueden encontrarse todas ellas en la siguiente página web:
  http://wwwoptima.amp.i.kyotou.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page364.htm

Tras las consideraciones realizadas en el capítulo precedente fijaremos para los parámetros los valores $\alpha = 0.005$, $\beta = 0.4$, $G = 10^5$ y **no los modificaremos en ningún caso.**

Mantendremos el valor $r_d = 0.000001$ tal y como hemos hecho en todas las experimentaciones.

Sí modificaremos los valores de n y n_iter en orden a mejorar la eficacia de los métodos cuando sea necesario y lo detallaremos en cada caso.

2. EL METAHEURÍSTICO S.G.O.- NELDER – MEAD

El método S.G.O.-Nelder-Mead consta de dos partes bien diferenciadas: en la primera fase actúa S.G.O. ejecutándose *dim+1* veces para así generar el *dim+1-simplex* que necesita Nelder-Mead como valor de entrada. A partir de ahí entra Nelder-Mead ejecutándose un número fijo de iteraciones determinado por el criterio de parada, *valnel*, y da como valor de salida el mejor óptimo encontrado y la abscisa donde se ha alcanzado.

Hemos preferido tener el mejor de dim+1 ejecuciones que tener los dim+1 mejores de solo una ejecución de S.G.O. porque en nuestra implementación previa de S.G.O. solo conservábamos el mejor valor obtenido y los tiempos totales de ejecución no han supuesto un problema ni siquiera en dimensiones altas. Llegado el caso podríamos plantearnos la opción de ejecutar S.G.O. una vez y conservar los *dim+1* mejores, lo que aumentaría en gran medida la eficiencia del método. No disponemos de datos empíricos que muestren cómo repercutiría tal decisión en su eficacia.

Ya que nuestra decisión de ejecutar S.G.O. *dim+1* veces incrementa el tiempo del metaheurístico, rebajaremos el valor de *n_iter,* respecto al usado en ocasiones precedentes, de *n_iter*=10^4 a *n_iter*=10 o 10^2, salvo en alguna excepción que comentaremos en su momento, manteniendo el número de asteroides lanzados en $n = 100$.

Si la región factible es muy grande, como sucede por ejemplo en el caso de la función SCHWEFEL[3], aumentaremos si es necesario el valor de *n* para poder explorar todo el espacio con pocos movimientos de los asteroides. Lo detallaremos en cada caso.

Como ya hemos comentado, elegiremos en todos los casos los valores $\alpha = 0.005$, $\beta = 0.4$, $G = 10^5$.

Aunque el algoritmo Nelder-Mead tiene sus propios parámetros no haremos un estudio previo de su variación, sino que nos limitaremos a elegir y fijar los valores tradicionalmente aceptados como los mejores (Lagarias 1998).

Realizaremos con cada instancia 3 tandas de 10 ejecuciones cada una, en las que solo variaremos *valnel*, generalmente en potencias consecutivas de 10. En dimensión 2 serán 10, 10^2 y 10^3 respectivamente. En mayores dimensiones modificaremos el rango de potencias, si es necesario, para obtener mejores resultados. Explicitaremos los datos en cada caso.

Hemos realizado las tres tandas para tener una primera idea del comportamiento de Nelder-Mead. Las diferencias entre los tiempos de ejecución respecto a la variación de *valnel* son tan pequeñas que no creemos que sea significativas.

---

[3] $x_i \in \prod_{i=1}^{\dim} [-500, 500]$

Hemos modificado las instancias elegidas (Laguna y Martí 2005) para que el óptimo[4] sea 0, con el fin de calcular más fácilmente los errores.

Todas las funciones están recogidas en el apéndice 1. Nuestro objetivo en esta ocasión será comprobar la eficacia de las hibridaciones planteadas al intentar optimizarlas.

En algunos casos de funciones cuyo óptimo no es un número entero, disponemos de aproximaciones redondeadas al óptimo[5] con menor número de dígitos significativos que el que nosotros utilizamos y que, por ello, nos han dificultado la labor de control de los errores, no obstante entendemos necesario fijar un criterio de alcance del óptimo que será el siguiente:

Consideraremos que hemos alcanzado el óptimo si el error de alguno de los resultados obtenidos es menor que $3 \cdot 10^{-6}$ y el error de alguno de los promedios es menor que $3 \cdot 10^{-3}$.

Puesto que vamos a realizar tandas de solo 10 ejecuciones, el objetivo de promedios implica que, en cuanto uno de los diez valores falle, los promedios pueden aumentar considerablemente[6]. Por ello estableceremos un segundo criterio que cumplir en el caso en que no se verifique el error de los promedios en el criterio anterior:

Consideraremos que hemos alcanzado el óptimo si el error de alguno de los resultados obtenidos es menor que $3 \cdot 10^{-6}$ y en alguna de las tandas de 10 ejecuciones hemos obtenido errores menores que $3 \cdot 10^{-6}$ en, al menos, 7 de ellas.

Hemos resaltado en amarillo los resultados en los que hemos alcanzado el óptimo según nuestro criterio en cada hoja Excel de resultados.

[4] Restando o sumando el valor del óptimo.
[5] He aquí la razón de los resultados negativos obtenidos.
[6] Ver por ejemplo la función de EASOM.

3. RESULTADOS DE S.G.O.-NELDER-MEAD

Muchas de las funciones de este apartado ya han sido abordadas con éxito usando solo S.G.O. pero con otros valores diferentes para los parámetros, elegidos específicamente para cada función.

Veremos cómo responde el método hibridado en las nuevas condiciones.

Todos los resultados obtenidos para las 40 funciones se han recogido en archivos de Excel que constan de 4 hojas: 3 de ellas con los resultados obtenidos en cada una de las 3 tandas de 10 ejecuciones con los correspondientes valores de *valnel* y *n_iter* elegidos, figura 5.1, y una última con los resultados generales obtenidos para cada instancia, figura 5.2.

En las hojas de cada ejecución hemos detallado los valores del óptimo logado por S.G.O., FOPT-PARC, y el óptimo logrado por el hibridado, FOPT, también hemos recogido los tiempos de ejecución de S.G.O. -TIEMP-GRAVI-, los de Nelder-Mead -TIEMP-NEL- y los tiempos totales -TIEMP. TOTAL- con el fin de constatar si los tiempos totales son del mismo orden que los de S.G.O. También hemos detallado los valores de evaluaciones de la función objetivo del hibridado -E.F.O.- que, consideramos, un parámetro más válido para cuantificar la eficiencia del método.

Además, hemos considerado los promedios, el mejor óptimo y el peor óptimo, tanto de los valores intermedios como de los finales de los óptimos obtenidos. Hemos denotado FMAX-FMIN el rango de los valores de FOPT.

FUNCION B2(2)-SGO-NELME

EJECUCION 3

N_EJEC	FOPT-PARC	FOPT	TIEMP-GRAVI	TIEMP-NEL	E.F.O TOTAL	TIEMP-TOTAL
1	0.104589459	0	0.27	0.16	19994	0.43
2	0.035700408	0	0.27	0.15	19998	0.42
3	0.144119969	0	0.26	0.15	19997	0.42
4	0.101186919	0	0.27	0.16	19995	0.42
5	0.185553822	0	0.27	0.16	19996	0.42
6	0.065820027	0	0.27	0.15	19995	0.42
7	0.161825306	0	0.26	0.15	19995	0.42
8	0.269925062	0	0.27	0.15	19996	0.42
9	1.37939E-05	0	0.27	0.15	20000	0.42
10	0.217016083	0	0.27	0.15	19992	0.42

PROMEDIO	0.128575085	0
DESVIACIÓN TÍPICA	0.083541576	0
VALOR MEJOR	1.37939E-05	0
VALOR PEOR	0.269925062	0

FMAX-FMIN	0

PARÁMETROS

ALFA	0.005
BETA	0.4
G	1.00E+05
N	1.00E+02
N_ITER	1.00E+01
VALNEL	1.00E+03
DIM	2

Figura 5.1. Hoja Excel B2-NELME ejecución 3.

En la hoja de RESULTADOS GENERALES recogemos los valores del mejor y peor óptimos y el mejor y peor promedios de las 3 tandas realizadas con cada función. También detallamos el valor de *valnel* del mejor óptimo para comprobar si Nelder-Mead mejora de forma efectiva al aumentar su número máximo de ejecuciones.

Hemos añadido un gráfico que represente la mejora efectiva del óptimo con el aumento de *valnel*, esto es, a la vista de la figura 5.2, sí se produce una mejora significativa del óptimo entre los valores *valnel=10* y *valnel=100*, no así al aumentar *valnel* hasta *1.000*. Por tanto *valnel=100* parece el valor más eficiente para esta hibridación. Esta misma situación se repite en casi todas las funciones analizadas, ver CD adjunto.

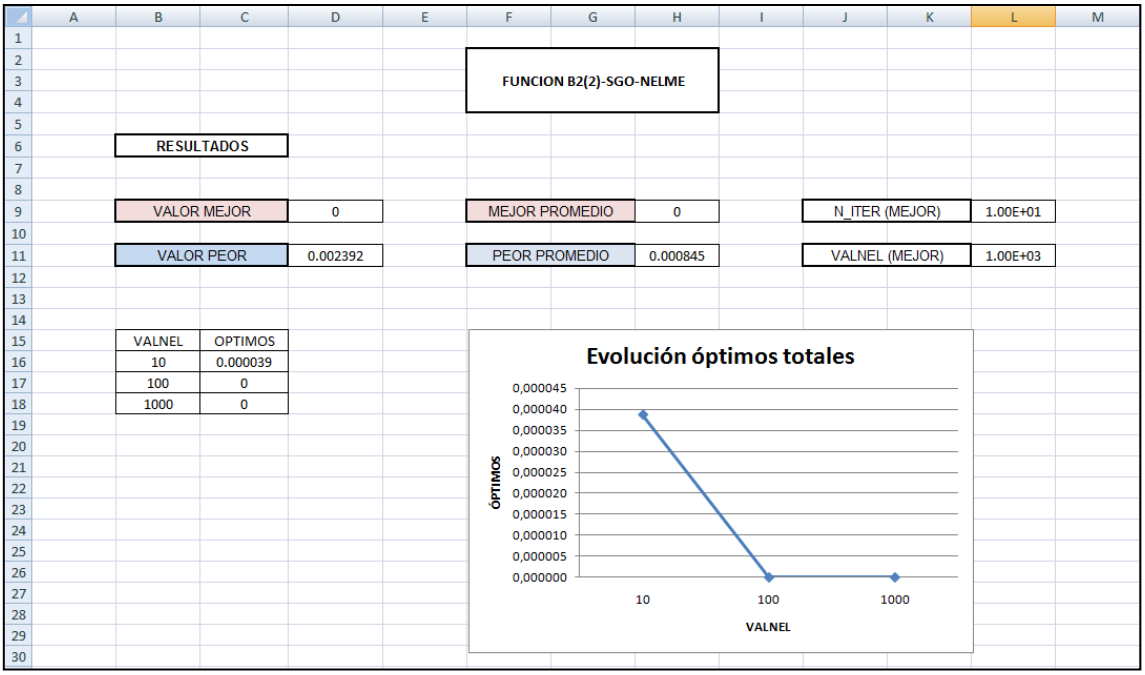


Figura 5.2. Hoja Excel B2-NELME RESULTADOS GENERALES.

Tras completar las ejecuciones con todas las instancias de la misma dimensión hemos recabado todos los valores de los óptimos obtenidos en un archivo Excel que consta de una sola hoja llamada genéricamente RESULTADOS S.G.O.-NELDER-MEAD DIMENSIÓN N. Expondremos todas ellas a continuación.

Todos los archivos están en el CD anexo al presente trabajo.

3.1. FUNCIONES DE DIMENSIÓN 2

Doce son las instancias de dimensión 2 con las que hemos probado S.G.O.-Nelder-Mead. En todas ellas hemos variado *valnel* en el rango *10, 10^2, 10^3*.

No expondremos cada una de las hojas Excel aunque pueden consultarse en el CD adjunto.

A continuación presentamos en la hoja Excel RESULTADOS S.G.O.- NELDER-MEAD DIMENSION 2, figura 5.3, los resultados obtenidos.

La columna TIEMPO y la columna E.F.O. detallan el tiempo y el número de evaluaciones de la función objetivo de la ejecución concreta, de las 30, en la que hemos obtenido el mejor óptimo.

RESULTADOS S.G.O.-NELDER MEAD DIMENSION 2

Nº	FUNCIÓN	ÓPTIMO DE S.G.O.	MEJOR OPTIMO	MEJOR PROMEDIO	TIEMPO (SEG.)	E.F.O.
1	BRANIN	1,122850E-05	-2,20056E-09	-2,20056E-09	0,45	19993
2	B2	1,379392E-05	0	0	0,43	19994
3	EASOM	8,952131E-01	0	0,19998378	0,30	15494
4	GOLDSTEIN AND PRICE	1,010260E+00	-7,81597E-14	-7,03437E-14	1,03	15485
5	SHUBERT	6,473697E+00	8,52651E-14	1,02318E-13	0,85	19990
6	BEALE	6,178690E-04	0	0	1,06	19991
7	BOOTH	1,249118E-02	0	0	0,56	19989
8	MATYAS	6,160015E-05	1,2009E-277	1,1482E-272	0,57	19946
9	SIXHUMPCAMELBACK	5,603342E-05	4,65101E-08	4,65101E-08	0,64	19991
10	SCHWEFEL	3,807695E+01	4,54747E-13	4,77485E-13	0,35	15495
11	ROSENBROCK	5,819848E-02	0	2,46519E-31	0,68	19990
12	ZAKHAROV	1,059730E-06	9,0244E-281	2,3818E-273	0,94	19943

Figura 5.3. Hoja Excel de RESULTADOS S.G.O.-NELDER-MEAD DIMENSIÓN 2.

En todos los casos hemos hallado el óptimo, ya que todos los errores son menores que $3 \cdot 10^{-6}$ y todos los promedios permanecen menores que $3 \cdot 10^{-3}$ salvo en una ocasión: la función EASOM. Dada la peculiar topología de esta función, los valores obtenidos han sido exclusivamente 0 y 1. En la EJECUCIÓN-2 de la función de EASOM podemos comprobar que de los 10 resultados hemos obtenido ocho valores 0 y dos valores 1, ver CD anexo. Por lo tanto se cumple nuestro segundo criterio de optimalidad.

Hemos obtenido mejores aproximaciones de los óptimos en los casos en que su valor no es entero, funciones SHUBERT y SIXHUMPCAMELBACK.

La eficiencia del método ha sido muy buena: con **menos de 20000 evaluaciones** de la función objetivo hemos hallado el óptimo en todos los casos con errores menores que una millonésima.

3.2. FUNCIONES DE DIMENSIÓN 3

En el artículo de Laguna y Martí (2005) sólo tenemos dos funciones de dimensión 3. En ambas hemos mantenido los valores *n_iter=10* y *n=100* y hemos variado *valnel* en el rango 10, 10^2 y 10^3. Presentamos los resultados en la figura 5.4.

Igual que en el caso anterior, la columna TIEMPO y la columna E.F.O. detallan el tiempo y el número de evaluaciones de la función objetivo de la ejecución concreta, de las 30, en la que hemos obtenido el mejor óptimo.

	A	B	C	D	E	F	G	H
1								
2				RESULTADOS S.G.O.-NELDER MEAD DIMENSION 3				
3								
4								
5								
6								
7	Nº	FUNCIÓN	ÓPTIMO DE S.G.O.	MEJOR OPTIMO	MEJOR PROMEDIO	TIEMPO (SEG.)	E.F.O.	
8	1	JONG	5,58383E-05	4,469E-164	4,4233E-152	0,694	33864	
9	2	HARTMANN(3,4)	0,038996385	3,9968E-15	4,39648E-15	1,466	33982	
10								

Figura 5.4. Hoja Excel de RESULTADOS S.G.O.-NELDER-MEAD DIMENSIÓN 3.

Se observa que hemos **hallado todos los óptimos** de forma muy satisfactoria con menos de 34000 evaluaciones de la función objetivo.

También hemos conseguido mejor aproximación del óptimo para la función de HARTMANN (3,4), ya que no es entero.

3.3. FUNCIONES DE DIMENSIÓN 4

Las instancias resueltas en esta ocasión son siete. Como en los casos anteriores en todas ellas hemos mantenido *n* y *n_iter* en los valores *100* y *10* respectivamente. El rango de variación de *valnel* ha sido 10^2, 10^3 y 10^4.

Los resultados se exponen en la figura 5.5.

Como ya hemos comentado, la columna TIEMPO y la columna E.F.O. detallan el tiempo y el número de evaluaciones de la función objetivo de la ejecución concreta, de las 30, en la que hemos obtenido el mejor óptimo.

	A	B	C	D	E	F	G	H
1								
2				RESULTADOS S.G.O.-NELDER MEAD DIMENSION 4				
3								
4								
5								
6								
7	Nº	FUNCIÓN	ÓPTIMO DE S.G.O.	MEJOR OPTIMO	MEJOR PROMEDIO	TIEMPO (SEG.)	E.F.O.	
8	1	COLVILLE	1,642649E+02	0	2,26748E-30	4,206	114785	
9	2	SHEKEL(5)	1,470252E-01	3,20942E-07	3,20942E-07	4,245	114916	
10	3	SHEKEL(7)	5,907452E-01	-4,05668E-05	-4,05679E-06	4,310	114920	
11	4	SHEKEL(10)	8,988007E-01	-4,79616E-14	-4,56524E-14	1,743	51938	
12	5	PERM(4,0.5)	1,836296E+01	3,39053E-28	0,000967104	3,581	114806	
13	6	PERM0(4,10)	1,423722E-01	0	0	3,120	114966	
14	7	POWERSUM	1,098647E-01	4,86357E-24	4,07753E-23	3,183	111977	
15								

Figura 5.5. Hoja Exce del RESULTADOS S.G.O.-NELDER-MEAD DIMENSIÓN 4.

Una vez más, **hemos alcanzado el óptimo en todos los casos**.

Las funciones de SHEKEL no tienen óptimo entero, así hemos mejorado la precisión en los 3 casos. Puede verse el valor de los óptimos hallados en las hojas de Excel de resultados del CD. En el resto de los casos no hemos tenido ese problema porque el óptimo es 0.

En el caso de la función de SHEKEL (10) el valor de E.F.O. es menor porque obtuvimos el mejor resultado para el caso *valnel=10^3*. No obstante, los resultados en el caso *valnel=10^4* también eran de orden 10^{-14}, ver CD adjunto.

El número de llamadas a la función objetivo ha sido algo inferior a 115000. A nuestro entender la hibridación es muy eficiente, no obstante, en muchos casos los resultados obtenidos para *valnel=10^3* han sido ya muy aceptables por lo que, si fuera necesario, podríamos acortar tiempos de ejecución perdiendo un poco de precisión.

3.4. FUNCIONES DE DIMENSIÓN 6

Las instancias tratadas en dimensión 6 no han supuesto tampoco problema para nuestro heurístico, salvo la función de SCHWEFEL 6. En el resto hemos variado *valnel* en los valores *10^2 , 10^3 y 10^4*.

En el caso de la función de SCHWEFEL6, los datos obtenidos con los valores anteriores han sido malos, por ello hemos incrementado la cantidad de asteroides lanzados y también los movimientos de cada asteroide hasta los valores *n=500* y *niter=500 y* valnel ha tomado el valor $6 \cdot 10^5$. Como puede apreciarse en los resultados, los asteroides han quedado atrapados en algunas ocasiones y Nelder-Mead los ha guiado a óptimos locales. Los promedios no han resultado menores que 10^{-3} aunque sí hemos

obtenido valores con errores menores que 10^{-6}, obviamente aumentando el valor de E.F.O. En esta ocasión no hemos alcanzado el óptimo.

A continuación presentamos los resultados generales de dimensión 6, figura 5.6. La columna TIEMPO y la columna E.F.O. detallan el tiempo y el número de evaluaciones de la función objetivo de la ejecución concreta, de las 30, en la que hemos obtenido el mejor óptimo.

	A	B	C	D	E	F	G	H
1								
2				RESULTADOS S.G.O.-NELDER MEAD DIMENSION 6				
3								
4								
5								
6								
7	Nº	FUNCIÓN	ÓPTIMO DE S.G.O.	MEJOR OPTIMO	MEJOR PROMEDIO	TIEMPO (SEG.)	E.F.O.	
8	1	HARTMANN6	1,586154E-01	1,98858E-06	1,98858E-06	8,143	180817	
9	2	SCHWEFEL	3,619091E-06	-4,346025E-07	136,2049016	315,521	28149791	
10	3	TRID6	2,996593E-01	-1,136868E-13	-6,82121E-14	3,640	180848	
11								

Figura 5.6. Hoja Excel de RESULTADOS S.G.O.-NELDER-MEAD DIMENSIÓN 6.

Como ya hemos comentado, **hemos hallado el óptimo para todos los casos a excepción de la función de SCHWEFEL.**

El elevado promedio de esta función indica que en varias ejecuciones **algunas coordenadas** de las abscisas de las soluciones **han quedado atrapadas** en óptimos locales, aunque no en todas las ejecuciones, a la vista del mejor resultado. Como veremos en capítulos posteriores, el algoritmo AGUJERO DE GUSANO solucionará este problema.

El comportamiento de S.G.O.-Nelder-Mead con las otras instancias ha sido excelente.

3.5. FUNCIONES DE DIMENSIÓN 10

En esta ocasión probamos S.G.O. con seis instancias bien diferenciadas: por un lado están las funciones “topológicamente rebeldes”: todas ellas son multimodales con grandes oscilaciones y gradientes altos en valor absoluto, que van tomando alternativamente valores positivos y negativos. A este grupo pertenecen las funciones de RASTRIGIN y de GRIEWANK. Por otro lado están las funciones “topológicamente buenas” que, aun siendo algunas multimodales, tienen gradientes pequeños permitiendo a los asteroides desplazarse libremente. Forman este grupo las funciones SUMSQUARES, de ROSENBROCK, de ZAKHAROV y de TRID

En el primer caso *n_iter* ha variado entre los valores *10* y *50*, *valnel* ha variado en el rango $10^4, 10^5 \; y \; 5 \cdot 10^5$. Para el segundo grupo *n_iter* ha permanecido en el valor *10* y *valnel* ha variado tomando los valores $10^2, 10^3 \; y \; 10^4$.

Presentamos los resultados obtenidos en la figura 5.7. Como en casos anteriores, la columna TIEMPO y la columna E.F.O. detallan el tiempo y el número de evaluaciones de la función objetivo de la ejecución concreta, de las 30, en la que hemos obtenido el mejor óptimo.

RESULTADOS S.G.O.-NELDER MEAD DIMENSION 10

Nº	FUNCIÓN	ÓPTIMO DE S.G.O.	MEJOR OPTIMO	MEJOR PROMEDIO	TIEMPO	E.F.O.
1	TRID10	6,496065E+00	-1,818989E-12	1,182343E-12	6,578	360345
2	RASTRIGIN10	7,241089E+01	1,989918E+00	1,691428E+01	10,629	360083
3	GRIEWANK10	2,616589E-01	0,000000E+00	1,479208E-03	47,696	1530068
4	SUM SQUARES10	1,581055E+02	4,261542E-161	2,070583E-146	6,799	355777
5	ROSENBROCK10	2,690914E+04	3,525222E-29	1,993290E+00	7,324	358577
6	ZAKHAROV10	5,166569E+01	3,407196E-164	1,242851E-152	7,120	355713

Figura 5.7. Hoja Excel de RESULTADOS S.G.O.-NELDER-MEAD DIMENSIÓN 10.

Hemos alcanzado el óptimo según nuestro criterio para todas las funciones a excepción de dos: de ROSENBROCK 10 y de RASTRIGIN 10.

En el primer caso sí hemos hallado el mínimo con un error muy pequeño pero no hemos conseguido obtener buenos promedios. Ésta función tiene gradientes muy bajos cerca del óptimo y algunas veces los asteroides no han tenido suficiente impulso para llegar, aunque en otras sí lo han logrado.

En el segundo caso nuestro mejor resultado ha distado considerablemente del óptimo por la topología de la función. No obstante, hemos buscado la dimensión más alta en la que el metaheurístico ha respondido favorablemente con *n_iter* = *10* y *valnel* = $5 \cdot 10^5$. Presentamos los resultados en la figura 5.8.

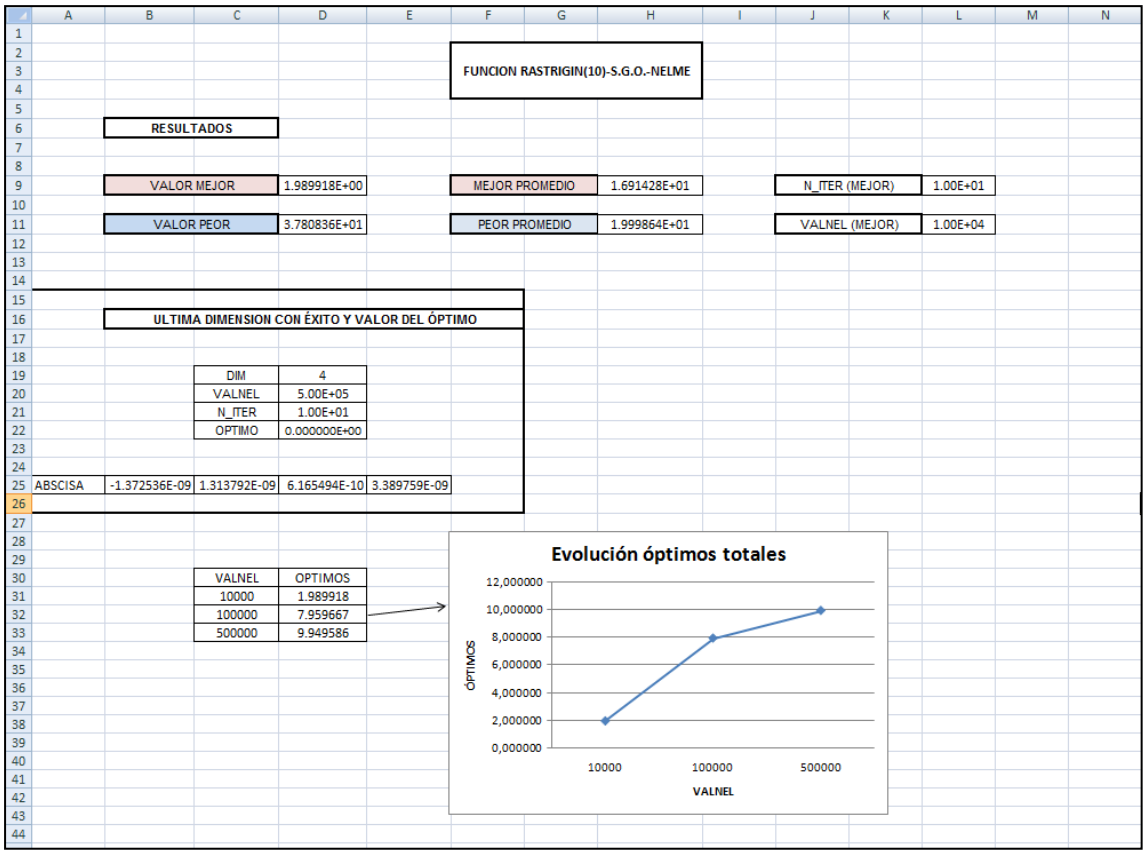


Figura 5.8. Hoja Excel de mayor dimensión exitosa para la función de RASTRIGIN.

El valor de *E.F.O*. para la función de RASTRIGIN 10 es similar al de las funciones "topológicamente buenas" porque el mejor óptimo fue obtenido para *n=10* y *n_iter=10*^{4}. En las ejecuciones con valores más altos sí mejoraron los promedios y disminuyeron las desviaciones típicas, ver CD anexo.

S.G.O-Nelder-Mead ha obtenido resultados muy buenos con la función de GRIEWANK porque, a pesar de su topología, los gradientes no son muy altos y los agentes de búsqueda han conseguido escapar de los óptimos locales. De hecho ya con *n=10* y *valnel=10*^{4} hemos obtenido el óptimo según nuestro criterio, aunque el mejor valor se ha obtenido para *n*=10 y *n_iter=10*^{5}, por eso el valor de *E.F.O.* es superior al resto, ver CD anexo.

3.6. FUNCIONES DE DIMENSIÓN 20

Hemos probado S.G.O.-Nelder-Mead con las mismas instancias elegidas en dimensión 10, salvo la función de TRID 10, obviamente considerando la región factible 20-dimensional. En todos los casos hemos considerado el valor *n_iter=10.*

Para las funciones de ZAKHAROV y SUMSQUARES hemos usado los valores *valnel*=10^3, 10^4 y 10^5. Para las funciones de GRIEWANK y de RASTRIGIN hemos usado el rango *valnel*=10^4, 10^5 y $5 \cdot 10^5$. Para la función de ROSENBROCK, el rango *valnel*=10^5 , $5 \cdot 10^5$, 10^6. Esta es la explicación para la disparidad en los valores de *E.F.O.*

Como era de esperar, los resultados para RASTRIGIN 20 no han sido buenos, además el promedio obtenido para GRIEWANK 20 ya no satisface nuestro criterio de alcance del óptimo, al aumentar el número de variables el algoritmo pierde eficacia, aunque sí hemos obtenido el mejor óptimo con un error excelente.

En el resto de las funciones hemos alcanzado el óptimo. Veamos los resultados generales en la figura 5.9. Como ya hemos comentado, la columna TIEMPO y la columna E.F.O. detallan el tiempo y el número de evaluaciones de la función objetivo de la ejecución concreta, de las 30, en la que hemos obtenido el mejor óptimo.

RESULTADOS S.G.O.-NELDER MEAD DIMENSION 20

Nº	FUNCIÓN	ÓPTIMO DE S.G.O.	MEJOR ÓPTIMO	MEJOR PROMEDIO	TIEMPO	E.F.O.
1	RASTRIGIN20	1,518094E+02	1,094455E+01	2,636635E+01	351,095	12349950
2	GRIEWANK20	1,486563E+02	2,109424E-15	1,919262E-02	273,577	12350321
3	SUM SQUARES20	7,003908E+01	0,000000E+00	0,000000E+00	61,562	3156198
4	ROSENBROCK20	4,687691E+05	3,453659E-26	9,074516E-26	245,061	12234229
5	ZAKHAROV20	2,482232E+02	1,639534E-200	1,490449E-180	54,487	3087388

Figura 5.9. Hoja Excel de RESULTADOS S.G.O.-NELDER-MEAD DIMENSIÓN 20.

3.7. FUNCIONES DE DIMENSIÓN MAYOR QUE 20

Cinco son las funciones elegidas en esta ocasión por Laguna y Martí (2005): tres de dimensión 30, de LEVY, de JONG y de ACKLEY; una de dimensión 24, de POWELL, y una de dimensión 25, de DIXON & PRICE.

En todos los casos salvo para la función de LEVY hemos usado los valores *n_iter=10* y hemos variado *valnel* en el rango 10^4 , 10^5 , $5 \cdot 10^5$. Para la función de LEVY hemos incrementado *n_iter* al valor *100* manteniendo los valores para *valnel* como en el resto de las instancias, por eso el valor *E.F.O.* es superior en este caso.

Sólo en dos casos el algoritmo no ha superado nuestro criterio de optimización. En la figura 5.10 presentamos los resultados

En los otros tres casos, que detallamos en la figura 5.10, **la hibridación ha alcanzado el óptimo satisfactoriamente.**

Recordamos que la columna TIEMPO y la columna E.F.O. detallan el tiempo y el número de evaluaciones de la función objetivo de la ejecución concreta, de las 30, en la que hemos obtenido el mejor óptimo.

RESULTADOS S.G.O.-NELDER MEAD DIMENSION MAYOR QUE 20

Nº	FUNCIÓN	ÓPTIMO DE S.G.O.	MEJOR ÓPTIMO	MEJOR PROMEDIO	TIEMPO	E.F.O.
1	POWELL24	3,609915E+03	4,258095E-13	1,441107E-06	724,275	14334875
2	DIXON AND PRICE25	4,197200E+05	6,666667E-01	8,708717E-01	372,134	15131912
3	LEVY30	1,817324E-01	6,581939E-07	9,673174E-04	1223,135	22128695
4	JOUNG30 (SPHERE)	1,143291E-02	3,171053E-16	9,448377E-06	126,238	5109712
5	ACKLEY30	1,647415E+01	3,786261E+00	5,721325E+00	66,596	2212993

Figura 5.10. Hoja Excel de RESULTADOS S.G.O.-NELDER-MEAD DIMENSIÓN MAYOR QUE 20.

La función de ACKLEY tiene una topología, en principio, similar a la de la función de EASOM, muy plana en casi todo su dominio y con un “embudo” de gradiente alto y negativo en el entorno del mínimo. Sin embargo, la función de ACKLEY es multimodal, lo que le aporta una textura “rugosa” que frena a los asteroides y los captura en óptimos locales, impidiendo que se deslicen por el “embudo” hasta el óptimo. Así los datos obtenidos no han sido muy buenos. La mayor dimensión en la que hemos alcanzado éxito ha sido *dim=9*. Presentamos la hoja de Excel con los resultados en la figura 5.11.

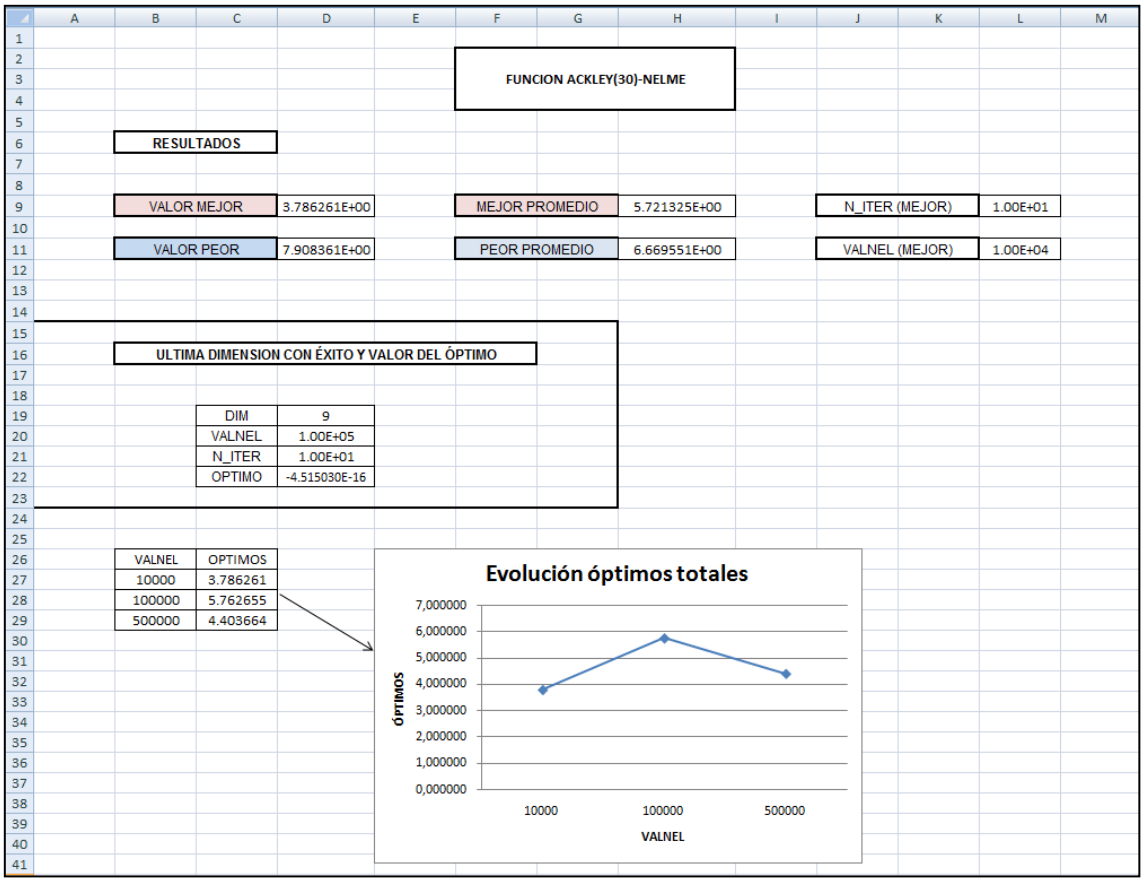


Figura 5.11. Hoja Excel de última dimensión exitosa para la función ACKLEY 30.

3.7.1. LA FUNCIÓN DE DIXON & PRICE

La función de DIXON & PRICE es la única del grupo en la que el óptimo, que es 0, no se obtiene en valores enteros de las abscisas sino en los valores de la forma

$$\vec{x}_{MIN} = \left(1, 2^{\frac{1}{2^1}-1}, 2^{\frac{1}{2^2}-1}, ..., 2^{\frac{1}{2^k}-1}, ..., \pm 2^{\frac{1}{2^{\dim-1}}-1}\right)$$

Se trata de una función polinómica que, independientemente de la dimensión, tiene un punto de silla ubicado en el punto $\vec{x}_{\max r} = \left(\frac{1}{3}, 0, ..., 0\right)$ de valor $\frac{2}{3}$ y dos mínimos relativos, que además son absolutos, en las abscisas anteriormente citadas, de valor 0. Fuera de esta zona la función crece con orden cuarto.

Veamos con detalle el caso *dim=2*.

$$F_{DIXON\,2}(x, y) = (x-1)^2 + 2\left(2y^2 - x\right)^2$$

Usando técnicas exactas de Análisis Matemático podemos conocer sus puntos estacionarios

$$\left.\begin{array}{l}\dfrac{\partial F_{DIXON2}}{\partial x}(x,y)=6x-8y^2-2\\ \dfrac{\partial F_{DIXON2}}{\partial y}(x,y)=32y^3-16xy\end{array}\right\};\quad \left.\begin{array}{l}6x-8y^2-2=0\\ 32y^3-16xy=0\end{array}\right\}\Rightarrow \begin{array}{r}P_1=\left(1,2^{\frac{1}{2}-1}\right)\\ P_2=\left(1,-2^{\frac{1}{2}-1}\right)\\ Q=\left(\dfrac{1}{3},0\right)\end{array}$$

Y estudiando las matrices Hessianas en los puntos estacionarios llegamos sin dificultad a la conclusión ya expuesta: P_i mínimos relativos y *Q* punto de silla.

Si pasamos a *dim=3* tenemos que

$F_{DIXON3}(x,y,z)=(x-1)^2+2\left(2y^2-x\right)^2+3\left(2z^2-y\right)^2$ y si planteamos el correspondiente sistema para hallar los puntos estacionarios llegamos a

$$\left.\begin{array}{l}\dfrac{\partial F_{DIXON3}}{\partial x}(x,y,z)=6x-8y^2-2=0\\ \dfrac{\partial F_{DIXON2}}{\partial y}(x,y,z)=32y^3-16xy-6\left(2z^2-y\right)=0\\ \dfrac{\partial F_{DIXON3}}{\partial z}(x,y,z)=24\left(2z^2-y\right)z=0\end{array}\right\}$$

que mantiene todas las ecuaciones iguales al sistema planteado para dimensión 2 salvo la última y obviamente presenta una ecuación más como resultado de introducir la nueva variable.

Si comenzamos a resolver el sistema tridimensional por la última ecuación tenemos

O bien $\begin{cases}z=0\\ 2z^2-y=0\end{cases}$

El caso $2z^2-y=0$ implica recuperar exactamente el sistema bidimensional pero con la variable y tomando solo valores positivos o nulo ya que $y=2z^2$.

Si $y=2^{\frac{1}{2}-1}$ tenemos

$$y = 2z^2 \Rightarrow z^2 = \frac{y}{2} = \frac{2^{\frac{1}{2}-1}}{2} = 2^{\frac{1}{2}-2} \Rightarrow z = \pm 2^{\frac{1}{2^2}-1}.$$

Si $y = 0$ obviamente $z = 0$

Por lo tanto tenemos las soluciones

$$\left(1, 2^{\frac{1}{2}-1}, \pm 2^{\frac{1}{2^2}-1}\right) \quad , \quad \left(\frac{1}{3}, 0, 0\right)$$

El caso $z = 0$ deja el sistema tridimensional en

$$\left.\begin{array}{l} 6x - 8y^2 - 2 = 0 \\ 32y^3 - 16xy + 6y = 0 \\ 0 = 0 \end{array}\right\} \Rightarrow \left.\begin{array}{c} 6x - 8y^2 - 2 = 0 \\ 2\left(16y^2 - 8x + 3\right) y = 0 \\ 0 = 0 \end{array}\right\} \Rightarrow$$

$$\Rightarrow \text{o bien} \begin{cases} y = 0 \ \text{ solución ya vista} \\ y^2 = \dfrac{8x-3}{16} = \dfrac{3x-1}{4} \Rightarrow x = \dfrac{1}{4} \Rightarrow y^2 = \dfrac{-1}{16} \ \text{ imposible} \end{cases}$$

Por inducción puede demostrarse que aunque aumentemos la dimensión, los puntos estacionarios y los valores de la función van a ser los detallados anteriormente.

A medida que aumenta la dimensión del espacio de soluciones los valores de las abscisas en las que se alcanza el mínimo se van aproximando al valor 0.5 ya que

$$\lim_{\dim \to \infty} 2^{\frac{1}{2^{\dim -1}}-1} = 2^{\lim_{\dim \to \infty} \frac{1}{2^{\dim -1}}-1} = 2^{-1}$$

Podemos observar la representación tridimensional de la función de DIXON & PRICE en la figura 5.12.

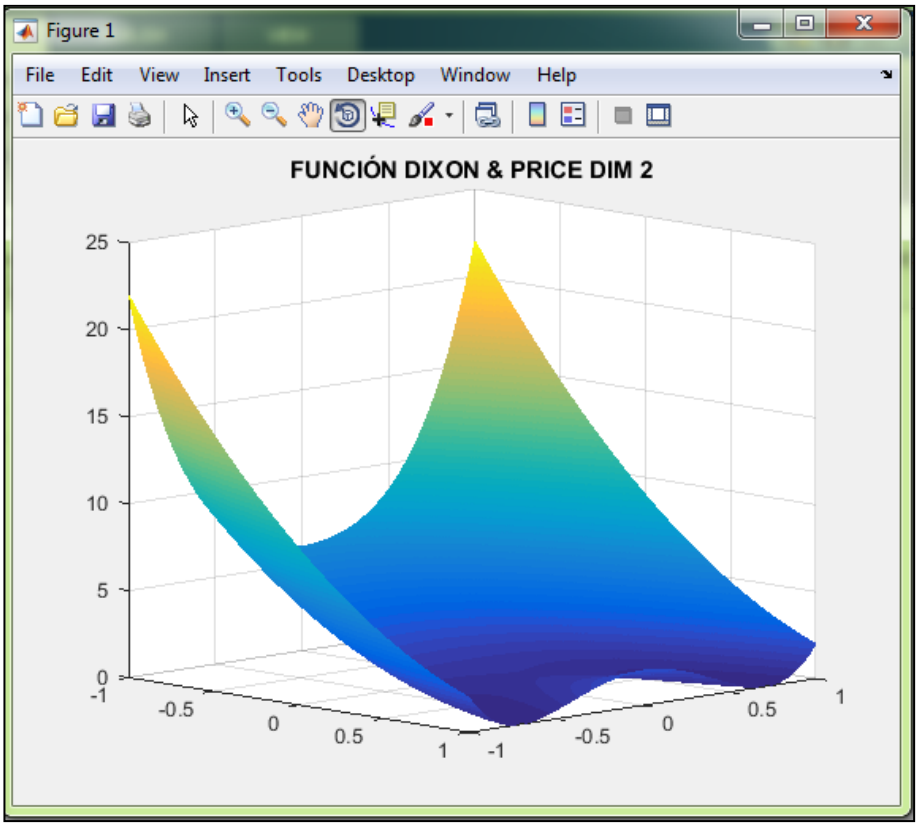


Figura 5.12. Representación con MATLAB de la función de DIXON & PRICE en DIMENSIÓN 2.

En dimensión 25 el algoritmo S.G.O.-Nelder-Mead ha guiado a los asteroides al punto de silla en todas las simulaciones por lo que, en la práctica, se comporta como un "mínimo local en casi todas las direcciones" evitando así que los agentes de búsqueda vayan hacia los mínimos globales, como se aprecia en los resultados de la figura 5.10. La mayor dimensión en la que hemos tenido éxito en la consecución del mínimo es *dim=12*. En este caso hemos detallado los valores de las abscisas encontrados que, como hemos comentado, se van aproximando al valor 0.5, figura 5.13.

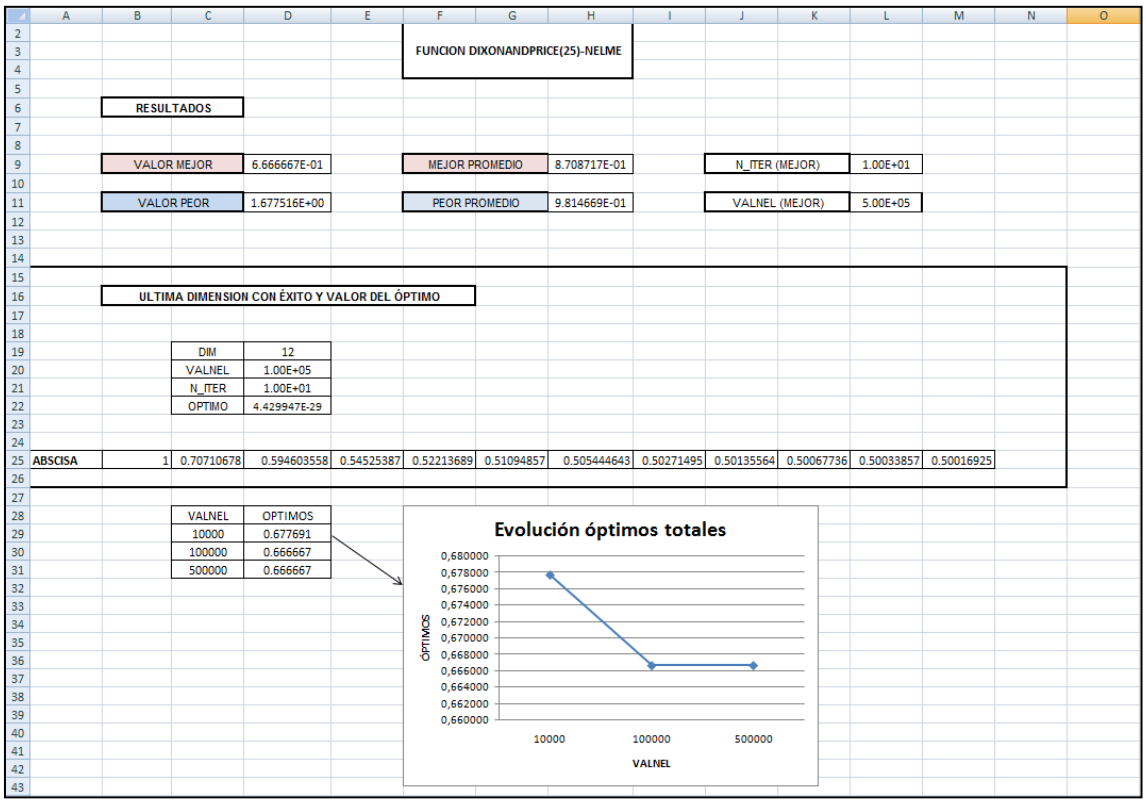


Figura 5.13. Hoja de RESULTADOS S.G.O.-NELDER-MEAD para DIXON & PRICE DIMENSIÓN 25.

3.8. CONCLUSIONES

De las 40 instancias evaluadas **hemos obtenido el óptimo, según nuestro criterio, en 33 de ellas** y sólo hemos fracasado en la consecución del óptimo en 7. **En las funciones de SCHWEFEL 6, de ROSENBROCK 10 Y de GRIEWANK 20 sí hemos conseguido el mejor resultado con errores menores que** $3 \cdot 10^{-6}$ pero han fallado los promedios obtenidos. En las funciones de RASTRIGIN 10, de RASTRIGIN 20, de DIXON & PRICE 25 y de ACKLEY 30 no hemos conseguido valores aceptable ni siquiera para el mejor óptimo.

Es curioso el caso de la función de ROSENBROCK. El comportamiento de la hibridación en dimensión 20 ha sido más aceptable que en dimensión 10.

En la mayor parte de los casos se trata de funciones pertenecientes al que hemos definido anteriormente como grupo de “topologías rebeldes” que sin duda son las más difíciles para S.G.O. porque los asteroides quedan atrapados en ellas.

En el resto de los casos, a pesar del hándicap que ha supuesto haber tenido que ejecutar S.G.O. *dim+1* veces para poder ejecutar tras él Nelder-Mead, los tiempos de ejecución han sido aceptables y los resultados muy satisfactorios.

A nuestro entender, la hibridación S.G.O.-Nelder-Mead se presenta por tanto, como un algoritmo muy eficaz, muy eficiente y muy recomendable si la topología de la función es buena. Si no es el caso y la dimensión del espacio de soluciones es grande creemos necesario buscar algún otro algoritmo con el que hibridar nuestro heurístico o nuestra hibridación para obtener buenos resultados. Recogemos el resumen de resultados en la figura 5.14.

RESUMEN DE RESULTADOS S.G.O.-NELDER MEAD

Nº	FUNCIÓN	DIM.	ÓPTIMO DE S.G.O.	MEJOR ÓPTIMO	MEJOR PROMEDIO	TIEMPO (SEG.)	E.F.O.
1	BRANIN	2	1.122850E-05	-2.200563E-09	-2.200563E-09	0.45	19993
2	B2	2	1.379392E-05	0.000000E+00	0.000000E+00	0.43	19994
3	EASOM	2	8.952131E-01	0.000000E+00	1.999838E-01	0.30	15494
4	GOLDSTEIN AND PRICE	2	1.010260E+00	-7.815970E-14	-7.034373E-14	1.03	15485
5	SHUBERT	2	6.473697E+00	8.526513E-14	1.023182E-13	0.85	19990
6	BEALE	2	6.178690E-04	0.000000E+00	0.000000E+00	1.06	19991
7	BOOTH	2	1.249118E-02	0.000000E+00	0.000000E+00	0.56	19989
8	MATYAS	2	6.160015E-05	1.200879E-277	1.148189E-272	0.57	19946
9	SIXHUMPCAMELBACK	2	5.603342E-05	4.651012E-08	4.651012E-08	0.64	19991
10	SCHWEFEL	2	3.807695E+01	4.547474E-13	4.774847E-13	0.35	15495
11	ROSENBROCK	2	5.819848E-02	0.000000E+00	2.465190E-31	0.68	19990
12	ZAKHAROV	2	1.059730E-06	9.024350E-281	2.381833E-273	0.94	19943
13	JONG	3	5.583834E-05	4.468989E-164	4.423258E-152	0.69	33864
14	HARTMANN(3,4)	3	3.899639E-02	3.996803E-15	4.396483E-15	1.47	33982
15	COLVILLE	4	1.642649E+02	0.000000E+00	2.267482E-30	4.21	114785
16	SHEKEL(5)	4	1,470252E-01	3,209418E-07	3,209418E-07	4,25	114916
17	SHEKEL(7)	4	5,907452E-01	-4,056682E-05	-4,056792E-06	4,31	114920
18	SHEKEL(10)	4	8,988007E-01	-4,796163E-14	-4,565237E-14	1,74	51938
19	PERM(4,0.5)	4	1,836296E+01	3,390530E-28	9,671042E-04	3,58	114806
20	PERM0(4,10)	4	1,423722E-01	0,000000E+00	0,000000E+00	3,12	114966
21	POWERSUM	4	1,098647E-01	4,863574E-24	4,077535E-23	3,18	111977
22	HARTMANN6	6	1,586154E-01	1,988584E-06	1,988584E-06	8,14	180817
23	SCHWEFEL	6	3,619091E-06	-4,346025E-07	1,362049E+02	315,52	28149791
24	TRID6	6	2,996593E-01	-1,136868E-13	-6,821210E-14	3,64	180848
25	TRID10	10	6,496065E+00	-1,818989E-12	1,182343E-12	6,58	360345
26	RASTRIGIN10	10	7,241089E+01	1,989918E+00	1,691428E+01	10,63	360083
27	GRIEWANK10	10	2,616589E-01	0,000000E+00	1,479208E-03	47,70	1530068
28	SUM SQUARES10	10	1,581055E+02	4,261542E-161	2,070583E-146	6,80	355777
29	ROSENBROCK10	10	2,690914E+04	3,525222E-29	1,993290E+00	7,32	358577
30	ZAKHAROV10	10	5,166569E+01	3,407196E-164	1,242851E-152	7,12	355713
31	RASTRIGIN20	20	1,518094E+02	1,094455E+01	2,636635E+01	351,10	12349950
32	GRIEWANK20	20	1,486563E+02	2,109424E-15	1,919262E-02	273,58	12350321
33	SUM SQUARES20	20	7,003908E+01	0,000000E+00	0,000000E+00	61,56	3156198
34	ROSENBROCK20	20	4,687691E+05	3,453659E-26	9,074516E-26	245,06	12234229
35	ZAKHAROV20	20	2,482232E+02	1,639534E-200	1,490449E-180	54,49	3087388
36	POWELL24	24	3,609915E+03	4,258095E-13	1,441107E-06	724,28	14334875
37	DIXON AND PRICE25	25	4,197200E+05	6,666667E-01	8,708717E-01	372,13	15131912
38	LEVY30	30	1,817324E-01	6,581939E-07	9,673174E-04	1223,13	22128695
39	JONG30 (SPHERE)	30	1,143291E-02	3,171053E-16	9,448377E-06	126,24	5109712
40	ACKLEY30	30	1,647415E+01	3,786261E+00	5,721325E+00	66,60	2212993

Figura 5.14. Hoja Excel de RESUMEN DE RESULTADOS S.G.O.-Nelder-Mead.

4. LA HIBRIDACIÓN S.G.O.- GRADIENTE

La hibridación S.G.O.-GRADIENTE consiste en una ejecución de S.G.O. con sus parámetros fijados como en el caso anterior y tras él, un número máximo de ejecuciones de Gradiente, *n_vecesgrad*, con el objeto de guiar al óptimo encontrado a su mínimo posible, sea este local o global. Pararemos el algoritmo en el momento en el que el óptimo no mejore.

Como ya hemos comentado, implementar el algoritmo Gradiente conlleva resolver un problema unidimensional para hallar la longitud del paso en cada iteración. En vez de ello, hemos optado por fijar un parámetro $\alpha_k = 1$ y posteriormente hallar los valores de la función en los puntos $x + \alpha_k \nabla f(x)$, $x + \frac{\alpha_k}{100} \nabla f(x)$, $x + \frac{\alpha_k}{1000} \nabla f(x)$ y $x + \frac{\alpha_k}{50000} \nabla f(x)$ y sustituir el punto x por el primer valor de los anteriores que mejore el valor de la función objetivo, entendemos que considerar las 4 longitudes abarca la mayor parte de los casos y si el paso real cumpliera $\alpha > 1$ solo repercutiría en mayor número de iteraciones**.** Lógicamente hemos considerado el gradiente de módulo unitario.

Computacionalmente la nueva hibridación es bastante menos costosa que la anterior, sobre todo en dimensiones altas, ya que Gradiente solo necesita una ejecución de S.G.O. y no *dim+1* como ocurría en el caso S.G.O.-Nelder-Mead, por ello podemos incrementar los parámetros de S.G.O. sin aumentar las evaluaciones totales de la función objetivo. De esta forma dejaremos que S.G.O. explore con mayor detalle la región factible, ya que Gradiente solo se limitará a llevar cada solución concreta obtenida con S.G.O. a su mínimo local.

Consideraremos *n=100* asteroides, salvo en los casos en que los dominios sean muy grandes en los que tendremos en cuenta *n=500* asteroides*,* moviéndose *n_iter=1000* movimientos. Hemos rebajado *n_iter* respecto a los valores del capítulo 3, *10000,* porque, al estar trabajando en dimensiones grandes las evaluaciones a la función objetivo de S.G.O. se incrementan en gran medida.

Dado que el algoritmo Gradiente conlleva muy poco coste computacional, hemos considerado *n_vecesgrad=*10^4 en casi todos los casos, teniendo en cuenta además, que en el momento en que no se mejore el óptimo, Gradiente parará la ejecución.

No obstante, en cada caso hemos procurado que el número *E.F.O.* fuera similar al de la hibridación anterior. Como veremos seguidamente en varios casos ha sido inferior.

Una vez elegidos todos los valores para cada instancia haremos una sola tanda de 10 ejecuciones y registraremos los datos obtenidos en una hoja de Excel llamada genéricamente FUNCIÓN S.G.O.-GRADIENTE. A continuación expondremos algunas de ellas.

Recogeremos los datos de todas las instancias en un archivo Excel de RESUMEN DE RESULTADOS S.G.O.-GRADIENTE que también expondremos en la presente sección.

Nuestro objetivo en esta ocasión es comprobar si la nueva hibridación es capaz de encontrar el óptimo, atendiendo al criterio que nos fijamos al principio del presente capítulo, en las instancias en las que S.G.O.-Nelder-Mead ha fallado, empleando un número de evaluaciones de la función objetivo similar o inferior a la hibridación S.G.O.-Nelder-Mead.

Recordamos nuestro criterio de consecución del óptimo:

Consideraremos que hemos alcanzado el óptimo si el error de alguno de los resultados obtenidos es menor que $3 \cdot 10^{-6}$ y el error de alguno de los promedios es menor que $3 \cdot 10^{-3}$.

O bien

Consideraremos que hemos alcanzado el óptimo si el error de alguno de los resultados obtenidos es menor que $3 \cdot 10^{-6}$ y en alguna de las tandas de 10 ejecuciones hemos obtenido errores menores que $3 \cdot 10^{-6}$ en, al menos, 7 de ellas.

5. RESULTADOS DE S.G.O.- GRADIENTE

Recordamos que hemos probado S.G.O.-Gradiente exclusivamente con las 7 instancias en las que hemos fracasado con S.G.O.-Nelder-Mead. Para cada instancia hemos rellenado una hoja de Excel con 10 ejecuciones en la que hemos recogido, como en la hibridación anterior, los óptimos y tiempos parciales, el mejor y peor valores, el rango, el promedio y la desviación típica correspondientes, figura 5.15.

En dos de ellas hemos obtenido el óptimo. A continuación exponemos los resultados obtenidos.

5.1. FUNCIÓN DE SCHWEFEL 6

Como se aprecia en la figura 5.15, los resultados son, en general, similares a los obtenidos con la hibridación anterior porque, tanto Nelder-Mead como Gradiente mantienen los datos obtenidos por S.G.O., es decir, si éstos eran buenos, los han mejorado, pero si no eran buenos, alguna abscisa atrapada en óptimos locales, ambas hibridaciones se han mostrado incapaces de sacar las abscisas atrapadas. *E.F.O.* es muy inferior en este caso porque hemos ejecutado S.G.O. con valores similares a la sección anterior pero, lógicamente solo una vez. Hemos comprobado empíricamente que aumentar *n_iter* empeora los resultados, ver CD anexo.

FUNCION SCHWEFEL(6)-S.G.O.-GRADIENTE

EJECUCION

N_EJEC	FOPT-PARC	FOPT	TIEMP-GRAVI	TIEMP-GRAD	E.F.O TOTAL	TIEMP-TOTAL
1	7.493974E+02	4.145450E+02	1.39	0.02	66047	1.41
2	9.006645E+02	5.527177E+02	1.39	0.02	66227	1.41
3	9.336523E+02	3.553150E+02	1.38	0.02	66069	1.40
4	6.187938E+02	3.553150E+02	1.39	0.03	66347	1.41
5	9.228355E+02	-4.241929E-07	1.38	0.02	66031	1.40
6	8.172548E+02	7.896326E+01	1.39	0.02	65873	1.40
7	7.282271E+02	2.961067E+02	1.39	0.02	65809	1.40
8	1.007249E+03	3.553150E+02	1.38	0.02	65945	1.40
9	6.870088E+02	7.896326E+01	1.38	0.02	65835	1.40
10	8.056881E+02	-4.220979E-07	1.38	0.02	65875	1.40

PROMEDIO	8.170772E+02	2.487241E+02
DESVIACIÓN TÍPICA	1.232728E+02	1.936789E+02
VALOR MEJOR	6.187938E+02	-4.241929E-07
VALOR PEOR	1.007249E+03	5.527177E+02

FMAX-FMIN	5.527177E+02

PARÁMETROS

ALFA	0.005
BETA	0.4
G	1.00E+05
N	5.00E+02
N_ITER	1.00E+01
N_GRAD	1.00E+04
DIM	6

Figura 5.15. Hoja Excel de RESULTADOS SCHWEFEL S.G.O.-GRADIENTE.

5.2. FUNCIÓN DE ROSENBROCK 10

Para esta función Gradiente se ha mostrado más potente que Nelder-Mead porque **ha guiado de forma efectiva a los asteroides**, que tenían poca aceleración**, hacia el óptimo global**. Hemos aumentado *n_vecesgrad* a 50000 manteniendo *n_iter* en *10* para no aumentar el número de evaluaciones de la función objetivo.

La diferencia de los valores de *E.F.O.* y los tiempos de ejecución se explica precisamente por este hecho. No obstante, las diferencias no son significativas, como se recoge en la figura 5.16.

FUNCION ROSENBROCK(10)-S.G.O.-GRADIENTE

EJECUCION

N_EJEC	FOPT-PARC	FOPT	TIEMP-GRAVI	TIEMP-GRAD	E.F.O TOTAL	TIEMP-TOTAL
1	1.284591E+05	8.814270E-08	9.56	0.42	618775	9.99
2	7.441501E+04	3.054720E-05	2.74	0.42	171469	3.16
3	1.070780E+05	8.814261E-08	9.72	0.42	628515	10.14
4	3.375678E+04	3.652793E-05	3.14	0.42	196819	3.56
5	4.457340E+04	8.814255E-08	9.67	0.42	627001	10.10
6	1.501047E+05	8.814275E-08	9.67	0.42	625219	10.09
7	9.991549E+04	8.814301E-08	9.38	0.42	606711	9.81
8	1.529956E+05	8.814267E-08	10.43	0.42	673253	10.85
9	1.523164E+05	8.814297E-08	9.40	0.42	608415	9.83
10	1.415609E+05	2.361560E-07	3.11	0.42	194781	3.53

PROMEDIO	1.085175E+05	6.792829E-06
DESVIACIÓN TÍPICA	4.476200E+04	1.416610E-05
VALOR MEJOR	3.375678E+04	8.814255E-08
VALOR PEOR	1.529956E+05	3.652793E-05

FMAX-FMIN 3.643979E-05

PARÁMETROS

ALFA	0.005
BETA	0.4
G	1.00E+05
N	1.00E+02
N_ITER	1.00E+01
N_GRAD	5.00E+05
DIM	10

Figura 5.16. Hoja Excel de RESULTADOS ROSENBROCK 10 S.G.O.-GRADIENTE.

5.3. LAS FUNCIONES DE RASTRIGIN 10 Y 20

En ambos casos los resultados obtenidos por S.G.O.-Gradiente han sido similares a los obtenidos en la hibridación anterior. El valor de *E.F.O.* es inferior porque, como en el caso de la función de SCHWEFEL 6, aumentar *n_iter* empeora los resultados. Las correspondientes hojas Excel pueden verse en el CD adjunto.

5.4. LA FUNCIÓN DE GRIEWANK 20

También para esta instancia S.G.O.-Gradiente se ha revelado más eficiente que la anterior hibridación porque hemos obtenido mejores promedios aunque para ello hemos incrementado los valores de *n* y *n_iter* como se ve en la figura 5.17.

A pesar de tratarse de una función multimodal, los asteroides no han quedado atrapados y con Gradiente han llegado al óptimo en más ocasiones.

Aunque el valor del promedio es 0.0032 **sí cumple nuestro segundo criterio de alcance del óptimo** ya que en 8 de las 10 ejecuciones hemos obtenido el mínimo con un error menor que 10^{-6}.

FUNCION GRIEWANK(20)-S.G.O.-GRADIENTE

EJECUCION

N_EJEC	FOPT-PARC	FOPT	TIEMP-GRAVI	TIEMP-GRAD	E.F.O TOTAL	TIEMP-TOTAL
1	7.928179E+01	7.204237E-13	188.10	0.26	20528515	188.36
2	1.092393E+02	2.705725E-12	186.97	0.30	20532663	187.27
3	9.990740E+01	1.722629E-02	188.28	0.29	20532535	188.58
4	1.089517E+02	1.125988E-12	187.65	0.28	20531197	187.93
5	1.020217E+02	7.674861E-12	188.24	0.31	20533959	188.55
6	1.166780E+02	1.240119E-12	188.01	0.31	20534035	188.32
7	7.256308E+01	2.652323E-12	188.43	0.24	20526271	188.67
8	1.172913E+02	1.477978E-02	187.27	0.31	20533467	187.58
9	1.296667E+02	5.014766E-11	189.35	0.25	20527883	189.61
10	1.092235E+02	2.324574E-11	189.70	0.29	20531221	189.98
PROMEDIO	1.044824E+02	3.200607E-03				
DESVIACIÓN TÍPICA	1.729812E+01	6.772068E-03		FMAX-FMIN	1.722629E-02	
VALOR MEJOR	7.256308E+01	7.204237E-13				
VALOR PEOR	1.296667E+02	1.722629E-02				

PARÁMETROS

ALFA	0.005
BETA	0.4
G	1.00E+05
N	5.00E+02
N_ITER	1.00E+03
N_GRAD	1.00E+04
DIM	20

Figura 5.17. Hoja Excel de RESULTADOS GRIEWANK 20 S.G.O.-GRADIENTE.

5.5. FUNCIONES DE DIXON & PRICE 25 y DE ACKLEY 30

No hemos conseguido el óptimo para ninguna de estas instancias. Los resultados, que se encuentran en el CD adjunto, no han sido mejores que los obtenidos con la anterior hibridación.

Resulta particularmente curioso el caso de la función de DIXON & PRICE 25. El algoritmo Gradiente ha guiado a **todos** los asteroides al punto de silla. Mostramos los resultados en la figura 5.18.

FUNCION DIXON & PRICE(25)-S.G.O.-GRADIENTE

EJECUCION

N_EJEC	FOPT-PARC	FOPT	TIEMP-GRAVI	TIEMP-GRAD	E.F.O TOTAL	TIEMP-TOTAL
1	7.582696E+05	6.666667E-01	0.44	0.23	79769	0.67
2	7.314388E+05	6.666667E-01	0.42	0.17	71707	0.59
3	6.359104E+05	6.666667E-01	0.42	0.38	97681	0.80
4	5.098672E+05	6.666667E-01	0.42	1.07	182605	1.48
5	6.190331E+05	6.666667E-01	0.42	0.09	62363	0.51
6	7.397390E+05	6.666667E-01	0.42	0.22	77677	0.63
7	5.176202E+05	6.666667E-01	0.42	0.13	66953	0.55
8	4.871359E+05	6.666667E-01	0.42	0.07	59753	0.49
9	5.332076E+05	6.666667E-01	0.42	0.60	124661	1.02
10	6.060850E+05	6.666667E-01	0.42	0.18	73285	0.60
PROMEDIO	6.138307E+05	6.666667E-01				
DESVIACIÓN TÍPICA	1.020557E+05	4.831992E-09		FMAX-FMIN	1.292930E-08	
VALOR MEJOR	4.871359E+05	6.666667E-01				
VALOR PEOR	7.582696E+05	6.666667E-01				

PARÁMETROS

ALFA	0.005
BETA	0.4
G	1.00E+05
N	1.00E+02
N_ITER	1.00E+01
N_GRAD	1.00E+04
DIM	25

Figura 5.18. Hoja Excel de RESULTADOS DIXON & PRICE 25 S.G.O.-GRADIENTE.

5.6. CONCLUSIONES

En la figura 5.19 podemos observar los resultados obtenidos para la hibridación S.G.O.-Gradiente en las siete instancias estudiadas. **Hemos obtenido el mínimo en dos de ellas.** En las otras cinco instancias los resultados han sido similares a los obtenidos en la sección anterior.

RESULTADOS GENERALES S.G.O.-GRADIENTE

Nº	FUNCIÓN	DIM	ÓPTIMO DE S.G.O.	MEJOR ÓPTIMO	PROMEDIO	TIEMPO (SEG.)	E.F.O.
1	SCHWEFEL 6	6	9,228355E+02	-4,241929E-07	2,487241E+02	1,40	66031
2	ROSENBROCK 10	10	4,457340E+04	8,814255E-08	6,792829E-06	10,10	627001
3	RASTRIGIN 10	10	4,788447E+01	9,949588E+00	2,139164E+01	8,88	210461
4	RASTRIGIN 20	20	1,732185E+02	5,273274E+01	7,163683E+01	160,57	4101029
5	GRIEWANK 20	20	7,928179E+01	7,204237E-13	3,200607E-03	188,36	20528515
6	DIXON & PRICE 25	25	6,060850E+05	6,666667E-01	6,666667E-01	0,60	73285
7	ACKLEY 30	30	1,776493E+01	1,697037E+01	6,835115E+01	2,41	62463

Figura 5.19. Hoja Excel de RESUMEN DE RESULTADOS S.G.O.-GRADIENTE.

En general, el metaheurístico resultante de la hibridación S.G.O.-Gradiente es más rápido **en nuestro planteamiento** que S.G.O.-Nelder-Mead ya que solo es necesario ejecutar S.G.O. una vez para ejecutar tras él Gradiente, mientras que en el caso anterior, esto no es posible, repetimos, en nuestro planteamiento.

Además, el algoritmo del Gradiente se ha presentado más eficaz guiando a los asteroides hasta “su mínimo posible”, sea éste local o global, -tal como hace una pelota que se deja libremente en la ladera de una montaña-, habiendo obtenido mejores promedios en los casos en que S.G.O.-Nelder-Mead alcanzaba el mínimo con mayor dificultad. Pero, por ello, adolece con mayor intensidad del mismo defecto que la hibridación anterior: no sólo no ha podido evitar que los asteroides queden atrapados en óptimos locales, sino que para las funciones “topológicamente malas” los resultados obtenidos han sido peores que hibridando S.G.O. con Nelder-Mead.

6. CONCLUSIONES GENERALES

Los métodos locales han aportado a S.G.O. mayor eficiencia, porque nos han permitido rebajar el número, *n,* de asteroides y *n_iter,* de iteraciones, y mayor eficacia a la vista de los promedios obtenidos en las funciones de topología "buena" y en una parte importante de las funciones de topología "mala" de dimensiones medias y bajas. Por ello creemos que hemos planteado unas metaheurísticas muy recomendables con las que **hemos obtenido éxito en 35 de las 40 funciones planteadas al inicio del presente capítulo.**

Sin embargo, las alianzas de S.G.O. con algoritmos locales no han mejorado significativamente la situación al enfrentarse a las cinco funciones multimodales de "topologías complicadas" en dimensiones altas. Obviamente, si los asteroides ya habían quedado atrapados en S.G.O. las hibridaciones locales no han sido capaces de guiarlos al óptimo global.

En la figura 5.20 exponemos una hoja de Excel recogiendo los tiempos de ejecución de ambas hibridaciones. Solo disponemos de datos para 7 de las 40 instancias estudiadas, ya que solo hemos probado la hibridación S.G.O.-Gradiente con las funciones con las que no cumplimos nuestro criterio de óptimo con la hibridación S.G.O.-Nelder-Mead.

COMPARATIVA TIEMPOS S.G.O.-NELME Y S.G.O.-GRADIENTE

Nº	FUNCIÓN	DIM.	ÓPTIMO S.G.O.-NELME	TIEMPO (SEG.)	E.F.O.	ÓPTIMO S.G.O.-GRAD	TIEMPO (SEG.)	E.F.O.	
1	SCHWEFEL 6	6	-4.346025E-07	315.52	28149791	-4.241929E-07	1.40	66031	(*)
2	ROSENBROCK 10	10	3.525222E-29	7.32	358577	8.814255E-08	10.10	627001	(*)
3	RASTRIGIN 10	10	1.989918E+00	10.63	360083	9.949588E+00	8.88	210461	
4	RASTRIGIN 20	20	1.094455E+01	351.10	12349950	5.273274E+01	160.57	4101029	
5	GRIEWANK 20	20	2.109424E-15	273.58	12350321	7.204237E-13	188.36	20528515	(*)
6	DIXON & PRICE 25	25	6.666667E-01	372.13	15131912	6.666667E-01	0.60	73285	
7	ACKLEY 30	30	3.786261E+00	66.60	2212993	1.697037E+01	1.21	62463	

(*) NO OBTUVIMOS BUENOS PROMEDIOS CON S.G.O.-NELME

Figura 5.20. Hoja Excel COMPARATIVA DE TIEMPOS.

Las disparidades en tiempo se deben al uso de distintos equipos para realizar las ejecuciones, por ello hemos incluido los datos de número de evaluaciones de la función objetivo.

La figura 5.20 no pretende realizar un estudio comparativo de tiempos entre ambas hibridaciones, ya que los valores dados a los parámetros no han sido los mismos en todos los casos y, por tanto, no pueden compararse.

Analizando las abscisas de los óptimos obtenidos hemos llegado a la conclusión de que. en general, en S.G.O. los asteroides no han quedado atrapados en óptimos locales sino que **son solo algunas de las abscisas del óptimo las que sí quedan atrapadas**. Además, aumentar los valores de *n* y *n_iter* no ha mejorado los resultados en estos casos sino que ha provocado que el número de abscisas fallidas sea superior, ver CD anexo con la función de SCHWEFEL por ejemplo.

Parece, por ello, conveniente, **a la hora de afrontar la tarea de optimizar las cinco funciones del presente capítulo en las que todavía no hemos hallado el óptimo,** mantener los valores de *n* y *n_iter* en rangos similares a los usados en el presente capítulo e intentar resolver el problema buscando algún otro heurístico con el que hibridar S.G.O..

Para generar las nuevas hibridaciones nos hemos basado en dos ideas completamente distintas entre sí, y diferentes a su vez de la idea de hibridación con algoritmos locales, usada en el presente capítulo, que han dado lugar a sendas vías:

- En la primera vía, **que ha originado la innovación y el desarrollo del algoritmo que hemos denominado SEGMENTACIÓN, SG.**, hemos considerado **técnicas de concentración de la región factible** para estudiar la posibilidad de fraccionarla realizando exploraciones en cada subregión sin elevar de forma exponencial el coste computacional.

- En la segunda vía hemos considerado **técnicas de intensificación en entornos de soluciones buenas.** Tomando la definición de "entornos" en base a dos ideas diferentes: la primera se basa en la búsqueda de mejores soluciones **en las rectas que unen las soluciones buenas** y lleva a la implementación del heurístico V.S.O. (Formato 2013)" que es, en realidad, una versión muy simplificada de Path Relinking y de S.S. La segunda vía se basa en la innovadora idea, basada en la experimentación anterior, que consiste en que "sólo algunas abscisas, no siempre las mismas, han quedado atrapadas en S.G.O.". Tomando así como entorno del óptimo **el conjunto de soluciones que comparten con él un número alto de coordenadas**, hemos llegado a la génesis y desarrollo del algoritmo, que hemos bautizado como AGUJERO DE GUSANO, A.G.

Desarrollaremos la hibridación de S.G.O. con SG. en el próximo capítulo. La hibridación con V.S.O. y A.G. será vista con detalle en el capítulo 7.

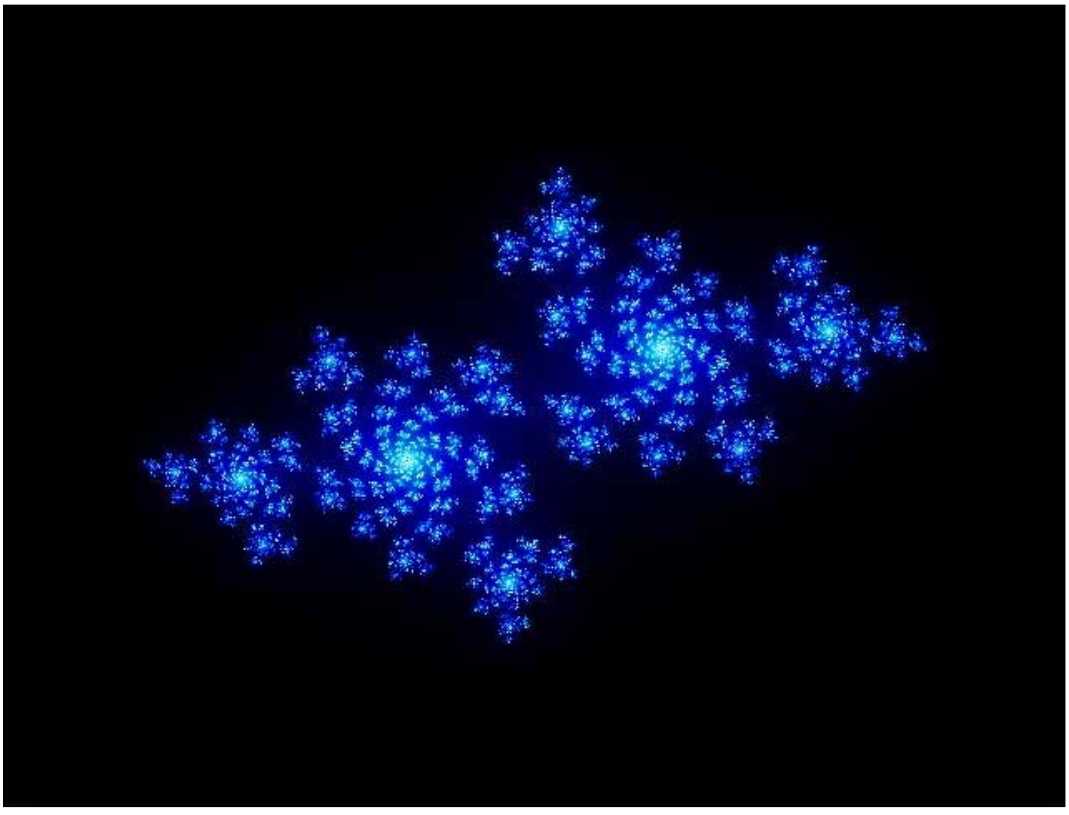
Nuestro objetivo en ambos capítulos será **encontrar el óptimo de las cinco instancias** que las hibridaciones locales no han sido capaces de conseguir.

"Divide y vencerás."

Julio César (100 A.C.-44 A.C.)

CAPÍTULO 6

HIBRIDACIÓN CON EL ALGORITMO SEGMENTACIÓN



1. INTRODUCCIÓN

A lo largo de todo el trabajo desarrollado hemos observado que la topología de la función objetivo es, sin duda, un factor fundamental a tener en cuenta a la hora de enfrentarse a una instancia de optimización continua, pero **no lo es menos la región factible**. Pensemos en la evidencia de que todo óptimo local se convierte en global en un entorno suficientemente pequeño y todo óptimo global queda aislado como óptimo local en una región factible suficientemente pequeña.

Ahondando en esta idea, un algoritmo basado en técnicas estadísticas de diseño de experimentos factoriales, m^{dim}, que dividiera cada intervalo de la región factible en *m subintervalos* de longitud suficientemente pequeña en cada variable, y buscara el óptimo en cada uno de los *dim-paralelepípedos* correspondientes, diseño factorial completo, tendría éxito garantizado pero, como es bien sabido, su crecimiento computacional es exponencial respecto a la dimensión de la función. Tal algoritmo es inoperante en la práctica.

La teoría de diseño de experimentos factoriales fraccionales de G. Taguchi, (Roy 1990) plantea una forma de resolver el problema computacional, reduciendo notable y estratégicamente el número de *dim-paralelepípedos* en los que ejecutar el algoritmo de búsqueda. De esta forma, se revela como una eficaz herramienta en la consecución del óptimo global y es implementada con éxito en el artículo de Laguna y Martí (2005) con Scatter Search.

Sin embargo, existen otras técnicas derivadas de la teoría de diseño de experimentos que también pueden resolver el problema computacional planteado: **son las técnicas de experimentación secuencial**, i.e. planificación de nuevos experimentos a partir de los resultados obtenidos en experimentos previos.

Basándonos en estas técnicas, y en nuestra amplia experiencia con S.G.O. en dimensiones bajas, en las que los resultados han sido muy buenos, hemos diseñado el heurístico SEGMENTACIÓN, SG., un algoritmo de concentración en el que realizamos una nueva elección de *dim-paralelepípedos* sobre los que ejecutar S.G.O.

En la lista de *funciones test* con las que hemos probado nuestro algoritmo se da la circunstancia de que hemos podido experimentar todas las instancias en dimensiones muy bajas en las que las hibridaciones S.G.O-Nelder Mead y S.G.O.-Gradiente se han comportado extraordinariamente bien con coste computacional pequeño. Ello nos permite diseñar experimentos en los que se explore con especial interés los *dim-paralelepípedos* donde se ha alcanzado el óptimo en dimensiones pequeñas, sin olvidar una búsqueda general en toda la región factible.

De este modo, en el heurístico **SEGMENTACIÓN,** planteamos **subdividir la región factible en un número fijo de *dim-paralelepípedos* apropiados,** que **no aumente con la**

dimensión de la función y en los que S.G.O. pueda incrementar su eficacia mediante técnicas de concentración.

El coste computacional va a ser 3*m* superior al de S.G.O., concretamente en las ejecuciones realizadas ha sido 15 veces superior en cualquier dimensión. Lo cual dista mucho de ser un incremento exponencial.

Sólo hemos probado S.G.O.-SEGMENTACIÓN para las cinco instancias con las que fracasamos en el capítulo anterior: las funciones de SCHWEFEL 6, de RASTRIGIN 10, de RASTRIGIN 20, de DIXON & PRICE 25 y de ACKLEY 30.

Recordamos a continuación nuestra definición de optimalidad:

Consideraremos que hemos alcanzado el óptimo si el error de alguno de los resultados obtenidos es menor que $3 \cdot 10^{-6}$ y el error de alguno de los promedios es menor que $3 \cdot 10^{-3}$.

O bien:

Consideraremos que hemos alcanzado el óptimo si el error de alguno de los resultados obtenidos es menor que $3 \cdot 10^{-6}$ y en alguna de las tandas de 10 ejecuciones hemos obtenido errores menores que $3 \cdot 10^{-6}$ en, al menos, 7 de ellas.

Hemos resaltado en amarillo los resultados en los que hemos alcanzado el óptimo según nuestro criterio en cada hoja Excel de resultados.

2. EL METAHEURÍSTICO S.G.O.- SEGMENTACIÓN

2.1. IMPLEMENTACIÓN DEL ALGORITMO

Las instancias planteadas como *benchmark functions* en el presente trabajo tienen definido en todos los casos el dominio de optimización de la forma

$$[r_1, r_2] \text{x} [r_1, r_2] \text{x} ... x [r_1, r_2] = [r_1, r_2]^{\dim}$$

y así es como hemos implementado S.G.O. Sin embargo, supone una fuerte restricción a la hora de diseñar las subregiones de la región factible en las que vamos a explorar, ya que no permite que cada coordenada varíe en un intervalo diferente. No obstante, hemos implementado SG. respetando esta condición. En un caso general podría diseñarse dejando que cada abscisa variara con más libertad.

Puesto que queremos crear subregiones factibles más pequeñas que la original pero sin olvidar la región completa, hemos decidido formar los *m dim-paralelepípedos* de forma que alguno de ellos en algún momento coincida con la totalidad de la región factible.

Enumeremos las fases de la hibridación S.G.O.-SG.

- **Fase 1**

En 1er lugar consideramos una partición fija de cada intervalo formada por *m+1* puntos equiespaciados, lo que da lugar a *m* subintervalos de longitud $h = \frac{r_2 - r_1}{m}$, de la forma

$$[r_1, r_1 + h], [r_1 + h, r_1 + 2h], ..., [r_1 + (m-1)h, r_2]$$

A continuación ejecutaremos S.G.O. en cada una de las subregiones factibles de la forma

$$R_i = [r_1, r_1 + ih]^{\dim} \; i = 1, ..., m.$$

Lo que arroja un total de *m* ejecuciones.

Como se muestra en la figura 6.1 para el caso *dim=2*, *m=3*, los *m* rectángulos van aumentando desde la esquina inferior izquierda hasta abarcar la totalidad de la región factible.

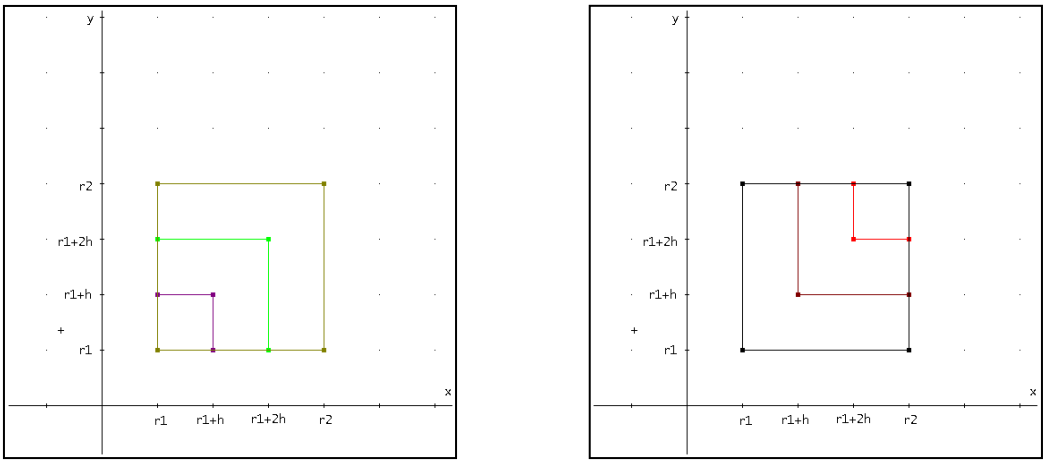


Figura 6.1. Figura de m-paralelepípedos expandidos.

- **Fase 2**

En 2º lugar ejecutaremos S.G.O. en cada una de las subregiones factibles de la forma

$$R_i = \left[r_1 + ih, r_2\right]^{\dim} \quad i = 0,...,m-1.$$

Lo que nuevamente arroja un total de *m* ejecuciones.

Como se muestra en la figura 6.1 para el caso *dim=2* y *m=3*, los *m* rectángulos van aumentando desde la esquina superior derecha hasta abarcar la totalidad de la región factible.

- **Fase 3**

En último lugar, dada la particular idiosincrasia de las funciones con las que trataremos en este apartado, ejecutaremos S.G.O. en cada una de las *m* subregiones factibles de la forma

$$R_i = \left[r_1 + ih, r_1 + (i+1)h\right]^{\dim} \quad i = 0,...,m-1.$$

De nuevo se trata de *m* ejecuciones de S.G.O. en total.

Como se muestra en la figura 6.2 para el caso *dim=2* y *m=3*, los *m* rectángulos tienen en esta ocasión el mismo tamaño y se desplazan a lo largo de la diagonal principal del paralelepípedo que forma la región factible.

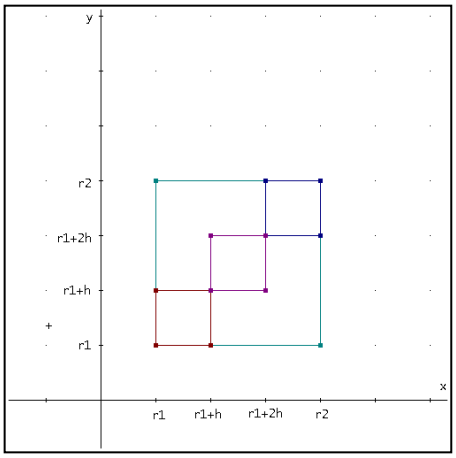


Figura 6.2. m-paralelepípedos de la diagonal

- **Fase Final**

A continuación se escoge el mejor de los resultados obtenidos en las *3m* iteraciones de S.G.O. y se ejecuta sobre él el algoritmo Gradiente para mejorarlo tanto como sea posible.

2.2. GENERALIZACIONES DE S.G.O.-SG.

La implementación anterior ha sido específica para el grupo de funciones que nos ocupa, pero puede generalizarse de múltiples formas: expandir, por ejemplo, los *dim-paralelepípedos* desde el central, formar rectángulos en la diagonal secundaria, barrer determinada columna o fila de la región factible, etc. La elección depende lógicamente de los experimentos previos realizados con cada instancia concreta. También puede realizarse un número mayor o menor de *3m* experimentos, aunque sí creemos necesario que alguno de los experimentos sea expansivo llegando a la totalidad de la región factible.

Obviamente, el número de experimentos realizados no va a variar en función de la dimensión de la instancia correspondiente.

3. RESULTADOS DE S.G.O.-SG.

Para las pruebas realizadas hemos elegido *m=5*, es decir, hemos dividido cada intervalo en 5 subintervalos del mismo tamaño, lo que ha supuesto ejecutar S.G.O. 15 veces antes de obtener el resultado final. Teniendo en cuenta que nos enfrentamos a funciones de dimensión alta y muy alta el coste computacional también lo es, pero permanece inferior al que supondría considerar valores muy altos para *n* o *n_iter.*

Detallaremos los valores en cada caso concreto.

3.1. FUNCIÓN DE SCHWEFEL 6

Los estudios iniciales en dimensiones 1 y 2, en las que sí obtenemos el óptimo, nos indican que se encuentra en las abscisas aproximadas de la forma $x_i \approx 421$ con lo que SG. buscará los puntos de abscisas iguales con los 3 tipos de rectángulos diseñados.

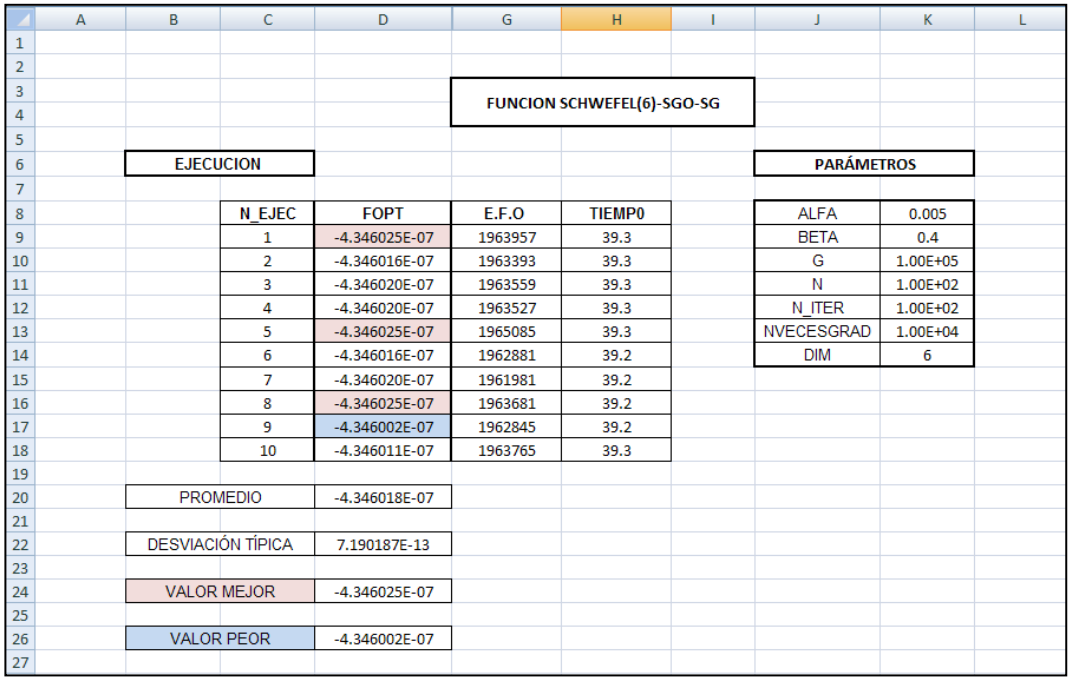
Hemos probado la hibridación en las condiciones comentadas previamente con *n=100*, *n_iter=100* y *nvecesgrad=10000*. Además, como en el capítulo anterior, hemos preparado el algoritmo gradiente para que pare una vez que compruebe que no puede mejorar el resultado, así las evaluaciones de la función objetivo varían en cada caso.

En esta ocasión la región factible era $[-500,500]^6$, por lo tanto los 5 subintervalos en cada una de las 6 variables serán

$$[-500,-300],[-300,-100],[-100,100],[100,300]\ y\ [300,500]\ .$$

Todos ellos de longitud $\frac{500+500}{5}=\frac{1000}{5}=200$

Como en casos anteriores recogemos los resultados en una hoja de Excel que mostramos en la figura. 6.3.



FUNCION SCHWEFEL(6)-SGO-SG

EJECUCION

N_EJEC	FOPT	E.F.O	TIEMP0
1	-4.346025E-07	1963957	39.3
2	-4.346016E-07	1963393	39.3
3	-4.346020E-07	1963559	39.3
4	-4.346020E-07	1963527	39.3
5	-4.346025E-07	1965085	39.3
6	-4.346016E-07	1962881	39.2
7	-4.346020E-07	1961981	39.2
8	-4.346025E-07	1963681	39.2
9	-4.346002E-07	1962845	39.2
10	-4.346011E-07	1963765	39.3

PROMEDIO	-4.346018E-07
DESVIACIÓN TÍPICA	7.190187E-13
VALOR MEJOR	-4.346025E-07
VALOR PEOR	-4.346002E-07

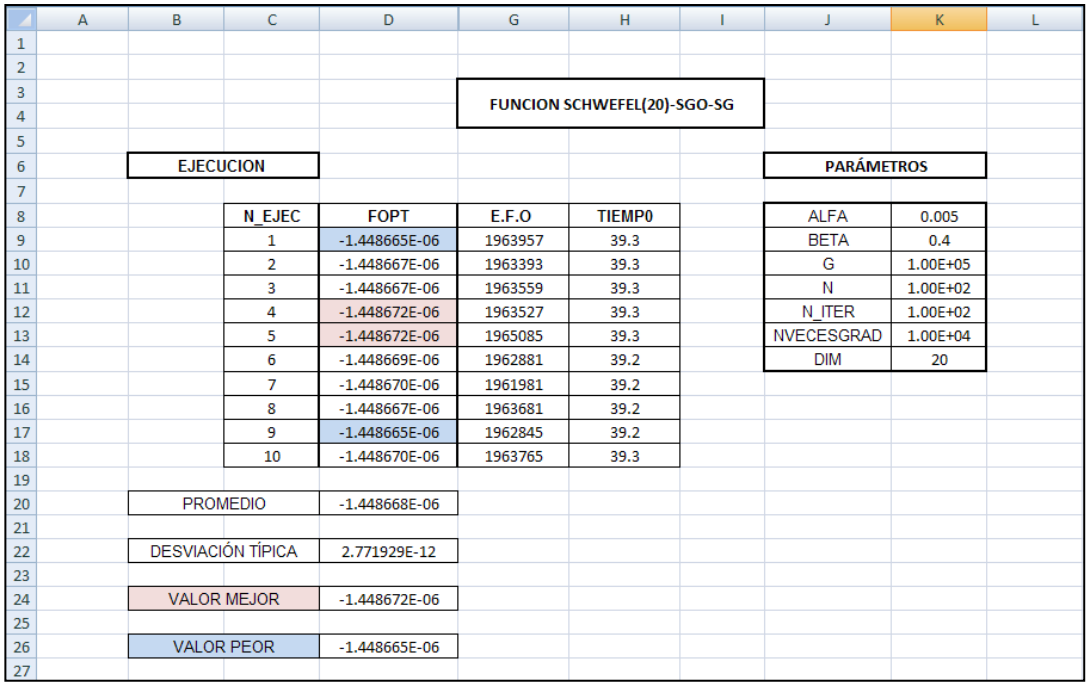
PARÁMETROS

ALFA	0.005
BETA	0.4
G	1.00E+05
N	1.00E+02
N_ITER	1.00E+02
NVECESGRAD	1.00E+04
DIM	6

Figura 6.3. Resultados S.G.O.-SG. con la función de SCHWEFEL 6.

A la vista de los resultados **hemos obtenido el óptimo atendiendo a nuestro criterio** con tiempos inferiores a los utilizados anteriormente con esta instancia, ver capítulo 5 hibridación S.G.O.-Nelder-Mead. El resultado es tan bueno que hemos probado el algoritmo en mayores dimensiones para comprobar su alcance.

Presentamos los **excelentes resultados** en dimensión 20, figura 6.4.



FUNCION SCHWEFEL(20)-SGO-SG

EJECUCION

N_EJEC	FOPT	E.F.O	TIEMP0
1	-1.448665E-06	1963957	39.3
2	-1.448667E-06	1963393	39.3
3	-1.448667E-06	1963559	39.3
4	-1.448672E-06	1963527	39.3
5	-1.448672E-06	1965085	39.3
6	-1.448669E-06	1962881	39.2
7	-1.448670E-06	1961981	39.2
8	-1.448667E-06	1963681	39.2
9	-1.448665E-06	1962845	39.2
10	-1.448670E-06	1963765	39.3

PROMEDIO	-1.448668E-06
DESVIACIÓN TÍPICA	2.771929E-12
VALOR MEJOR	-1.448672E-06
VALOR PEOR	-1.448665E-06

PARÁMETROS

ALFA	0.005
BETA	0.4
G	1.00E+05
N	1.00E+02
N_ITER	1.00E+02
NVECESGRAD	1.00E+04
DIM	20

Figura 6.4. Resultados S.G.O.-SG. con la función de SCHWEFEL 20.

En dimensión 50 t**ambién hemos obtenido el óptimo,** figura 6.5.

FUNCION SCHWEFEL(50)-SGO-SG

EJECUCION

N_EJEC	FOPT	E.F.O	TIEMPO
1	-3.621655E-06	15763707	365.13
2	-3.621655E-06	16764767	387.79
3	-3.621648E-06	15756227	365.39
4	-3.621655E-06	15759747	365.65
5	-3.621641E-06	15734165	364.90
6	-3.621648E-06	16752863	387.80
7	-3.621655E-06	15712905	364.32
8	-3.621630E-06	15742589	365.14
9	-3.621670E-06	15723791	364.66
10	-3.621655E-06	15742033	365.13

PROMEDIO	-3.621651E-06
DESVIACIÓN TÍPICA	1.063413E-11
VALOR MEJOR	-3.621670E-06
VALOR PEOR	-3.621630E-06

PARÁMETROS

ALFA	0.005
BETA	0.4
G	1.00E+05
N	1.00E+02
N_ITER	1.00E+02
NVECESGRAD	1.00E+04
DIM	50

Figura 6.5. Resultados S.G.O.-SG. con la función de SCHWEFEL 50.

En dimensión 80, la última verificada, **hemos obtenido nuevamente el óptimo con 8 resultados excelentes**, figura 6.6.

FUNCION SCHWEFEL(80)-SGO-SG

EJECUCION

N_EJEC	FOPT	E.F.O	TIEMPO
1	-5.794594E-06	242690917	2909.80
2	-5.794587E-06	242733045	2910.75
3	2.171397E+02	242749043	2910.74
4	-5.794587E-06	242709877	2910.71
5	-5.794398E-06	410663949	4852.09
6	-5.794573E-06	242705779	2910.76
7	-5.794602E-06	242737869	2910.72
8	2.171397E+02	242756783	2910.80
9	-5.794594E-06	242748769	2910.75
10	-5.794602E-06	410674767	4910.69

PROMEDIO	4.342793E+01
DESVIACIÓN TÍPICA	9.155412E+01
VALOR MEJOR	-5.794602E-06
VALOR PEOR	2.171397E+02

PARÁMETROS

ALFA	0.005
BETA	0.4
G	1.00E+05
N	1.00E+03
N_ITER	1.00E+02
NVECESGRAD	1.00E+06
DIM	80

Figura 6.6. Resultados de S.G.O.-SG. con la función SCHWEFEL 80.

3.2. FUNCIÓN DE RASTRIGIN 10

Los datos obtenidos en dimensiones inferiores nos corroboran que el óptimo se alcanza para todas las abscisas nulas. El diseño del S.G.O.-SG. por el que nos hemos decantado también parece indicado para esta instancia.

Ejecutamos el algoritmo para los mismos valores que en el caso anterior.

En la hoja de Excel podemos ver que en este caso los resultados han sido buenos ya que **hemos hallado el óptimo en una ocasión** pero los promedios no llegan a cumplir nuestro criterio de optimalidad, figura 6.7.

FUNCION RASTRIGIN(10)-SGO-SG

EJECUCION

N_EJEC	FOPT	E.F.O	TIEMP0
1	3.979836E+00	3171389	50.37
2	1.989918E+00	3171785	50.08
3	1.989918E+00	3171035	50.04
4	1.342880E-08	3170457	50.06
5	2.984877E+00	3172245	50.03
6	1.989918E+00	3169275	50.01
7	2.984877E+00	3170257	50.04
8	3.979836E+00	3172481	50.09
9	1.989918E+00	3173143	50.09
10	1.989918E+00	3170935	50.07

PROMEDIO	2.387902E+00
DESVIACIÓN TÍPICA	1.167871E+00
VALOR MEJOR	1.342880E-08
VALOR PEOR	3.979836E+00

PARÁMETROS

ALFA	0.005
BETA	0.4
G	1.00E+05
N	1.00E+02
N_ITER	1.00E+02
NVECESGRAD	1.00E+04
DIM	10

Figura 6.7. Resultados S.G.O.-SG. con la función de RASTRIGIN10.

3.3. FUNCIÓN DE RASTRIGIN 20

Si ya en dimensión 10 S.G.O.- SG. no se comportaba satisfactoriamente, pues no hemos obtenido un buen promedio, podemos esperar que las cosas empeoren en dimensión 20. Así es efectivamente.

Veamos la hoja Excel de resultados, figura 6.8.

FUNCION RASTRIGIN(20)-SGO-SG

EJECUCION

N_EJEC	FOPT	E.F.O	TIEMP0
1	6.964713E+00	6189467	101.7
2	3.979836E+00	6191723	101.4
3	7.959672E+00	6203273	101.6
4	4.974795E+00	6196699	101.5
5	7.959672E+00	6190091	101.3
6	5.969754E+00	6192307	101.5
7	4.974795E+00	6198267	101.6
8	5.969754E+00	6192369	101.4
9	4.974795E+00	6195533	101.4
10	7.959672E+00	6193985	101.4

PROMEDIO	6.168746E+00
DESVIACIÓN TÍPICA	1.468291E+00
VALOR MEJOR	3.979836E+00
VALOR PEOR	7.959672E+00

PARÁMETROS

ALFA	0.005
BETA	0.4
G	1.00E+05
N	1.00E+02
N_ITER	1.00E+02
NVECESGRAD	1.00E+04
DIM	20

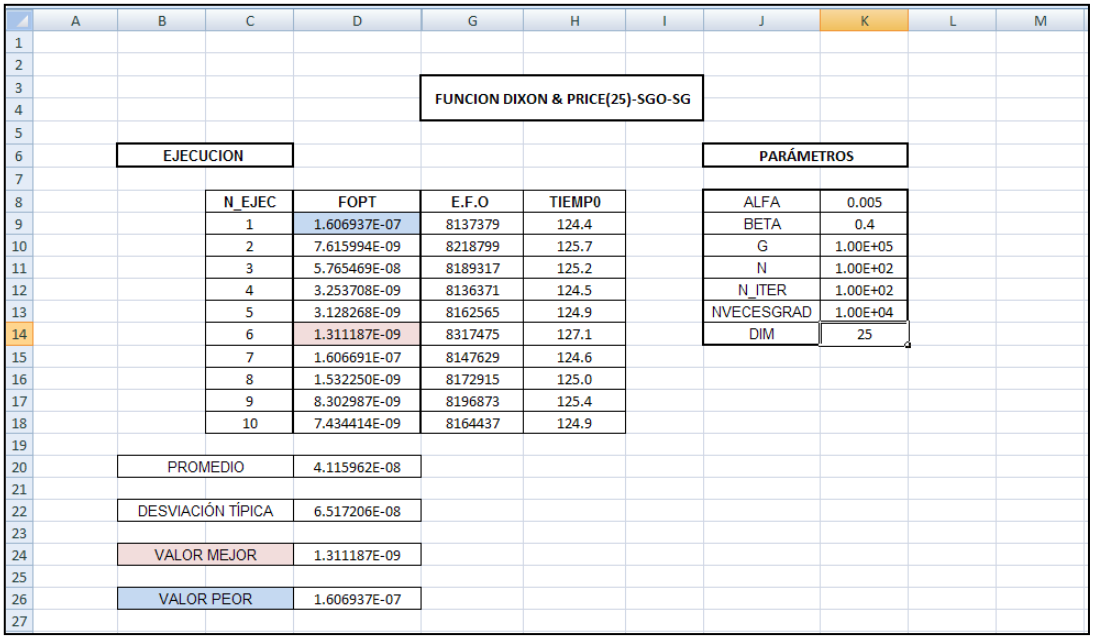
Figura 6.8. Resultados S.G.O.-SG con la función de RASTRIGIN 20.

3.4. FUNCIÓN DE DIXON & PRICE 25

Aunque las abscisas en las que esta función alcanza su mínimo absoluto son diferentes entre sí, hemos demostrado en el capítulo anterior que todas ellas varían entre 0 y 1. También lo hemos comprobado empíricamente en dimensiones inferiores, por ello, la elección de las subregiones parece también apropiada para esta instancia y, a la vista de los resultados, lo ha sido ya que **ha cumplido holgadamente nuestro criterio de alcance del óptimo**.

La región factible para esta instancia es $[-10,10]^{25}$, por lo tanto las subregiones serán $[-10,-6],[-6,-2],[-2,2],[2,6],[6,10]$.

En ningún caso el hibridado S.G.O.-SG. ha dado el punto de silla, $x_i = 0.\widehat{6} \quad \forall i = 1,...,25$, ver capítulo 5, como óptimo global, figura 6.9.



FUNCION DIXON & PRICE(25)-SGO-SG

EJECUCION			
N_EJEC	FOPT	E.F.O	TIEMP0
1	1.606937E-07	8137379	124.4
2	7.615994E-09	8218799	125.7
3	5.765469E-08	8189317	125.2
4	3.253708E-09	8136371	124.5
5	3.128268E-09	8162565	124.9
6	1.311187E-09	8317475	127.1
7	1.606691E-07	8147629	124.6
8	1.532250E-09	8172915	125.0
9	8.302987E-09	8196873	125.4
10	7.434414E-09	8164437	124.9

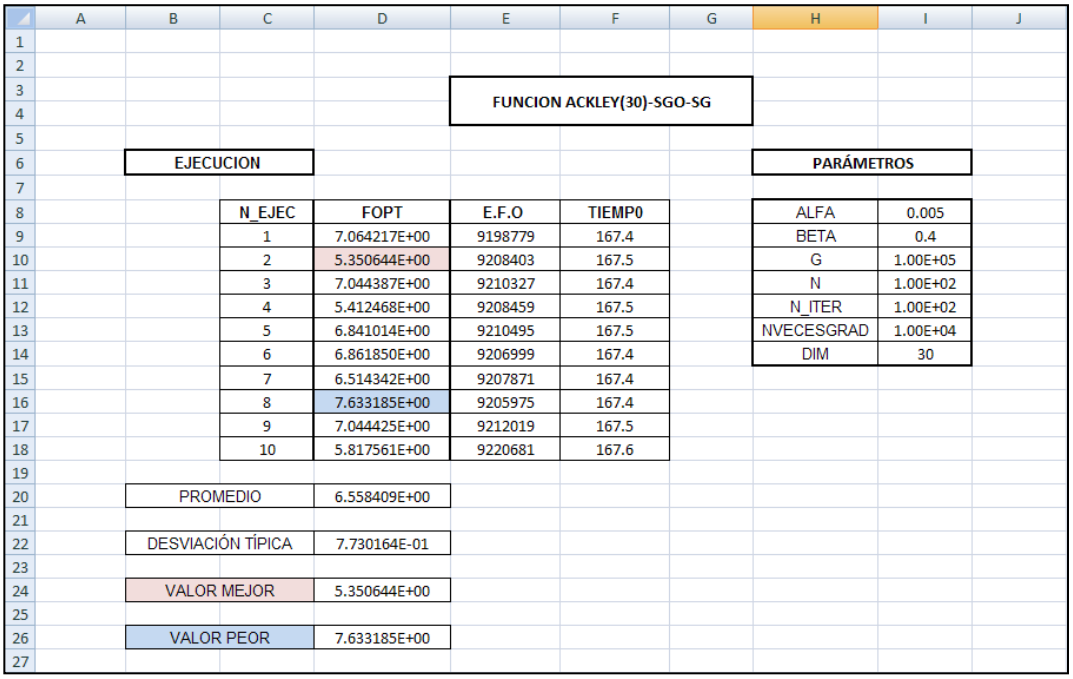
PROMEDIO	4.115962E-08
DESVIACIÓN TÍPICA	6.517206E-08
VALOR MEJOR	1.311187E-09
VALOR PEOR	1.606937E-07

PARÁMETROS	
ALFA	0.005
BETA	0.4
G	1.00E+05
N	1.00E+02
N_ITER	1.00E+02
NVECESGRAD	1.00E+04
DIM	25

Figura 6.9. Resultados S.G.O.-SG con la función de DIXON & PRICE 25.

3.5. FUNCIÓN DE ACKLEY 30

Como se ve en la figura 6.10 no hemos alcanzado el óptimo ni una sola vez. Además, los resultados son peores que los obtenido con la hibridación S.G.O.-Nelder-Mead, ver capítulo 5. Esta función alcanza su mínimo, 0, en todas las abscisas nulas. Por ello, la elección de los rectángulos también parecía a priori apropiada aunque a posteriori no lo ha sido.



FUNCION ACKLEY(30)-SGO-SG

EJECUCION			
N_EJEC	FOPT	E.F.O	TIEMP0
1	7.064217E+00	9198779	167.4
2	5.350644E+00	9208403	167.5
3	7.044387E+00	9210327	167.4
4	5.412468E+00	9208459	167.5
5	6.841014E+00	9210495	167.5
6	6.861850E+00	9206999	167.4
7	6.514342E+00	9207871	167.4
8	7.633185E+00	9205975	167.4
9	7.044425E+00	9212019	167.5
10	5.817561E+00	9220681	167.6

PROMEDIO	6.558409E+00
DESVIACIÓN TÍPICA	7.730164E-01
VALOR MEJOR	5.350644E+00
VALOR PEOR	7.633185E+00

PARÁMETROS	
ALFA	0.005
BETA	0.4
G	1.00E+05
N	1.00E+02
N_ITER	1.00E+02
NVECESGRAD	1.00E+04
DIM	30

Figura 6.10. Resultados S.G.O.-SG con la función de ACKLEY 30.

3.6. RESUMEN DE RESULTADOS

Hemos probado el heurístico S.G.O.–SG. con cinco instancias y hemos obtenido el óptimo, según nuestro criterio, en dos de ellas: de SCHWEFEL 6 y de DIXON & PRICE 25.

En el caso de la función de RASTRIGIN 10 sí hemos sido capaces de alcanzar el óptimo con un error muy aceptable, $1.34 \cdot 10^{-8}$, en una de las diez ocasiones. pero es, a nuestro juicio, insuficiente. Para dos de ellas, las funciones de RASTRIGIN 20 y de ACKLEY 30, no hemos obtenido el óptimo en ninguna ocasión.

Veamos la hoja Excel de RESULTADOS GENERALES S.G.O.-SG., figura 6.11.

RESULTADOS S.G.O.- SG.

Nº	FUNCIÓN	ÓPTIMO DE S.G.O.	MEJOR ÓPTIMO	PROMEDIO	E.F.O.	TIEMPO
1	SCHWEFEL-6	9.870026E+01	-4.346025E-07	-4.346018E-07	1963957	39.33
2	RASTRIGIN-10	1.790923E+01	1.342880E-08	2.387902E+00	3170457	50.06
3	RASTRIGIN-20	6.069238E+01	3.979836E+00	6.168746E+00	6191723	101.38
4	DIXON & PRICE-25	6.666667E-01	1.311187E-09	4.115962E-08	8317475	127.11
5	ACKLEY-30	1.604187E+01	5.350644E+00	6.558409E+00	9208403	167.46

Figura 6.11. RESULTADOS GENERALES S.G.O. –SG.

4. CONCLUSIONES

Al introducir la experimentación secuencial de la teoría de diseño de experimentos hemos conseguido desarrollar una inédita hibridación muy versátil y potente con poco coste computacional.

Obviamente, dicha implementación requiere un conocimiento previo de la instancia con la que estamos trabajando, si fuera posible, en dimensiones inferiores.

En el caso de que no tengamos la posibilidad de explorar las instancias en dimensiones bajas podemos hacer una serie de experimentaciones previas con S.G.O. solo aumentando los parámetros *n* y *n_iter*. De esta forma, tendremos mejores resultados y así seleccionaremos los *dim-paralelepípedos* con mayor probabilidad de contener al óptimo. No obstante, siempre existe la posibilidad de acudir a técnicas de diseños factoriales fraccionales.

En algunas funciones multimodales y con "topologías malas", como la función de SCHWEFEL, **hemos alcanzado el óptimo en dimensiones muy altas, incluso 80,** y en una región factible mayor comparada con la casi totalidad de las instancias evaluadas.

También **hemos obtenido el óptimo con la función de DIXON & PRICE 25** para la que el punto de silla, ver capítulo 5, no ha supuesto ningún problema.

Los tiempos de ejecución han sido 15 veces superiores a los empleados por una ejecución de S.G.O. Pensamos que el esfuerzo computacional es asumible.

Aunque los resultados han sido, en general, muy buenos, no hemos conseguido alcanzar el óptimo, según nuestro criterio, en tres instancias: las funciones de RASTRIGIN 10, de RASTRIGIN 20 Y de ACKLEY 30. No obstante, para la función de RASTRIGIN 10 **sí hemos encontrado un excelente valor entre las 10 ejecuciones**.

Intentaremos en el próximo capítulo la hibridación con otros heurísticos, basados en una filosofía diferente, para obtener el óptimo en ellas.

El heurístico SG. puede adaptarse fácilmente a cualquier otro algoritmo de optimización sea éste el que sea, ya que sólo cambian las variables de entrada relativas a la región factible, para incrementar su potencia. Se trata en realidad de una estrategia innovadora bastante eficiente para diseñar las subregiones en que realizar la concentración de la exploración.

"La Ley de Murphy no significa que algo malo va a pasar. Significa que todo lo que puede suceder, sucederá "

Cooper, Interstellar (Christopher Nolan 2014)

CAPÍTULO 7

HIBRIDACIÓN CON ALGORITMOS DE INTENSIFICACIÓN

1. INTRODUCCIÓN

Hasta el momento hemos hibridado S.G.O. con algoritmos de búsqueda local, Nelder-Mead y Gradiente, y con algoritmos de concentración, segmentando la región factible, SG. Con estas alianzas hemos obtenido muy buenos resultados con todas las funciones de “topología buena” planteadas, incluso en dimensiones altas y con algunas funciones de "topología mala", con moderados esfuerzos computacionales. Sin embargo las hibridaciones planteadas no han sido efectivas con algunas funciones multimodales en dimensiones altas, concretamente con las funciones de RASTRIGIN 10, de RASTRIGIN 20 y de ACKLEY 30.

En el presente capítulo vamos a plantear dos nuevas hibridaciones para S.G.O. con sendos algoritmos basados en técnicas de intensificación en entornos de soluciones muy buenas.

El primero es un algoritmo que hemos bautizado con el nombre “Agujero de Gusano”, A.G., inspirado en la homónima idea de la mecánica relativista. Consiste en construir túneles “curvando la región factible” que permitan viajar a las soluciones muy buenas hasta el óptimo global.

Tal y como ocurre en la teoría de Einstein dichos túneles solo podrán existir, en nuestro símil serán efectivos, si se ubican en las inmediaciones de un punto que concentre gran atracción gravitatoria: el óptimo global. Figura 7.1

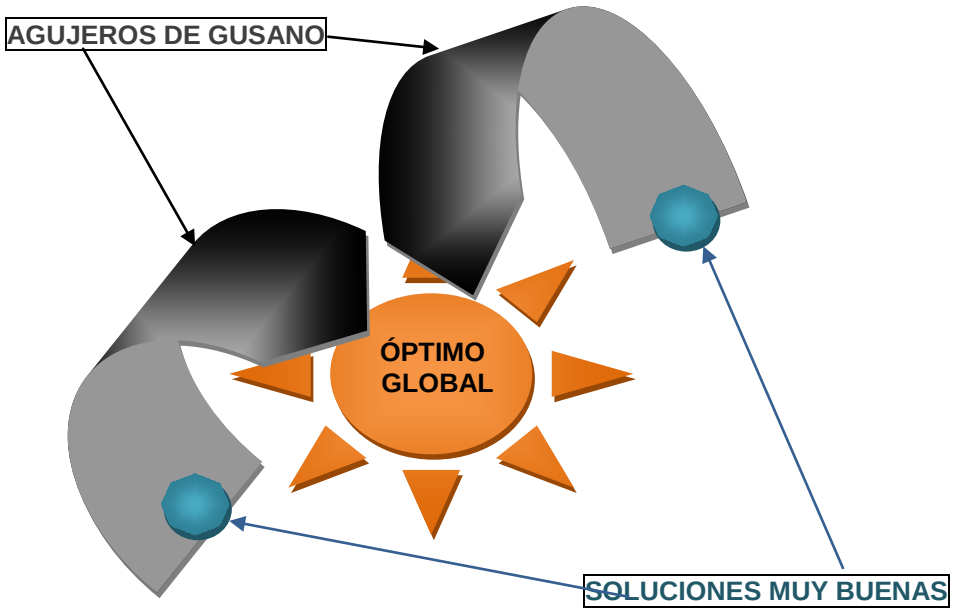


Figura 7.1. Representación del algoritmo Agujero de Gusano.

El segundo es el heurístico V.S.O. (Formato 2005), ya expuesto en el capítulo 2. Se trata de un algoritmo en el que, de forma análoga a Path Relinking, partiendo de una semilla de "soluciones buenas" se busca mejorar los valores encontrados evaluando la función objetivo en puntos más o menos aleatorios, dependiendo de la implementación, pertenecientes a las rectas que unen los distintos puntos de la semilla con el mejor, figura 7.2.

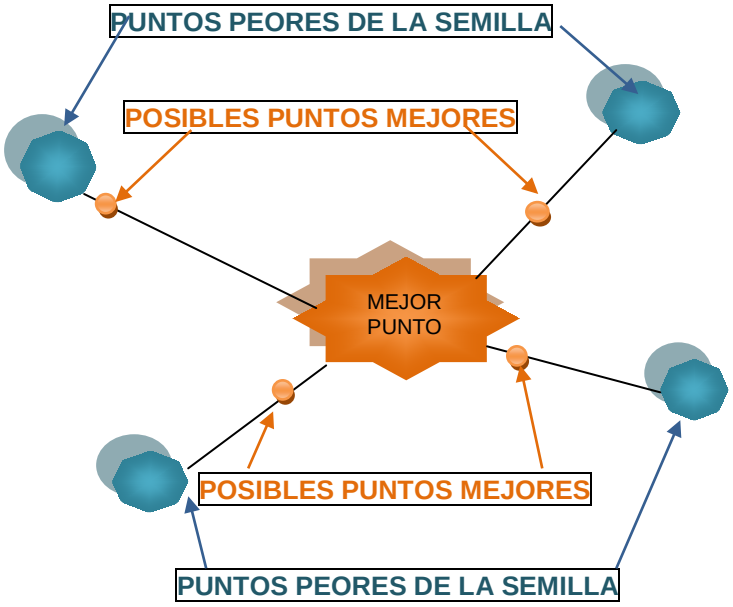


Figura 7.2. Representación del algoritmo V.S.O.

Nuestro objetivo en el presente capítulo será presentar las implementaciones de las nuevas hibridaciones planteadas y utilizarlas para alcanzar los óptimos globales de las tres instancias en las que han fracasado las hibridaciones anteriores en los capítulos 5 y 6.

Recordemos que son las funciones de RASTRIGIN 10, de RASTRIGIN 20 Y de ACKLEY 30.

Así como nuestro criterio de optimalidad:

Consideraremos que hemos alcanzado el óptimo si el error de alguno de los resultados obtenidos es menor que $3 \cdot 10^{-6}$ y el error de alguno de los promedios es menor que $3 \cdot 10^{-3}$.

O bien:

Consideraremos que hemos alcanzado el óptimo si el error de alguno de los resultados obtenidos es menor que $3 \cdot 10^{-6}$ y en alguna de las tandas de 10 ejecuciones hemos obtenido errores menores que $3 \cdot 10^{-6}$ en, al menos, 7 de ellas.

2. EL HEURÍSTICO AGUJERO DE GUSANO

Atendiendo a la definición de Wikipedia, un agujero de gusano, puente de Einstein-Rosen, es “una hipotética característica topológica de un espacio-tiempo que esencialmente consiste en un atajo a través del espacio y el tiempo” (Wikipedia, entrada: Agujero de gusano, 2015). Estos hipotéticos **cambios topológicos** forman una red que se origina merced a **determinadas probabilidades** que varían constantemente y que requieren la existencia cercana de potentes atractores gravitatorios (Kip Thorne 2014).

La forma en que hemos planteado la implementación del cambio topológico es muy simple y surge a raíz del estudio pormenorizado de los valores de las abscisas obtenidas en multitud de experimentos muy buenos, pero no óptimos, de S.G.O. con funciones de dimensión alta: observando dichos valores comprobamos que el heurístico no quedaba realmente atrapado en óptimos locales sino que solo un pequeño número de las coordenadas no alcanzaba el valor de la correspondiente abscisa del óptimo, mientras que un porcentaje muy elevado de las coordenadas llegaba con mucha precisión al valor de las del óptimo global.

Al repetir las experiencias, comprobamos que en todos los casos fallaba alguna coordenada pero, afortunadamente, no eran siempre las mismas. Así pues, **el cambio topológico** consiste, evidentemente, en **cambiar el valor erróneo de la correspondiente coordenada por el valor correcto.**

Para plantear **las probabilidades** con que dichos cambios deben producirse partimos de un grupo de resultados necesariamente muy buenos, de tamaño *tamgrupo* y analizamos **la moda estadística**, i.e. el valor más repetido, **de cada componente**. Si la moda muestra suficiente peso, es decir, su frecuencia absoluta supera estrictamente la mitad de *tamgrupo*, o la mitad de *tamgrupo*-1 en caso de que sea impar, **cambiamos todos los valores de la abscisa correspondiente al valor de la moda**. Dado que en las simulaciones es extraordinariamente difícil que dos valores sean exactamente iguales, redondeamos previamente el valor de las abscisas a 5 cifras significativas; pero al realizar el cambio recuperamos exactamente el valor de la primera de las abscisas del grupo en la que se alcanzó la moda.

Expongamos el algoritmo con detalle para analizar las distintas posibilidades:

Partimos de un grupo formado por *tamgrupo* soluciones.

- **Paso 1: Para cada coordenada** calculamos la moda:

  O bien el número de veces que aparece la moda supera estrictamente la mitad de los elementos del grupo en cuyo caso **todos los valores se cambian a la moda**

así, **de *tamgrupo* valores pasamos a tener solo uno en la correspondiente coordenada.**

O bien la frecuencia de la moda es inferior o igual a la mitad de los elementos del grupo, la moda no tiene suficiente peso, en cuyo caso trabajamos en dos vías diferentes simultáneamente:

- **Paso 2:**

  - **Vía 1:** dejamos los valores de la coordenada correspondiente tal y como estaban inicialmente, bloqueamos el agujero de gusano, **dejando *tamgrupo* valores diferentes en la correspondiente coordenada.**

  - **Vía 2: cambiamos todos los valores de la abscisa correspondiente al valor de su media estadística**, creamos un nuevo agujero de gusano que saque las coordenadas con moda débil de eventuales óptimos locales. **En este caso pasamos a tener un solo valor en la correspondiente coordenada.**

.Una vez que hayamos terminado de analizar todas las coordenadas tendremos **dos grupos de soluciones:**

mejor_1: formado por *tamgrupo* soluciones en las que en algunas coordenadas tendremos el mismo valor, modas fuertes, y en otras coordenadas, modas débiles, los valores de partida, obviamente distintos. **El resultado de la vía 1.**

mejor_2: formado por una única solución que tendrá en cada coordenada el resultado de la moda o la media de la correspondiente componente de *tamgrupo* dependiendo del peso de la moda. **El resultado de la vía 2.**

- **Paso 3:**

Pasamos todas las soluciones de mejor_1 y la única solución de mejor_2 por el algoritmo Gradiente para mejorarlas cuanto sea posible.

Por último, nos quedamos con el mejor de todos los elementos de mejor_1 y del elemento de mejor_2: mejor_total

Si no hemos mejorado el mejor valor del grupo inicial entendemos que el grupo no estaba lo suficientemente cerca del óptimo global y el agujero de gusano no ha dado resultado.

- **Fin del algoritmo.**

Un posible organigrama del algoritmo A.G. es el de la figura 7.3.

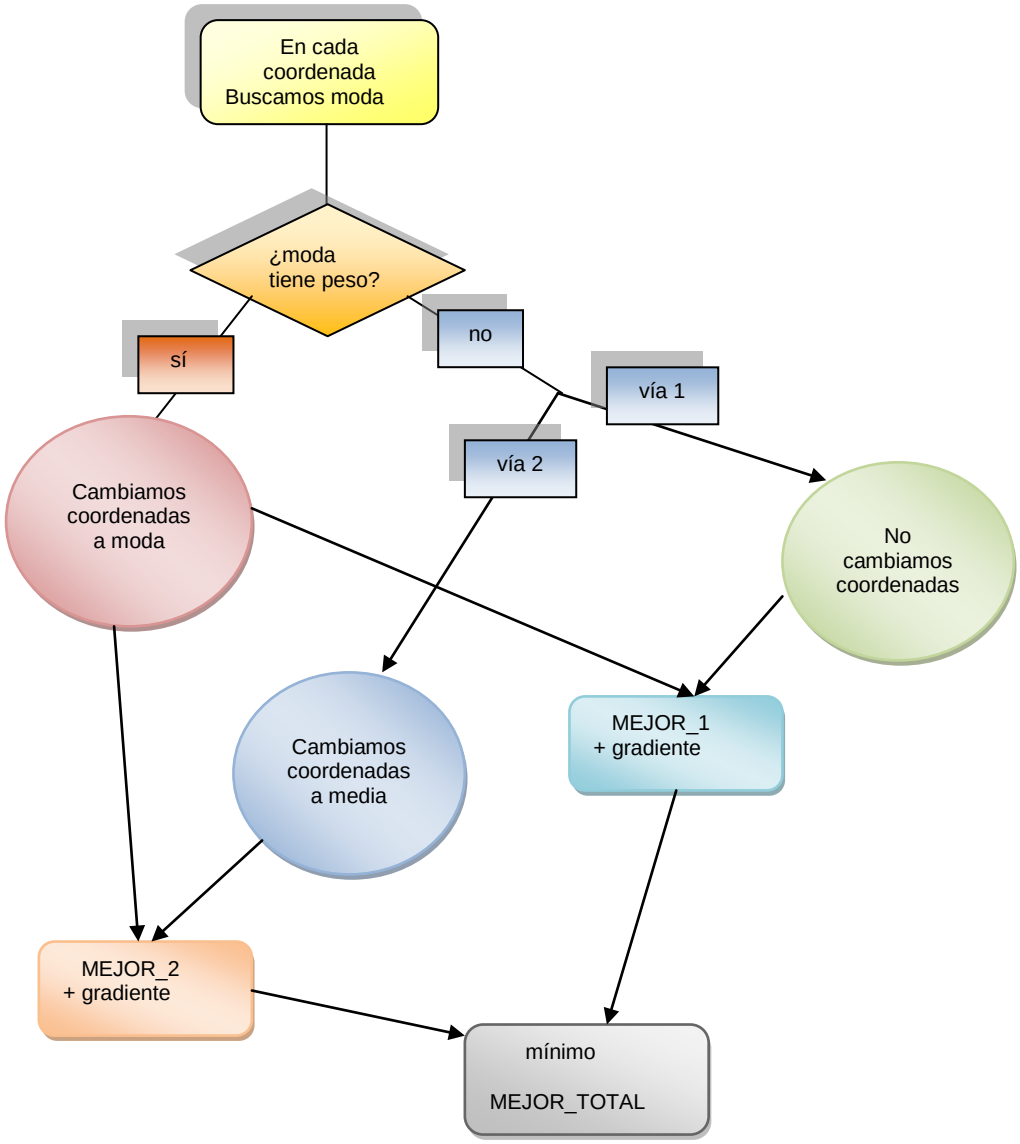


Figura 7.3. Organigrama del algoritmo A.G.

Recordemos que MEJOR_1 está formado por *tamgrupo* soluciones, mientras que MEJOR_2 está formado por una única solución. Nos quedaremos con MEJOR_TOTAL, que será la mejor de todas ellas, tras haberlas pasado por el algoritmo Gradiente.

Pongamos un ejemplo:

Supongamos que estamos optimizando la función $f\left(x,y,z\right)=x^2+y^2+z^2$ y partimos del grupo $\left\{s_1=(0.2,1,3),\ s_2=(0.2,2,-1),\ s_3=(0.2,-1,-3),\ s_4=(1,-1,3)\right\}$

en este caso solo tiene moda fuerte la primera componente, por tanto

mejor_1 = $\left\{(0.2,1,3),\ (0.2,2,-1),\ (0.2,-1,-3),\ (0.2,-1,3)\right\}$

mejor_2 = $\left\{(0.2,0.25,0.5)\right\}$ (medias de las componentes 2ª y 3ª).

Por último (en este caso, sin pasar por Gradiente)

mejor_total = mejor_2

3. IMPLEMENTACIÓN DE V.S.O.

Para implementar el heurístico Very Simple Optimization partimos de una semilla, generalmente formada por 20 soluciones obtenidas respectivamente de 20 evaluaciones de S.G.O., como hicimos en la hibridación S.G.O.-Nelder-Mead para obtener el *dim+1 simplex* inicial, ver capítulo 5. Tal como ocurría entonces, dicho criterio aumenta el coste computacional, aunque en esta ocasión el incremento es independiente de la dimensión de la instancia estudiada.

En el caso de que el tiempo de ejecución llegara a ser muy alto podría variarse esta estrategia tomando los 20 mejores de una sola ejecución de S.G.O. o planteando alguna opción intermedia.

Para hallar las nuevas soluciones hemos realizado en cada vuelta, hasta un total de *valvso* vueltas, 10 intentos en los que **cortamos las rectas que unen el mejor con el resto de la semilla** con **10 proporciones elegidas al azar** respectivamente entre 0 y 1. En cuanto encontramos una solución mejor nos vamos directamente a la siguiente vuelta y si tras los 10 intentos no hemos conseguido mejorar el mejor punto, paramos. De esta forma no se llega a dar *valvso* vueltas si en algún momento no hemos podido obtener mejores soluciones.

En todas las ejecuciones realizadas hemos considerado *valvso=100*.

La salida es el mejor valor obtenido en todos los intentos de todas las vueltas.

Un pseudocódigo de nuestra implementación es el siguiente, figura 7.4

Inicio: semilla= $\{s_i, 1 \le i \le 20\}$

$fs^* = \min\{f(s_i) / 1 \le i \le 20\}$; $s^* = s_i$ / $fs^* = f(s_i)$

vuelta=0 intento =0

Desarrollo:

(1) vuelta =vuelta+1 (mientras vuelta ≤ valvso)

(2) intento=intento+1

$\alpha \in U(0,1)$; $t_i = \alpha s^* + (1-\alpha) s_i \quad 1 \le i \le 20$

$ft^* = \min\{f(t_i) / 1 \le i \le 20\}$

si $ft^* < fs^* \Rightarrow s_i = t_i \quad 1 \le i \le 20$ vamos directamente a (1)

si $ft^* = fs^*$

si intento ≤ 9 vamos a (2)

si intento=10 SALIMOS DEL PROGRAMA

Salida:

s^*, fs^*

Figura 7.4. Pseudocódigo de la implementación de V.S.O.

4. RESULTADOS DE LOS METAHEURÍSTICOS DE S.G.O. CON V.S.O. , SG. Y A.G.

Hemos probado la hibridación de S.G.O. con los algoritmos propuestos en el presente capítulo únicamente con las 3 instancias con las que fracasaron las anteriores hibridaciones: funciones de RASTRIGIN 10, de RASTRIGIN 20 Y de ACKLEY 30.

El coste computacional de A.G. es mínimo pero, para tener garantías de éxito debemos ejecutarlo con un grupo de aproximaciones muy buenas, lo que necesariamente va a incrementar el tiempo de ejecución.

Para la búsqueda del grupo de soluciones muy buenas hemos actuado hibridando S.G.O. con algunas de las heurísticas desarrolladas a lo largo del presente trabajo, concretamente con V.S.O. y SG., ver capítulo 6, adaptando cada hibridación a la correspondiente instancia de forma que obtengamos un grupo de soluciones suficientemente buenas con el mínimo coste posible.

En las ejecuciones llevadas a cabo no ha sido necesario contar con un grupo numeroso de soluciones muy buenas como punto de partida para A.G. Hemos tomado un grupo de *tamgrupo=10* soluciones. Además lo hemos "reciclado" extrayendo todos los posibles subconjuntos del grupo de *tamgrupo=9* elementos, obviamente 10. Así, hemos ejecutado A.G. un total de 11 veces sobre el mismo grupo de partida de soluciones muy buenas, lo que no incrementa de forma apreciable el tiempo total, ya que, la mayor parte del tiempo de la hibridación se emplea en buscar el grupo, y nos permite obtener 10 ejecuciones diferentes para completar una hoja de Excel, como hemos hecho en las experimentaciones anteriores.

En cada caso concreto detallaremos la hibridación usada para obtener el grupo de muy buenas soluciones.

Para la última fase de A.G., mejora de las soluciones con Gradiente, hemos aumentado nvecesgrad al valor 1000000. Recordemos que el algoritmo está diseñado para parar si en algún momento no consigue mejorar la solución.

Exponemos a continuación los resultados obtenidos.

4.1. LA FUNCIÓN DE RASTRIGIN10

En esta ocasión, para obtener el grupo hemos usado la alianza S.G.O.-V.S.O. Hemos ejecutado S.G.O. 20 veces, con los valores habituales para los parámetros y los valores *n=100* y *n_iter=100,* obteniendo así una semilla de 20 soluciones usada como entrada para V.S.O. con *valvso=100* -recordemos que el algoritmo acaba cuando no consigue mejorar la solución-. La salida de V.S.O. es una sola solución, por tanto, para obtener el

grupo con el que alimentar A.G. hemos tenido que ejecutar S.G.O.-V.S.O. un total de 10 veces.

Hemos optimizado la función en el paralelepípedo $R = [-5.12\,,\,5.12]^{10}$.

En la hoja de Excel de la figura 7.5 recogemos 10 de los resultados obtenidos aplicando A.G. a los distintos subconjuntos del grupo con tamgrupo=10 y tamgrupo=9. En esta ocasión son excelentes y **verifican ampliamente nuestro criterio establecido para la búsqueda del óptimo.**

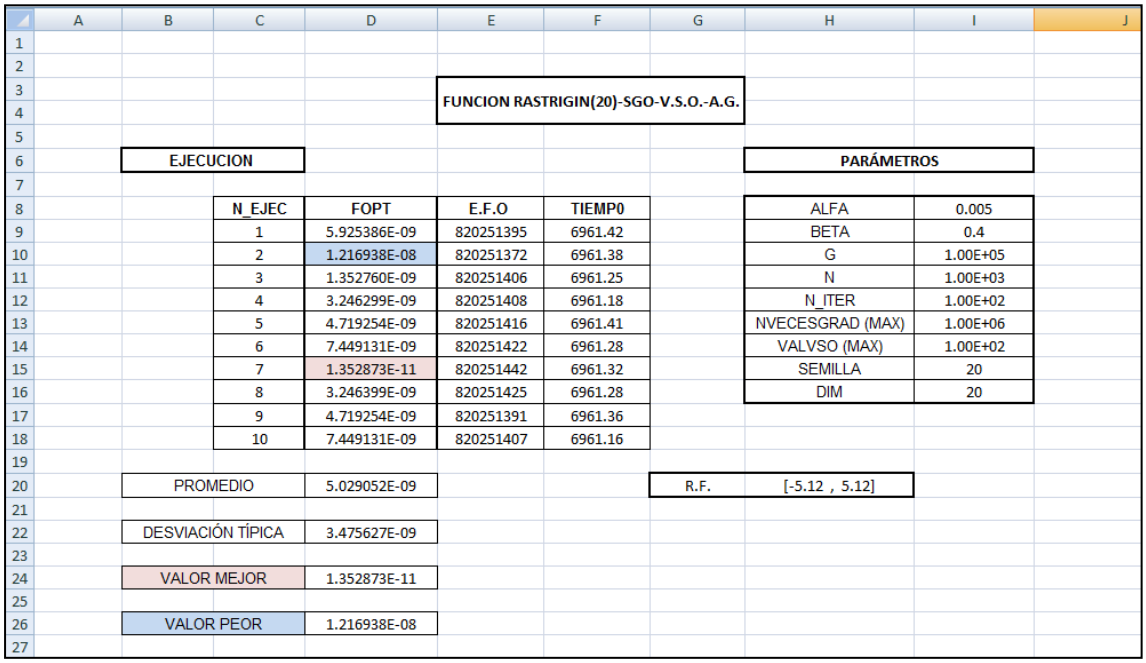
	A	B	C	D	E	F	G	H	I	J
1										
2										
3					FUNCION RASTRIGIN(10)-SGO-V.S.O.-A.G.					
4										
5										
6		EJECUCION						PARÁMETROS		
7										
8			N_EJEC	FOPT	E.F.O	TIEMP0		ALFA	0.005	
9			1	1.580214E-08	41968536	364.34		BETA	0.4	
10			2	2.881364E-09	41968541	364.39		G	1.00E+05	
11			3	1.013433E-08	41968546	364.40		N	1.00E+02	
12			4	5.268802E-09	41968539	364.36		N_ITER	1.00E+02	
13			5	1.217771E-08	41968540	364.36		NVECESGRAD (MAX)	1.00E+06	
14			6	5.955414E-09	41968533	366.30		VALVSO (MAX)	1.00E+02	
15			7	1.109801E-08	41968560	364.69		SEMILLA	20	
16			8	3.613380E-09	41968538	365.33		DIM	10	
17			9	2.746248E-10	41968589	364.82				
18			10	1.048576E-09	41968575	364.68				
19										
20		PROMEDIO		6.825435E-09			R.F.	[-5.12 , 5.12]		
21										
22		DESVIACIÓN TÍPICA		5.204379E-09						
23										
24		VALOR MEJOR		2.746248E-10						
25										
26		VALOR PEOR		1.580214E-08						
27										

Figura 7.5. Resultados RASTRIGIN10 S.G.O.-V.S.O.-A.G.

4.2. LA FUNCIÓN DE RASTRIGIN 20

Inicialmente hemos repetido la misma hibridación, con idénticos valores para los parámetros a los usados para la instancia de RASTRIGIN 10. Sin embargo, al duplicar la dimensión, el grupo ha empeorado considerablemente y A.G. no ha llevado a las soluciones a alcanzar el óptimo.

Con la intención de perfeccionar el grupo hemos mejorado la semilla usada en V.S.O. incrementado el número de asteroides de las 20 ejecuciones de S.G.O. a *n=1000*. El resto de los valores de parámetros y de número de veces de ejecuciones se han mantenido iguales al caso anterior. Ha bastado con este cambio **para obtener el óptimo según nuestro criterio** como se recoge en la hoja Excel de la figura 7.6.



FUNCION RASTRIGIN(20)-SGO-V.S.O.-A.G.

EJECUCION				
N_EJEC	FOPT	E.F.O	TIEMP0	
1	5.925386E-09	820251395	6961.42	
2	1.216938E-08	820251372	6961.38	
3	1.352760E-09	820251406	6961.25	
4	3.246299E-09	820251408	6961.18	
5	4.719254E-09	820251416	6961.41	
6	7.449131E-09	820251422	6961.28	
7	1.352873E-11	820251442	6961.32	
8	3.246399E-09	820251425	6961.28	
9	4.719254E-09	820251391	6961.36	
10	7.449131E-09	820251407	6961.16	

PROMEDIO	5.029052E-09
DESVIACIÓN TÍPICA	3.475627E-09
VALOR MEJOR	1.352873E-11
VALOR PEOR	1.216938E-08

PARÁMETROS	
ALFA	0.005
BETA	0.4
G	1.00E+05
N	1.00E+03
N_ITER	1.00E+02
NVECESGRAD (MAX)	1.00E+06
VALVSO (MAX)	1.00E+02
SEMILLA	20
DIM	20

R.F.	[-5.12 , 5.12]

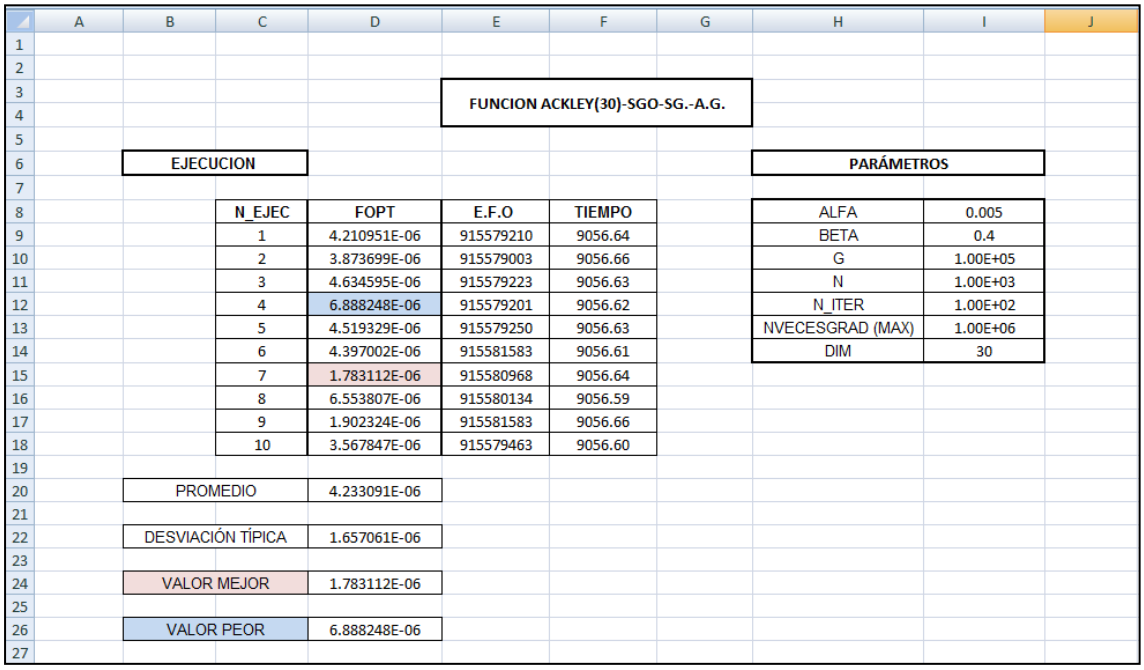
Figura 7.6. Resultados RASTRIGIN20 S.G.O.-V.S.O.-A.G.

4.3. LA FUNCIÓN DE ACKLEY 30

La hibridación S.G.O.-V.S.O. no se ha mostrado tan eficaz para esta instancia como en los casos anteriores, por ello hemos optado por usar la alianza S.G.O.-SG. para obtener la semilla de A.G. En este caso hemos ejecutado S.G.O.-SG. 10 veces para obtener la semilla de entrada para A.G. El esfuerzo computacional es un poco inferior en este caso ya que evaluamos S.G.O. en 15 ocasiones para obtener cada solución del grupo, mientras que en el caso anterior realizábamos 20 ejecuciones, aunque, por otra parte la dimensión es 30 y, por ello, el tiempo de computación aumenta. Los valores de los parámetros de S.G.O se detallan en la figura 7.7, observemos que *n=5000*.

La segmentación de la región factible ha resultado mejor estrategia ya que con ella hemos obtenido un grupo suficientemente bueno sobre el que la actuación de Agujero de Gusano ha sido un éxito.

Hemos reciclado el grupo obtenido como en los casos anteriores. Presentamos 10 de los resultados obtenidos en la figura 7.7. **Hemos alcanzado el óptimo según nuestro criterio también en este caso.**



FUNCION ACKLEY(30)-SGO-SG.-A.G.

EJECUCION

N_EJEC	FOPT	E.F.O	TIEMPO
1	4.210951E-06	915579210	9056.64
2	3.873699E-06	915579003	9056.66
3	4.634595E-06	915579223	9056.63
4	6.888248E-06	915579201	9056.62
5	4.519329E-06	915579250	9056.63
6	4.397002E-06	915581583	9056.61
7	1.783112E-06	915580968	9056.64
8	6.553807E-06	915580134	9056.59
9	1.902324E-06	915581583	9056.66
10	3.567847E-06	915579463	9056.60

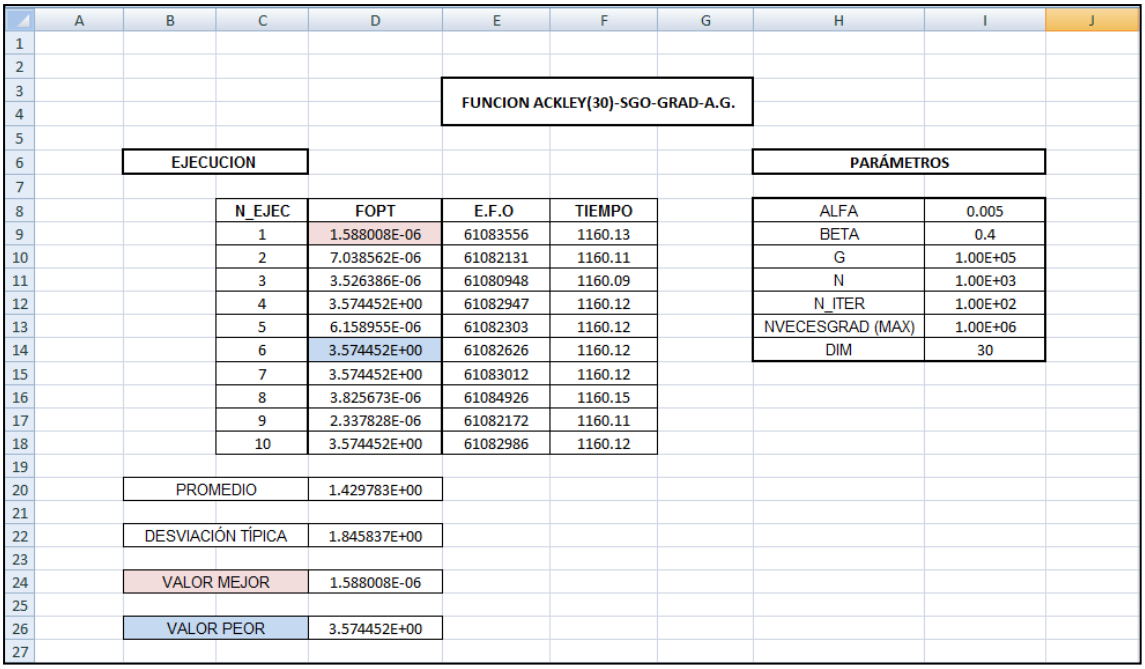
PROMEDIO	4.233091E-06
DESVIACIÓN TÍPICA	1.657061E-06
VALOR MEJOR	1.783112E-06
VALOR PEOR	6.888248E-06

PARÁMETROS

ALFA	0.005
BETA	0.4
G	1.00E+05
N	1.00E+03
N_ITER	1.00E+02
NVECESGRAD (MAX)	1.00E+06
DIM	30

Figura 7.7. Resultados ACKLEY30 S.G.O.-SG-A.G.

También hemos probado el metaheurístico S.G.O.-Gradiente-A.G. con esta instancia y los valores *n=1000, niter=100* y *nvecesgad=1000000.* En la figura 7.8 puede observarse que **hemos obtenido el óptimo en 6 de las 10 ocasiones** con menor esfuerzo computacional.



FUNCION ACKLEY(30)-SGO-GRAD-A.G.

EJECUCION

N_EJEC	FOPT	E.F.O	TIEMPO
1	1.588008E-06	61083556	1160.13
2	7.038562E-06	61082131	1160.11
3	3.526386E-06	61080948	1160.09
4	3.574452E+00	61082947	1160.12
5	6.158955E-06	61082303	1160.12
6	3.574452E+00	61082626	1160.12
7	3.574452E+00	61083012	1160.12
8	3.825673E-06	61084926	1160.15
9	2.337828E-06	61082172	1160.11
10	3.574452E+00	61082986	1160.12

PROMEDIO	1.429783E+00
DESVIACIÓN TÍPICA	1.845837E+00
VALOR MEJOR	1.588008E-06
VALOR PEOR	3.574452E+00

PARÁMETROS

ALFA	0.005
BETA	0.4
G	1.00E+05
N	1.00E+03
N_ITER	1.00E+02
NVECESGRAD (MAX)	1.00E+06
DIM	30

Figura 7.8. Resultados ACKLEY30 S.G.O.-GRADIENTE-A.G.

5. RESUMEN DE RESULTADOS CON LAS FUNCIONES DE SCHWEFEL 6, DE RASTRIGIN 10, DE RASTRIGIN 20, DE DIXON & PRICE 25 Y DE ACKLEY 30

Dado que hemos usado distintas hibridaciones, sobre todo con las cinco funciones más complicadas (ver capítulo 5), hemos decidido presentar una hoja Excel, figura 7.9, para tener recogidos los resultados obtenido en los distintos casos para las cinco funciones estudiadas en el presente y el anterior capítulos.

Hemos resaltado en amarillo los resultados que han cumplido nuestro criterio de alcance del óptimo.

RESULTADOS DE LAS DISTINTAS HIBRIDACIONES CON LAS 5 FUNCIONES MÁS COMPLICADAS

SCHWEFEL 6

HIBRIDACIÓN	MEJOR VALOR	MEJOR PROMEDIO
S.G.O.-NELME	-4.34602E-07	1.362049E+02
S.G.O.-GRAD	-4.24193E-07	2.487241E+02
S.G.O.-SG.	-4.34602E-07	-4.34602E-07
S.G.O.	9.870026E+01	

DIXON & PRICE 25

HIBRIDACIÓN	MEJOR VALOR	MEJOR PROMEDIO
S.G.O.-NELME	6.666667E-01	8.708717E-01
S.G.O.-GRAD	6.666667E-01	6.666667E-01
S.G.O.-SG.	1.311187E-09	4.115962E-08
S.G.O.	6.666667E-01	

RASTRIGIN 10

HIBRIDACIÓN	MEJOR VALOR	MEJOR PROMEDIO
S.G.O.-NELME	1.989918E+00	1.691428E+01
S.G.O.-GRAD	9.949588E+00	2.139164E+01
S.G.O.-SG.	1.342880E-08	2.387902E+00
S.G.O.-V.S.O.-A.G.	2.746248E-10	6.825435E-09
S.G.O.	1.790923E+01	

ACKLEY 30

HIBRIDACIÓN	MEJOR VALOR	MEJOR PROMEDIO
S.G.O.-NELME	3.786261E+00	5.721325E+00
S.G.O.-GRAD	1.697037E+01	6.835115E+01
S.G.O.-SG.	5.350644E+00	6.558409E+00
S.G.O.-SG.-A.G.	1.783112E-06	4.233091E-06
S.G.O.	1.604187E+01	

RASTRIGIN 20

HIBRIDACIÓN	MEJOR VALOR	MEJOR PROMEDIO
S.G.O.-NELME	1.094455E+01	2.636635E+01
S.G.O.-GRAD	5.273274E+01	7.163683E+01
S.G.O.-SG.	3.979836E+00	6.168746E+00
S.G.O.-V.S.O.-A.G.	1.352873E-11	5.029052E-09
S.G.O.	6.069238E+01	

Figura 7.9. Resumen de resultados para las 5 funciones más complicadas.

6. CONCLUSIONES

El algoritmo V.S.O. se ha revelado como un eficaz aliado de S.G.O. para generar muy buenas soluciones en los casos de las funciones de RASTRIGIN 10 y de RASTRIGIN 20. Sin embargo, en el caso de la función de ACKLEY 30, ha sido la hibridación con SG. la que nos ha dado el grupo de muy buenas soluciones.

Si bien las soluciones intermedias halladas en ambos casos, eran aceptables, no llegaban a cumplir nuestro criterio de optimalidad. Sólo la ayuda del algoritmo que hemos diseñado y desarrollado, Agujero de Gusano, ha sido capaz de guiarlas hasta obtener rotundamente el objetivo planteado. Así A.G. se presenta como una sencilla y muy eficaz herramienta que puede hibridarse con cualquier heurístico de optimización continua para obtener excelentes resultados cuando los algoritmos utilizados aportan buenas soluciones pero no llegan al óptimo global.

Los tiempos de computación han aumentado considerablemente, ya que generar una semilla suficientemente buena ha conllevado muchas ejecuciones de S.G.O. Los tiempos de ejecución, tanto de V.S.O., como de A.G. han sido insignificantes.

Por ejemplo:

- De los 41968589 de evaluaciones de la función objetivo, 364.82 segundos- de una ejecución de la hibridación S.G.O.-V.S.O.-A.G. con la función de RASTRIGIN 10, solo 7124, -0.11 segundos- se emplean en la ejecución de A.G. El resto se usa para la obtención de la semilla, cuyo mejor valor es 29.423214.

- De los 915580968 de evaluaciones de la función objetivo , -9056.64 segundos- de una ejecución de la hibridación S.G.O-SG.-A.G. para la función de ACKLEY 30, solo 3315, -0.064 segundos- se emplean en la ejecución de A.G. El resto se usa para obtener la semilla, cuyo mejor valor es 15.622567.

Es obvio que la mejora en los óptimos es grande, pero también lo es el incremento en los tiempos de computación.

Las tres instancias del presente capítulo, representan, sin duda, el conjunto más difícil con el que hemos enfrentado a nuestro algoritmo hasta el momento y **haber logrado éxito en la consecución del óptimo, según nuestro criterio, en todos los casos** nos lleva a pensar que S.G.O. con las distintas hibridaciones realizadas genera buenos metaheurísticos, robustos, eficientes y eficaces.

""La filosofía está escrita en ese vasto libro que es el universo.
Está escrito en lenguaje matemático, y las letras son triángulos,
círculos y otras figuras geométricas, sin las cuales es humanamente
imposible entender una sola palabra."

Galileo Galilei (1564-1642)

CAPÍTULO 8

PRUEBAS CON LA INSTANCIA CASSINI 2

1. INTRODUCCIÓN

En los últimos veinte años, los ingenieros aeroespaciales han formulado sus problemas de diseño de trayectorias interplanetarias en términos de problemas de optimización continua global. Los problemas de la Agencia Espacial Europea, E.S.A., denominados "Problemas de Optimización de Trayectorias Globales", G.T.O.P., que pueden consultarse en la página de la E.S.A. http://www.esa.int/gsp/ACT/inf/projects/gtop/gtop.html, conforman un extenso grupo de tales diseños.

Sin embargo, actualmente, dichos problemas se formulan, no solo buscando tener éxito en sus misiones, sino conseguirlo con los mínimos costes posibles, en tiempo, combustible, etc. usando asistencia gravitatoria. Se denominan " Multiple Gravity Assist missions with the possibillity of using Deep Space Maneuver, M.G.A.D.S.M. y en ellos puede modificarse, con propulsión química, la trayectoria en vuelo, fly-by, para aprovechar mejor los recursos disponibles.

Las soluciones de estos problemas de optimización continua con restricciones se utilizan para planificar trayectorias reales en misiones espaciales, pero implican la resolución de problemas en dimensiones altas con funciones multimodales. Por ello, la E.S.A. los presenta como problemas abiertos para que la comunidad científica en general aporte los óptimos que vaya encontrando.

Además, dichas instancias se presentan como "cajas-negras", implementadas a base de diferentes subrutinas que se encargan de las distintas variables a considerar en la función total. Las variables tienen naturaleza, y dimensiones, muy diferentes: velocidades, tiempos, distancias, ángulos... lo que imposibilita realizar pruebas en dimensiones inferiores.

En la misma página se expone un amplio número de heurísticos que se han utilizado en la optimización de los distintos problemas. Entre ellos están: Particle Swarm, Simulated Annealing, Artificial Bee Colony etc. y cobra una especial relevancia el algoritmo M.I.D.A.C.O., Mixed Integer Distributed Ant Colony Optimization, http://www.midaco-solver.com/ , que ostenta el record en la optimización de varios de los problemas planteados.

Todas las instancias, así como los códigos fuente de las funciones en el programa MATLAB, y en los lenguajes C++ y Pyton 2.7 pueden encontrarse en la página de la E.S.A. http://www.esa.int/gsp/ACT/inf/projects/gtop/gtop.html,

Entre los problemas M.G.A.D.S.M. planteados por la E.S.A., hemos elegido uno de ellos: la trayectoria de la sonda automática CASSINI 2, que fue lanzada en 1997, para probar en ella los heurísticos desarrollados a lo largo del presente trabajo. En primer lugar haremos una breve descripción de la historia de la sonda CASSINI 2. Posteriormente expondremos los resultados obtenidos con las distintas hibridaciones de S.G.O.

2. LA TRAYECTORIA CASSINI 2

CASSINI 2 es un problema M.G.A.D.S.M. para hallar la trayectoria interplanetaria de la misión CASSINI-HUYGENS, una sonda automática lanzada finalmente en Octubre de 1997 por la N.A.S.A. en colaboración con la E.S.A. con el fin de explorar Saturno y algunas de sus lunas, en especial Titán, por poseer atmósfera propia, (Marín 2014). Su nombre se debe al astrónomo italiano Giovanni Domenico Cassino, que en los siglos XVII y XVIII estudió Saturno y llegó a descubrir cuatro de sus lunas: Jápeto, Dione, Rea y Tetis. La nave está preparada para realizar distintas maniobras de asistencia gravitatoria con el fin de ahorrar al máximo combustible y tiempo. En julio de 2004 llegó finalmente a Saturno. Su trayectoria interplanetaria fue EVVEJS[1] con dos sobrevuelos de Venus, uno de la Tierra, uno de Júpiter para finalizar en Saturno, figura 8.1.

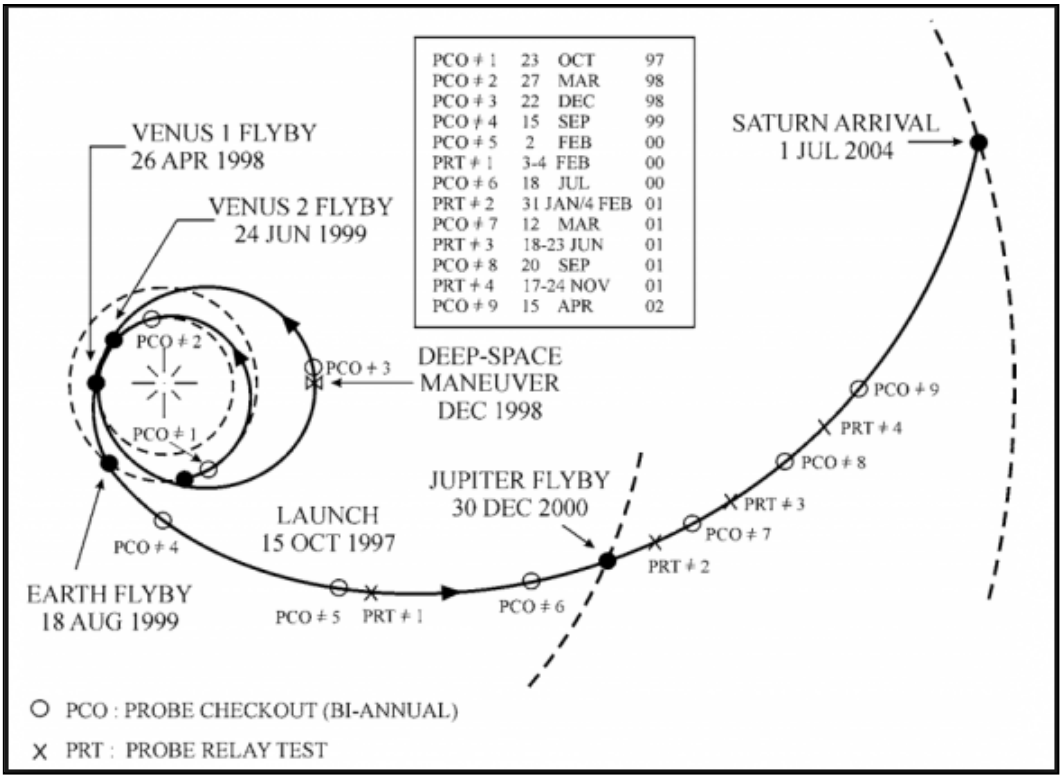


Figura 8.1. Trayectoria de CASSINI tomada de danielmarin.naukas.com

El problema CASSINI 2 implica la optimización de una función 22-dimensional, formada por varias subfunciones que exponemos en el apéndice 3, cuyas variables son

$$\overline{x} = \left(t_0, V_0, \alpha_0, \beta_0, \mathrm{T}_1, \mathrm{T}_2, \mathrm{T}_3, \mathrm{T}_4, \mathrm{T}_5, \eta_1, \eta_2, \eta_3, \eta_4, \eta_5, r_{p1}, r_{p2}, r_{p3}, r_{p4}, b_{incl1}, b_{incl2}, b_{incl3}, b_{incl4},\right)$$

donde

[1] Los planetas se definen con su nombre en inglés.

$t_0, V_0, \alpha_0, \beta_0$ son las condiciones de tiempo, medido en seg., velocidad, medida en km/seg y determinadas constantes de inicio del vuelo.

T_1, T_2, T_3, T_4, T_5 son los tiempos de duración, en días, de las trayectorias E-V , V-V ,V-E, E-J y J-S respectivamente.[2]

$\eta_1, \eta_2, \eta_3, \eta_4, \eta_5$ y s $b_{incl1}, b_{incl2}, b_{incl3}, b_{incl4}$ son ángulos de rotación e inclinación para hacer factibles las trayectorias anteriores, medidos en radianes.

$r_{p1}, r_{p2}, r_{p3}, r_{p4}$ son los radios de las órbitas que debe seguir la nave para ser capturada por los distintos planetas en su recorrido, medido en Km.

Los dominios de optimización de cada variable se exponen en la figura 8.2. A partir de este momento las denominaremos genéricamente $x(i)$.

	A	B	C	D	E	F	G
1							
2							
3							
4		ORDEN	C.INFERIOR	C.SUPERIOR	UNIDAD		
5		x(1)	-1000	0	seg.		
6		x(2)	3	5	km/seg		
7		x(3)	0	1	-		
8		x(4)	0	1	-		
9		x(5)	100	400	dias		
10		x(6)	100	500	dias		
11		x(7)	30	300	dias		
12		x(8)	400	1600	dias		
13		x(9)	800	2200	dias		
14		x(10)	0.01	0.9	rad		
15		x(11)	0.01	0.9	rad		
16		x(12)	0.01	0.9	rad		
17		x(13)	0.01	0.9	rad		
18		x(14)	0.01	0.9	rad		
19		x(15)	1.05	6	km		
20		x(16)	1.05	6	km		
21		x(17)	1.15	6.5	km		
22		x(18)	1.7	291	km		
23		x(19)	-π	π	rad		
24		x(20)	-π	π	rad		
25		x(21)	-π	π	rad		
26		x(22)	-π	π	rad		
27							

Figura 8.2. Dominios de optimización del problema CASSINI 2.

[2] Los nombres de los planetas se expresan en inglés.

Un exhaustivo estudio sobre los problemas M.G.A.D.S.M., así como las maniobras de asistencia gravitatoria y otras cuestiones relacionadas con las trayectorias de sondas espaciales puede encontrarse en Hamdy (2011).

También puede encontrarse información en la página web de la E.S.A. y en un buen número de blogs, entre los que destaca el blog de Daniel Marín EUREKA (Marín 2014) perteneciente a la red NAUKAS. Daniel Marín es un astrofísico y gran divulgador que ofrece multitud de artículos sobre sondas espaciales.

El mejor resultado admitido por la E.S.A. en la optimización de la trayectoria CASSINI 2 corresponde al valor $f(\overline{x}_M) = 8.383184 \quad \frac{Kg \cdot Km}{seg}$ ubicado en el punto (Danoy y otros 2012)

$$\begin{aligned}\overline{x}_M = (&-779.629801566988\,,\,3.265804135361\,,\;0.528440291493\,,\,0.382390419772\,,\ldots\\&,167.937610148996\,,\,424.032204472497\,,\,53.304869390732\,,\,589.767895836123\,,\ldots\\&,\,2199.961911685212\,,\,0.772877728290\,,\,0.531757418755\,,\,0.010789195916\,,\ldots\\&,\,0.167388829033\,,\,0.010425709182\,,\,1.358596310427,1.050001151443\,,\ldots\\&,1.306852313623\,,\,69.813404643644\,,\,-1.593310577644\,,\,-1.959572311812\,,\ldots\\&,\,-1.554796022348\,,\,-1.513432303179)\end{aligned}$$

Ha sido encontrado usando el heurístico MIDACO dentro del proyecto "Non-linear mixed-integer-based Optimisation Technique for Space Applications" co-fundado por E.S.A. Networking Partnership Initiative, Astrium Limited (Stevenage, UK) y the School of Mathematics University of Birmingham, UK. La información puede encontrarse consultando en la página web de la E.S.A., ya expuesta,

http://www.esa.int/gsp/ACT/inf/projects/gtop/gtop.html.

Para su obtención, en Mayo del año 2009, fue necesario ejecutar M.I.D.A.C.O. durante 50 días en un single core (Danoy y otros 2012).

En Danoy (2012) puede encontrarse una aproximación que mejora la aportada por M.I.D.A.C.O., $f(\overline{x}_{MDanoy}) = 8.383091 \quad \frac{Kg \cdot Km}{seg}$, obtenida usando la hibridación de un algoritmo genético con el simplex de Nelder-Mead. Sin embargo, no tenemos noticia de que haya sido registrada en la página web de la E.S.A.

El mejor valor hallado por la propia E.S.A. es $f(\overline{x}_{M(ESA)}) = 8.924 \quad \frac{Kg \cdot Km}{seg}$. En la página web no se explicitan los valores concretos de las abscisas.

Con el objetivo de hacer una primera observación de la variación de la función objetivo, hemos tomado sus valores en las cotas inferior y superior, figura 8.2, que denominaremos, por facilitar la notación, $C.INFERIOR = \overline{r_1}$, $C.SUPERIOR = \overline{r_2}$ respectivamente, y en algunos otros puntos, tomando como referencia, tanto las cotas, como el óptimo aportado por M.I.D.A.C.O., $\overline{x}_M$.

He aquí los datos registrados:

- Resultados en las cotas

$$f(\overline{r_1}) = 516.7052070411061 \qquad f(\overline{r_2}) = 296.6019679970260$$

- Resultados en algunas rectas que unen los puntos definidos por las cotas

$$f\left(\frac{\overline{r_1}+\overline{r_2}}{2}\right) = 209.5259300043539$$

$$f\left(\frac{\overline{r_1}+2\overline{r_2}}{3}\right) = 202.4180826931125 \;\; ; \;\; f\left(\frac{2\overline{r_1}+\overline{r_2}}{3}\right) = 73.02434451633918$$

$$f\left(\frac{\overline{r_1}+3\overline{r_2}}{4}\right) = 221.4979348280654 \; ; f\left(\frac{2\overline{r_1}+2\overline{r_2}}{4}\right) = 209.5259300043539$$

$$f\left(\frac{3\overline{r_1}+\overline{r_2}}{4}\right) = 177.9381770008948$$

- Resultados en entornos del óptimo

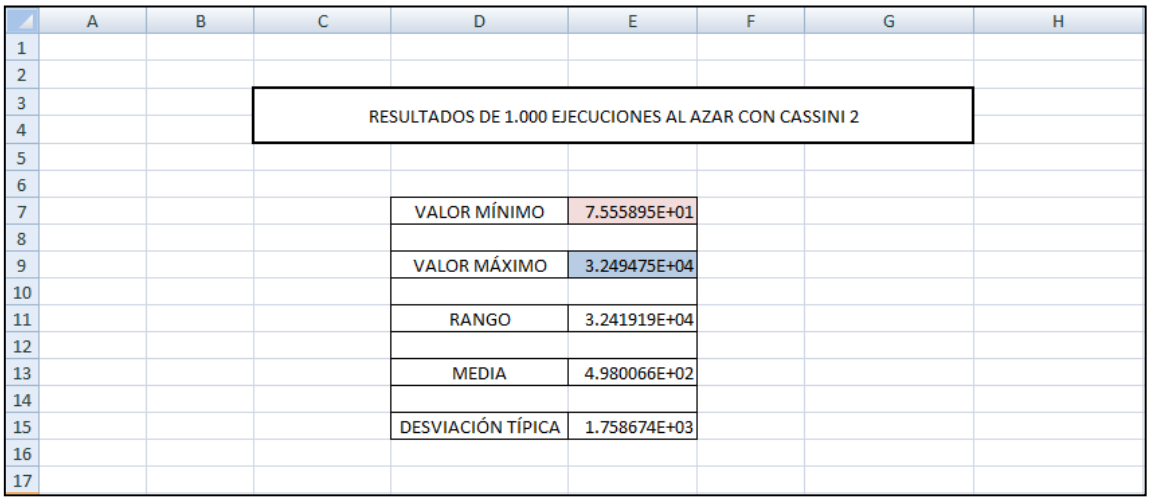
[3] $\overline{x}_M + 0.5 > r_2 \qquad ; \qquad f\left(\overline{x}_M - 0.5\right) = 218.8799598149091$

$$f\left(\overline{x}_M + 0.01\right) = 24.97825305129862 \;\; ; \;\; f\left(\overline{x}_M - 0.01\right) = 34.32454834604529$$

$$f\left(\overline{x}_M + 0.001\right) = 10.80981374163367 \;\; ; \;\; f\left(\overline{x}_M - 0.001\right) = 10.75486389415364$$

[3] Entendiendo 0.5, 0.01 y 0.001 como vectores 22-dimensionales cuyas componentes son todas iguales a 0.5, 0.01 y 0.001 respectivamente.

- También hemos elegido 1000 puntos al azar en la región factible y en la siguiente hoja de Excel, figura 8.3, hemos recogido los valores mínimo, máximo, rango, media y desviación típica de los valores de la función evaluada en ellos. La hoja de Excel completa, PRUEBAS ALEATORIAS CASSINI 2, puede consultarse en el CD adjunto.



RESULTADOS DE 1.000 EJECUCIONES AL AZAR CON CASSINI 2	
VALOR MÍNIMO	7.555895E+01
VALOR MÁXIMO	3.249475E+04
RANGO	3.241919E+04
MEDIA	4.980066E+02
DESVIACIÓN TÍPICA	1.758674E+03

Figura 8.3. Hoja Excel de resultados aleatorios con CASSINI 2.

A la vista de los resultados obtenidos en las distintas pruebas realizadas, en particular la última con un rango y una desviación típica tan elevados, y una media de casi 500, podemos intuir que la instancia CASSINI 2 va a suponer un difícil reto para nuestros algoritmos.

3. RESULTADOS DE LAS HIBRIDACIONES DE S.G.O.

Como hemos visto, la instancia CASSINI 2 conforma un problema 22-dimensional en el que cada variable representa una magnitud diferente, obviamente con intervalos de variación diferentes, por ello hemos tenido que modificar la implementación de los distintos algoritmos para esta nueva situación. La complejidad de la función objetivo ha disparado los tiempos de ejecución y ha supuesto un hándicap grande para realizar las pruebas, dado que las hemos realizado con un procesador Intel(R) Core(TM) i5-4460 CPU @ 3.20 GHz.

En todos los casos hemos hecho tandas de 23 ejecuciones como máximo con el fin de tener un grupo sobre el que probar las distintas hibridaciones.

Hemos comenzado probando solo S.G.O. con distintos valores para *n* y *niter*. Posteriormente hemos probado las hibridaciones S.G.O.-Nelder-Mead y S.G.O.-Gradiente con los valores que se detallan.

También hemos probado la hibridación S.G.O.-SG. y las hibridaciones S.G.O.- V.S.O. y S.G.O.-A.G.

Exponemos a continuación los resultados obtenidos.

MEJORES RESULTADOS OBTENIDOS PARA CASSINI 2

	MEJOR VALOR	E.F.O.	TIEMPO (seg.)	PARÁMETROS
S.G.O. n=10 , niter=10	5.78623414E+01	4060	13.11	
S.G.O. n=100 , niter=10	4.87421957E+01	40600	127.63	
S.G.O. n=100 , niter=100	4.73496904E+01	445600	1411.97	
S.G.O.-Gradiente	3.24221859E+01	468909	1510.53	n=100, niter=100, nvecesgrad= 1000
S.G.O.-Nelder-Mead	1.90006024E+01	25633435	60449.78	n=100, niter=10, valnel=10000
S.G.O.-SG.	3.02995908E+01	1583890	6958.80	n=100, niter=10, nvecesgrad=1000
S.G.O.-V.S.O.	2.88299013E+01	1380	2.78	(*) n=100, niter=100
S.G.O.-A.G.	2.00012248E+01	3653761	18114.86	(*) n=100, niter=100, nvecesgrad=10000

(*) tiempos de una sola ejecución en un grupo de 30 resultados obtenidos con S.G.O.

Figura 8.4. Hoja Excel de mejores resultados para CASSINI 2.

Hemos realizado una comparación de los resultados obtenidos por nosotros, FMEJOR, con el mejor dato obtenido por M.I.D.A.C.O., FMEJOR_GLOBAL, y con los datos obtenidos en la figura 8.3. Con ellos hemos elaborado la siguiente hoja Excel de resultados.

DATOS RELATIVOS A PRUEBAS ALEATORIAS CASSINI 2		
MEJOR DATO OBTENIDO: **FMEJOR**	1.90006024E+01
MEJOR DATO GLOBAL: **FMEJOR_GLOBAL**	8.383184E+00
FMEJOR-FMEJOR_GLOBAL	1.06174184E+01
FMEJOR-MEDIA	-4.79005984E+02
(**FMEJOR-FMEJOR_GLOBAL**)/RANGO	3.27504093E-04
(**FMEJOR**-MEDIA)/RANGO	-1.47753827E-02

Figura 8.5. Hoja Excel de comparación de resultados de CASSINI 2.

La diferencia entre nuestro mejor dato y el mejor global es 10 y es una diferencia grande. Sin embargo cuando relativizamos los datos al rango aleatoriamente obtenido en la figura 8.3, obtenemos datos más favorables.

Teniendo en cuenta que es un estudio inicial y que los tiempos de ejecución no han sido muy elevados, consideramos que es un resultado interesante para nuestras heurísticas.

4. CONCLUSIONES

El trabajo realizado debe entenderse como un primer estudio del comportamiento de S.G.O. y sus distintas hibridaciones para la optimización de la trayectoria de la sonda CASSINI 2.

Nos planteamos la optimización de la instancia CASSINI 2 en la fase final de nuestro trabajo de investigación. Los elevados tiempos de ejecución necesarios para llevar a cabo las distintas ejecuciones han dificultado extraordinariamente la obtención de resultados aceptables. Teniendo en cuenta, además, las primeras observaciones, creemos que obtener un mínimo de valor 19 en las condiciones en que hemos trabajado, es un resultado interesante.

La instancia CASSINI 2 parece un difícil reto para algoritmos de optimización con amplia experiencia en la literatura de optimización en trayectorias de sondas espaciales, como MIDACO.

En definitiva, si bien los resultados obtenidos no son buenos, son inferiores a los que se obtienen en un entorno de una centésima del óptimo y, en todo caso, nos aportan un interesante punto de partida para realizar posteriores investigaciones.

"Hacer predicciones es muy difícil,

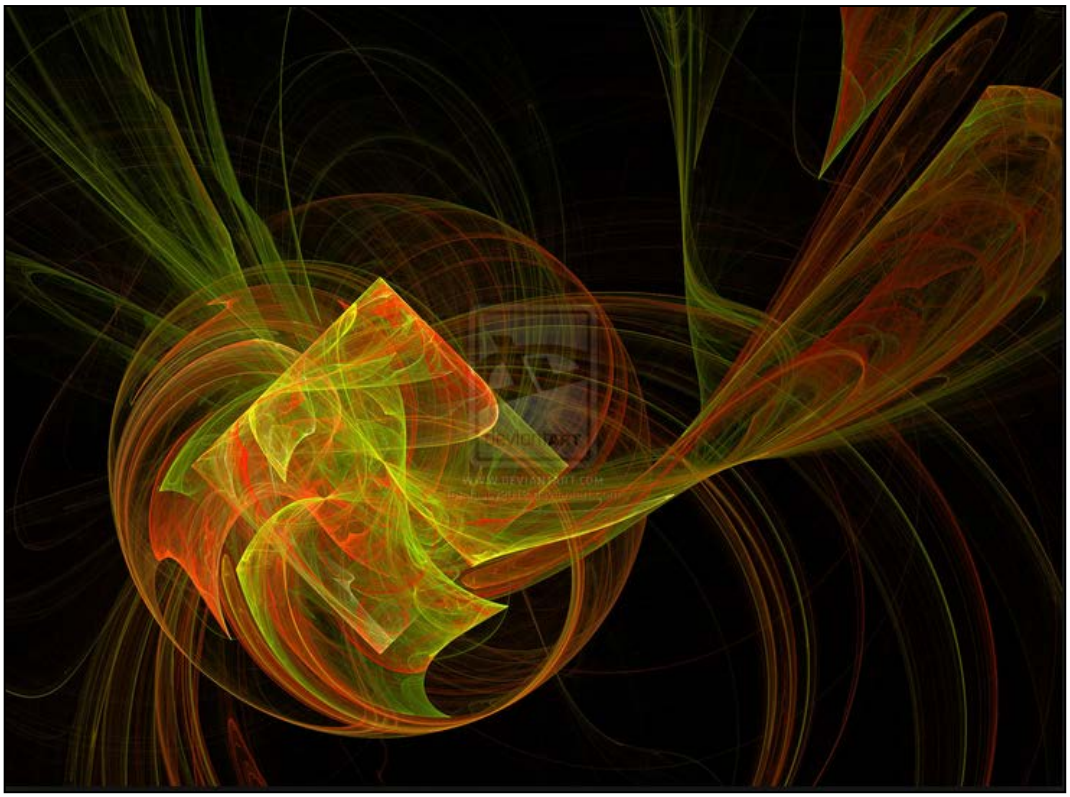
especialmente sobre el futuro."

Niels Bohr (1885-1962)

CAPÍTULO 9

CONCLUSIONES GENERALES

FUTURAS LÍNEAS DE INVESTIGACIÓN



1. INTRODUCCIÓN

El eje principal que ha articulado la presente investigación abarca **en primer lugar, el estudio a fondo del algoritmo S.G.O: de sus motivaciones físicas, su estructura y parámetros principales y su convergencia con funciones de distintas topologías y dimensiones. Y en segundo lugar la evaluación de su eficacia y eficiencia** probándolo con una serie de **40 *benchmark functions*** habituales en la literatura de optimización, varias de ellas multimodales. Parte importante de nuestro objetivo principal ha sido **alcanzar el óptimo en todas ellas** potenciando S.G.O. con diferentes hibridaciones. También hemos comprobado el alcance de las distintas hibridaciones en un **problema práctico real**: la instancia CASSINI 2.

Sin embargo, el trabajo realizado ha trascendido el objetivo central de la investigación, llevándonos a **diseñar y desarrollar nuevos heurísticos**, **basados en ideas inéditas**, que nos han llevado a replantearnos algunos conceptos en optimización continua, como el de buenas soluciones y con él, el de entornos de la solución óptima.

Tras nuestras experimentaciones hemos llegado a la conclusión de que una buena solución no tiene por qué estar en un entorno, tal como está definido matemáticamente, del óptimo global. Sobre todo en dimensiones altas puede suceder que alguna coordenada de una aproximación esté "lejos" de la correspondiente coordenada del óptimo y, sin embargo, ser una buena solución que, guiada por el algoritmo adecuado, nos lleve al óptimo global.

A lo largo de los siguientes apartados iremos exponiendo las diferentes aportaciones realizadas en cada etapa del estudio y desgranando los diferentes puntos de nuestro objetivo.

2. PRINCIPALES APORTACIONES

El algoritmo Space Gravitational Optimimization fue diseñado por Ying-Tung Hsiao, Cheng-Long Chuang, Joe-Air Jiang y Cheng-Chih Chien en el año 2005. Desde entonces, ya lo hemos comentado en el capítulo 1, no tenemos constancia de algún intento de justificación de su motivación o de su convergencia, o de algún estudio sobre la variación de sus parámetros. Tampoco tenemos constancia de que haya sido usado, solo o hibridado, para resolver algún problema, con o sin aplicación práctica, de optimización[1].

En ese sentido, nuestra aportación es extensa: en primer lugar, hemos llevado a cabo un **amplio estudio sobre su fundamentación física y matemática**. A continuación, hemos probado su **robustez**, tanto respecto a los valores de ejecución que representan el número de agentes de búsqueda, como el número de movimientos permitidos a cada agente, ***n*** **y** ***n_iter***. Una vez comprobada, hemos pasado a **buscar** una terna de valores, inexistente hasta el momento, para los parámetros específicos de S.G.O. α , β y G, con los que el algoritmo se mostrara robusto y eficaz, para **fijarlos de forma universal**, con independencia de la instancia elegida. Conseguido este objetivo, hemos reforzado S.G.O. **hibridándolo** con algoritmos de etiología diversa, algunos de búsqueda local: **Nelder-Mead y Gradiente**, otro de concentración de la región factible: **Segmentación** y los últimos de intensificación: **V.S.O. y Agujero de Gusano**. Con ellos hemos **alcanzado el óptimo en 40** ***benchmark functions*** con las que hemos probado las distintas hibridaciones. Como aportaciones adicionales debemos reseñar que, tanto **Segmentación** como **Agujero de Gusano**, **son algoritmos inéditos íntegramente diseñados y desarrollados por nosotros.** También hemos probado las distintas hibridaciones con un problema práctico real: la instancia CASSINI 2.

A continuación detallamos los capítulos específicos en los que puede encontrarse cada una de las aportaciones concretas:

- En el capítulo 2 hemos llevado a cabo un extenso análisis sobre sus motivaciones y hemos comprobado que el heurístico S.G.O. está bien fundamentado física y matemáticamente.
- En el capítulo 3 hemos realizado un estudio empírico del comportamiento de S.G.O. con 9 funciones de distintas topologías en dimensiones 2 y 4. Además, hemos realizado un primer estudio de robustez del algoritmo en función del parámetro *G*, constante de gravitación universal, usando las 9 funciones anteriores. También hemos llevado a cabo un análisis de robustez del heurístico en función de los parámetros de ejecución *n* y *n_iter*, número de asteroides y número de movimientos, con un grupo de 5 funciones bidimensionales.

[1] Hemos realizado una búsqueda en la base de datos del Web of Knowledge sobre S. G. O. en los últimos 5 años y no hemos encontrado ningún trabajo que lo utilice.

- En el capítulo 4 hemos desarrollado un pormenorizado estudio estadístico de variación de los parámetros principales de S.G.O. α, β y G, con el doble fin de determinar la robustez del algoritmo en función de su variación, por un lado, y, por otro hallar y fijar definitivamente la terna más consistente de ejecución para S.G.O. Hemos considerado un grupo de 6 instancias de diversas topologías, con dimensiones 3, 4, 6, 10, 20 y 30. Concluimos que la terna formada por los parámetros $\alpha = 0.005$, $\beta = 0.4$, $G = 10^5$ se muestra como la más robusta y eficaz para la implementación del algoritmo.

- En el capítulo 5, hemos elegido un grupo de 40 *benchmark functions* de muy diversas topologías y dimensiones para probar la eficacia de S.G.O. Para alcanzar el óptimo hemos hibridado S.G.O. con dos algoritmos tradicionales de búsqueda local: Nelder-Mead y Gradiente En primer lugar hemos considerado la hibridación S.G.O.-Nelder-Mead, con la que hemos conseguido encontrar el óptimo, atendiendo a nuestro criterio, para 33 de las 40 instancias planteadas. Posteriormente hemos probado la hibridación S.G.O.-Gradiente solo en los 7 casos restantes, obteniendo el óptimo en otras 2 instancias.

- En el capítulo 6 hemos innovado y desarrollado un inédito algoritmo de concentración, Segmentación, basado en técnicas de diseño secuencial de experimentos, con el que hemos segmentando la región factible. Lo hemos probado con las 5 instancias con las que no habíamos encontrado el óptimo según nuestro criterio. Con la nueva hibridación, no solo hemos logrado el óptimo en dos nuevas instancias, sino que hemos obtenido excelentes resultados para la función de SCHWEFEL[2] en dimensiones muy altas.

- En el capítulo 7 hemos desarrollado e implementado un innovador algoritmo basado en técnicas de intensificación denominado Agujero de Gusano. También hemos desarrollado los metaheurísticos S.G.O.-V.S.O.-A.G. y S.G.O.-SG.-A.G. con los que, finalmente, hemos obtenido el óptimo en las tres instancias en las que no habíamos obtenido éxito con las hibridaciones anteriores.

- En el capítulo 8 hemos probado las distintas hibridaciones con la instancia real obtenida de la E.S.A. CASSINI 2.

[2] En el capítulo 2 puede verse su representación gráfica tridimensional.

3. CONCLUSIONES GENERALES

S.G.O. es un algoritmo basado en una **inspiración consistente** y **bien fundamentada** en conceptos **físicos y matemáticos**.

Tras la ingente cantidad de ejecuciones llevadas a cabo, con instancias de muy diversa topología y dimensión, concluimos que S.G.O. es un **heurístico eficiente**. Además, las distintas hibridaciones planteadas lo potencian y le permiten subsanar eventuales deficiencias que pueda presentar con instancias sin suficientemente regularidad.

Con los valores determinados por nosotros para los parámetros α, β y G, se presenta como un algoritmo **eficaz y robusto**.

Hemos comprobado que los asteroides no quedan atrapados en óptimos locales ya que, hibridándolo con algoritmos locales ha conseguido llevar a los agentes de búsqueda al óptimo global en un porcentaje superior al 87% de las instancias planteadas. Posteriormente, con las hibridaciones con los algoritmos de concentración e intensificación, **el éxito ha llegado al 100%.**

Haber conseguido **fijar** los valores de los parámetros, α, β y G, para solo variar *n* y *n_iter* en caso necesario, resuelve las incertidumbres de cualquier usuario que desconozca S.G.O. y lo convierte en un **algoritmo universal** aplicable a cualquier instancia en cualquier dimensión con garantías de éxito. Así, sus posibilidades de aplicación se incrementan en gran medida.

En el presente trabajo hemos expuesto **cinco posibles hibridaciones**, tomando siempre como programa principal S.G.O. que, solas o combinadas, han potenciado su eficacia y eficiencia.

Para determinadas instancias, la hibridación con el algoritmo **Segmentación** puede resultar de gran utilidad y en todo caso proporciona una exploración inicial que puede ayudar a acotar la región en la que se encuentra el óptimo. Si bien, la filosofía que inspira esta heurística no es nueva, sí lo es la **forma de desarrollarla** y abre la puerta a **múltiples implementaciones** derivadas de ella.

La heurística **Agujero de Gusano** es una sencilla e inédita herramienta, perfectamente adaptada a S.G.O. con la que hemos conseguido resolver los casos más complicados. La idea que la inspira, así como su desarrollo, **son completamente innovadores** e inicia un nuevo campo de heurísticos basados en **manipulación de coordenadas**.

Respecto a los **tiempos de ejecución**.- A lo largo de este proyecto, hemos usado varios equipos y distintas versiones de MATLAB para realizar las experimentaciones; por ello los tiempos de ejecución presentan una **apariencia tan dispar**. Tras obtener los primeros resultados empezamos a profundizar en su eficiencia y comenzamos a tomar medidas sobre tiempos y evaluaciones de la función objetivo. Las diferencias en los tiempos de ejecución eran tan significativas, dependiendo sobre todo del equipo, pero también de la versión de MATLAB, que hemos llegado a la conclusión de que **no puede ser un parámetro fiable** para cuantificar la eficiencia de un algoritmo. En este sentido, creemos que el **número de evaluaciones de la función objetivo es el único posible cuantificador a tener en cuenta a la hora de evaluar la eficiencia de un algoritmo**. A partir del capítulo 3 en todos los resultados obtenidos hemos registrado tal valor.

4. FUTURAS LÍNEAS DE INVESTIGACIÓN

El trabajo presentado abre varias líneas de investigación que consideramos interesantes:

En primer lugar, dado que en la implementación de S.G.O. se utilizan valores aproximados para las derivadas parciales, nos parece interesante **realizar un estudio** de su comportamiento con funciones de **distintas regularidades**.

Además, el estudio de parámetros realizado en el capítulo 4 puede ampliarse y completarse con su **ajuste respecto al incremento de dimensión**.

Hemos expuesto cinco posibles hibridaciones para S.G.O., pero, obviamente, existen **muchas combinaciones de hibridación entre ellas**. Por ejemplo, hibridar Segmentación con Nelder-Mead, o con V.S.O., en vez de hibridarlo con Gradiente, etc. Desarrollarlas abre un ingente campo de actuación.

La aplicación de la **teoría de diseño de experimentos**, ya sea secuencial, como en el caso de Segmentación, o de diseño de experimentos factoriales fraccionales, herramienta inexplorada en nuestro estudio, a S.G.O. abre una línea de investigación, que, intuimos, puede llegar a mostrar un gran potencial.

La versatilidad de los dos algoritmos de concentración e intensificación diseñados en el presente trabajo de investigación, desarrollo e innovación, **SG.** y **A.G.**, permite que **puedan hibridarse a cualquier algoritmo** de optimización mejorando sus resultados. Las posibles futuras vías de desarrollo en este campo son amplias.

La filosofía que inspira el algoritmo Agujero de Gusano es innovadora y abre la puerta a **distintos algoritmos** en los que utilizar nuevos parámetros estadísticos, como la moda y nuevas ideas, como la **manipulación de coordenadas**.

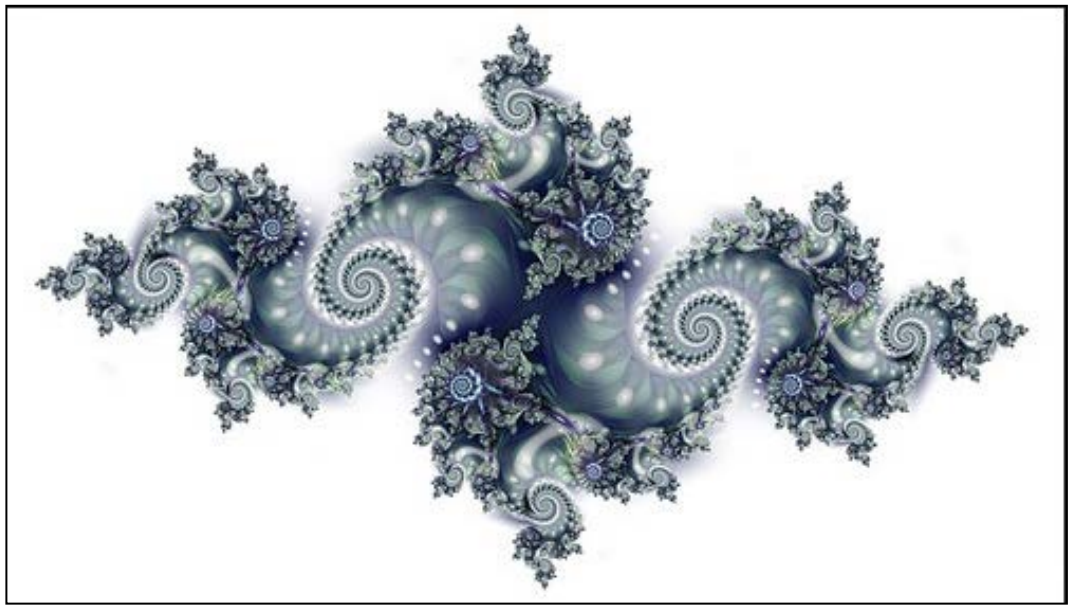
En definitiva, los resultados obtenidos nos hacen pensar en una interesante heurística rápida y eficaz que, ya sea sola o hibridada con los algoritmos expuestos, **puede resolver una extensa gama de problemas de optimización continua.**

Por último, la **aplicación** de S.G.O. y sus distintas hibridaciones en **problemas reales,** como la desarrollada en el capítulo 8 con la instancia CASSINI 2, abre un interesante campo de acción en el que realizar un estudio sobre su eficacia y con él obtener mejores resultados.

"¡Qué pobre memoria es aquélla que sólo funciona hacia atrás!"

Lewis Carroll (1832-1898)

APÉNDICES



APÉNDICE 1 *BENCHMARK FUNCTIONS*

FUNCIONES TEST UTILIZADAS

Nº	DIM	NOMBRE	EXPRESIÓN	DOMINIO OPTIMIZACIÓN	ÓPTIMO	ABSCISA
1	n	DE ACKLEY	$f(\bar{x}) = 20 + e - 20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)}$	$[-15,30]^n$	0	$(0,..,0,..,0)$
2	2	DE BEALE	$f(x_1,x_2) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2)^2 + (2.65 - x_1 + x_1x_2^3)^2$	$[-4.5,4.5]\times[-4.5,4.5]$	0	$(3, 0.5)$
3	2	DE BOOTH	$f(x_1, x_2) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$	$[-10,10]\times[-10,10]$	0	$(1,3)$
4	2	DE BRANIN	$f(x_1,x_2) = \left(x_2 - \frac{5}{4\pi^2}x_1^2 + \frac{5x_1}{\pi} - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)cos(x_1) + 10$	$[-5,15]\times[-5,15]$	0.397887	$(9.424646, 2.249820)^*$
5	2	B2	$f(x_1,x_2) = x_1^2 + 2x_2^2 - 0.3cos(3\pi x_1) - 0.4cos(4\pi x_2) + 0.7$	$[-50,100]\times[-50,100]$	0	$(0,0)$
6	4	DE COLVILLE	$f(x_1,x_2,x_3,x_4) = 100(x_2 - x_1^2)^2 + (1-x_1)^2 + 90(x_4 - x_3^2)^2 + (1-x_3)^2 + 10.1((x_2-1)^2 + (x_4-1)^2) + 19.8(x_2-1)(x_4-1)$	$[-100,100]\times[-100,100]$	0	$(1,1,1,1)$
7	n	DE DIXON & PRICE	$f(\bar{x}) = \sum_{i=1}^{n} i(2x_i^2 - x_{i-1})^2 + (x_1 - 1)^2$	$[-10,10]^n$	0	$\bar{x}_{MIN} = \left(1, 2^{\frac{1}{2^1}-1}, 2^{\frac{1}{2^2}-1}, ..., 2^{\frac{1}{2^i}-1}, ..., \pm 2^{\frac{1}{2^{dim}-1}}\right)$
8	2	DE EASOM	$f(x_1,x_2) = -cos(x_1)cos(x_2)e^{-(x_1-\pi)^2-(x_2-\pi)^2}$	$[-100,100]\times[-100,100]$	-1	$(-\pi, \pi)$
9	2	DE GOLDSTEIN & PRICE	$f(x_1,x_2) = \left(1 + (x_1+x_2+1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right)$ $\left(30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right)$	$[-2,2]\times[-2,2]$	3	$(0,-1)$
10	n	DE GRIEWANK	$f(\bar{x}) = \sum_{I=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-300,600]^n$	0	$(0,..,0,..,0)$
11	3	DE HARTMANN (3,4)	$f(\bar{x}) = -\sum_{I=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$	$[0,1]^3$	0	$(0.114614, 0.555649, 0.852547)$
	i	a_{ij}	c_i	p_{ij}		
	1	3.0 10.0 30.0	1	0.3689 0.1170 0.2673		
	2	0.1 10.0 35.0	1.2	0.4699 0.4387 0.7470		
	3	3.0 10.0 30.0	3	0.1091 0.8732 0.5574		
	4	0.1 10.0 35.0	3.2	0.0381 0.5743 0.8828		
12	6	DE HARTMANN (6,4)	$f(\bar{x}) = -\sum_{I=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$	$[0,1]^6$	0	(0.20169 , 0.150011 , 0.47687,... ,...0.275332 , 0.311652 , 0.6573)
	i	a_{ij}	c_i	p_{ij}		
	1	10.0 3.0 17.0 3.5 1.7 8.0	1	0.1312 0.1696 0.5569 0.0124 0.8283 0.5886		
	2	0.05 10.0 17.0 0.1 8.0 14.0	1.2	0.2329 0.4135 0.8307 0.3736 0.1004 0.9991		
	3	3.0 3.5 1.7 10.0 17.0 8	3	0.2348 0.1451 0.3522 0.2883 0.3047 0.6650		
	4	17.0 8.0 0.05 10.0 0.1 14	3.2	0.4047 0.8828 0.8732 0.5743 0.1091 0.0381		
13	n	DE JONG (SPHERE)	$f(\bar{x}) = \sum_{I=1}^{n} x_i^2$	$[-2.56,5.12]^n$	0	$(0,..,0,..,0)$
14	n	DE LEVY	$f(\bar{x}) = \sin^2\left(\pi\left(1 + \frac{x_1-1}{4}\right)\right)^2 + \sum_{i=1}^{n-1}\left(\frac{x_i-1}{4}\right)^2\left(1 + 10\sin^2\left(\pi\left(1+\frac{x_i-1}{4}\right)+1\right)\right) +$ $+\left(\frac{x_n-1}{4}\right)^2(1+\sin^2(2\pi x_n))$	$[-10,10]^n$	0	$(1,..,1,..,1)$
15	2	DE MATYAS	$f(x_1,x_2) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$	$[-5,10]\times[-5,10]$	0	$(0,0)$
16	4	PERM0 (4,10)	$f(\bar{x}) = \sum_{k=1}^{n}\left(\sum_{i=1}^{n}(i+10)\left(x_i^k - \left(\frac{1}{i}\right)^k\right)\right)^2$	$[-4,4]^4$	0	$(1, 1/2, 1/3, 1/4)$
17	4	PERM (4,0.5)	$f(\bar{x}) = \sum_{k=1}^{n}\left(\sum_{i=1}^{n}(i^k + 0.5)\left(\left(\frac{x_i}{i}\right)^k - 1\right)\right)^2$	$[-4,4]^4$	0	$(1,2,3,4)$

APÉNDICE 1 *BENCHMARK FUNCTIONS*

18	n	DE POWELL	$f(\bar{x})=\sum_{j=1}^{n/4}(x_{4j-3}+10x_{4j-2})^2+5(x_{4j-1}-x_{4j})^2+$ $+(x_{4j-2}-2x_{4j-1})^4+10(x_{4j-3}-x_{4j})^4$	$[-4,5]^n$	0	$(3,-1,0,1,3..,3,-1,0,1)$
19	4	POWER SUM(8,18,44,114)	$f(x_1,x_2,x_3,x_4)=\left(\sum_{i=1}^{4}x_i-8\right)^2+\left(\sum_{i=1}^{4}x_i^2-18\right)^2+$ $+\left(\sum_{i=1}^{4}x_i^3-44\right)^2+\left(\sum_{i=1}^{4}x_i^4-114\right)^2$	$[0,4]^4$	0	$(1,2,2,3)$
20	n	DE RASTRIGIN	$f(\bar{x})=10n+\sum_{i=1}^{n}\left(x_i^2-10\cos(2\pi x_i)\right)$	$[-5.12,5.12]^n$	0	$(0,..,0,..,0)$
21	n	DE ROSENBROCK	$f(\bar{x})=\sum_{i=1}^{n/2}100\left(x_{2i}-x_{2i-1}^2\right)^2+(1-x_{2i-1})^2$	$[-10,10]^n$	0	$(1,..,1,..,1)$
22	n	DE SCHWEFEL	$f(\bar{x})=418.9829n+\sum_{i=1}^{n}(-x_i\sin\sqrt{\lvert x_i\rvert})$	$[-500,500]^n$	0	$x_i=420.9687469543275\ \ \forall i=1,...,n$
	4	DE SHEKEL(n)	$f(\bar{x})=-\sum_{i=1}^{n}\left(\sum_{j=1}^{4}(x_i-a_{ij})^2+c_i\right)^{-1}$	$[0,10]^4$		
	i	a_{ij}	c_i			
	1	4.0 4.0 4.0 4.0	0.1			
	2	1.0 1.0 1.0 1.0	0.2			
	3	8.0 8.0 8.0 8.0	0.2			
	4	6.0 6.0 6.0 6.0	0.4			
	5	3.0 7.0 3.0 7.0	0.4			
	6	2.0 9.0 2.0 9.0	0.6			
	7	5.0 5.0 3.0 3.0	0.3			
	8	8.0 1.0 8.0 1.0	0.7			
	9	6.0 2.0 6.0 2.0	0.5			
	10	7.0 3.6 7.0 3.6	0.5			
23	4	DE SHEKEL(5)			-10.1532	$(4,4,4,4)$
24	4	DE SHEKEL(7)			-10.4029	$(4,4,4,4)$
25	4	DE SHEKEL(10)			-10.5364	$(4,4,4,4)$
26	2	DE SHUBERT	$f(x_1,x_2)=\left(\sum_{i=1}^{5}i\cos((i+1)x_1+i)\right)\left(\sum_{i=1}^{5}i\cos((i+1)x_2+i)\right)$	$[-10,10]\times[-10,10]$	-186.7309	$(0.0217,-0.9527)$
27	2	SIX HUMP CAMEL BACK	$f(x_1,x_2)=4x_1^2-2.1\cdot x_1^4+0.33x_1^6+x_1x_2-4x_2^2+4x_2^4$	$[-5,5]\times[-5,5]$	-1.0316285	$(0.08984,-0.712659)$
28	2	SPACE-PAPER	$f(x_1,x_2)=x_1^4-16x_1^2+0.5x_1+x_2^4-16x_2^2+0.5x_2$	$[-50,50]\times[-50,50]$	-130.832323	$(-2.836207,-2.836207)$
29	n	SUM SQUARES	$f(\bar{x})=\sum_{I=1}^{n}ix_i^2$	$[-5,10]^n$	0	$(0,..,0,..,0)$
	n	DE TRID (n)	$f(\bar{x})=\sum_{i=1}^{n}(x_i-1)^2-\sum_{i=2}^{n}x_ix_{i-1}$	$[-n^2,n^2]^n$		
30	6	DE TRID (6)			-50	$(6,10,12,12,10,6)$
31	10	DE TRID (10)			-210	$(10,18,24,28,30,30,28,24,18,10)$
32	n	DE ZAKHAROV	$f(\bar{x})=\sum_{i=1}^{n}x_i^2+\left(\sum_{i=1}^{n}0.5\,i\,x_i\right)^2+\left(\sum_{i=1}^{n}0.5\,i\,x_i\right)^4$	$[-5,10]^n$	0	$(0,..,0,..,0)$

APÉNDICE 2. CÓDIGOS EN MATLAB DE LOS ALGORITMOS USADOS

2.1. SUBFUNCIONES S.G.O.

```
function
[OPTIMOS,TIEMPO,efo,CONT,alfa,beta,g,n,niter]=gravi4(nveces)

%programa principal gravi
%
%
%xopt es la posicion del optimo y fopt es el valor de la funcion
objetivo en
%el  optimo
%guardo los optimos en XFopt
%r1 y r2 definen la zona de factibilidad
%rd es el incremento para aproximar el gradiente
%n es el numero de asteroides
%xc son las coordenadas del centro de gravedad
%alfa , beta y g son los parametros de S.G.O.
%niter es el numero de iteraciones
%dim es la dimension de la funcion objetivo
%x es una matriz de tamaño nx2 con las posiciones de los asteroides
%v es una matriz de tamaño nx2 con las velocidades de los
asteroides
%
%val es el vector resultado de evaluar f en cada fila de x
%indice es el lugar donde se alcanza el optimo
%
%x1 y v1 guardan los valores iniciales por si es interesante
repetir con valores fijos
% y en ese caso ejecutamos gravi2.
%
%contador indica cuantos asteroides se van fuera de la zona de
factibilidad
%nveces es el número de veces que ejecutamos el algoritmo completo
%
% Tiempo guarda los tiempos en cada fila
% OPTIMOs es una matriz de p filas que guarda las abscisas y
ordenadas
%
% CONT guarda las infactibilidades en cada ejecución
%
%
%
%[OPTIMOS,TIEMPO,efo,CONT,alfa,beta,g,n,niter]=gravi4(nveces)
%
% MARÍA JOSÉ ZAPATERO MORENO

% Inicializacion de los parámetros HAY QUE CAMBIARLOS EN CADA
CASO!!!!

global EFO
dim=30;'nº de variables';
n=100;'nº de asteroides';
niter=1000;' nº de movimientos de cada asteroide';
alfa=0.005;
beta=0.99;
g=1000;
r1=-15;
```

```
r2=30;
rd=0.000001;


TIEMPO=zeros(nveces,1);
CONT=zeros(nveces,1);
OPTIMOS=zeros(nveces,dim+1);
efo=zaros(nveces,1);


for p=1:nveces
cont=0;
tic
EFO=0;


'funcion de inicializacion aleatoria de espacios y velocidades';
[x,v]=inicial(n,r1,r2,dim);



'primer valor del optimo';

val=zeros(n,1);

for i=1:n
   val(i)=f(x(i,:));
end


[fopt,indice]=min(val);
xopt=x(indice,:);

'en XFopt guardamos POR ESTE ORDEN x y el óptimo';

XFopt=[xopt,fopt];

'calculo del centro de gravedad';

[xc]=cengrav(x,n,dim);


'calculo de las aceleraciones';

a=acelera(x,dim,g,alfa,xc,n,rd);


'calculo de las sucesivas velocidades y posiciones';


  for j=2:niter


    'actualizamos veloc y posiciones empiezo las iteraciones';

    [x,v]=velpos(beta,a,x,v);
```

```
      'comprobacion de la factibilidad';

      [x,v,cont]=facti(x,v,r1,r2,n,cont,dim);


       'nuevo optimo';

       for i=1:n
         [val(i)]=f(x(i,:));
       end


        [fopt1,indice1]=min(val);
        xopt1=x(indice1,:);

         if fopt1<fopt
             fopt=fopt1;
             xopt=xopt1;
       %XFopt=[XFopt;[xopt,fopt]]; por si necesito todos los óptimos
             XFopt=[xopt,fopt];
         end




' actualizamos aceleraciones';

    'calculo del nuevo centro de gravedad';

    [xc]=cengrav(x,n,dim);


    'calculo de las nuevas aceleraciones';

    a=acelera(x,dim,g,alfa,xc,n,rd);

  end
TIEMPO(p)= toc;
OPTIMOS(p,:)=XFopt;
CONT(p)=cont;
efo(p)=EFO;

 end



-------------------------------------------------------------------------------------------------------------------------------



function a=acelera(x,dim,g,alfa,xc,n,rd)
% rest es el vector resta entre cada posicion y el centro de
gravedad.
% dist es su norma
```

```
% a es una matriz con las aceleraciones de cada asteroide por tanto
tiene
% el mismo tamaño que x y que v
%
% MARÍA JOSÉ ZAPATERO MORENO


id=eye(dim);
a=zeros(n,dim);

'calculo de la distancia de cada x al centro de gravedad';
for i=1:n

    rest=x(i,:)-xc;
    dist=norm(rest);

    for j=1:dim


    a(i,j)=g*(f(x(i,:)-rd*id(j,:))-
f(x(i,:)+rd*id(j,:)))+alfa*xc(j)/dist;


    end

end
```

```
function [xc]=cengrav(x,n,dim)
% calculo del centro de gravedad de las posiciones
% xc es un vector fila que contiene la componentes (x,y,x,...) del
centro
% de gravedad
%
% MARÍA JOSÉ ZAPATERO MORENO


xc=zeros(1,dim);

for i=1:dim
xc(i)=sum(x(:,i))./n;
```

```
function [x,v,cont]=facti(x,v,r1,r2,n,cont,dim)
% esta funcion comprueba la factibilidad del punto encontrado y si
no lo es
% la reinicia de forma aleatoria
% guarda un contador con el numero de asteroides escapados.
```

```
% solo cambio el que se escapa
%
% MARÍA JOSÉ ZAPATERO MORENO



for i=1:n



            if(min(x(i,:)))<r1  ;'asteroide fuera
factibilidad!!!!!!!!!!  ' ;
                'lo inicializamos';cont=cont+1;

             x(i,:)=r1+(r2-r1)*rand(1,dim);
             v(i,:)=r1+(r2-r1)*rand(1,dim);

                elseif (max(x(i,:)))>r2 ;'asteroide fuera
factibilidad!!  ' ;
                'lo inicializamos';cont=cont+1;


             x(i,:)=r1+(r2-r1)*rand(1,dim);
             v(i,:)=r1+(r2-r1)*rand(1,dim);
             end

 end
```

```
function [x,v]=inicial(n,r1,r2,dim)
%esta funcion inicializa las posiciones y velocidades de los
asteroides
%n es el numero de asteroides
%r1 y r2 son los limites de factibilidad TIENEN QUE SER NºENTEROS
(supongo r1 menor que r2)
%x es una matriz nxdim con las posiciones de los n asteroides
%v es una matriz nxdim con las velocidades iniciales de los n
asteroides
% en principio acoto las velocidades entre -2 y 3
%
%[x,v]=inicial(n,r1,r2,dim)
%
% MARÍA JOSÉ ZAPATERO MORENO



%se supone que todas las variables varian en el mismo intervalo

x=r1+(r2-r1)*rand(n,dim);
v=r1+(r2-r1)*rand(n,dim);
```

```
function [x,v]=velpos(beta,a,x,v)


v=a+beta*v;
x=x+v;
```

```
%function y=f(x)
function Z=f(X)
%
% EN ESTE ARCHIVO ESTÁN REGISTRADAS TODAS LAS FUNCIONES DE LAS
PRUEBAS CON
% S.G.O.
% HAY QUE MODIFICAR Y GUARDAR EL ARCHIVO EN CADA CASO
% HAY 2 FUNCTION DEPENDIENDO DE CÓMO ESTÉ METIDA LA VARIABLE
DEPENDIENTE
%
% EFO es una variable global que cuenta las llamadas a la función
objetivo
%
%
%Z=f(X) O y=f(x)
%
% MARÍA JOSÉ ZAPATERO MORENO
global EFO
EFO=EFO+1;



% *********FUNCIONES DE DIMENSION 2******************

%funcion space-paper
%Y=X.^4-16.*X.^2+0.5.*X;
%Z=sum(Y);

% funcion Branin
%Z=(X(:,2)-(5/(4*pi^2)).*X(:,1).^2+(5/pi).*X(:,1)-6).^2+10*(1-
1/(8*pi))*cos(X(:,1))+10-0.397887359930301;

% funcion Schwefel
Y=(X.*sin(sqrt(abs(X))));
Z=-sum(Y)+837.965774544868;
%Z=-sum(Y);

%funcion easom
%Z=-cos(X(:,1)).*cos(X(:,2)).*exp(-((X(:,1)-pi).^2+(X(:,2)-
pi).^2))+1;

%funcion shubert (2)
%Z=(cos(2*X(:,1)+1)+2*cos(3*X(:,1)+2)+3*cos(4*X(:,1)+3)+4*cos(5*X(:
,1)+4)+5*cos(6*X(:,1)+5)).*(cos(2*X(:,2)+1)+2*cos(3*X(:,2)+2)+3*cos
(4*X(:,2)+3)+4*cos(5*X(:,2)+4)+5*cos(6*X(:,2)+5))+
186.730908831024;
```

```
%funcion rosenbrock
%Z=100*(X(:,2)-X(:,1).^2).^2+(1-X(:,1)).^2;

%funcion Griewank
%Z=1/4000*(X(:,1).^2+X(:,2).^2)-
(cos(X(:,1)).*cos(X(:,2))/sqrt(2))+1;


%funcion B2
%Z=X(:,1).^2+2*X(:,2).^2-0.3*cos(3*pi*X(:,1))-
0.4*cos(4*pi*X(:,2))+0.7;

%funcion Goldstein and Price
%Z=(1+(X(:,1)+X(:,2)+1).^2*(19-14*X(:,1)+3*X(:,1).^2-
14*X(:,2)+6*X(:,1)*X(:,2)+3*X(:,2).^2))*(30+(2*X(:,1)-
3*X(:,2)).^2*(18-32*X(:,1)+12*X(:,1).^2+48*X(:,2)-
36*X(:,1)*X(:,2)+27*X(:,2).^2))-3;

%funcion Beale
%Z=(1.5-X(:,1)+X(:,1)*X(:,2)).^2+(2.25-
X(:,1)+X(:,1)*X(:,2).^2).^2+(2.625-X(:,1)+X(:,1)*X(:,2).^3).^2;

%funcion Booth
%Z=(X(:,1)+2*X(:,2)-7).^2+(2*X(:,1)+X(:,2)-5).^2;

%funcion Matyas
%Z=0.26*(X(:,1).^2+X(:,2).^2)-0.48*X(:,1)*X(:,2);

%funcion SixHumpCamelBack
%Z=4*X(:,1).^2-2.1*X(:,1).^4+1/3*X(:,1).^6+X(:,1).*X(:,2)-
4*X(:,2).^2+4*X(:,2).^4+1.0316285;

%'funcion Zakharov (2)'
%Y=1:length(X);
%Z=sum(X.^2)+sum(0.5*X.*Y)^2+sum(0.5*X.*Y)^4;



%********FUNCIONES DE DIMENSION 3***********

% funcion Hartman(3,4)
%Z=-exp(-(3.*(X(:,1)-0.6899).^2+10.*(X(:,2)-0.1170).^2+30.*(X(:,3)-
0.2673).^2))-1.2*exp(-(0.1*(X(:,1)-0.4699).^2+10.*(X(:,2)-
0.4387).^2+35.*(X(:,3)-0.7470).^2))-3.*exp(-(3*(X(:,1)-
0.1091).^2+10*(X(:,2)-0.8732).^2+30*(X(:,3)-0.5547).^2))-3.2.*exp(-
(0.1*(X(:,1)-0.0381).^2+10*(X(:,2)-0.5743).^2+35*(X(:,3)-
0.8828).^2));
%function y = hart3(x)
%
% Hartmann function
% Matlab Code by A. Hedar (Sep. 29, 2005).
% The number of variables n = 3.
%
%a(:,2)=10.0*ones(4,1);
%for j=1:2;
%   a(2*j-1,1)=3.0; a(2*j,1)=0.1;
%   a(2*j-1,3)=30.0; a(2*j,3)=35.0;
%end
%c(1)=1.0;c(2)=1.2;c(3)=3.0;c(4)=3.2;
```

```
%p(1,1)=0.36890;p(1,2)=0.11700;p(1,3)=0.26730;
%p(2,1)=0.46990;p(2,2)=0.43870;p(2,3)=0.74700;
%p(3,1)=0.10910;p(3,2)=0.87320;p(3,3)=0.55470;
%p(4,1)=0.03815;p(4,2)=0.57430;p(4,3)=0.88280;
%s = 0;
%for i=1:4;
%   sm=0;
%   for j=1:3;
%      sm=sm+a(i,j)*(x(j)-p(i,j))^2;
%   end
%   s=s+c(i)*exp(-sm);
%end
%y = -s+3.86278214782076;



%funcion De Joung (3)
%Z=sum(X.^2);



%*********FUNCIONES DE DIMENSION 4********

%funcion Colville
%Z=100*(X(:,2)-X(:,1).^2).^2+(1-X(:,1)).^2+90*(X(:,4)-
X(:,3).^2).^2+(1-X(:,3)).^2+10.1*((X(:,2)-1).^2+(X(:,4)-
1).^2)+19.8*(X(:,2)-1)*(X(:,4)-1);


%funcion Shekel
%
% Shekel function
% Matlab Code by A. Hedar (Nov. 23, 2005).
% The number of variables n = 4
% The parameter m should be adjusted m = 5,7,10.
% The default value of m = 10.
%
%m = 10;
%a = ones(10,4);
%a(1,:) = 4.0*a(1,:);
%a(2,:) = 1.0*a(2,:);
%a(3,:) = 8.0*a(3,:);
%a(4,:) = 6.0*a(4,:);
%for j = 1:2;
%  a(5,2*j-1) = 3.0; a(5,2*j) = 7.0;
%   a(6,2*j-1) = 2.0; a(6,2*j) = 9.0;
%   a(7,j)     = 5.0; a(7,j+2) = 3.0;
%   a(8,2*j-1) = 8.0; a(8,2*j) = 1.0;
%   a(9,2*j-1) = 6.0; a(9,2*j) = 2.0;
%   a(10,2*j-1)= 7.0; a(10,2*j)= 3.6;
%end
%c(1) = 0.1; c(2) = 0.2; c(3) = 0.2; c(4) = 0.4; c(5) = 0.4;
%c(6) = 0.6; c(7) = 0.3; c(8) = 0.7; c(9) = 0.5; c(10)= 0.5;
%s = 0;
%for j = 1:m;
%   p = 0;
%   for i = 1:4
%      p = p+(x(i)-a(j,i))^2;
%   end
%   s = s+1/(p+c(j));
```

```
%end

%Shekel 5
%y = -s+10.1532;

%Shekel 7
%y=-s+10.4029405668187;

%Shekel 10
%y = -s+10.536409816692;


%funcion Perm
%n=4;
%b=0.5;
%s_out=0;
%for k=1:n
%    s_in=0;
%    for j=1:n
%        s_in=s_in+(j^k+b)*((x(j)/j)^k-1);
%    end
%    s_out=s_out+s_in^2;
%end
%y=s_out;


%funcion perm0
%n=2;
%b=10;
%s_out=0;
%for k=1:n
%    s_in=0;
%    for j=1:n
%        s_in=s_in+(j+b)*(x(j)^k-(1/j)^k);
%    end
%    s_out=s_out+s_in^2;
%end
%y=s_out;


%funcion powersum
%n=4;
%b=[8,18,44,114];
%s_out=0;
%for k=1:n
%    s_in=0;
%    for j=1:n
%        s_in=s_in+x(j)^k;
%    end
%    s_out=s_out+(s_in-b(k))^2;
%end
%y=s_out;


%funcion rosenbrock 4
%Z=100*(X(:,2)-X(:,1).^2).^2+(1-X(:,1)).^2+100*(X(:,3)-
X(:,2).^2).^2+(1-X(:,2)).^2+100*(X(:,4)-X(:,3).^2).^2+(1-
X(:,3)).^2;
```

```
%funcion colville
%Z= 100*(X(:,1).^2-X(:,2)).^2+(X(:,1)-1).^2+(X(:,3)-
1).^2+90*(X(:,3).^2-X(:,4)).^2+10.1.*((X(:,2)-1).^2+(X(:,4)-
1).^2)+19.8*(X(:,2).^-1).*(X(:,4)-1);



%***************FUNCIONES DE DIMENSION 5***********************

%Funcion michalewics
 %n = 5;
 %s = 0*ones(size(X,1),1);
 %for i = 1:n
  %  s = s+sin(X(:,i)).*(sin(i*X(:,i).^2/pi)).^20;
  %end
 %Z = -s;


 %*****************FUNCIONES DE DIMENSION
6************************
%function y = hart6(x)
%
% Hartmann function
% Matlab Code by A. Hedar (Sep. 29, 2005).
% The number of variables n = 6.
%

%a(1,1)=10.0;   a(1,2)=3.0;     a(1,3)=17.0;    a(1,4)=3.5;
a(1,5)=1.7;     a(1,6)=8.0;
%a(2,1)=0.05;   a(2,2)=10.0;    a(2,3)=17.0;    a(2,4)=0.1;
a(2,5)=8.0;     a(2,6)=14.0;
%a(3,1)=3.0;        a(3,2)=3.5;     a(3,3)=1.7;     a(3,4)=10.0;
a(3,5)=17.0;    a(3,6)=8.0;
%a(4,1)=17.0;   a(4,2)=8.0;     a(4,3)=0.05;    a(4,4)=10.0;
a(4,5)=0.1;     a(4,6)=14.0;
%c(1)=1.0;c(2)=1.2;c(3)=3.0;c(4)=3.2;
%p(1,1)=0.1312; p(1,2)=0.1696;  p(1,3)=0.5569;  p(1,4)=0.0124;
p(1,5)=0.8283;  p(1,6)=0.5886;
%p(2,1)=0.2329; p(2,2)=0.4135;  p(2,3)=0.8307;  p(2,4)=0.3736;
p(2,5)=0.1004;  p(2,6)=0.9991;
%p(3,1)=0.2348; p(3,2)=0.1451;  p(3,3)=0.3522;  p(3,4)=0.2883;
p(3,5)=0.3047;  p(3,6)=0.6650;
%p(4,1)=0.4047; p(4,2)=0.8828;  p(4,3)=0.8732;  p(4,4)=0.5743;
p(4,5)=0.1091;  p(4,6)=0.0381;
%s = 0;
%for i=1:4;
%   sm=0;
%   for j=1:6;
%      sm=sm+a(i,j)*(x(j)-p(i,j))^2;
%   end
%   s=s+c(i)*exp(-sm);
%end
%y = -s+3.32237;


% FUNCION TRID 6
% function  y = trid(x)
%
```

```
% Trid function
% Matlab Code by A. Hedar (Nov. 23, 2005).
% The number of variables n should be adjusted below.
% The default value of n = 10.
%
%n = 6;
%s1 = 0;
%s2 = 0;
%for j = 1:n;
%   s1 = s1+(x(j)-1)^2;
%end
%for j = 2:n;
%   s2 = s2+x(j)*x(j-1);
%end
%y = s1-s2;


 % funcion Schwefel 6
 %Y=sum(X.*sin(sqrt(abs(X))));
 %Z=418.9828872*6-Y;
 %Z=2513.8973232-Y;


%**********************FUNCIONES DE DIMENSION
10**********************
% FUNCION TRID 10
% function  y = trid(x)
%
% Trid function
% Matlab Code by A. Hedar (Nov. 23, 2005).
% The number of variables n should be adjusted below.
% The default value of n = 10.
%
%n = 10;
%s1 = 0;
%s2 = 0;
%for j = 1:n;
%    s1 = s1+(x(j)-1)^2;
%end
%for j = 2:n;
%    s2 = s2+x(j)*x(j-1);
%end
%y = s1-s2+210;


% FUNCION RASTRIGIN-n
%function y = rast(x)
%
% Rastrigin function
% Matlab Code by A. Hedar (Nov. 23, 2005).
% The number of variables n should be adjusted below.
% The default value of n = 2.
%
%n = 20;
%s = 0;
%for j = 1:n
%   s = s+x(j)^2-10*cos(2*pi*x(j));
%end
%y = 10*n+s;
```

```
%FUNCION GRIEWANK n

%function y = griewank(x)
%
% Griewank function
% Matlab Code by A. Hedar (Sep. 29, 2005).
% The number of variables n should be adjusted below.
% The default value of n =2.
%
%n = 20;
%fr = 4000;
%s = 0;
%p = 1;
%for j = 1:n; s = s+x(j)^2; end
%for j = 1:n; p = p*cos(x(j)/sqrt(j)); end
%y = s/fr-p+1;


% FUNCION SUM SQUARES 10
%y=1*x(1)^2;
%for i=2:10
%    y=y+i*x(i)^2;
%end


%FUNCION ZAKHAROV n
%
% Zakharov function
% Matlab Code by A. Hedar (Nov. 23, 2005).
% The number of variables n should be adjusted below.
% The default value of n = 2.
%
%n = 2;
%s1 = 0;
%s2 = 0;
%for j = 1:n;
%    s1 = s1+x(j)^2;
%    s2 = s2+0.5*j*x(j);
%end
%y = s1+s2^2+s2^4;



%************FUNCIONES DE DIMENSION 20***********
%
%
%function y = rosen(x)
%
% Rosenbrock function n
% Matlab Code by A. Hedar (Nov. 23, 2005).
% The number of variables n should be adjusted below.
% The default value of n = 2.
%
%n = 10;
%sum = 0;
%for j = 1:n-1;
%    sum = sum+100*(x(j)^2-x(j+1))^2+(x(j)-1)^2;
```

```
%end
%y = sum;

%***************FUNCIONES DE DIMENSION 30*****************

%funcion De Joung (30)
%Z=sum(X.^2);


%***************FUNCIONES DE DIMENSION MAYOR QUE
20*****************

%function y = ackley(x)n
%
% Ackley function.
% Matlab Code by A. Hedar (Sep. 29, 2005).
% The number of variables n should be adjusted below.
% The default value of n =2.
%
%n = 30;
%a = 20; b = 0.2; c = 2*pi;
%s1 = 0; s2 = 0;
%for i=1:n;
%   s1 = s1+x(i)^2;
%   s2 = s2+cos(c*x(i));
%end
%y = -a*exp(-b*sqrt(1/n*s1))-exp(1/n*s2)+a+exp(1);
%quito la raíz cuadrada a ver cómo va.
%y = -a*exp(-b*(1/n*s1))-exp(1/n*s2)+a+exp(1);



%funcion powell n
%
% Powell function
% Matlab Code by A. Hedar (Nov. 23, 2005).
% The number of variables n should be adjusted below.
% The default value of n = 24.
%n = 24;m = n;
%for i = 1:m/4
%    fvec(4*i-3) = x(4*i-3)+10*(x(4*i-2));
%    fvec(4*i-2) = sqrt(5)*(x(4*i-1)-x(4*i));
%    fvec(4*i-1) = (x(4*i-2)-2*(x(4*i-1)))^2;
%    fvec(4*i)   = sqrt(10)*(x(4*i-3)-x(4*i))^2;
%end;
%fvec = fvec';
%y = norm(fvec)^2;


%FUNCION DIXON AND PRICE n
%function y = dp(x)
%
% Dixon and Price function.
% Matlab Code by A. Hedar (Sep. 29, 2005).
% The number of variables n should be adjusted below.
% The default value of n = 25.
%
%n = 25;
%s1 = (x(1)-1)^2;
```

```
%for j = 2:n;
%   s1 = s1+j*(2*x(j)^2-x(j-1))^2;
%end
%y = s1;



%function y = levy(x) n
%
% Levy function
% Matlab Code by A. Hedar (Nov. 23, 2005).
% The number of variables n should be adjusted below.
% The default value of n =2.
%
%n = 30;
%for i = 1:n; z(i) = 1+(x(i)-1)/4; end
%s = sin(pi*z(1))^2;
%for i = 1:n-1
%    s = s+(z(i)-1)^2*(1+10*(sin(pi*z(i)+1))^2);
%end
%y = s+(z(n)-1)^2*(1+(sin(2*pi*z(n)))^2);
```

2.2. SUBFUNCIONES NELDER-MEAD

```
function [OPTIMO]=nelme(v,valnel)
% esta funcion calcula el algoritmo Nelder-Mead para una funcion
objetivo
% que se encuentra en el fichero f.m y que tiene n variables
% v es una matriz (dim+1 x dim) que contiene, por filas los dim+1
puntos de
% dimension dim
% para construir cada simplex
% punto es el optimo obtenido vector de tamaño n y funcion el valor
de la funcion objetivo
%
%v_nel es el número de veces que se hace nelme
% [OPTIMO]=nelme(v)
%
% MARÍA JOSÉ ZAPATERO MORENO


% Valores a los parametros de reflexion, contraccion, expansion y
% encogimiento

ro=1;chi=2; gama=1/2; sigma=1/2;
w=zeros(length(v));

%error=1000;
tic

for k=1:valnel


    % iteracion k-esima (hago valnel iteraciones)
```

```
    % construccion del simplex ordenando los puntos

    n=length(v)-1;

        for i=1:length(v)
            w(i)=f(v(i,:));
        end


    % el vector waux da los valores de f ordenados de menor a mayor
    % en la matriz indice se guardan los indices.
    % en vaux estan los nodos ordenados

        [waux,indice]=sort(w);

        for i=1:length(indice)
            vaux(i,:)=v(indice(i),:);
        end

     % calculo del centro de gravedad de los buenos

            c=mean(vaux(1:length(vaux)-1,:));

    % Reflexion

     [xrefle,frefle]=reflexion(vaux,c,ro);


        if  frefle < waux(1) %hemos mejorado estrictamente al mejor

            % expansion

            [xexpan,fexpan]=expansion(xrefle,c,chi);


            if fexpan < frefle %hemos mejorado estrictamente al
reflejado

                % cambiamos al peor por el expandido y volvemos al
                % principio

                v(indice(length(indice)),:)=xexpan;

            else %no hemos mejorado estrictamente al reflejado

                % cambiamos al peor por el reflejado y volvemos al
                % principio

                v(indice(length(indice)),:)=xrefle;
            end

        elseif frefle >= waux(length(waux)-1) %hemos empeorado o
igualado al ultimo de los buenos
```

```
            %contraccion que puede ser hacia adentro o hacia afuera

            if frefle < waux(length(waux)) %hemos mejorado
estrictamente al peor

                % contraccion hacia afuera (contraemos con el
reflejado)

                [xcontracf,fcontracf]=contracfu(xrefle,c,gama);

                if fcontracf <= frefle %hemos igualado o mejorado
al reflejado

                    % nos quedamos con el contraido y volvemos al
principio

                     v(indice(length(indice)),:)=xcontracf;

                 else %hemos empeorado estrictamente al reflejado

                     % hacemos un encogimiento(cambiamos casi
                     % completamente el vector)

                     v=encoger(vaux,sigma);


                 end


            else %no hemos mejorado estrictamente al peor

                                % contraccion hacia dentro
(contraemos con el peor)

                [xcontracd,fcontracd]=contracd(c,vaux(n+1,:),gama);


                if fcontracd < waux(length(waux)) %hemos mejorado
estrictamente al peor

                    % nos quedamos con el contraido y volvemos al
principio

                    v(indice(length(indice)),:)=xcontracd;

                else %no hemos mejorado estrictamente al peor

                    % hacemos un encogimiento (cambiamos casi
completamente
                    % el vector)

                    v=encoger(vaux,sigma);


                 end
```

```
            end

        else %hemos igualado o empeorado al mejor pero hemos
mejorado estrictamente al ultimo de los buenos

            % cambiamos el peor por el reflejado y volvemos al
principio'
            v(indice(length(indice)),:)=xrefle;
        end

end

 xopt=vaux(1,:);
 fopt=waux(1);

 OPTIMO=[xopt,fopt];




 % por si conocemos el exacto calculamos el error, hay que
cambiarlo en
 % cada caso

 %error=abs(fopt);
```

```
 function [xcontracd,fcontracd]=contracd(c,v,gama)
 %
 % MARÍA JOSÉ ZAPATERO MORENO

 % punto contraido

 xcontracd=c-gama*(c-v);
 fcontracd=f(xcontracd);

 ------------------------------------------------------------------



 function [xcontracf,fcontracf]=contracfu(xrefle,c,gamma)
 %
 % MARÍA JOSÉ ZAPATERO MORENO


     % punto contraido

     xcontracf=c+gamma*(xrefle-c);
     fcontracf=f(xcontracf);
```

```
function v=encoger(vaux,sigma)
%
% MARÍA JOSÉ ZAPATERO MORENO


% cambiamos el vector v (el simplex), solo mantenemos el mejor
punto

v(1,:)=vaux(1,:);

    for i=2:length(vaux)

           v(i,:)=vaux(1,:)+sigma*(vaux(i,:)-vaux(1,:));
    end
```

```
function [xexpan,fexpan]=expansion(xrefle,c,chi)
 %
 % MARÍA JOSÉ ZAPATERO MORENO


    % punto expandido

    xexpan=c+chi*(xrefle-c);
    fexpan=f(xexpan);
```

```
function [xrefle,frefle]=reflexion(vaux,c,ro)
%
% MARÍA JOSÉ ZAPATERO MORENO




        % punto reflejado

            xrefle=c+ro*(c-vaux(length(vaux),:));
            frefle=f(xrefle);
```

2.3. FUNCIÓN GRADIENTE

```
function [OPTIMO]=gradiente(x,dim,rd,nvecesgrad,alfak)
%
%
% Esta funcion calcula el metodo del gradiente con alfa fijo
alfak,, es decir, no
```

```
% resolvemos el problema de minimizacion unidimensional.
% OPTIMO es un vector que contiene en cada fila las abscisas y el
valor de
% f dado por gradiente
% El metodo hara el gradiente en cada fila.
% TIEMPO es el tiempo de ejecucion
% rd es el rango de aproximacion para calcular el gradiente
% dim es la dimension de la funcion
% x es el optimo de S.G.O. (un punto)
%
% [OPTIMO]=gradiente(x,dim,rd,nvecesgrad,alfak)
%
% MARÍA JOSÉ ZAPATERO MORENO




id=eye(dim);
grad=zeros(1,dim);
%tic
for pe=1:nvecesgrad

'calculo del gradiente (con signo negativo)';



    for j=1:dim;


    grad(j)=(f(x(1,:)-rd*id(j,:))-f(x(1,:)+rd*id(j,:)))/(2*rd);

    end

    modulo=norm(grad);


 'Miro si el módulo es nulo';


    if modulo==0
        disp('gradiente nulo');
        disp(pe);
        OPTIMO=[x,f(x)];
        return
    end

    'Calculo el nuevo punto con alfak fijo';


    x1=x+alfak*grad/modulo;

    if f(x1)<=f(x);
        x=x1;
    else
        x1=x+alfak/100*grad/modulo;
        disp('optimo no mejorado');
        disp(pe);
```

```
          if f(x1)<=f(x);
             x=x1;
          else
             x1=x+alfak/1000*grad/modulo;
               if f(x1)<=f(x);
                   x=x1;
               else
                  x1=x+alfak/50000*grad/modulo;

                     if f(x1)<=f(x);
                        x=x1;
                     else
                       disp('optimo no mejorado');
                       disp(pe);

                       OPTIMO=[x,f(x)];
                       return
                     end
               end
          end
     end


end

OPTIMO=[x,f(x)];
```

2.4. FUNCIÓN SEGMENTACIÓN

```
function
[OPTIMOS,OPTIMOS1,OPTIMOS2,OPTIMOS3,alfa,beta,g,n,niter,nvecesgrad,
dim,efo,TIEMPO]=graviseg(nveces)

%programa principal gravi
%
%ESTE PROGRAMA HIBRIDA S.G.O CON SEGMENTACIÓN1 Y GRADIENTE
%xopt es la posicion del optimo y fopt es el valor de la funcion
objetivo en
%el  optimo
%guardo los optimos en XFopt
%r1 y r2 definen la zona de factibilidad
%rd es el incremento para aproximar el gradiente
%n es el numero de asteroides
%xc son las coordenadas del centro de gravedad
%alfa , beta y g son los parametros de S.G.O.
%niter es el numero de iteraciones
%dim es la dimension de la funcion objetivo
%x es una matriz de tamaño nx2 con las posiciones de los asteroides
%v es una matriz de tamaño nx2 con las velocidades de los
asteroides
%val es el vector resultado de evaluar f en cada fila de x
%indice es el lugar donde se alcanza el optimo
%XFopt es un vector que va guardando los sucesivos optimos:
posicion, valor
```

```
%de f
%x1 y v1 guardan los valores iniciales por si es interesante
repetir con valores fijos
% y en ese caso ejecutamos gravi2.
%
% nveces es el número de veces que ejecuto el algoritmo
% Tiempo guarda los tiempos en cada fila
% efo guarda las evaluaciones a la función objetivo
% OPTIMOS es una matriz de p filas que guarda en cada fila el XFopt
% correspondiente
% numseg ees el número de subintervalos en que dividimos cada
intervalo
%
% La búsqueda es con intervalos expansivos y los de la diagonal
principal
%
% OPTIMOS 1, 2 Y 3 es el óptimo en cada una de las 3 subdivisiones
% planteadas respectivamente
%
%[OPTIMOS,OPTIMOS1,OPTIMOS2,OPTIMOS3,alfa,beta,g,n,niter,nvecesgrad
,dim,efo,TIEMPO]=graviseg(nveces)
%
% MARÍA JOSÉ ZAPATERO MORENO



% inicializacion de los parámetros HAY QUE CAMBIARLOS EN CADA
CASO!!!

dim=2;'nº de variables';
n=100;'nº de asteroides';
niter=100;' nº de movimientos de cada asteroide';
alfa=0.005;
beta=0.4;
g=100000;
r1=-500;
r2=500;
rd=0.000001;
alfak=1;
nvecesgrad=10^6;
numseg=5;

global EFO



OPTIMOS=zeros(nveces,dim+1);
OPTIMOS1=zeros(nveces,dim+1);
OPTIMOS2=zeros(nveces,dim+1);
OPTIMOS3=zeros(nveces,dim+1);


OPTIMOPAR=zeros(numseg,dim+3);
efo=zeros(nveces,1);
TIEMPO=zeros(nveces,1);
pasoabs=abs(r2-r1)/numseg;
```

```
for p=1:nveces

   tic

    EFO=0;
cont=0;


% DIVIDO LA REGIÓN FACTIBLE EN 5 CUADRADOS EN LOS QUE EJECUTO S.G.O
CADA
% VEZ MÁS GRANDES (DEJO r1 FIJO Y MUEVO r2)

    for segx=1:numseg
        abscisa=r1+pasoabs:pasoabs:r2;


'funcion de inicializacion aleatoria de espacios y velocidades';

    [x,v]=inicial(n,r1,abscisa(segx),dim);


'primer valor del optimo';

val=zeros(n,1);

for i=1:n
   val(i)=f(x(i,:));
end

[fopt, indice2]=min(val);

xopt=x(indice2,:);

'en XFopt guardamos POR ESTE ORDEN x y el óptimo';



'calculo del centro de gravedad';

[xc]=cengrav(x,n,dim);


'calculo de las aceleraciones';

a=acelera(x,dim,g,alfa,xc,n,rd);


'calculo de las sucesivas velocidades y posiciones';


  for j=2:niter


    'actualizamos veloc y posiciones empiezo las iteraciones';

    [x,v]=velpos(beta,a,x,v);
```

```
      'comprobacion de la factibilidad';

      [x,v,cont]=facti(x,v,r1,abscisa(segx),n,cont,dim);


       'nuevo optimo';

       for i=1:n
         val(i)=f(x(i,:));
       end


        [fopt1,indice1]=min(val);
        xopt1=x(indice1,:);

         if fopt1<fopt
             fopt=fopt1;
             xopt=xopt1;
             %XFopt=[XFopt;[xopt,fopt]]; por si necesito todos los
óptimos
             %XFopt=[xopt,fopt];
         end




' actualizamos aceleraciones';

    'calculo del nuevo centro de gravedad';

    [xc]=cengrav(x,n,dim);


    'calculo de las nuevas aceleraciones';

    a=acelera(x,dim,g,alfa,xc,n,rd);

  end
        'LAMAMOS AL GRADIENTE';

        OPTIMO=gradiente(xopt,dim,rd,nvecesgrad,alfak);

        OPTIMOPAR(segx,:)=[r1,abscisa(segx),OPTIMO];

  end


% busco el mínimo de las 5 vueltas y lo guardo en OPTIMOS1

[~,indice]=min(OPTIMOPAR(:,dim+3));

OPTIMOS1(p,:)=OPTIMOPAR(indice,3:dim+3);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%

% A CONTINUACIÓN DIVIDO LA REGIÓN FACTIBLE EN 5 CUADRADOS EN LOS
QUE EJECUTO S.G.O CADA
% VEZ MÁS GRANDES (DEJO r2 FIJO Y MUEVO r1)

    for segx=1:numseg

     abscisa=r1:pasoabs:r2-pasoabs;

'funcion de inicializacion aleatoria de espacios y velocidades';

    [x,v]=inicial(n,abscisa(segx),r2,dim);


'primer valor del optimo';

val=zeros(n,1);

for i=1:n
   val(i)=f(x(i,:));
end

[fopt, indice2]=min(val);

xopt=x(indice2,:);



'calculo del centro de gravedad';

[xc]=cengrav(x,n,dim);


'calculo de las aceleraciones';

a=acelera(x,dim,g,alfa,xc,n,rd);


'calculo de las sucesivas velocidades y posiciones';


  for j=2:niter


    'actualizamos veloc y posiciones empiezo las iteraciones';

    [x,v]=velpos(beta,a,x,v);


     'comprobacion de la factibilidad';

     [x,v,cont]=facti(x,v,abscisa(segx),r2,n,cont,dim);
```

```
        'nuevo optimo';

        for i=1:n
          val(i)=f(x(i,:));
        end


         [fopt1,indice1]=min(val);
         xopt1=x(indice1,:);

          if fopt1<fopt
              fopt=fopt1;
              xopt=xopt1;
              %XFopt=[XFopt;[xopt,fopt]]; por si necesito todos los
óptimos
              %XFopt=[xopt,fopt];
          end




' actualizamos aceleraciones';

    'calculo del nuevo centro de gravedad';

    [xc]=cengrav(x,n,dim);


    'calculo de las nuevas aceleraciones';

    a=acelera(x,dim,g,alfa,xc,n,rd);

  end
          'LAMAMOS AL GRADIENTE';

          OPTIMO=gradiente(xopt,dim,rd,nvecesgrad,alfak);

          OPTIMOPAR(segx,:)=[abscisa(segx),r2,OPTIMO];

  end



% busco el mínimo de las 5 vueltas y lo guardo en OPTIMOS2

[~,indice]=min(OPTIMOPAR(:,dim+3));

OPTIMOS2(p,:)=OPTIMOPAR(indice,3:dim+3);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% A CONTINUACIÓN DIVIDO LA REGIÓN FACTIBLE EN 5 CUADRADOS EN LOS
QUE EJECUTO S.G.O CADA
% VEZ MÁS GRANDES (CAMBIO r1 Y r2 CON CUADRADOS PEQUEÑOS POR LA
DIAGONAL)

    for segx=1:numseg

    % OJO AHORA ABSCISA TIENE TAMAÑO 6
     abscisa=r1:pasoabs:r2;


'funcion de inicializacion aleatoria de espacios y velocidades';

    [x,v]=inicial(n,abscisa(segx),abscisa(segx+1),dim);


'primer valor del optimo';

val=zeros(n,1);

for i=1:n
   val(i)=f(x(i,:));
end

[fopt, indice2]=min(val);

xopt=x(indice2,:);


'calculo del centro de gravedad';

[xc]=cengrav(x,n,dim);


'calculo de las aceleraciones';

a=acelera(x,dim,g,alfa,xc,n,rd);


'calculo de las sucesivas velocidades y posiciones';


  for j=2:niter


    'actualizamos veloc y posiciones empiezo las iteraciones';

    [x,v]=velpos(beta,a,x,v);


     'comprobacion de la factibilidad';


[x,v,cont]=facti(x,v,abscisa(segx),abscisa(segx+1),n,cont,dim);
```

```
        'nuevo optimo';

        for i=1:n
          val(i)=f(x(i,:));
        end


         [fopt1,indice1]=min(val);
         xopt1=x(indice1,:);

          if fopt1<fopt
              fopt=fopt1;
              xopt=xopt1;
              %XFopt=[XFopt;[xopt,fopt]]; por si necesito todos los
óptimos
              %XFopt=[xopt,fopt];
          end




' actualizamos aceleraciones';

    'calculo del nuevo centro de gravedad';

    [xc]=cengrav(x,n,dim);


    'calculo de las nuevas aceleraciones';

    a=acelera(x,dim,g,alfa,xc,n,rd);

  end
        'LAMAMOS AL GRADIENTE';

        OPTIMO=gradiente(xopt,dim,rd,nvecesgrad,alfak);

        OPTIMOPAR(segx,:)=[abscisa(segx),abscisa(segx+1),OPTIMO];

  end



% busco el mínimo de las 5 vueltas y lo guardo en OPTIMOS3

  [~,indice]=min(OPTIMOPAR(:,dim+3));

  OPTIMOS3(p,:)=OPTIMOPAR(indice,3:dim+3);

  % escojo como OPTIMOS el mínimo de OPTIMOS1,2 Y 3.

   if OPTIMOS1(p,dim+1)<OPTIMOS2(p,dim+1)&&
OPTIMOS1(p,dim+1)<OPTIMOS3(p,dim+1)

      OPTIMOS(p,:)=OPTIMOS1(p,:);
   elseif OPTIMOS3(p,dim+1)<OPTIMOS2(p,dim+1)
```

```
      OPTIMOS(p,:)=OPTIMOS3(p,:);

   else
      OPTIMOS(p,:)=OPTIMOS2(p,:);
   end

   efo(p)=EFO;
   TIEMPO(p)=toc;
   end
```

2.5. FUNCIÓN V.S.O.

```
function [OPTIMO, TIEMPO, EFO]=vso(v,valvso)

% esta funcion calcula el algoritmo vso para una funcion objetivo
% que se encuentra en el fichero f.m
% v es una matriz (semilla x dim) que contiene, por filas los
semilla puntos de
% dimension dim (la semilla)
%
% hago las rectas cortadas por corte (un número al azar) durante 5
veces como máximo.
% si no mejora salimos del bucle.
%
%
%
% [OPTIMO]=vso3(v,valvso)
%
% MARÍA JOSÉ ZAPATERO MORENO


% Valor de corte de las rectas que unen el mejor con todos


global EFO
dim=size(v,2);
semilla=size(v,1);
w=zeros(semilla,1);
faux=zeros(semilla,1);
tic;
EFO=0;


for k=1:valvso


    % iteracion k-esima (hago valvso iteraciones)


       for j=1:10 % hago 10 intentos de cortes al azar en las
           %rectas que unen el resto de la semilla con el mejor

           waux=ones(semilla,dim);
```

```
            for i=1:semilla
             w(i)=f(v(i,:));
            end

            % el numero valor se guarda el valor minimo de f
            % en el numero indice se guarda el indice del mejor.

            [valor,indice]=min(w);

            for i=1:semilla
                waux(i,:)=waux(i,:).*v(indice,:);
            end


            corte=rand;

            vaux=corte*waux+(1-corte)*v;

            for i=1:semilla
             faux(i)=f(vaux(i,:));
            end


            if min(faux)<min(w)
                v=vaux;

                for i=1:semilla
                  w(i)=f(v(i,:));
                end
                break
            end

        end


        if j==10 % en 10 cortes al azar no hemos mejorado
            disp('optimo no mejorado')
            disp(k)
            break
        end
    end




  [fopt,indice]=min(w);

 xopt=v(indice,:);

 OPTIMO=[xopt,fopt];

 TIEMPO=toc;
-----------------------------------------------------------------------
```

2.6. FUNCIÓN AGUJERO DE GUSANO

```
function [OPTIMO,TIEMPO,EFO]=agujerodegusano(grupo,fgrupo)
% ESTE PROGRAMA CALCULA EL HEURÍSTICO AGUJERO DE GUSANO EN UN GRUPO
DE
% ABSCISAS
% PRIMERO: BUSCA ABSCISAS REPETIDAS Y CAMBIA LAS ABSCISAS DISTINTAS
CON LA
% MODA
% EL RESTO LAS DEJA COMO ESTÁ
% LUEGO FILTRA LOS RESULTADOS CON GRADIENTE Y GUARDA EL VALOR
%
%
% SEGUNDO: BUSCA ABSCISAS REPETIDAS Y CAMBIA LAS ABSCISAS DISTINTAS
CON LA
% MODA
% EN LAS QUE LA MODA NO ES SUFICIENTEMENTE GRANDE CAMBIA LAS
ABSCISAS POR LA
% MEDIA
% LUEGO FILTRA LOS RESULTADOS CON GRADIENTE Y GUARDA EL VALOR
% EN GRUPO ESTÁN SOLO LAS ABSCISAS
% EN fgrupo ESTÁ LO QUE VALE LA FUNCIÓN EN GRUPO
%
%
% SE QUEDA CON EL MEJOR SI LO HAY Y SI NO DEVUELVE UN MENSAJE Y EL
VALOR DE
% ENTRADA.
%[OPTIMO]=agujerodegusano(grupo)
%
% MARÍA JOSÉ ZAPATERO MORENO



tic
global EFO
EFO=0;
dim=size(grupo,2);
tamgrupo=size(grupo,1);
OPTIMOPAR1=grupo;
OPTIMOPAR2=grupo;
rd=0.000001;
alfak=1;
nvecesgrad=10^6;


%CALCULAMOS EL OPTIMO DEL GRUPO

foptgrupo=min(fgrupo);


if rem(tamgrupo,2)==0
    mitad=tamgrupo/2;
else
    mitad=(tamgrupo-1)/2;
end


% redondeo a 6 dígitos de precisión
```

```
grupo1=grupo*10^5;
grupo1=round(grupo1);



% Busco si hay valores de moda

for i=1:dim  % barro las columnas buscando modas
    [moda,cantidad]=mode(grupo1(:,i));
    media=mean(grupo(:,i));

    if cantidad>mitad
        for j=1:tamgrupo
            if grupo1(j,i)==moda
                indice=j;

                break

            end
        end

        OPTIMOPAR1(:,i)=ones(tamgrupo,1)*grupo(indice,i);
        OPTIMOPAR2(:,i)=ones(tamgrupo,1)*grupo(indice,i);
    else
        OPTIMOPAR2(:,i)=ones(tamgrupo,1)*media;
    end

end

%OBVIAMENTE HE CONVERTIDO LA SEMILLA EN UN SOLO VECTOR EN EL CASO
DE
% OPTIMOPAR2


OPTIMOPAR3=OPTIMOPAR2(1,:);


% AHORA MEJORO LOS PUNTOS QUE ME QUEDAN CON GRADIENTE

for k=1:tamgrupo

OPTIMOPAR1(k,1:dim+1)=gradiente(OPTIMOPAR1(k,1:dim),dim,rd,nvecesgr
ad,alfak);
end

OPTIMOPAR3(1,1:dim+1)=gradiente(OPTIMOPAR3,dim,rd,nvecesgrad,alfak)
;


% EVALÚO TODOS LOS ÓPTIMOS

[OPTIMOPAR11,INDICE]=min(OPTIMOPAR1(:,dim+1));

if min([OPTIMOPAR11,OPTIMOPAR3(1,dim+1),foptgrupo])==OPTIMOPAR11
    OPTIMO=OPTIMOPAR1(INDICE,:);
```

```
elseif
min([OPTIMOPAR11,OPTIMOPAR3(1,dim+1),foptgrupo])==OPTIMOPAR3(1,dim+
1)
    OPTIMO=OPTIMOPAR3;
else
    disp('la semilla no está lo suficientemente cerca del óptimo')
    OPTIMO=[grupo,fgrupo];
end

TIEMPO=toc;

-----------------------------------------------------------------------
```

APÉNDICE 3. CÓDIGOS EN MATLAB DE LA FUNCIÓN CASSINI 2

```
function [J] = cassini2(t,problem)
[J] = mga_dsm(t,problem);

-----------------------------------------------------------------------------------------------



------------------------------------------------------------------------
% This source file is part of the 'ESA Advanced Concepts Team's
% Space Mechanics Toolbox' software.
%
% The source files are for research use only,
% and are distributed WITHOUT ANY WARRANTY. Use them on your own
risk.
%
% Copyright (c) 2004-2007 European Space Agency
% ------------------------------------------------------------------
%

%Programmed by:         Dario Izzo          (ESA/ACT)
%                       Claudio Bombardelli (ESA/ACT)

%Date:                  15/03/2007
%Revision:              4
%Tested by:             D.Izzo and C.Bombardelli
%
%
%  Computes the DeltaV cost function of a Multiple Gravity Assist
trajectory
%  with a Deep Space Maneuver between each planet pair
%  N.B.: All swing-bys are UNPOWERED (thrust is only present at
each dsm)
%  It takes as input a sequence of planets P1,Pn and a decision
vector t
%
%  The flyby/dsm sequence is:  P1/d1/P2/d2/../dn-1/Pn
%

% DECISION VECTOR DEFINITION:
%
%  t(1)=  epoch of departure MJD2000 from first planet (not
necessarily earth)
%  t(2)=  magnitude hyperbolic escape velocity from first planet
%  t(3,4)= u,v variables for the hyperbolic velocity orientation
wrt Earth
%  velocity at departure
%  t(5..n+3)= planet-to-planet Time of Flight [ToF] (days)
%  t(n+4..2n-2)= fraction of ToF at which the DSM occurs
%  t(2n+3..3n) = perigee fly-by radius for planets P2..Pn-1, non-
dimensional wrt planetary radii
%  t(3n+1..4n-2) = rotation gamma of the bplane-component of the
swingby outgoing velocity (v_rel_out)
%  [take n_r=cross(v_rel_in,v_planet_helio) if you rotate n_r by
+gamma around v_rel_in
%  you obtain the projection of v_rel_out on the b-plane]
%  Vector (Vout) around the axis of the incoming swingby velocity
vector (Vin)
%
```

```
%Usage:     [J,DVvec,DVarr] = mga_dsm(t,MGADSMproblem)
%Outputs:
%           J:      Cost function = depends on the problem:
%                   orbit insertion: total DV from propulsion system
(V_launcher not counted)
%                   gtoc1= (s/c final mass)*v_asteroid'*vrel_ast_sc
%                   asteroid deflection-> 1/d with d=deflecion on
the earth-asteroid lineofsight
%           DVvec: vector of all DV maneuvers, including the escape
C3 and
%                   the arrival DV evaluaed according to the
objective function
%           DVarr: Relative velocity at the arrival planet
%
%
%Inputs:
%           t:          decision vector
%               MGADSMproblem = struct array defining the problem,
i.e.
%               MGADSMproblem.sequence:  planet sequence. Example
[3 3 4]= [E E M]. A
%                       negative sign represents a retrograde orbit
after the fly-by
%               MGADSMproblem.objective.type: type of objective
function (e.g.
%                        'orbit insertion','rndv','gtoc1')
%               MGADSMproblem.objective.rp: pericentre radius of
the
%                        target orbit if type = 'orbit insertion'
%               MGADSMproblem.objective.e:  eccentricity of the
target
%                        orbit if type = 'orbit insertion'
%               MGADSMproblem.bounds: decision vector upper and
lower
%               bounds for the optimiser
%               MGADSMproblem.yplot:     1-> plot trajectory, 0->
don't
%
%
%*********  IMPORTANT NOTE (SINGULARITY)   ************
%
% The routine is singular when the S/C relative incoming  velocity
to a planet v_rel_in
% is parallel to the heliocentric velocity of that planet.
%
% One can move the singularity elsewhere by changing the definition
of the
% angle gamma.
% For example one possibility is to define gamma as follows:
% [take n_r=cross(v_rel_in,r_planet_helio) and rotate n_r by +gamma
around v_rel_in
% in this case is singular for v_rel_in parallel to r_planet_helio
%
%
function [J,DVvec,DVrel] = mga_dsm(t,problem)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%

sequence= problem.sequence;  % THE PLANETs SEQUENCE
```

```
switch problem.objective.type
    case 'orbit insertion'
        rp_target= problem.objective.rp; % radius of pericentre at
capture
        e_target=problem.objective.e;   % Eccentricity of the
target orbit at capture
    case 'total DV orbit insertion'
        rp_target= problem.objective.rp; % radius of pericentre at
capture
        e_target=problem.objective.e;   % Eccentricity of the
target orbit at capture
    case 'gtoc1'
        Isp=problem.objective.Isp;
        mass = problem.objective.mass;
    case 'time to AUs'
        AUdist = problem.objective.AU;
        DVtotal = problem.objective.DVtot;
        DVonboard = problem.objective.DVonboard;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%

%*********************************************************************
********
%Definition of the gravitational constants of the various planets
%(used in the powered swing-by routine) and of the sun (used in the
lambert
%solver routine)
%*********************************************************************
********

mu(1)=22321;          %                           Mercury
mu(2)=324860;         %Gravitational constant of Venus
mu(3)=398601.19;      %                           Earth
mu(4)=42828.3;        %                           Mars
mu(5)=126.7e6;        %                           Jupiter
mu(6)=37.93951970883e6; %                         Saturn

muSUN=1.32712428e+11; %Gravitational constant of Sun


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%  Definition of planetari radii
%
RPL(1)=2440; % Mercury
RPL(2)=6052; % Venus
RPL(3)=6378; % Earth
RPL(4)=3397; % Mars
RPL(5)=71492;% Jupiter
RPL(6)=60330;% Saturn


%*********************************************************************
********
% Decision vector definition
```

```
tdep=t(1);         % departure epoch (MJD2000)
VINF=t(2);         %Hyperbolic escape velocity (km/sec)
udir=t(3);            %Hyperbolic escape velocity var1 (non dim)
vdir=t(4);            %Hyperbolic escape velocity var2 (non dim)
N=length(sequence);

%Preallocating memory increase speed
tof = zeros(N-1,1);
alpha = zeros(N-1,1);

for i=1:N-1
    tof(i)=t(i+4); %planet-to-planet Time of Flight [ToF] (days)
    alpha(i)=t(N+i+3); %fraction of ToF at which the DSM occurs
end


%If we are optimising to reach a given distance in the shortest
time ('time
%to AUs) then the decision vector needs to include also r_p and
bincl at
%the last planet, otherwise not

if strcmp(problem.objective.type,'time to AUs')
    rp_non_dim=zeros(N-1,1);        %initialization gains speed
    gamma=zeros(N-1,1);
    for i=1:N-1
        rp_non_dim(i)=t(i+2*N+2); % non-dim perigee fly-by radius
of planets P2..Pn (i=1 refers to the second planet)
        gamma(i)=t(3*N+i);        % rotation of the bplane-
component of the swingby outgoing
        % velocity  Vector (Vout) around the axis of the incoming
swingby velocity vector (Vin)
    end
else
    rp_non_dim=zeros(N-2,1);        %initialization gains speed
    gamma=zeros(N-2,1);
    for i=1:N-2
        rp_non_dim(i)=t(i+2*N+2); % non-dim perigee fly-by radius
of planets P2..Pn-1 (i=1 refers to the second planet)
        gamma(i)=t(3*N+i);        % rotation of the bplane-
component of the swingby outgoing
        % velocity  Vector (Vout) around the axis of the incoming
swingby velocity vector (Vin)
    end
end

%*****************************************************************
********
%Evaluation of position and velocities of the planets
%*****************************************************************
********


N=length(sequence);

r= zeros(3,N);
v= zeros(3,N);

muvec=zeros(N,1);
```

```
Itime = zeros(N);
dT=zeros(1,N);

seq = abs (sequence);

T=tdep;
dT(1:N-1)=tof;

for i=1:N
    Itime(i)=T;
    if seq(i)<10
        [r(:,i),v(:,i)]=pleph_an( T , seq(i)); %positions and
velocities of solar system planets
        muvec(i)=mu(seq(i)); %gravitational constants
    else
        [r(:,i),v(:,i)]=CUSTOMeph( mjd20002jed(T) , ...
            problem.customobject(seq(i)).epoch, ...
            problem.customobject(seq(i)).keplerian , 1); %positions
and velocities of custom object
        muvec(i)=problem.customobject(seq(i)).mu; %gravitational
constant of custom object

    end

    T=T+dT(i);

end



if strcmp(problem.objective.type,'time to AUs')
    rp=zeros(N-1,1);        %initialization gains speed
    for i=1:N-1
        rp(i)= rp_non_dim(i)*RPL(seq(i+1)); %dimensional flyby
radii (i=1 corresponds to 2nd planet)
    end
else
    rp=zeros(N-2,1);        %initialization gains speed
    for i=1:N-2
        rp(i)= rp_non_dim(i)*RPL(seq(i+1)); %dimensional flyby
radii (i=1 corresponds to 2nd planet)
    end
end


%*****************************************************************
********
%%%% FIRST BLOCK (P1 to P2)

%Spacecraft position and velocity at departure

vtemp= cross(r(:,1),v(:,1));

iP1= v(:,1)/norm(v(:,1));
zP1= vtemp/norm(vtemp);
jP1= cross(zP1,iP1);
```

```
theta=2*pi*udir;         %See Picking a Point on a Sphere
phi=acos(2*vdir-1)-pi/2; %In this way: -pi/2<phi<pi/2 so phi can be
used as out-of-plane rotation

%vinf=VINF*(-
sin(theta)*iP1+cos(theta)*cos(phi)*jP1+sin(phi)*cos(theta)*zP1);
vinf=VINF*(cos(theta)*cos(phi)*iP1+sin(theta)*cos(phi)*jP1+sin(phi)
*zP1);

v_sc_pl_in(:,1)=v(:,1); %Spacecraft absolute incoming velocity at
P1
v_sc_pl_out(:,1)=v(:,1)+vinf; %Spacecraft absolute outgoing
velocity at P1

%Days from P1 to DSM1
tDSM(1)=alpha(1)*tof(1);

%Computing S/C position and absolute incoming velocity at DSM1
[rd(:,1),v_sc_dsm_in(:,1)]=propagateKEP(r(:,1),v_sc_pl_out(:,1),tDS
M(1)*24*60*60,muSUN);

%Evaluating the Lambert arc from DSM1 to P2

lw=vett(rd(:,1),r(:,2));
lw=sign(lw(3));
if lw==1
    lw=0;
else
    lw=1;
end
[v_sc_dsm_out(:,1),v_sc_pl_in(:,2)]=lambertI(rd(:,1),r(:,2),tof(1)*
(1-alpha(1))*24*60*60,muSUN,lw);

%First Contribution to DV (the 1st deep space maneuver)
DV=zeros(N-1,1);
DV(1)=norm(v_sc_dsm_out(:,1)-v_sc_dsm_in(:,1));


%**************************************
% INTERMEDIATE BLOCK

tDSM=zeros(N-1,1);
for i=1:N-2

    %Evaluation of the state immediately after Pi

    v_rel_in=v_sc_pl_in(:,i+1)-v(:,i+1);

    e=1+rp(i)/muvec(i+1)*v_rel_in'*v_rel_in;

    beta_rot=2*asin(1/e);              %velocity rotation

    ix=v_rel_in/norm(v_rel_in);
    % ix=r_rel_in/norm(v_rel_in);  % activating this line and
disactivating the one above
    % shifts the singularity for r_rel_in parallel to v_rel_in

    iy=vett(ix,v(:,i+1)/norm(v(:,i+1)))';
```

```
        iy=iy/norm(iy);
        iz=vett(ix,iy)';
        iVout = cos(beta_rot) * ix + cos(gamma(i))*sin(beta_rot) * iy +
    sin(gamma(i))*sin(beta_rot) * iz;
        v_rel_out=norm(v_rel_in)*iVout;

        v_sc_pl_out(:,i+1)=v(:,i+1)+v_rel_out;


        %Days from Pi to DSMi
        tDSM(i+1)=alpha(i+1)*tof(i+1);


        %Computing S/C position and absolute incoming velocity at DSMi

    [rd(:,i+1),v_sc_dsm_in(:,i+1)]=propagateKEP(r(:,i+1),v_sc_pl_out(:,
    i+1),tDSM(i+1)*24*60*60,muSUN);


        %Evaluating the Lambert arc from DSMi to Pi+1

        lw=vett(rd(:,i+1),r(:,i+2));
        lw=sign(lw(3));
        if lw==1
            lw=0;
        else
            lw=1;
        end

    [v_sc_dsm_out(:,i+1),v_sc_pl_in(:,i+2)]=lambertI(rd(:,i+1),r(:,i+2)
    ,tof(i+1)*(1-alpha(i+1))*24*60*60,muSUN,lw);

        %DV contribution
        DV(i+1)=norm(v_sc_dsm_out(:,i+1)-v_sc_dsm_in(:,i+1));

    end

    %*****************************************************************
    ******
    % FINAL BLOCK
    %
    %1)Evaluation of the arrival DV
    %
    DVrel=norm(v(:,N)-v_sc_pl_in(:,N)); %Relative velocity at target
    planet


    switch problem.objective.type
        case 'orbit insertion'
            DVper=sqrt(DVrel^2+2*muvec(N)/rp_target);  %Hyperbola
            DVper2=sqrt(2*muvec(N)/rp_target-muvec(N)/rp_target*(1-
    e_target)); %Ellipse
            DVarr=abs(DVper-DVper2);
        case 'total DV orbit insertion'
            DVper=sqrt(DVrel^2+2*muvec(N)/rp_target);  %Hyperbola
            DVper2=sqrt(2*muvec(N)/rp_target-muvec(N)/rp_target*(1-
    e_target)); %Ellipse
            DVarr=abs(DVper-DVper2);
        case 'rndv'
```

```
            DVarr = DVrel;
        case 'total DV rndv'
            DVarr = DVrel;
        case 'gtoc1'
            DVarr = DVrel;
        case 'time to AUs'  %no DVarr is considered
            DVarr = 0;
    end

    DV(N)=DVarr;


    %
    %*****************************************************************
    ********
    %Evaluation of total DV spent by the propulsion system
    %*****************************************************************
    ********

    switch problem.objective.type
        case 'gtoc1'
            DVtot=sum(DV(1:N-1));
        case 'deflection demo'
            DVtot=sum(DV(1:N-1));
        otherwise
            DVtot=sum(DV);
    end


    DVvec=[VINF ; DV];




    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%
    %Finally our objective function is:
    switch problem.objective.type
        case 'total DV orbit insertion'
            J= DVtot+VINF;
        case 'total DV rndv'
            J= DVtot+VINF;
        case 'orbit insertion'
            J= DVtot;
        case 'rndv'
            J= DVtot;
        case 'gtoc1'
            mass_fin = mass * exp (- DVtot/ (Isp/1000 * 9.80665));
            J = 1/(mass_fin * abs((v_sc_pl_in(:,N)-v(:,N))'* v(:,N)));
        case 'deflection demo'
            mass_fin = mass * exp (- DVtot/ (Isp/1000 * 9.80665));
            %non-dimensional units (mu=1)
            AU=149597870.66;
            VEL=sqrt(muSUN/AU);
            TIME=AU/VEL;
            %calculate the DV due to the impact
            relV = (V(:,N,1)-v(:,N));
            impactDV = (relV * mass_fin/astmass)/VEL;
            %calculate phi (see defl_radial.m)
```

```
            ir = r(:,N)/norm(r(:,N));
            iv = v(:,N)/norm(v(:,N));
            ih = vett(ir,iv)';
            ih = ih/norm(ih);
            itheta = vett(ih,ir)';
            impactDV = (impactDV'*ir) * ir + (impactDV'*itheta) *
    itheta; %projection on the orbital plane
            phi = acos((impactDV/norm(impactDV))'*ir);
            if (impactDV'*itheta) < 0
                phi = phi +pi;
            end
            %calculate ni0
            r0 = r(:,N)/AU;
            v0 = v(:,N)/VEL;
            E0 = IC2par( r0 , v0 , 1 );
            ni0 = E2ni(E0(6),E0(2));
            %calcuate the deflection projected in covenient coordinates
            a0 = E0(1);
            e0 = E0(2);
            M0 = E2M(E0(6),E0(2));
            numberofobs = 50;
            obstime = linspace(0,obstime,numberofobs);
            M = M0 + obstime*60*60*24/TIME*sqrt(1/a0);
            theta=zeros(numberofobs,1);
            for jj = 1:numberofobs
                theta(jj) = M2ni(M(jj),E0(2));
            end
            theta = theta-ni0;
            [dumb,dr] =
    defl_radial(a0,e0,ni0,phi,norm(impactDV),theta);

            [dumb,dtan] =
    defl_tangential(a0,e0,ni0,phi,norm(impactDV),theta);

            %calculate the deflecion on the Earth-asteroid lineofsight
            defl=zeros(3,numberofobs);
            temp=zeros(numberofobs,1);
            for i=1:numberofobs
                Tobs = T + obstime(i);
                [rast,vast]=CUSTOMeph( mjd20002jed(Tobs) , ...
                    problem.customobject(seq(end)).epoch, ...
                    problem.customobject(seq(end)).keplerian , 1);
                [rearth,vearth]=pleph_an( Tobs , 3);
                lineofsight=(rearth-rast)/norm((rearth-rast));
                defl(:,i) = rast / norm(rast) * dr(i) + ...
                    vast / norm(vast) * dtan(i);
                temp(i) = norm(lineofsight'*(defl(:,i)));

            end

            J = 1./abs(temp)/AU;
            [J,index]=min(J);
        case 'time to AUs'
            %non dimensional units
            AU  = 149597870.66;
            V = sqrt(muSUN/AU);
            T = AU/V;
            %evaluate the state of the spacecraft after the last fly-by
            v_rel_in=v_sc_pl_in(:,N)-v(:,N);
            e=1+rp(N-1)/muvec(N)*v_rel_in'*v_rel_in;
            beta_rot=2*asin(1/e);              %velocity rotation
```

```
        ix=v_rel_in/norm(v_rel_in);
        % ix=r_rel_in/norm(v_rel_in);  % activating this line and
disactivating the one above
        % shifts the singularity for r_rel_in parallel to v_rel_in
        iy=vett(ix,v(:,N)/norm(v(:,N)))';
        iy=iy/norm(iy);
        iz=vett(ix,iy)';
        iVout = cos(beta_rot) * ix + cos(gamma(N-1))*sin(beta_rot)
* iy + sin(gamma(N-1))*sin(beta_rot) * iz;
        v_rel_out=norm(v_rel_in)*iVout;
        v_sc_pl_out(:,N)=v(:,N)+v_rel_out;
        t = time2distance(r(:,N)/AU,v_sc_pl_out(:,N)/V,AUdist);
        DVpen=0;
        if sum(DVvec)>DVtotal
            DVpen=DVpen+(sum(DVvec)-DVtotal);
        end
        if sum(DVvec(2:end))>DVonboard
            DVpen=DVpen+(sum(DVvec(2:end))-DVonboard);
        end

        J= (t*T/60/60/24 + sum(tof))/365.25 + DVpen*100;
        if isnan(J)
            J=100000;
        end
end
```

```
%-----------------------------------------------------------------
--------
function [r,v] = propagateKEP(r0,v0,t,mu)
%
%Usage: [r,v] = propagateKEP(r0,v0,t)
%
%Inputs:
%           r0:    column vector for the non dimensional position
%           v0:    column vector for the non dimensional velocity
%           t:     non dimensional time
%
%Outputs:
%           r:    column vector for the non dimensional position
%           v:    column vector for the non dimensional velocity
%
%Comments:  The function works in non dimensional units, it takes
an
%initial condition and it propagates it as in a kepler motion
analytically.
%
%The matrix DD will be almost always the unit matrix, except for
orbits
%with little inclination in which cases a rotation is performed so
that
%par2IC is always defined
DD=eye(3);
h=vett(r0,v0);
ih=h/norm(h);
if abs(abs(ih(3))-1)<1e-3         %the abs is needed in cases in
which the orbit is retrograde,
                                  %that would held ih=[0,0,-1]!!
```

```
    DD=[1,0,0;0,0,1;0,-1,0];      %Random rotation matrix that make
the Euler angles well defined for the case
    r0=DD*r0;                     %For orbits with little
inclination another ref. frame is used.
    v0=DD*v0;
end

E=IC2par(r0,v0,mu);

M0=E2M(E(6),E(2));
if E(2)<1
    M=M0+sqrt(mu/E(1)^3)*t;
else
    M=M0+sqrt(-mu/E(1)^3)*t;
end
E(6)=M2E(M,E(2));
[r,v]=par2IC(E,mu);

r=DD'*r;
v=DD'*v;


%-------------------------------------------------------------------
--------%
function E=IC2par(r0,v0,mu)
%
%Usage: E = IC2par(r0,v0,mu)
%
%Inputs:
%           r0:    column vector for the position
%           v0:    column vector for the velocity
%
%Outputs:
%           E:     Column Vectors containing the six keplerian
parameters,
%                  (a (negative for hyperbolas),e,i,OM,om,Eccentric
Anomaly
%                  (or Gudermannian whenever e>1))
%
%Comments:  The parameters returned are, of course, referred to the
same
%ref. frame in which r0,v0 are given. Units have to be consistent,
and
%output angles are in radians
%The algorithm used is quite common and can be found as an example
in Bate,
%Mueller and White. It goes singular for zero inclination and for
ni=pi
%Note also that the anomaly in output ranges from -pi to pi
%Note that a is negative for hyperbolae

k=[0,0,1]';
h=vett(r0,v0)';
p=h'*h/mu;
n=vett(k,h)';
n=n/norm(n);
R0=norm(r0);
evett=vett(v0,h)'/mu-r0/R0;
e=evett'*evett;
E(1)=p/(1-e);
```

```
E(2)=sqrt(e);
e=E(2);
E(3)=acos(h(3)/norm(h));
E(5)=(acos((n'*evett)/e));
if evett(3)<0
    E(5)=2*pi-E(5);
end
E(4)=acos(n(1));
if n(2)<0
    E(4)=2*pi-E(4);
end
ni=real(acos((evett'*r0)/e/R0)); %real is to avoid problems when
ni~=pi
if (r0'*v0)<0
    ni=2*pi-ni;
end
EccAn=ni2E(ni,e);
E(6)=EccAn;


%-------------------------------------------------------------------
--------
function E=ni2E(ni,e)
if e<1
    E=2*atan(sqrt((1-e)/(1+e))*tan(ni/2)); %algebraic kepler's
equation
else
    E=2*atan(sqrt((e-1)/(e+1))*tan(ni/2)); %algebraic equivalent of
kepler's equation in terms of the Gudermannian
end



%-------------------------------------------------------------------
--------
function [r0,v0]=par2IC(E,mu)
%
%Usage: [r0,v0] = IC2par(E,mu)
%
%Outputs:
%           r0:    column vector for the position
%           v0:    column vector for the velocity
%
%Inputs:
%           E:     Column Vectors containing the six keplerian
parameters,
%                  (a (negative fr hyperbolas),e,i,OM,om,Eccentric
Anomaly
%                    or Gudermannian if e>1)
%           mu:    gravitational constant
%
%Comments:  The parameters returned are, of course, referred to the
same
%ref. frame in which r0,v0 are given. a can be given either in kms
or AUs,
%but has to be consistent with mu.All the angles must be given in
radians.

a=E(1);
e=E(2);
```

```
i=E(3);
omg=E(4);
omp=E(5);
EA=E(6);


% Grandezze definite nel piano dell'orbita
if e<1
    b=a*sqrt(1-e^2);
    n=sqrt(mu/a^3);

    xper=a*(cos(EA)-e);
    yper=b*sin(EA);

    xdotper=-(a*n*sin(EA))/(1-e*cos(EA));
    ydotper=(b*n*cos(EA))/(1-e*cos(EA));
else
    b=-a*sqrt(e^2-1);
    n=sqrt(-mu/a^3);
    dNdzeta=e*(1+tan(EA)^2)-
(1/2+1/2*tan(1/2*EA+1/4*pi)^2)/tan(1/2*EA+1/4*pi);

    xper = a/cos(EA)-a*e;
    yper = b*tan(EA);

    xdotper = a*tan(EA)/cos(EA)*n/dNdzeta;
    ydotper = b/cos(EA)^2*n/dNdzeta;
end

% Matrice di trasformazione da perifocale a ECI

R(1,1)=cos(omg)*cos(omp)-sin(omg)*sin(omp)*cos(i);
R(1,2)=-cos(omg)*sin(omp)-sin(omg)*cos(omp)*cos(i);
R(1,3)=sin(omg)*sin(i);
R(2,1)=sin(omg)*cos(omp)+cos(omg)*sin(omp)*cos(i);
R(2,2)=-sin(omg)*sin(omp)+cos(omg)*cos(omp)*cos(i);
R(2,3)=-cos(omg)*sin(i);
R(3,1)=sin(omp)*sin(i);
R(3,2)=cos(omp)*sin(i);
R(3,3)=cos(i);

% Posizione nel sistema inerziale

r0=R*[xper;yper;0];
v0=R*[xdotper;ydotper;0];



%-------------------------------------------------------------------
--------
function [v1,v2,a,p,theta,iter]=lambertI(r1,r2,t,mu,lw,N,branch)
%
%
%This routine implements a new algorithm that solves Lambert's
problem. The
%algorithm has two major characteristics that makes it favorable to
other
%existing ones.
%
```

```
%   1) It describes the generic orbit solution of the boundary
condition
%   problem through the variable X=log(1+cos(alpha/2)). By doing so
the
%   graphs of the time of flight become defined in the entire real
axis and
%   resembles a straight line. Convergence is granted within few
iterations
%   for all the possible geometries (except, of course, when the
transfer
%   angle is zero). When multiple revolutions are considered the
variable is
%   X=tan(cos(alpha/2)*pi/2).
%
%   2) Once the orbit has been determined in the plane, this
routine
%   evaluates the velocity vectors at the two points in a way that
is not
%   singular for the transfer angle approaching to pi (Lagrange
coefficient
%   based methods are numerically not well suited for this
purpose).
%
%   As a result Lambert's problem is solved (with multiple
revolutions
%   being accounted for) with the same computational effort for all
%   possible geometries. The case of near 180 transfers is also
solved
%   efficiently.
%
%   We note here that even when the transfer angle is exactly equal
to pi
%   the algorithm does solve the problem in the plane (it finds X),
but it
%   is not able to evaluate the plane in which the orbit lies. A
solution
%   to this would be to provide the direction of the plane
containing the
%   transfer orbit from outside. This has not been implemented in
this
%   routine since such a direction would depend on which
application the
%   transfer is going to be used in.
%
%Usage: [v1,v2,a,p,theta,iter]=lambertI(r1,r2,t,mu,lw,N,branch)
%
%Inputs:
%           r1=Position vector at departure (column)
%           r2=Position vector at arrival (column, same units as
r1)
%           t=Transfer time (scalar)
%           mu=gravitational parameter (scalar, units have to be
%           consistent with r1,t units)
%           lw=1 if long way is chosen
%           branch='l' if the left branch is selected in a problem
where N
%           is not 0 (multirevolution)
%           N=number of revolutions
%
%Outputs:
%           v1=Velocity at departure        (consistent units)
```

```
%           v2=Velocity at arrival
%           a=semi major axis of the solution
%           p=semi latus rectum of the solution
%           theta=transfer angle in rad
%           iter=number of iteration made by the newton solver
(usually 6)
%
%
%Preliminary control on the function call
if nargin==5
    N=0;
end
if t<=0
    warning('Negative time as input')
    v1=[NaN;NaN;NaN];
    v2=[NaN;NaN;NaN];
    return
end


tol=1e-11;  %Increasing the tolerance does not bring any advantage
as the
%precision is usually greater anyway (due to the rectification of
the tof
%graph) except near particular cases such as parabolas in which
cases a
%lower precision allow for usual convergence.


%Non dimensional units
R=norm(r1);
V=sqrt(mu/R);
T=R/V;

%working with non-dimensional radii and time-of-flight
r1=r1/R;
r2=r2/R;
t=t/T;

%Evaluation of the relevant geometry parameters in non dimensional
units
r2mod=norm(r2);
theta=acos(r1'*r2/r2mod);

if lw
    theta=2*pi-theta;
end
c=sqrt(1+r2mod^2-2*r2mod*cos(theta)); %non dimensional chord
s=(1+r2mod+c)/2;                      %non dimensional semi-
perimeter
am=s/2;                               %minimum energy ellipse semi
major axis
lambda=sqrt(r2mod)*cos(theta/2)/s;    %lambda parameter defined in
BATTIN's book



%We start finding the log(x+1) value of the solution conic:
%%NO MULTI REV --> (1 SOL)
```

```
if N==0
    inn1=-.5233;    %first guess point
    inn2=.5233;     %second guess point
    x1=log(1+inn1);
    x2=log(1+inn2);
    y1=log(x2tof(inn1,s,c,lw,N))-log(t);
    y2=log(x2tof(inn2,s,c,lw,N))-log(t);

    %Newton iterations
    err=1;
    i=0;
    while ((err>tol) && (y1~=y2))
        i=i+1;
        xnew=(x1*y2-y1*x2)/(y2-y1);
        ynew=log(x2tof(exp(xnew)-1,s,c,lw,N))-log(t);
        x1=x2;
        y1=y2;
        x2=xnew;
        y2=ynew;
        err=abs(x1-xnew);
    end
    iter=i;
    x=exp(xnew)-1;


    %%MULTI REV --> (2 SOL) SEPARATING RIGHT AND LEFT BRANCH
else
    if branch=='l'
        inn1=-.5234;
        inn2=-.2234;
    else
        inn1=.7234;
        inn2=.5234;
    end
    x1=tan(inn1*pi/2);
    x2=tan(inn2*pi/2);
    y1=x2tof(inn1,s,c,lw,N)-t;

    y2=x2tof(inn2,s,c,lw,N)-t;
    err=1;
    i=0;

    %Newton Iteration
    while ((err>tol) && (i<60) && (y1~=y2))
        i=i+1;
        xnew=(x1*y2-y1*x2)/(y2-y1);
        ynew=x2tof(atan(xnew)*2/pi,s,c,lw,N)-t;
        x1=x2;
        y1=y2;
        x2=xnew;
        y2=ynew;
        err=abs(x1-xnew);
    end
    x=atan(xnew)*2/pi;
    iter=i;
end

%The solution has been evaluated in terms of log(x+1) or
tan(x*pi/2), we
%now need the conic. As for transfer angles near to pi the lagrange
```

```
%coefficient technique goes singular (dg approaches a zero/zero
that is
%numerically bad) we here use a different technique for those
cases. When
%the transfer angle is exactly equal to pi, then the ih unit vector
is not
%determined. The remaining equations, though, are still valid.


a=am/(1-x^2);                       %solution semimajor axis
%calcolo psi
if x<1 %ellisse
    beta=2*asin(sqrt((s-c)/2/a));
    if lw
        beta=-beta;
    end
    alfa=2*acos(x);
    psi=(alfa-beta)/2;
    eta2=2*a*sin(psi)^2/s;
    eta=sqrt(eta2);
else %iperbole
    beta=2*asinh(sqrt((c-s)/2/a));
    if lw
        beta=-beta;
    end
    alfa=2*acosh(x);
    psi=(alfa-beta)/2;
    eta2=-2*a*sinh(psi)^2/s;
    eta=sqrt(eta2);
end
p=r2mod/am/eta2*sin(theta/2)^2;     %parameter of the solution
sigma1=1/eta/sqrt(am)*(2*lambda*am-(lambda+x*eta));
ih=vers(vett(r1,r2)');
if lw
    ih=-ih;
end

vr1 = sigma1;
vt1 = sqrt(p);
v1  = vr1 * r1   +   vt1 * vett(ih,r1)';

vt2=vt1/r2mod;
vr2=-vr1+(vt1-vt2)/tan(theta/2);
v2=vr2*r2/r2mod+vt2*vett(ih,r2/r2mod)';
v1=v1*V;
v2=v2*V;
a=a*R;
p=p*R;



%-------------------------------------------------------------------
--------
%Subfunction that evaluates the time of flight as a function of x
function t=x2tof(x,s,c,lw,N)

am=s/2;
a=am/(1-x^2);
if x<1 %ELLISSE
    beta=2*asin(sqrt((s-c)/2/a));
```

```
    if lw
        beta=-beta;
    end
    alfa=2*acos(x);
else   %IPERBOLE
    alfa=2*acosh(x);
    beta=2*asinh(sqrt((s-c)/(-2*a)));
    if lw
        beta=-beta;
    end
end
t=tofabn(a,alfa,beta,N);


%-----------------------------------------------------------------------
--------
function t=tofabn(sigma,alfa,beta,N)
%
%subfunction that evaluates the time of flight via Lagrange
expression
%
if sigma>0
    t=sigma*sqrt(sigma)*((alfa-sin(alfa))-(beta-sin(beta))+N*2*pi);
else
    t=-sigma*sqrt(-sigma)*((sinh(alfa)-alfa)-(sinh(beta)-beta));
end

%-----------------------------------------------------------------------
--------
function v=vers(V)
%subfunction that evaluates unit vectors
v=V/sqrt(V'*V);



%-----------------------------------------------------------------------
--------
function ansd = vett(r1,r2)
%
%subfunction that evaluates vector product
ansd(1)=(r1(2)*r2(3)-r1(3)*r2(2));
ansd(2)=(r1(3)*r2(1)-r1(1)*r2(3));
ansd(3)=(r1(1)*r2(2)-r1(2)*r2(1));




%-----------------------------------------------------------------------
--------
function  [r,v,E]=pleph_an ( mjd2000, planet);
%
%The original version of this file was written in Fortran 77
%and is part of ESOC library. The file has later been translated
under the
%name uplanet.m in Matlab, but that version was affected by several
%mistakes. In particular Pluto orbit was affected by great
uncertainties.
%As a consequence its analytical eph have been substituted by a
fifth order
```

```
%polynomial least square fit generated by Dario Izzo (ESA ACT).
JPL405
%ephemerides (Charon-Pluto barycenter) have been used to produce
the
%coefficients. WARNING: Pluto ephemerides should not be used
outside the
%range 2000-2100;
%
%The analytical ephemerides of the nine planets of our solar system
are
%returned in rectangular coordinates referred to the ecliptic J2000
%reference frame. Using analytical ephemerides instead of real ones
(for
%examples JPL ephemerides) is faster but not as accurate,
especially for
%the outer planets
%
%Date:                  28/05/2004
%Revision:              1
%Tested by:             ----------
%
%
%Usage: [r,v,E]=pleph_an ( mjd2000 , IBODY );
%
%Inputs:
%           mjd2000:    Days elapsed from 1st of January 2000
counted from
%           midnight.
%           planet:     Integer form 1 to 9 containing the number
%           of the planet (Mercury, Venus, Earth, Mars, Jupiter,
Saturn,
%           Uranus, Neptune, Pluto)
%
%Outputs:
%           r:         Column vector containing (in km) the
position of the
%                      planet in the ecliptic J2000
%           v:         Column vector containing (in km/sec.) the
velocity of
%                      the planet in the ecliptic J2000
%           E:         Column Vectors containing the six keplerian
parameters,
%                      (a,e,i,OM,om,Eccentric Anomaly)
%
%Xref:      the routine needs a variable mu on the workspace whose
10th
%           element should be the sun gravitational parameter in
KM^3/SEC^2

mu = 1.327124400180000e+011;
RAD=pi/180;
KM=1.495978706910000e+008;


%
%  T = JULIAN CENTURIES SINCE 1900
%
T   = (mjd2000 + 36525.00)/36525.00;
TT  = T*T;
TTT = T*TT;
%
```

```
%  CLASSICAL PLANETARY ELEMENTS ESTIMATION IN MEAN ECLIPTIC OF DATE
%
switch planet
    %
    %  MERCURY
    %
    case 1
        E(1) = 0.38709860;
        E(2) = 0.205614210 + 0.000020460*T - 0.000000030*TT;
        E(3) = 7.002880555555555560 + 1.86083333333333333e-3*T -
1.83333333333333333e-5*TT;
        E(4) = 4.71459444444444444e+1 + 1.185208333333333330*T +
1.73888888888888889e-4*TT;
        E(5) = 2.87537527777777778e+1 + 3.70280555555555556e-1*T
+1.20833333333333333e-4*TT;
        XM   = 1.49472515288888889e+5 + 6.38888888888888889e-6*T;
        E(6) = 1.02279380555555556e2 + XM*T;
        %
        %  VENUS
        %
    case 2
        E(1) = 0.72333160;
        E(2) = 0.006820690 - 0.000047740*T + 0.0000000910*TT;
        E(3) = 3.393630555555555560 + 1.00583333333333333e-3*T -
9.72222222222222222e-7*TT;
        E(4) = 7.57796472222222222e+1 + 8.9985e-1*T + 4.1e-4*TT;
        E(5) = 5.43841861111111111e+1 + 5.08186111111111111e-1*T -
1.38638888888888889e-3*TT;
        XM   = 5.8517803875e+4 + 1.28605555555555556e-3*T;
        E(6) = 2.12603219444444444e2 + XM*T;
        %
        %  EARTH
        %
    case 3
        E(1) = 1.000000230;
        E(2) = 0.016751040 - 0.000041800*T - 0.0000001260*TT;
        E(3) = 0.00;
        E(4) = 0.00;
        E(5) = 1.01220833333333333e+2 + 1.7191750*T +
4.52777777777777778e-4*TT + 3.33333333333333333e-6*TTT;
        XM   = 3.599904975e+4 - 1.50277777777777778e-4*T -
3.33333333333333333e-6*TT;
        E(6) = 3.58475844444444444e2 + XM*T;
        %
        %  MARS
        %
    case 4
        E(1) = 1.5236883990;
        E(2) = 0.093312900 + 0.0000920640*T - 0.0000000770*TT;
        E(3) = 1.850333333333333330 - 6.75e-4*T +
1.26111111111111111e-5*TT;
        E(4) = 4.87864416666666667e+1 + 7.70991666666666667e-1*T -
1.38888888888888889e-6*TT - 5.33333333333333333e-6*TTT;
        E(5) = 2.85431761111111111e+2 + 1.06976666666666670*T +
1.3125e-4*TT + 4.13888888888888889e-6*TTT;
        XM   = 1.91398585e+4 + 1.80805555555555556e-4*T +
1.19444444444444444e-6*TT;
        E(6) = 3.19529425e2 + XM*T;

        %
        %  JUPITER
```

```
        %
    case 5
        E(1) = 5.2025610;
        E(2) = 0.048334750 + 0.000164180*T  - 0.00000046760*TT -
0.00000000170*TTT;
        E(3) = 1.308736111111111110 - 5.69611111111111111e-3*T +
3.88888888888888889e-6*TT;
        E(4) = 9.94433861111111111e+1 + 1.010530*T +
3.52222222222222222e-4*TT - 8.51111111111111111e-6*TTT;
        E(5) = 2.73277541666666667e+2 + 5.99431666666666667e-1*T +
7.0405e-4*TT + 5.07777777777777778e-6*TTT;
        XM   = 3.03469202388888889e+3 - 7.21588888888888889e-4*T +
1.78444444444444444e-6*TT;
        E(6) = 2.25328327777777778e2 + XM*T;
        %
        %  SATURN
        %
    case 6
        E(1) = 9.5547470;
        E(2) = 0.055892320 - 0.00034550*T - 0.0000007280*TT +
0.000000000740*TTT;
        E(3) = 2.492519444444444440 - 3.91888888888888889e-3*T -
1.54888888888888889e-5*TT + 4.44444444444444444e-8*TTT;
        E(4) = 1.12790388888888889e+2 + 8.73195138888888889e-1*T -
1.52180555555555556e-4*TT - 5.30555555555555556e-6*TTT;
        E(5) = 3.38307772222222222e+2 + 1.085220694444444440*T +
9.78541666666666667e-4*TT + 9.91666666666666667e-6*TTT;
        XM   = 1.22155146777777778e+3 - 5.01819444444444444e-4*T -
5.19444444444444444e-6*TT;
        E(6) = 1.75466216666666667e2 + XM*T;
        %
        %  URANUS
        %
    case 7
        E(1) = 19.218140;
        E(2) = 0.04634440 - 0.000026580*T + 0.0000000770*TT;
        E(3) = 7.72463888888888889e-1 + 6.25277777777777778e-4*T +
3.95e-5*TT;
        E(4) = 7.34770972222222222e+1 + 4.98667777777777778e-1*T +
1.31166666666666667e-3*TT;
        E(5) = 9.80715527777777778e+1 + 9.85765e-1*T -
1.07447222222222222e-3*TT - 6.05555555555555556e-7*TTT;
        XM   = 4.28379113055555556e+2 + 7.88444444444444444e-5*T +
1.11111111111111111e-9*TT;
        E(6) = 7.26488194444444444e1 + XM*T;
        %
        %  NEPTUNE
        %
    case 8
        E(1) = 30.109570;
        E(2) = 0.008997040 + 0.0000063300*T - 0.0000000020*TT;
        E(3) = 1.77924166666666670 - 9.54361111111111111e-3*T -
9.11111111111111111e-6*TT;
        E(4) = 1.30681358333333333e+2 + 1.0989350*T +
2.49866666666666667e-4*TT - 4.71777777777777778e-6*TTT;
        E(5) = 2.76045966666666667e+2 + 3.25639444444444444e-1*T +
1.4095e-4*TT + 4.11333333333333333e-6*TTT;
        XM   = 2.18461339722222222e+2 - 7.03333333333333333e-5*T;
        E(6) = 3.77306694444444444e1 + XM*T;
        %
        %  PLUTO
```

```
        %
    case 9
        %Fifth order polynomial least square fit generated by Dario
Izzo
        %(ESA ACT). JPL405 ephemerides (Charon-Pluto barycenter)
have been used to produce the coefficients.
        %This approximation should not be used outside the range
2000-2100;
        T=mjd2000/36525;
        TT=T*T;
        TTT=TT*T;
        TTTT=TTT*T;
        TTTTT=TTTT*T;
        E(1)=39.34041961252520 + 4.33305138120726*T -
22.93749932403733*TT + 48.76336720791873*TTT -
45.52494862462379*TTTT + 15.55134951783384*TTTTT;
        E(2)=0.24617365396517 + 0.09198001742190*T -
0.57262288991447*TT + 1.39163022881098*TTT - 1.46948451587683*TTTT
+ 0.56164158721620*TTTTT;
        E(3)=17.16690003784702 - 0.49770248790479*T +
2.73751901890829*TT - 6.26973695197547*TTT + 6.36276927397430*TTTT
- 2.37006911673031*TTTTT;
        E(4)=110.222019291707 + 1.551579150048*T -
9.701771291171*TT + 25.730756810615*TTT - 30.140401383522*TTTT +
12.796598193159 * TTTTT;
        E(5)=113.368933916592 + 9.436835192183*T -
35.762300003726*TT + 48.966118351549*TTT - 19.384576636609*TTTT -
3.362714022614 * TTTTT;
        E(6)=15.17008631634665 + 137.023166578486*T +
28.362805871736*TT - 29.677368415909*TTT - 3.585159909117*TTTT +
13.406844652829 * TTTTT;
end
%
%  CONVERSION OF AU INTO KM, DEG INTO RAD
%
E(1)     =     E(1)*KM;
for  I = 3: 6
    E(I)     = E(I)*RAD;
end
E(6)     = mod(E(6), 2*pi);

%Conversion from mean anomaly to eccentric anomaly via Kepler's
equation
EccAnom=M2E(E(6),E(2));
E(6)=EccAnom;

%Calcolo velocità e posizione nel sistema J2000
[r,v]=conversion(E);



%-------------------------------------------------------------------
--------
function [r,v] = conversion (E)
%
% Parametri orbitali

muSUN=  1.327124280000000e+011;       %gravitational parameter for
the sun
```

```
a=E(1);
e=E(2);
i=E(3);
omg=E(4);
omp=E(5);
EA=E(6);


% Grandezze definite nel piano dell'orbita

b=a*sqrt(1-e^2);
n=sqrt(muSUN/a^3);

xper=a*(cos(EA)-e);
yper=b*sin(EA);

xdotper=-(a*n*sin(EA))/(1-e*cos(EA));
ydotper=(b*n*cos(EA))/(1-e*cos(EA));

% Matrice di trasformazione da perifocale a ECI

R(1,1)=cos(omg)*cos(omp)-sin(omg)*sin(omp)*cos(i);
R(1,2)=-cos(omg)*sin(omp)-sin(omg)*cos(omp)*cos(i);
R(1,3)=sin(omg)*sin(i);
R(2,1)=sin(omg)*cos(omp)+cos(omg)*sin(omp)*cos(i);
R(2,2)=-sin(omg)*sin(omp)+cos(omg)*cos(omp)*cos(i);
R(2,3)=-cos(omg)*sin(i);
R(3,1)=sin(omp)*sin(i);
R(3,2)=cos(omp)*sin(i);
R(3,3)=cos(i);

% Posizione nel sistema inerziale

r=R*[xper;yper;0];
v=R*[xdotper;ydotper;0];



%-------------------------------------------------------------------
--------
function E=M2E(M,e)
%
i=0;
tol=1e-10;
err=1;
E=M+e*cos(M);   %initial guess
while err>tol && i<100
    i=i+1;
    Enew=E-(E-e*sin(E)-M)/(1-e*cos(E));
    err=abs(E-Enew);
    E=Enew;
end


%-------------------------------------------------------------------
--------
function M=E2M(E,e)
%
```

```
%Transforms the eccentric anomaly to mean anomaly. All i/o in
radians
%
%Usage: M=E2M(E,e)
%
%Inputs :   E : Eccentric anomaly or Gudermannian if e>1
%           e : Eccentricity of considered orbit
%
%Output :   M : Mean anomaly or N if e>1

if e<1 %Ellipse, E is the eccentric anomaly
    M=E-e*sin(E);
else  %Hyperbola, E is the Gudermannian
    M=e*tan(E)-log(tan(E/2+pi/4));
end




%-------------------------------------------------------------------
--------
function [r,v]=CUSTOMeph(jd,epoch,keplerian,flag)
%
%Returns the position and the velocity of an object having
keplerian
%parameters epoch,a,e,i,W,w,M
%
%Usage:     [r,v]=CUSTOMeph(jd,name,list,data,flag)
%
%Inputs:    jd: julian date
%           epoch: mjd when the object was observed (referred to M)
%           keplerian: vector containing the keplerian orbital
parameters
%
%Output:    r = object position with respect to the Sun (km if
flag=1, AU otherwise)
%           v = object velocity ( km/s if flag=1, AU/days otherways
)
%
%Revisions :    Function added 04/07

global AU mu

muSUN = mu(11);
     a=keplerian(1)*AU; %in km
     e=keplerian(2);
     i=keplerian(3);
     W=keplerian(4);
     w=keplerian(5);
     M=keplerian(6);
     jdepoch=mjd2jed(epoch);
     DT=(jd-jdepoch)*60*60*24;
     n=sqrt(muSUN/a^3);
     M=M/180*pi;
     M=M+n*DT;
     M=mod(M,2*pi);
     E=M2E(M,e);
     [r,v]=par2IC([a,e,i/180*pi,W/180*pi,w/180*pi,E],muSUN);
     if flag~=1
         r=r/AU;
```

```
        v=v*86400/AU;
    end


%-------------------------------------------------------------------
--------
function t = time2distance(r0,v0,rtarget)
%
%Usage: t = time2distance(r0,v0,rtarget)
%
%Inputs:
%           r0:    column vector for the position (mu=1)
%           v0:    column vector for the velocity (mu=1)
%           rtarget: distance to be reached
%
%Outputs:
%           t:     time taken to reach a given distance
%
%Comments:  everything works in non dimensional units

r0norm = norm(r0);
if r0norm < rtarget
    out = sign(r0'*v0);
    E = IC2par(r0,v0,1);
    a = E(1); e = E(2); E0 = E(6); p = a * (1-e^2);
    %If the solution is an ellipse
    if e<1
        ra = a * (1+e);
        if rtarget>ra
            t = NaN;
        else %we find the anomaly where the target distance is
reached
            ni = acos((p/rtarget-1)/e);         %in 0-pi
            Et = ni2E(ni,e);          %in 0-pi
            if out==1
                t = a^(3/2)*(Et-e*sin(Et)-E0 +e*sin(E0));
            else
                E0 = -E0;
                t = a^(3/2)*(Et-e*sin(Et)+E0 - e*sin(E0));
            end
        end
    else %the solution is a hyperbolae
        ni = acos((p/rtarget-1)/e);         %in 0-pi
        Et = ni2E(ni,e);          %in 0-pi
        if out==1
                t = (-a)^(3/2)*(e*tan(Et)-log(tan(Et/2+pi/4))-
e*tan(E0)+log(tan(E0/2+pi/4)));
            else
                E0 = -E0;
                t = (-a)^(3/2)*(e*tan(Et)-
log(tan(Et/2+pi/4))+e*tan(E0)-log(tan(E0/2+pi/4)));
        end
    end
else
    t=12;
end


%-------------------------------------------------------------------
--------
```

```
function jd = mjd20002jed(mjd2000)
%This function converts mean julian date 2000 to julian date
jd=mjd2000+2451544.5;


%-------------------------------------------------------------------------
--------
function jd = mjd2jed(mjd)
% This function converts mean Julian date into Julian date
jd = mjd +2400000.5;


-------------------------------------------------------------------------
```

APÉNDICE 4. IMÁGENES DE LAS PORTADAS DE LOS CAPÍTULOS

- PORTADA: Guillermocracia.blogspot.com
- CAPÍTULO 1: Tania-temasfísica.blogspot.com
- CAPÍTULO 2: actualimatica.wordpress.com
- CAPÍTULO 3: davidparcerisa.org
- CAPÍTULO 4: aperez4.blogspot.com
- CAPÍTULO 5: es.paperblog.com
- CAPÍTULO 6: www.circpau.org
- CAPÍTULO 7: editwallpaper.com
- CAPÍTULO 8: danielmarin.naukas.com
- CAPÍTULO 9: jassy2012.deviantart.com
- BIBLIOGRAFÍA: www.cosassencillas.com
- APÉNDICES: www.drgen.com.ar

"Que otros se jacten de las páginas que han escrito; a mí me enorgullecen las que he leído."

Jorge Luis Borges (1899-1986)

REFERENCIAS BIBLIOGRÁFICAS

1. REFERENCIAS BIBLIOGRÁFICAS

- Alegre Martínez, Jesús Francisco (2004). “Metaheurísticos. Aplicaciones a los modelos logísticos en la industria del automóvil”. Servicio de publicaciones de la Universidad de Burgos. ISBN: 84-95211-94-7. Primera edición 2004.

- Alonso, Sergio; Cordón, Óscar; Fernández de Viana, Iñaki; Herrera, Francisco (2002). “La metaheurística de optimización basada en colonias de hormigas: modelos y nuevos enfoques”. Trabajo realizado en el marco del proyecto “Mejora de metaheurísticas mediante hibridación y sus aplicaciones” publicación de la Universidad de Granada. 2002.

- Apóstol, Tom M.(1980). “Calculus” Volumen 2. Editorial Reverté, s.a. ISBN 84-291-5003 Cuarta edición 1980.

- Battiti R. (1996). “Reactive Search: Toward Self-Tuning Heuristics”. In V. J. Rayward-Smith, editor, Modern Heuristic Search Methods, chapter 4, págs. 61-83. John Wiley and Sons Ltd, 1996.

- Binjie Gu_ and Feng Pan (2013). “Modified Gravitational Search Algorithm with Particle Memory ability and its application”. International Journal of Innovative Computing, Information and Control ICIC International Volume 9, (11), Nov. 2013 págs. 4531-4544 2013 ISSN 1349-4198.

- Burden, Richard L., Faires J. Douglas. (1985) “Análisis numérico”. Grupo Editorial Iberoamérica. ISBN. 968-7270-09-8. Primera edición 1985.

- Catalá de Alemany. (1966). “Física general”. Editorial Guerra. Nº de registro V. 843-65 Cuarta edición1966.

- Casado, Silvia (2005) “Planificación de turnos en un aeropuerto: uso de simulación y metaheurísticos”. Servicio de publicaciones de la Universidad de Burgos. ISBN 84-96394-12-3. Primera edición 2005.

- Chuang, CL; Jiang, Joe-Air. (2007) “Integrated radiation optimization inspired by the gravitational radiation in the curvatureof space-time”.IEEE congress on evolutionary computation. Vols 1-10, 2007. págs. 3157-3164.

- Crowston W.B., Glover F., Thompson G.L. y J.D. (1963). “Probabilistic and Parametric Learning Combinations of Local Job Shop Scheduling Rules”. ONR Research Memorandum No. 117, GSIA, Carnegie Mellon University. Pittsburgh, PA.

- Danoy, G., Dorronsoro, B. y Bouvry P. (2012). "New-State-of-theárt Results for Cassini 2 Global Trajectory Optimization Problem". Acta Futura 5 págs. 65-72. DOI:10.2420/AF05.2012.65.

- Ding, DS (Ding, Dongsheng); Qi, DL (Qi, Donglian); Luo, XP (Luo, Xiaoping); Chen, JF (Chen, Jinfei); Wang, XJ (Wang, Xuejie); Du, PY (Du, Pengyin) (2012). " Convergence analysis and performance of an extended central force optimization algorithm". APPLIED MATHEMATICS AND COMPUTATION. Vol. 219. nº 4. págs. 2246-2259.

- Dorigo, M. Colorni, A. y Maniezzo, V. (1991) "Distributed Optimization by Ant Colonies". Actas de La première conférence européenne sur la vie artificielle, París Elvesier Publishing, págs. 134-142.

- Dorigo M., Maniezzo V. y Colorni A. (1996). “The Ant System: Optimization by a Colony of Cooperating”*. Agents*. *IEEE Transactions on Systems*, Man, and Cybernetics-Part B, 26(1): págs. 29-41.

- “The Ant System: Optimization by a Colony of Cooperating”*. Agents*. *IEEE Transactions on Systems*, Man, and Cybernetics-Part B, 26(1):29-41.

- Dowsland K.A. (1993).“Some experiments with Simulated Annealing techniques for Packing problems”. European Journal of Operations Research. Vol. 68, num.1, 3, págs. 389-399.

- Einstein, Albert (25 de Noviembre de 1915). “Die Feldgleichungun der Gravitation”. Sitzungsberichte der Preussichen Akademie der Wissenschaften zu. Berlín (en alemán): págs. 844-847.

- Feo T.A. y Resende M.G.C. (1989). "A Probabilistic heuristic for a computationally difficult Set Covering Problem". *Operations Research Letters*, 8, págs. 67-71.

- Feo T.A. y Resende M.G.C. (1995). "Greedy Randomized Adaptive Search Procedures". *Journal of Global Optimization.* Vol. 2, págs. 1-27.

- Feynman, Richard P.; Leighton, Robert B. and Sands, Matthew (1977). “The Feynman lectures on physics”. Vol 1. Addison-Wesley publishing company. ISBN 0-201-02116-1-P. Sixth printing, February 1977.

- Fisher H. y Thompson G. L. (1963). “Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules”. Industrial Scheduling, J.F. Muth and G. L. Thompson (eds.) Prentice-Hall, págs. 225-251.

- Formato R.A. (2007) “Central Force Optimization: A new metaheuristic with applications in applied electromagnetics”. Progress in Electromagnetics Research. Vol PIER 77 (2007). págs. 425-491.

- Formato, R.A. (2010) “Improved C.F.O. algorithm for antenna optimization”. Progress in Electromagnetics Research B, Vol. 19, págs. 405-425, 2010.

- Formato, R.A. (2010 b) "Pseudorandomness in Central Force Optimization". arXiv:1001.0317v2 [es.NE], www.arXiv.org

- Formato, R.A. (2013) “Dipole-Loaded Monopole optimized using VSO, v.2”. available on line at http://arxiv.org/abs/1307.0220

- Gauci, M . ; Dodd, T.J. y Gross, R. (2012). " Why 'GSA: a gravitational search algorithm' is not genuinely based on the law of gravity". NATURAL COMPUTING. nº 4. (número especial). pág.s 719-720.

- Glover F. (1989). “Tabú Search: Part I”. *ORSA Journal on Computing*, Vol 1, págs. 190-206.

- Glover F. (1990). “Tabú Search: Part II”. *ORSA Journal on Computing*, Vol 2, págs. 4-32.

- Glover F. (1994).“Genetic Algorithms and Scatter Search: Unsuspected Potencials”. *Statistics and Computing*, 4, págs. 131-140.

- Glover F. (1996-a). “Búsqueda Tabú” en Optimización Heurística y Redes Neuronales. Adenso Díaz (coordinador). Paraninfo, Madrid. págs. 105-143.

- Glover F. (1998). “A template for Scatter Search and Path Relinking”, in *Lecture Notes in Computer Science*, 1363, J.k.Hao, E. Lutton, E. Ronald, M. Schoenauer, D. Snyers (Eds.), págs. 13-54.

- Glover Fred. Páginas Web:

- http://spot.colorado.edu/~glover/ y http://faculty.bus.olemiss.edu/fglover/

- Glover F. y Laguna M. (1993). “Tabu Search” en *Modern Heuristic Techniques for Combinatorial Problems*. Reeves C. (ed.). Blackwell Scientific Publishing, págs., 70-141.

- Glover F. y Laguna,M. (1997). “Tabu Search“. Kluver Academic Publishers, Boston.

- Glover F. y Laguna M. (2002). “Tabu Search”. En *Handbook of Applied Optimization*, Paradalos, P.M. and Resende, M.G.S. (eds), págs. 194-208. Oxford University Press.

- Glover F., Laguna M. y Martí R. (2000). "Fundamentals of Scatter Search and Path Relinking". *Control and Cybernetics*, volume 29, number 3, págs. 653-684.

- Glover F., Laguna M. y Martí R. (2002). "Scatter Search". Aparecerá en *Theory and Applications of Evolutionary Computation: Recent Trends*, A.Ghosh and S. Tsutsui (Eds.), Springer-Verlag.

- Guzman M. (1975) "Ecuaciones diferenciales ordinarias. Teoría de estabilidad y control". Editorial Alhambra, S.A. Primera edición 1975. ISBN 84-205-0554-4.

- Hamdy A. (2011) “Space Trajectories Optimization using Variable-Chromosome-Length Genetic Algotihms”. Trabajo de Tesis Doctoral. Michigan Technological University 2011.

- Hansen P., Mladenovic N. y Pérez-Britos D. (2001). “Variable Neighborhood Decomposition Search”. *Journal Of Heuristics*, vol 7, nº4 págs. 335-350.

- Harkiolakis N. (2008). “Global optimization of non-lineal systems of Equations by simulating the flight of a projectile in the conformational space” Mathematicam methods, computational techniques, non-linear systems, Intelligent systems. 2008 págs. 414-420.

- Hatamlou, A (Hatamlou, Abdolreza); Abdullah, S (Abdullah, Salwani); Nezamabadi-pour, H (Nezamabadi-pour, Hossein) 2011. " Application of Gravitational Search Algorithm on Data Clustering". ROUGH SETS AND KNOWLEDGE TECHNOLOGY. Lecture Notes in Artificial Intelligence. Vol 6954. págs. 337-346.

- Hernández Oliva, Gonzalo. (2006) “Métodos clásicos de optimización para problemas no-lineales sin restricciones”. Publicación del departamento de Ingeniería matemática de la Universidad de Chile. Mayo 2006.

- Holland J. (1975). “Adaptation in Natural and Artificial Systems”. University of Michigan Press, Ann Abor.

- Hsiao, Ying-Tung; Chuang, Cheng-Long; Jiang, Joe-Air and Chien, Cheng-Chih. (2005) “A Novel Optimization Algorithm: Space Gravitational Optimization” Systems, Man and Cybernetics, IEEE International Conference on Volume 3, Issue, 10-12 October 2005 págs. 2323-2328.

- Kennedy James; Russell Eberhart (1995). "Particle Swarm Optimization" Neural Networks, 1995. Proceedings., IEEE International Conference on Volume 4, Issue, Nov/Dec 1995 págs 1942-1948. print ISBN 0-7803-2768-3.

- Kilinç N. y otros (2013). “Space Gravity Optimization applied to the Feasible Design Target Space Required for a Wide-band Front-end Amplifier”. Progress in Electromagnetics Research Symposium Proceedings, Stockholm, Sweden, Aug. 12-15, 2013 pág. 1495.

- Kincaid, David; Cheney Ward. (1994). "Análisis Numérico. Las matemáticas del cálculo científico" Ed. ADDISON-WESLEY IBEROAMERICANA ISBN 0-201-60130-3. 1994.

- Kindelán, Ultano. (2007). “Resolución de sistemas lineales de ecuaciones: Método del gradiente conjugado” Trabajo realizado para Open Course Ware de la Universidad Politécnica de Madrid (2007).

- Kip Thorne. (2014). "The Sciece of Interstellar" W.W. Norton & Company 2014.

- Kirkpatrick S., Gelatt C. D. y Vecchi M. P. (1983). “Optimization by Simulated Annealing”. *Science*, Vol. 220, págs. 671-680.

- Lagarias, J. C. ; y otros. (1988). "Convergence properties of the Nelder-Mead simplex method in low dimensions" SIAM Journal Optimization. Volume 9, No 1, págs. 112-147.

- Laguna Manuel. (2002). "Scatter Search". En *Handbook of Applied Optimization*, P. M. Pardalos and M. G. C. Resende (eds), págs. 183-193. Oxford University Press.

- Laguna Manuel. Página web: http://bus.colorado.edu/Faculty/Laguna/

- Laguna M. y Armentano V.A. (2002). “Lessons from Applying and Experimenting with Scatter Search”. Aparecerá en Adaptive Memory and Evolution: Tabu Search and Scatter Search, Cesar Rego and Bahram Alidaee (Eds.)

- Laguna, M. and Martí. R. (2005) .“Experimental Testing of Advanced Scatter Search Designs for Global Optimization of Multimodal Functions”. Journal of Global Optimization Vol 33. ISSUE 2. Oct. 2005. págs. 235-255.

- Ludvigsen, Malcolm. (1999). “General Relativity. A geometric approach”. Editorial Cambridge University press. ISBN 0 521 63019 3.Primera edición 1999.

- Marín, D. (2014). "Crónicas desde el gigante anillado: diez años de la Cassini en Saturno". Blog Eureka danielmarin.naukas.com.

- Martí, R. Universidad de Valencia, Spain Juan-José Pantrigo, Universidad Rey Juan Carlos, Spain Abraham Duarte, Universidad Rey Juan Carlos, Spain Vicente Campos, Universidad de Valencia, Spain Fred Glover, OptTek Systems, Inc., USA. 2011. " Scatter Search and Path Relinking: A Tutorial on the Linear Arrangement Problem".

International Journal of Swarm Intelligence Research, 2(2), 1-21, April-June 2011. DOI: 10.4018 / jsir. 2011040101.

- Martínez García, Javier; Moreno Pérez, José Antonio (2007). “Optimización por enjambre para la p-mediana continua y discreta”. Actas del Quinto Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados. MAEB 2007. Puerto de la Cruz, Tenerife Febrero 2007. ISBN: 978-84-690-3470-5.

- Martínez González, J.L. (2001). "Optimización y ajuste de parámetros mediante el método simplex (Nelder-Mead)". 1ª Reunión de EcosimPro, UNED, Madrid 3-4 Mayo 2001.

- Mercado, V. y otros (2013). "Hibridación de Metaheurísticas aplicadas al Problema de Ruteo de Vehículos". ICT-UNPA-70-2013 ISSN: 1852-4516 Aprobado por Resolución num. 0862/13-R-UNPA.

- Menéndez Alonso, Evelyn (2009). “Metaheurística de optimización mediante colonia de hormigas y aplicaciones”. http:// www.monografías. Com. Publicada en Junio 2009.

- Mirjalili S., Zaiton S. (2010). “A New Hybrid PSOGSA Algorithm for Function Optimization”. International Conference on Computer and Information Application (ICCIA 2010). 3-5 Dec. 2010. 978-1-4244-8598-7 (10) 2010 IEEE.

- Mladenovic, N. (1995). “A Variable Neighborhood Algorithm – A New Metaheuristic for Combinatorial Optimization”. Abstracts of papers presented at *Optimization Days*, Montreal, pág.112.

- Mladenovic, N. y Hansen P. (1997). “Variable Neighborhood Search”. *Computers and Operations Research*, 24, págs. 1097-1100.

- Moscato, P. (1989). “On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms”. *Caltech Concurrent Computation Program*, C3P Report 826.

- Moscato, P. y Norman M. G.(1992). “A Memetic Approach for the Traveling Salesman Problem Implementation of a Computational Ecology for Combinatorial Optimization on Message-Passing Systems” en *Parallel Computing and Transputer Applications*, págs. 177-186, Amsterdam, 1992. Editores: M. Valero, E. Onate, M. Jane, J. L. Larriba, y B. Suarez, IOS Press.

- Moscato, P. y Cotta C. (2002). “A Gentle Introduction to Memetic Algorithms” que aparecerá en *Handbook of Metaheuristics*. Kluwer.

- Nelder,J.A. y Mead, R. (1965) "A simplex method for function minimization". Computer Journal, 7, págs. 308-313.

- Newton, I. (1687, 1987) “Principios matemáticos de la filosofía natural”. Volumen I. Alianza Editoria. ISBN 84-206-2511-6. Primera edición 1987.

- Nyew, H.M. ; Abdelkhalik, O. y Onder, N. (2015) " Structured-Chromosome Evolutionary Algorithms for Variable-Size Autonomous Interplanetary Trajectory Planning Optimization". JOURNAL OF AEROSPACE INFORMATION SYSTEMS. Vol 12, nº 3, págs. 314-328.

- Osman I.H. y Kelly J.P. (1996). *Meta-Heuristics: Theory & Applications* Kluwer Academic Publisher, Boston.

- Pacheco J.A. (1997). “El Temple Simulado: Descripción y Aplicaciones en Optimización Combinatoria.” *Lección de la Cátedra* de Escuela de la Universidad de Burgos, Julio 1997.

- Palanikkumar, D. y otros (2012). “A Gravitational Search Algorithm for effective Web Service Selection for Composition with enhanced QoS in SOA”. *International Journal of Computer Applications (0975 – 8887) Volume 42– No.8, March 2012.*

- Pérez, Jesús R., Basterrechea, José (2005). “Contribución a los métodos de optimización basados en procesos naturales y su aplicación a la medida de antenas en campo póximo”. Servicio de publicaciones de la Universidad de Cantabria. ISBN SA. 231-2007 / 978-84-690-5924-1. Oct. 2005.

- Rashedi, E., Hossein Nezamabadi-pour; Saeid Saryazdi.(2009) “GSA : a gravitational search.” Information Sciences 179 (13) 2009 págs. 2232-2248.

- Rashedi, E. , Hossein Nezamabadi-pour; Saeid Saryazdi.(2010) “BGSA: binary gravitational search algorithm”. Natural Computing 9 (3): págs. 727-745.

- Resende Mauricio. Página web **http://www.research.att.com/~mgcr/**

- Rosing K.E. (1997). “Heuristic Concentration: An Introduction with Examples”. *The Tenth Meeting of the European Chapter on Combinatorial Optimization*. Tenerife. Spain. May, 1997.

- Rosing K.E. y Revelle C.S. (1997). “Heuristic Concentration: Two Stage solution Construction”. *European Journal of Operational Research 97*, págs.75-86.

- Roy,R.K. (1990). "A Primer on the Taguchi Method". Van Nostrand Reinhold, New York.

- Silver E.A., Vidal R.V. y Werra (de) D. (1980). “A Tutorial on Heuristic Methods”. European Journal of Operational Research, Vol. 5.1980.

- Simmons, G.F. (1977)."Ecuaciones diferenciales con aplicaciones y notas históricas" . Editorial McGraw-Hill . ISBN-0-07-057375-1.

- Sudin, S. y otros (2013). “A Modified Gravitational Search Algorithm for Discrete Optimization Problem”. IJSSST, Vol. 15, No.1 ISSN: 1 51 473-804x online, 1473-8031 print.

- Tu, Zhenguo and Lu, Yong. (2004). “A robust stochastic genetic algorithm (StGA) for global numerical optimization”. IEEE Transactions on evolutionary Computation. Vol 8. Nª 5. October 2004.

- Voudouris C. y Tsang E. (1999). "Guided Local Search for the Traveling Salesman Problem". *European Journal of Operations Research*. Vol. 113, págs. 469-499.

- Yao, Xin; Liu Yong and Lin Guangming (1999).”Evolutionary programming made faster”. IEEE Transactions on evolutionary computation. Vol 3. Nº 2. July 1999. págs. 82-102.

- Webs con información de las funciones

- **http://wwwoptima.amp.i.kyotou.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page364.htm**

- **http://solon.cma.univie.ac.at/glopt.html**

- Web de la agencia espacial europea

  http://www.esa.int/gsp/ACT/inf/projects/gtop/gtop.html