UNIVERSIDAD DEL BÍO-BÍO

Facultad de Ciencias Empresariales
Departamento de Sistemas de Información

Profesor Guía
Sr. Pedro Campos Soto

Estudio, Desarrollo e Implementación de Algoritmos para la Paralelización de una Biblioteca de Cálculo Científico basada en Aritmética Intervalar

INFORME FINAL DE HABILITACIÓN PROFESIONAL

Fecha de Presentación
30 de diciembre del 2008

Nombres de los Alumnos
Esteban Erices Peña
Renato Llanos Soto

AGRADECIMIENTOS

Con mucho cariño dedicado a mi linda mama Patricia, a mi sabio padre Reinaldo, mi querido hermano Sebastián y mi gran amor Daniela.

Gracias por apoyarme y creer en mí siempre.

Renato.

Dedicado especialmente a mis padres María y Juan por el apoyo y la confianza, como también a mis hermanos Cesar, Scarlette y Constanza. A los familiares y amigos que siempre desearon un exitoso termino de esta etapa.

Un gran abrazo.

Esteban.

ÍNDICE GENERAL

RESUMEN

Cada día existen problemas de gran dificultad y que necesitan mayor poder de procesamiento para su resolución. Algunas áreas con fuerte demanda por la computación de alto desempeño son: química molecular (industria farmacéutica), astrofísica, biología genética, predicción del tiempo, computación gráfica, etc. (Rodríguez, 2004)

La necesidad de determinar óptimos globales se presenta en numerosas disciplinas que modelan sistemas del mundo real, como la ingeniería, la química, sistemas financieros, medicina, física, tecnología. (Sahinidis, 2004) (Penot, 1989) (Campos y Valdés-González, 2006). Muchos problemas como los de matemática e ingeniería requieren una búsqueda completa.

La optimización global trata de encontrar la mejor solución a un problema de optimización, realizando búsquedas completas sobre un conjunto de soluciones factibles, esto quiere decir que abarca todos los puntos existentes en la región de búsqueda dada. Encontrar un mínimo global es mucho más costoso que encontrar un mínimo local.

Los problemas en donde se aplica la optimización global en general no pueden ser resueltos aplicando métodos de optimización local, ya que pueden haber muchos mínimos de la función dada en el dominio considerado. Bajo estas condiciones los mínimos globales solo pueden ser localizados con ayuda computacional.

El rendimiento de algunos computadores determina el desempeño en la resolución de problemas, existen problemas tan complejos, que su solución podría tomar muchos años (Rodríguez, 2004). Para este tipo de problemas usamos paralelización, dado que la computación paralela es la forma más barata de mejorar el tiempo de respuesta para ciertos problemas puntuales. (Peña y Muñoz, 2007)

Se entiende como computación paralela a una técnica de computación que descompone un gran problema en tareas menores, donde estás puedan ser computadas en diferentes máquinas o elementos de proceso al mismo tiempo. (Wilkinson, 2005).

Se conoce como balanceo de carga a una técnica que acrecienta los recursos, explotando el paralelismo, y acortando el tiempo de respuesta mediante una distribución apropiada de la aplicación. (Bozyigita, 2000).

La estrategia previamente implementada (Peña y Muñoz, 2007) se basa en la división y asignación de datos implementados de dos maneras: por división del espacio inicial de búsqueda, y por división de cajas resultado obtenidas mediante una optimización inicial.

Se desarrollaron y testearon nuevas estrategias para implementar el algoritmo de optimización global, en donde se cree obtener mejores resultados. Se diseñaron e implementaron cuatro versiones paralelizadas, con y sin balanceo de carga, las cuales fueron sometidas a un sinnúmero de pruebas para evaluar su comportamiento a distintos problemas. Se utilizó como base al algoritmo de división de cajas resultado, se trabajó en una arquitectura multi-computador y en una red de interconexión de computadores (clúster). La comunicación se lleva a cabo por medio de la librería de programación paralela MPI, la cual gestiona la comunicación por medio de paso de mensajes entre los distintos procesadores.

Los algoritmos fueron desarrollados bajo el lenguaje de programación C++.

Se utilizaron herramientas bajo los preceptos del software libre, proporcionando mayor libertad a la comunidad interesada en este campo de investigación, contribuyendo así al desarrollo de la investigación científica en la comunidad universitaria, proporcionando una base para futuras investigaciones y desarrollo en un campo de interés creciente en el ámbito internacional.

CAPÍTULO 1. FUNDAMENTACIÓN DE LA HABILITACIÓN PROFESIONAL

1.1 Introducción al Problema

La optimización es un aspecto muy relevante en el desarrollo de muchas áreas, como la química, física, matemática, economía, producción etc., y las personas encargadas de estas áreas consumen grandes porciones de tiempo, esfuerzo y dinero en lograr resultados lo más aproximados posibles al optimo.

Lo que busca la optimización es la solución de problemas que se aplican a las actividades que se realizan en el mundo real como por ejemplo la optimización de una función de producción para una fábrica de autos, optimizar la confección de una fórmula para la desinfección de algún tipo de virus, minimizar el daño producido por el roce a 300 k/h que se provocan al manejar autos de formula 1, maximizar ganancias de alguna cartera de inversión etc.

La optimización global es un tema importante pero a la vez complejo, no existiendo soluciones eficientes que resuelvan el problema general de optimización global (Campos, 2004). La solución de un problema de optimización se compone de uno o muchos óptimos, que pueden ser mínimos o máximos, locales, lo cual es una solución optima para un conjunto de soluciones cercanas. El óptimo global es la mejor solución de todos los óptimos locales encontrados y representa a la mejor solución de una función dada. Resulta difícil poder identificar si un óptimo representa realmente una solución global a alguna función dada.

Existe una biblioteca de optimización global por intervalos basados en el algoritmo de (Hansen, 1992) descrita en (Campos, 2004), la cual permite una identificación de óptimos globales basados en aritmética de intervalos, y además presenta pruebas de selección que permiten descartar regiones de un espacio de soluciones que no contienen óptimos globales (Moore, 1976) (Skelboe, 1974) (Ichida, 1979)

La aritmética de intervalos permite garantizar los resultados obtenidos, ya que considera el efecto de los errores de redondeo del procesador en operaciones aritméticas, que es el truncamiento que realiza cada computador dependiendo de su arquitectura, provocando muchas veces la pedida de precisión en los resultados obtenidos, cuando este no es capaz de representar un numero completo. El procesador redondea ya sea por izquierda o por derecha como también los trunca, ósea corta el numero hasta el decimal que pueda representar. Al utilizar números intervalares nos aseguramos que el resultado obtenido está en dicho intervalo, sin pérdidas en la precisión de este número.

Un número intervalo es un conjunto cerrado, acotado y conexo de números reales (Moore, 1992) por ejemplo [5,20], que contiene todos los números entre 5 y 20 incluyendo éstos. En la matemática de Intervalos se hace una generalización en donde los números reales son reemplazados por números intervalos, lo mismo pasa con la aritmética de intervalos.

También existe un primer acercamiento a la paralelización de la biblioteca de calculo científico basada en aritmética intervalar (Muñoz y Peña, 2007), la cual no considera la opción de comunicación entre tareas. A partir de esto se realizaron estudios y pruebas que permitieron definir que procesos pueden ser paralelizados o mejorar en las técnicas ya aplicadas de paralelización a esta biblioteca.

El rendimiento de algunos computadores determina el desempeño en la resolución de problemas, existen problemas tan complejos, que su solución podría tomar muchos años (Rodríguez, 2004). Para este tipo de problemas se puede usar la paralelización, dado que la computación paralela permite en muchos casos ahorrar recursos, mejorando la capacidad de computación con varios procesadores o computadores, haciendo que éstos trabajen de forma similar a un supercomputador, que en términos económicos es mucho más caro que una implementación paralela.

Para las cargas de los distintos procesadores, es donde aparece el concepto de Balanceo de carga, que es una técnica que acrecienta los recursos, explotando el paralelismo, y acortando el tiempo de respuesta mediante una distribución apropiada de la aplicación. (Bozyigita, 2000).Existen 2 formas de balance de carga; el estático y el dinámico. El estático se caracteriza por un conocimiento previo de la aplicación, las características del

sistema y las cargas de trabajo total que se desea paralelizar. El dinámico por su parte se puede adaptar a los cambios que se presenten en el sistema, acorde a un protocolo propuesto para detectar y enfrentar esos cambios. El balanceo de carga dinámico es requerido en una variedad de problemas de las ciencias de la computación, como sistemas operativos, problemas de optimización combinatoria y en general problemas de paralelismo en que no se conoce a priori la naturaleza de los trabajos a paralelizar.

Todo esto se propone con el objetivo de obtener un menor gasto del recurso tiempo que pueda tomar la librería de optimización global por intervalos en realizar un cálculo de forma secuencial, como paralelizada con la implementación anteriormente mencionada, buscando una paralización nueva de tareas trabajando en un sistema multi-computador con memoria distribuida, la cual nos permite hacer escalar algún sistema compuesto por varios sistemas más pequeños logrando rendimientos similares a una maquina de mayor desempeño. También se puede hacer énfasis a la distribución de tareas en tiempo de ejecución, que sería la distribución de cargas a través del balanceo de carga dinámico, el cual en muchos casos permite un mejor aprovechamiento del paralelismo utilizado, haciendo que pueda adaptarse más fácilmente a problemas cuyo comportamiento no se puede determinar fehacientemente al inicio de su resolución .

Dado los fundamentos expuestos en los párrafos anteriores se procede a definir los lineamientos generales de la presente habilitación profesional.

Objetivos

A continuación los objetivos generales y específicos a desarrollar en esta habilitación profesional:

1.1.1 Objetivo General

El objetivo principal de la habilitación profesional es el estudio , desarrollo e implementación de algoritmos de paralelización de la biblioteca de cálculo científico basada en aritmética de intervalos, basada en la biblioteca de optimización global por intervalos existente (Campos, 2004).

Se busca desarrollar nuevos métodos de paralelización de algoritmos para aplicarlos a la biblioteca de optimización global, utilizando la librería de programación paralela MPI y la comunicación por medio de paso de mensajes y los conceptos de balanceo de carga, logrando así mejorar la distribución del tiempo entre los procesos, el tiempo de respuesta y desempeño general de la biblioteca.

1.2.2 Objetivos Específicos

- Investigación de aritmética de intervalos.
- Estudio y comprensión de la librería de optimización disponible, así como de los algoritmos implementados.
- Estudio de estrategias de paralelización de algoritmos
- Análisis de herramientas orientadas a la paralelización de algoritmos.
- Examinar las metodologías de diseño de algoritmos paralelos.
- Investigar técnicas de balanceo dinámico.
- Implementación del Algoritmo
- Evaluación del Algoritmo
- Establecimiento de mejoras obtenidas con la implementaciones realizadas
- Conclusiones del trabajo realizado

1.3 Descripción de Capítulos

Capitulo 1 **"**Fundamentación de la Habilitación Profesional***":*** Se fundamentan las razones existentes para es el estudio, desarrollo e implementación de algoritmos de paralelización de la biblioteca de cálculo científico basada en aritmética de intervalos. Se establecen los objetivos de la habilitación profesional.

Capítulo 2 "Aritmética de Intervalos***":*** Definición de los números intervalos, definición de la aritmética de intervalos, operaciones y sus propiedades básicas, el error de redondeo, funciones y extensión por intervalos.

Capítulo 3 "Optimización Global***":*** Definición formal del problema de la optimización global y métodos de optimización global, se describe el algoritmo global por cotas y el algoritmo de optimización por intervalos y las mejoras introducidas a éste que lo hacen más eficiente.

Capítulo 4 "Paralelismo, estudio de estrategias y análisis de paralelización de algoritmos***":*** Se presentan los conceptos fundamentales de la programación paralela, junto con las distintas arquitecturas y el análisis de estrategias de paralelización de algoritmos.

Capítulo 5 "Descripción de C++ y MPI, Cluster**"**: Se define y esquematiza el sistema de paso de mensajes, se describe C++se analiza los clusters y se describe la librería de paralelización MPI

Capítulo 6 "Proceso de Implementación del Cluster de alto rendimiento en la facultad de ciencias empresariales (FACE) de la Universidad del Bio-Bío**"**. Se explica el proceso de reconfiguración del cluster, características y la importancia de este para la habilitación profesional

Capítulo 7. "Análisis de Estrategias Implementadas para el Algoritmo de Optimización Global**"** En el presente capítulo se analizarán las distintas estrategias de paralización para el algoritmo de optimización global por intervalos, luego se profundizará en las nuevas estrategias de paralelización de intervalos desarrolladas en la presente habilitación profesional

Capítulo 8. "Resultados y Conclusiones Obtenidas al Proceso de Depuración y Pruebas de los Algoritmos en Problemas Clásicos**".** En este capítulo se muestran las distintas pruebas aplicadas a los algoritmos basados en el marco teórico de los problemas clásicos de prueba.

CAPÍTULO 2. ARITMÉTICA DE INTERVALOS

Los primeros estudios realizados sobre la aritmética de intervalos fueron en Norteamérica en la década de 1960. El exponente más destacado es el físico y doctor en matemáticas Ramon E. Moore, quien desarrolló los principios de la teoría en su tesis de doctorado "Estimación de errores en el análisis numérico" (Moore, 1962). En sus estudios, Moore definió los números intervalos como un nuevo tipo de números representados por un par de números reales, los extremos inferior o izquierdo y superior o derecho, respectivamente:

$$\mathbf{I} = [a, b],\ a, b \in \mathbf{P}$$

Es decir, se tiene un intervalo cerrado de números reales, I = [a, b], que consiste en el conjunto $\{x: a \leq x \leq b\}$, o sea, el conjunto de todos los números reales entre a y b, incluidos a y b.

Se pueden mencionar algunas definiciones básicas como:

Un intervalo $X = [a,b]$ es *positivo* (o no negativo) si $a \geq 0$,*estrictamente positivo* si $a > 0$, *negativo* (o no positivo) si $b \leq 0$, y *estrictamente negativo* si $b < 0$.

Dos intervalos $[a,b]$ y $[c,d]$ son *iguales* si y solo si $a = c$ y $b = d$.

El ancho ("width") de un intervalo $[a,b]$ es un número real no negativo definido como:

$$w([a,b]) = b - a$$

Se debe notar que un intervalo degenerado tiene largo igual a cero.

El *centro* de un intervalo es definido como:

$$c([a,b]) = \frac{a+b}{2}$$

A partir de las definiciones anteriores, se puede deducir que si se conocen el ancho y el centro de un intervalo, se puede definir completamente dicho intervalo, de la siguiente manera (Campos, 2004):

$$X = [a,b] \begin{cases} a = c(X) - \frac{w(X)}{2} \\ b = c(X) + \frac{w(X)}{2} \end{cases}$$

Moore definió las propiedades y operaciones de la aritmética de intervalos, mediante las cuales podemos obtener las cotas inferior y superior del intervalo, considerando el error de redondeo presente en dicho cálculo.

Se pueden clasificar los intervalos según sus características topológicas (intervalos abiertos, cerrados y semi-abiertos) o según su características métricas (su longitud: nula, finita no nula, o infinita).

La regla del corchete invertido resulta más intuitiva, si se intuye que el corchete que esta al extremo del intervalo es una mano que tira hacia fuera o empuja hacia dentro respectivamente, como puede observarse en la figura 1.1.

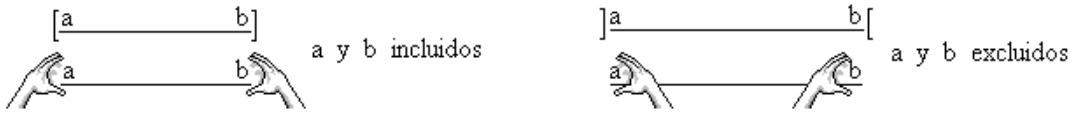


Figura 1.1 Regla del corchete invertido (Romero, 2003).

2.1 Error de Redondeo

Al momento de realizar una operación computacional utilizando números reales, el procesador realiza el cálculo utilizando aritmética de punto flotante de precisión fija, reduciendo de manera abrupta los números decimales, esto se debe a que el procesador no es capaz de representar todos los decimales del número resultado. Como solución el procesador redondea o trunca a un número representable, provocando muchas veces la perdida de precisión e introduciendo así un error de redondeo.

Considérese el siguiente ejemplo para ilustrar el error: El valor de π se conoce como:

$$\pi = 3,1415926\ldots$$

Considérese 7 cifras significativas luego del punto decimal, si truncamos □ en el 4 decimal, se obtiene un error de:

$$\pi = 3,1415926\tilde{5}3,1415\square$$

$$\Delta = 0,0000926$$

En cambio en el caso de la aproximación, se tiene que el decimal siguiente al corte es mayor que 5, entonces se aproxima el último decimal, quedando el número como 3, 1416, en este caso el error seria:

$$\Delta \pi = 3,1415926\tilde{5}3,1416\square$$

$$\Delta = 0,0000074$$

Lo que es menor que la diferencia anterior, pero el error sigue existiendo.

Muchas veces se tiende a subestimar el error de redondeo, ya que actualmente los procesadores utilizan cálculos de precisión extendida para solucionar este problema, la cual logra disminuir el error. Por ejemplo para calcular un resultado específico, se utiliza precisión simple y doble. Si los resultados coinciden en un número determinado de dígitos, se asume que el resultado específico es correcto (Hansen, 1992). Sin embargo, en ocasiones no basta con utilizar cálculos de precisión extendida, debido que la aritmética de punto flotante no es suficiente para representar el resultado exacto.

Por ejemplo considérese la función: (Leclerc, 1992)

$$f(x,y) = 333{,}75y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + 5{,}5y^8 + x/2y$$

Esta función fue evaluada en $x = 77617$ e $y = 33096$, utilizando FORTRAN en un computador SPARCStation SLC utilizando precisiones simple, doble y extendida (single, double and extended) que corresponden a 6, 14 y 35 dígitos decimales respectivamente.

Mediante multiplicaciones se obtuvieron los siguientes resultados:

$f = 6.33825 * 10^{29}$ (precisión simple)

$f = 1.1726039400532$ (precisión doble)

$f = 1.172603940053178631858834904520183 8$ (precisión extendida)

A simple vista se aprecia la gran diferencia entre el resultado evaluado en precisión simple, doble o extendida, esto puede llevar a pensar que el resultado está errado, pero al comparar la gran similitud entre los resultados de precisión doble y extendida se observa que concuerdan 13 dígitos entre ellos, lo que hace pensar que el resultado es correcto. Sin embargo si se evalúa la función utilizando aritmética de intervalos de precisión variable, se concluye que los 3 valores están errados. Como se observa en los siguientes resultados:

$$I = [\alpha, \beta]$$

$$\alpha = -0.827396059946821368141165095479816292005$$

$$\beta = -0.827396059946821368141165095479816291986$$

El resultado exacto se contiene en el intervalo anterior, siendo este ejemplo muy ilustrativo para darnos cuenta que el error de redondeo puede afectar significativamente los cálculos de punto flotante con precisión fija, aún cuando utilicemos precisión extendida y dos resultados puedan coincidir en varios dígitos.

2.2 Aritmética de Intervalos

La Aritmética de Intervalos fue definida por Ramón E. Moore (Moore, 1966) y constituye una técnica con precisión infinita de resolución de problemas con restricciones, basada en dos operaciones fundamentales: aproximación y reducción. Su utilización se refleja recientemente en temas diversos como la resolución de ecuaciones diferenciales ordinarias, sistemas lineales, verificación y optimización global (Hansen, 1992) (Leclerc, 1992). Para cualquier operación binaria θ, donde θpuede ser +,-, x o /, y X e Y son dos intervalos cualesquiera, se desea que: (Van Iwaarden, 1996)

$$X \theta Y = \begin{bmatrix} \min_{x \in X, y \in Y} x \theta y & \max_{x \in X, y \in Y} x \theta y \end{bmatrix}$$

El intervalo resultante de X θ Y debe contener cada número que puede resultar de x θ y para cada x∈ X y cada y∈ Y. Considérese el siguiente ejemplo:

Si a ≤ x ≤b y c ≤ y ≤d (X= [a, b] e Y= [c, d]), entonces a + c ≤ x + y ≤ b + d.

De la misma forma las cuatro operaciones básicas por intervalos son definidas como muestra la siguiente tabla:

Suma	X + Y = [a + c, b + d]
Resta	X - Y = [a - d, b – c]
Multiplicación	X * Y = [min (ac, ad, bc, bd), max (ac, ad, bc, bd)]
División	$\frac{X}{Y} = [a,b] * \left[\frac{1}{d},\frac{1}{c}\right] si \quad 0 \notin [c,d]$

Tabla 1. Operaciones básicas por intervalos

Si se utilizan intervalos degenerados (de ancho cero), la aritmética de intervalos se reduce a aritmética ordinaria, por ende se concluye que la aritmética de intervalos resulta ser una extensión de la aritmética de números reales.

2.2.1 Propiedades Básicas

Las operaciones básicas de la teoría de conjuntos para números reales son también aplicables a los números intervalos, por ende las operaciones de adición y de multiplicación de intervalos son también asociativas y conmutativas, al igual que los números reales 0 y 1 son identidades para la adición y la multiplicación respectivamente. Otra propiedad importante de los intervalos es la inclusión monótona. Descritas en el anexo K.

2.3 Funciones por Intervalos

La extensión natural es una de las formas existentes de extensión, que consiste en reemplazar las variables reales por variables intervalos de forma directa. Considérese la función real $f(x_1,x_2,\ldots,x_n)$ la cual puede generalizarse a una función por intervalos $F(X_1,X_2,\ldots,Xn)$ a través de la extensión natural si $\exists\ B \subset IP^n$, donde B es llamado caja n-

dimensional y IP^n corresponde al conjunto de los vectores intervalos, siendo éstos el producto cartesiano de n-intervalos. Las operaciones básicas vistas pueden extenderse a los vectores intervalos definidos en IP^n (Hansen, 1992)(Jaulin et al., 2001).

Considérese la extensión de $f(x_1,x_2,\ldots,xn)$ como:

$$F(X_1, X_2,\ldots, Xn) = \{f(x_1, x_2,\ldots, xn) \mid \forall xi \in Xi, i=1,\ldots,n\}$$

Si se reemplazan las variables reales por variables intervalos y la aritmética real por la intervalar se puede calcular el valor de una función por intervalos, obteniendo así las condiciones necesarias para poder realizar la evaluación de la función.

2.3.1 Problema de dependencia

Cuando una variable dada tiene más de una ocurrencia en un cálculo con intervalos, es tratada como una variable diferente en cada ocurrencia, lo que produce un grave inconveniente conocido como problema de dependencia (Hansen, 1992).

Considérese el intervalo X= [a, b] en donde restamos el intervalo a sí mismo, X – X, dando como resultado el intervalo [a – b, b - a], o [a – b, - (a - b)], cuando lo que en realidad deberíamos obtener es [0, 0]. Sin embargo esto no ocurre (a menos que b = a). El resultado es $\{x – y: x \in X, y \in Y\}$ en lugar de $\{x – x: x \in X\}$. Obsérvese el siguiente ejemplo:

Sea X= [-1, 1] si calculamos X–X, obtenemos: X– X= [-1, 1] - [-1, 1] = [-2, 2].

Así, X X es lo mismo que X – Y, con Y igual, pero independiente de X.

Por otro lado, considérese la función $f(x)=x^3 + x^2$, la evaluación sobre el intervalo X=[-1,1] sería $F([-1, 1]) = [-1, 1]^3 + [-1, 1]^2 = [0,4]$

El intervalo solución obtenido es [0,4], el cual no corresponde a la solución más acotada posible que en este caso correspondería a [0, 2], pero si es una solución válida porque incluye todos los máximos y mínimos posibles para $F(X)=X^3 + X^2$ evaluados en X=[-1, 1], como se observa en la Figura 2.2.

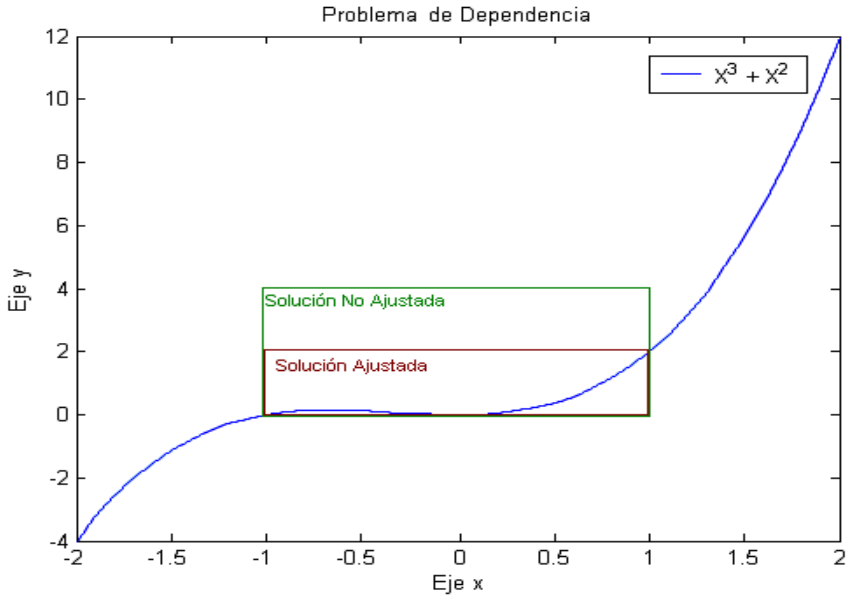


Figura 2.2 Problema de dependencia; solución acotada y solución no acotada.

Esto causa un "ensanchamiento" de los intervalos calculados y hace más difícil obtener un resultado ajustado, porque la variable tiene más de una ocurrencia en un cálculo con intervalos, y es tratada como una variable diferente en cada ocurrencia. Se debe tomar las precauciones adecuadas para reducir su efecto, ya que se busca obtener una evaluación más ajustada de una función, es decir, una forma de evaluar la función que evite el problema de dependencia.

2.4 Extensión de Funciones

Se puede extender una función real a una función por intervalos de varias formas, diferenciándose entre ellas básicamente por la mayor o menor aproximación a una extensión óptima. Entre las principales formas de extensión se encuentran: la *aproximación algebraica*, que consiste en utilizar factorización con el fin de conseguir la mínima repetición de variables, la *aproximación numérica*, la cual se vale de la subdivisión de intervalos más pequeños, su resolución y posterior unión para obtener un resultado parcial, y la extensión natural, que consiste en la forma más simple de evaluar una función por intervalos, donde se reemplazan las variables reales por variables de intervalo en f. Considérese el siguiente ejemplo:

Una función real *f(x)*= x^2–x y su representación en intervalos equivalente F(X)= X^2–X y se evalúa en el intervalo X= [0, 1] por extensión natural, se obtiene:

F[0, 1]= $(0, 1)^2$–(0, 1)= (0, 1)–(0, 1)= [-1, 1]

Esto quiere decir que al evaluar la función F(X)= X^2–X en el intervalo X= [0, 1] se obtiene como solución el intervalo cerrado Y= [-1, 1], resultado que no es ajustado. Esto sucede debido al problema de dependencia, lo que produce un ensanchamiento del intervalo calculado y hace más difícil obtener un resultado ajustado. El intervalo solución obtenido al evaluar la función F(X) por extensión natural, si bien entrega un intervalo mayor que el obtenido en la solución ajustada, sigue siendo válido y útil, porque comprende totalmente a ella. Obsérvese la figura 2.4

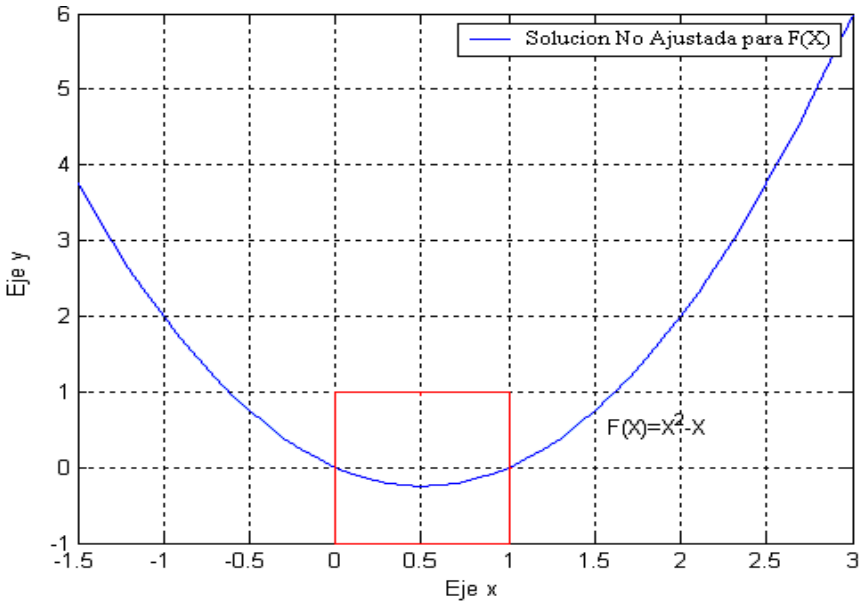


Figura 2.4 Evaluación de F(X) = X^2 – X en el intervalo [0, 1] por extensión natural, siendo esta solución no ajustada.

La solución ajustada para F(0, 1) es [-0.25, 0], porque este intervalo solución es el más acotado que se puede obtener y sigue conteniendo todos los mínimos y máximos de la

curva en el intervalo X= [0, 1] por ende es la solución más ajustada a nuestra función $F(X)= X^2-X$.

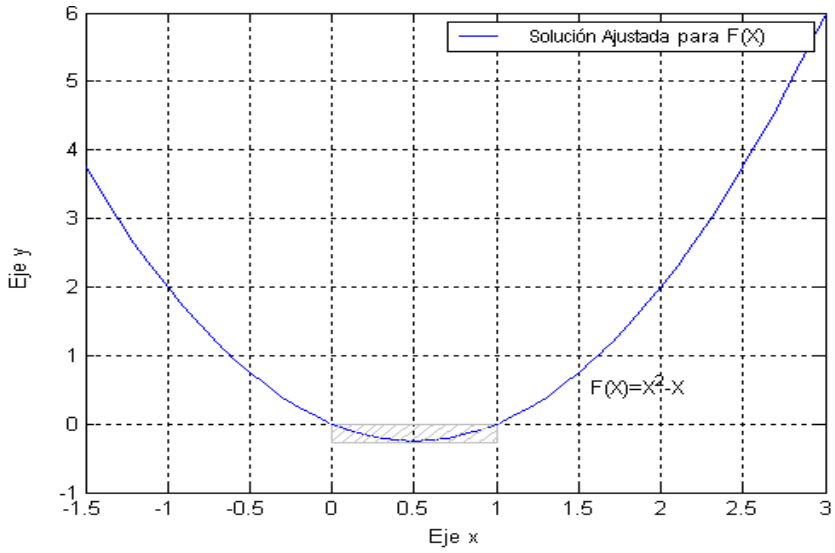


Figura 2.5 Solución más ajustada a la evaluación de $F(X) = X^2 – X$ en el intervalo [0, 1].

CAPÍTULO 3. OPTIMIZACIÓN GLOBAL

La optimización global trata de encontrar la mejor solución a un problema de optimización, realizando búsquedas completas sobre un conjunto de soluciones factibles, esto quiere decir que abarca todos los puntos existentes en la región de búsqueda dada.

Muchos problemas de matemática e ingeniería requieren una búsqueda completa. Un ejemplo de ello es la conjetura de Kepler, consistente en una conjetura matemática acerca de la disposición de esferas idénticas en el espacio euclidiano de tres dimensiones (Hales, 2005), o el problema del diseño de un brazo robótico (Lee y Mavroidis, 2002), donde los métodos locales no entregan información útil, ya que generalmente quedan atascados en minimizadores locales, o en el sistema de análisis de control y diseño, donde si no se aplica una búsqueda detallada se puede caer en el error de subestimar los verdaderos riesgos (Balakrishnan y Boyd, 1992).

3.1 Métodos de Optimización Global

Se conoce como proceso de optimización global a la técnica que encuentra el (o los) óptimo(s) global(es) de cualquier función, asumiendo cotas iniciales de los valores de las variables, aceptando también como cotas que las variables sean infinitas. Esto es, los valores aceptables para las variables pueden ser $(-\infty, \infty)$, $[\kappa, \infty)$, o $(-\infty, \kappa]$ donde k es cualquier número real.

Un optimizador global ideal es una caja negra, a la cual se le entrega la descripción de la función, sus restricciones, y las cotas de las variables (Van Iwaarden, 1996). La salida de esta caja negra es una combinación de las siguientes declaraciones:

- Los valores aproximados en los cuales la función alcanza el Óptimo, con cotas de error verificadas matemáticamente.
- Valores que son una aproximación al Óptimo global, con error máximo verificado matemáticamente.
- La función no está acotada.

Los métodos de optimización global pueden clasificarse en métodos estocásticos, que se

caracterizan por evaluar las funciones en puntos elegidos aleatoriamente, y métodos determinísticos, que desechan cualquier elemento aleatorio en el cálculo del Óptimo.

Los algoritmos pueden ser catalogados como confiables o no confiables. Cada categoría posee ventajas y desventajas, por lo que el algoritmo que se utilizará dependerá del problema en particular.

3.2 Definición Formal de Optimización Global

Considérese una función *f*(x) cualquiera, continua y real, como función objeto o la función a minimizar, el problema de optimización global es definido como:

$$f^{o} = \min_{x \in X} f(x)$$

Tal que f: $P^n \rightarrow P^1$ siendo f la función continua real a minimizar, el espacio o dominio donde la función será definida se denota como $X \subset P^n$, donde X es la región factible. Esta definición incluye tanto la búsqueda de un mínimo global como de un máximo global, ya que es equivalente minimizar *f*(x) y maximizar –*f*(x).

El conjunto de todos los puntos para los cuales la función objetivo posee un mínimo global se denota como X^o. En este conjunto se encuentran todos los puntos x^o tales que $f(x^o) = f^o$ y se llama comúnmente conjunto de minimizadores globales.

La cantidad de mínimos y máximos globales en un función puede variar, quedando claro que es perfectamente posible que exista más de un mínimo o máximo global, ya que en más de un punto la función puede alcanzar su valor mínimo (o máximo), obsérvese la figura 2.5.

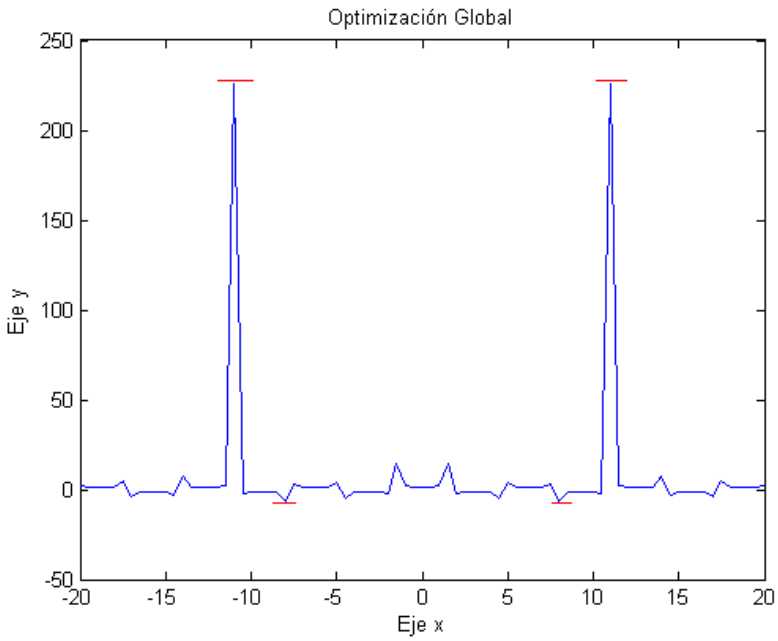


Figura 3.1 Gráfico de (x)= sec(x), donde se aprecian Multiples máximos y mínimos globales.

3.3 El Problema de la Optimización Global

El problema de la optimización global es complejo, de hecho una gran cantidad de autores están convencidos que no tiene solución en un número finito de pasos.

Se tiene una función continua en donde existe un punto x y una región factible de búsqueda, así para cualquier punto x no se puede garantizar que este no es el mínimo global sin evaluar antes la función en al menos un punto de cada región factible de x. Como la región factible puede ser elegida arbitrariamente pequeña, se deduce que cualquier método diseñado para resolver el problema de optimización global requiere un número de pasos ilimitado.

Este argumento es válido si se quiere garantizar que un punto exacto es minimizador global, si el minimizador global fuese un número irracional, es obviamente imposible, en un número finito de pasos, representar numéricamente esa solución. Lo que es posible de representar, en un tiempo finito, es que *f(x)* está a lo más a una distancia ε del mínimo global (Moore, 1991).

Un ε-minimizador global es un punto x donde $f(x) < f^{o}+\varepsilon$.

El problema de optimización global tiene solución en un tiempo finito, entendiendo que la solución es acercarse arbitrariamente al Óptimo, lo cual no es fácil de resolver de ninguna forma.

Está demostrado que algunos problemas de optimización son del tipo NP-completos (Pardalos y Vavasis, 1991) y que actualmente, todos los algoritmos confiables y aplicables universalmente para resolver el problema de optimización global poseen tiempos computaciones que crecen exponencialmente a medida que aumenta el número de variables.

3.4 Métodos Estocásticos o Probabilísticos

En la Optimización Global los métodos estocásticos o probabilísticos son procesos infinitos, esto quiere decir que la probabilidad de haber visitado al Óptimo global se aproxima a 1 a medida que el número de pasos tiende a infinito. Dicho lo anterior no se puede garantizar, en un número finito de pasos, haber localizado el Óptimo global.

La Fortaleza de estos métodos es la eficiencia, ya que, en general, se puede encontrar rápidamente algún Óptimo local. Por ahora, los problemas de gran escala pueden ser "resueltos" de mejor forma estocásticamente (Leclerc, 1992), (Luenberg, 1989), (Iwaarden, 1996).Se revisa una breve descripción de los principales métodos estocásticos o probabilísticos en el anexo R.

3.5 Métodos Determinísticos

Entre los algoritmos determinísticos y los estocásticos existe una gran diferencia, porque los algoritmos estocásticos no pueden garantizar en un 100% haber encontrado el óptimo global, ya que dicha garantía requeriría que el algoritmo se ejecutara por tiempo infinito.

Los algoritmos determinísticos garantizan que todos los óptimos globales están contenidos en las regiones que retornan al finalizar su ejecución.

Los algoritmos determinísticos siempre se ejecutan en tiempo finito (aunque puede ser una gran cantidad de tiempo), y retornan un conjunto de regiones en las que está garantizado que se encuentran todos los óptimos globales. Además, los algoritmos estocásticos

garantizan encontrar (en tiempo infinito) un óptimo global, cuando pueden existir varios óptimos globales alternativos.

3.5.1 Métodos de Cotas

Los métodos de cotas calculan cotas de la función sobre conjuntos compactos, si son implementados de forma apropiada y toman en consideración el error de redondeo, pueden producir rigurosas soluciones de optimización global, intentando producir al menos una cota inferior para el rango de valores de f sobre un cierto conjunto compacto, si la cota inferior de la función es asintóticamente precisa, como en la aritmética de intervalos, entonces el método puede obtener un □ -mínimo global.

La mayoría de las técnicas existentes para construir cotas inferiores de funciones operan sobre conjuntos compactos convexos, los cuales son rectángulos de n-dimensiones (hiper-rectángulos). Una de las técnicas más eficaces es la aproximación de intervalos.

La aproximación de intervalos consiste en el cálculo de cotas inferiores y superiores usando los principios de la aritmética de intervalos. Por ejemplo se puede escribir una extensión de intervalos natural, F, para cualquier función programable, f, en donde tal función de intervalos opera sobre hiper-rectángulos y entrega resultados en intervalos, los cuales acotan el rango de la función sobre el conjunto de entrada.

La ventaja al trabajar con intervalos es que se toma en cuenta el error de redondeo.

3.6 Algoritmo de Optimización Global por Cotas

Se tiene una función (F_L) que permite obtener la cota inferior en el rango de la función para una región sobre una función, es decir, si evaluamos la función sobre todo el intervalo, esta función nos entrega la cota inferior. Fácilmente se puede implementar un algoritmo de optimización global. De forma sencilla:

I. Se particional el conjunto inicial X en subconjuntos S_i^X,

II. Luego se calcula una cota inferior del valor de la función sobre cada subconjunto $F_L(S_i^X)$ y se mantiene una cota superior del valor mínimo global calculado hasta el momento, U_{f^o}.

III. Cualquier subconjunto de S_i^X donde $F_L(S_i^X) > U_{f^o}$ es rechazado, ya que no contiene un mínimo global (el valor mínimo de f sobre el subconjunto es mayor que el mínimo global).

Este proceso de particionar, acotar y posiblemente rechazar continúa sobre los subconjuntos generados sucesivamente hasta que se cumple algún criterio de detención. La unión de los conjuntos no rechazados contendrá el conjunto de todos los minimizadores globales de *f*.

La efectividad de este simple algoritmo se relaciona con la habilidad de la función que calcula las cotas inferiores de entregar cotas inferiores "ajustadas" para el mínimo global de la función sobre un conjunto dado. Si el valor de la cota inferior calculada es muy lejano al verdadero valor del mínimo, la capacidad de rechazar conjuntos se ve obstruida, en cambio, si la función de acotamiento entrega la cota inferior más "ajustada" para el mínimo global (hasta lo que permite la precisión de la máquina), entonces el problema de optimización tendrá una ε-solución.

Así, el algoritmo general que implementan todos los métodos por cotas es el siguiente:

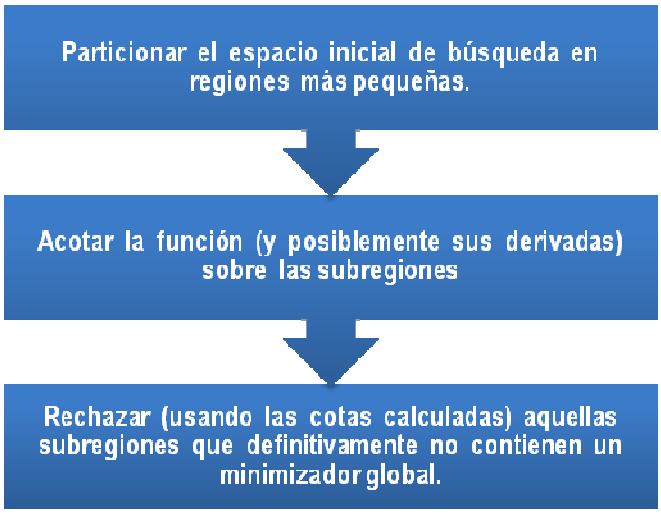


Figura 3.2 Pasos que utiliza el algoritmo general que implementan todos los métodos de cotas.

La unión de las regiones no rechazadas contendrá todos los minimizadores globales.

Usualmente se elige particionar el espacio de búsqueda en hiper-rectángulos. Si el espacio factible de búsqueda inicial no es un hiper-rectángulo, se usa un hiper-rectángulo con superposición como espacio inicial de búsqueda. Las ecuaciones que caracterizan el

verdadero espacio factible funciones de restricción) son usadas adicionalmente en la fase de rechazo para eliminar aquellas subregiones que, estando en el hiper-rectángulo inicial, no pertenecen a la región factible.

En cuanto al paso de rechazo, puede tomar las siguientes formas (Leclerc, 1992):

1. Rechazo de las subregiones cuyas cotas inferiores calculadas en el paso dos del algoritmo sean mayores que la cota superior del mínimo global conocido hasta ese momento.

2. Rechazo de las subregiones que no estén en el espacio factible (es decir, que no cumplen las restricciones)

3. Si la función objetivo es diferenciable:

- Rechazo de las subregiones fuera del borde del espacio inicial de búsqueda donde $0 \notin \nabla f$ (∇f es el gradiente de la función objetivo).

- Rechazo de las subregiones donde la función objetivo no es convexa en ningún lugar de la subregión.

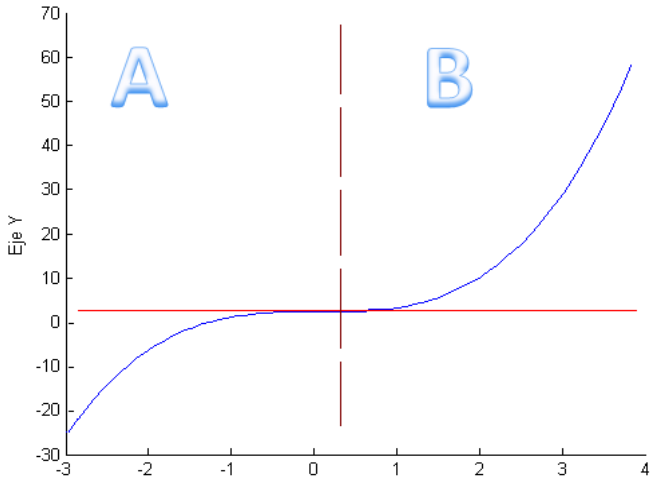


Figura 3.3 Gráfico f(x)= x^2+ 2, donde la cota superior de la región A es menor a la cota inferior de la región B.

Como se observa en la figura 3.3 la cota inferior de la región B es mayor a la cota superior de la región A, y si el mínimo global conocido hasta ese momento es la cota superior de A, se puede afirmar, según el paso 1 de las reglas de rechazo, el rechazo de la región B.

3.7 Optimización Global por Intervalos

Moore (Moore, 1966) fue posiblemente la primera persona que usó la aritmética de intervalos como herramienta para calcular el rango de una función sobre un hiper-rectángulo, por lo que posee el crédito de haber desarrollado e implementado el primer algoritmo de optimización global por intervalos.

Skelboe (Skelboe, 1974) combinó posteriormente la metodología por intervalos de "determinación del rango" con el principio de branch and bound (Leclerc, 1992). Este principio posee dos características:

- No se realiza una búsqueda uniforme en el espacio factible. Por el contrario, se realiza la búsqueda primero y con mayor profundidad en ciertas subregiones (o branches), que en otras.
- Se debe calcular una cota inferior de la subregión.

El algoritmo de Moore-Skelboe funciona de la siguiente manera:

- Se particional el conjunto inicial X en subconjuntos S_i^X,
- luego se busca el mínimo global f^o prefiriendo aquellas subregiones S_i^X para las cuales la cota inferior del valor de la función $F_L(S_i^X)$ es menor. (Se espera que las cajas seleccionadas contengan mejor probabilidad de contener un minimizador global, x^o).
- Se hace la selección de la sub-región S_i^X donde $F_L(S_i^X)$ es menor, se puede realizar rápidamente usando y manteniendo una cola de prioridades (o lista). En una cola de espera con m hiper-rectángulos, la sub-región con la menor $F_L(S_i^X)$, digamos $S^X m$, estará al frente.
- El valor $FL\ (S^X m)$ es una cota inferior del valor mínimo global calculado hasta ese momento, a la cual llamaremos L_{f^o}

El algoritmo de Moore-Skelboe no emplea ninguna prueba para eliminar aquellas sub-regiones que definitivamente no contienen ningún mínimo global. Tales pruebas son llamadas pruebas de rechazo. El algoritmo simplemente continúa hasta que se cumple algún criterio de detención.

3.8 Algoritmo de Optimización Global por Intervalos

El algoritmo global por intervalos resuelve problemas de optimización global de la forma $\min_{x \in B} f(x)$ sujeto a un conjunto de restricciones, pi $\lneq$ 0, i = 1,…, k, que permiten definir el espacio o región factible, es decir, la región donde es válido buscar Óptimos.

B es cualquier hiper-rectángulo inicial dado, o caja, definido como el siguiente intervalo n-dimensional (vector de intervalos):

$$B = \{x : a_i \leq x_i \leq b_i\} \text{ Para todo i=1, 2,…,n}$$

$$= \begin{bmatrix} [a_1, b_1] \\ [a_2, b_2] \\ \vdots \\ [a_n, b_n] \end{bmatrix}$$

Se busca una cota arbitrariamente ajustada, en la caja B, para el conjunto X^o de todos los minimizadores globales x^o que están en la región factible, se desea también una cota arbitrariamente ajustada del valor del mínimo global, f^o, de la función objetivo dada, *f*.

$$f(x^o) = f^o \text{ y } f^o \lneq f(x) \text{ para todo } x \in B$$

El algoritmo básico encuentra una lista de pequeñas cajas cuya unión contiene el conjunto X^o de todos los minimizadores globales x^o. El algoritmo termina cuando el ancho máximo de todas las cajas negras en la lista es menor que una tolerancia prescrita, ε_x. El algoritmo también entrega cotas superiores e inferiores para el valor mínimo $f^o = f(x^o)$.

Este algoritmo procede rechazando las partes de la caja inicial B que no pueden contener un minimizador global, dejando una lista de sub-cajas (de B) cuya unión contiene el conjunto de todos los minimizadores globales de *f*(x).

Se describen a continuación los métodos utilizados para rechazar aquellas partes de la caja y para encontrar las cotas superiores e inferiores del valor de *f*(x) para puntos factibles *x*.

3.8.1 Prueba de Rechazo: Prueba de Factibilidad

Sea X una sub-caja de B y un punto en B es representado como una caja degenerada.

Se pueden evaluar las funciones de restricción $p_1(X),\ldots, p_k(X)$ utilizando la aritmética de intervalos y afirmar que X es *certeramente factible* si $p_i(X) \leq 0 \ \forall i = 1,\ldots, k.$

Si X es certeramente factible, entonces cada punto $x \in X$ es factible. De lo anterior se desprende que:

- X es certeramente no factible (no contiene puntos factibles)

$\exists i$ tal que, i=1,…, k, pi(X) > 0.

En tal caso, X puede ser rechazada, no tomándosele en cuenta en lo sucesivo.

3.8.2 Prueba de Rechazo: Prueba del Punto Medio

Sea mX el punto medio (o cualquier otro punto) de una sub-caja X de B y mX es certeramente factible, entonces la función objetivo es evaluada en mX para obtener un intervalo:

$$f(mx) = \lfloor L_{f_{mx}}, U_{f_{mx}} \rfloor$$

Si $f^{\circ} \leq U_{f_{mx}}$, esto significa que, $U_{f_{mx}}$ es una cota superior del valor mínimo de $f(x)$ sobre la región factible. Si f es evaluado sobre otra sub-caja Y de B, teniendo como resultado $f(Y) = \lfloor L_{fr}, U_{fr} \rfloor$, entonces se puede realizar la siguiente prueba. Sea $U_{f^{\circ}}$ el menor $U_{f_{mx}}$ encontrado hasta el momento:

Si $L_{fr} > U_{f^{\circ}}$ entonces Y no puede contener un minimizador global factible en B, por lo tanto Y puede ser rechazada, no tomándose en cuenta en lo sucesivo.

Una cola de espera consiste en un contenedor en donde se alojan las cajas a las cuales se les aplicará las pruebas de rechazo, estas se aplican al momento de sacar una caja de la cola. Los elementos son añadidos al final de la cola y son removidos del inicio de la misma.

Aplicando las pruebas de rechazo la caja removida de la cola es biseccionada a lo largo de la dirección (eje) de ancho máximo. Luego se aplican las pruebas a cada mitad. Si con las pruebas no se puede rechazar una caja, entonces es añadida al final de la cola. Como resultado, la caja más ancha que queda en la cola es siempre la primera. Si el ancho de ésta es menor que ε_x entonces también lo son todo el resto. Véase figura 3.4.

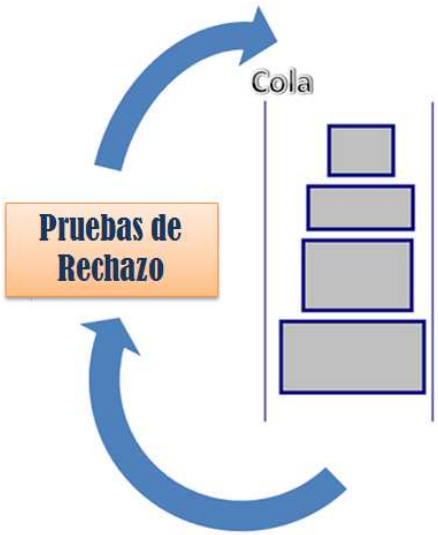


Figura 3.4 Se selecciona una caja del final de la cola, se le aplican las pruebas de rechazo, si se rechaza es removida de la cola, si no es rechazada se añade al inicio de la cola.

Con las pruebas vistas se puede formular un algoritmo de optimización global con restricciones. Este es válido tanto para funciones diferenciables como si no lo son. Se debe destacar, como una de las propiedades más importantes de este algoritmo, que en cualquier instante del proceso, las cajas que se encuentran en la cola contienen todos los puntos mínimos globales factibles.

3.8.3 El Algoritmo

1. Ingresar la caja inicial, B, y la tolerancia de ancho de caja, ε_x.
2. Añadir (B, Lf_B) a la cola y actualizar U_{f°.
3. Ciclo:
   a. Remover la primera caja, *X*, de la cola.

b. Biseccionar $X = X_1 \cup X_2$ a través de la dirección (eje) de ancho máximo.

c. Rechazar X_1 o añadir $(X_1,\ Lf_{X1})$ al final de la cola.

Si X_1 es añadido a la cola, entonces actualizar U_{f^o} si mX_1 es factible.

d. Rechazar X_2 o añadir $(X_2,\ Lf_{X2})$ al final de la cola.

Si X_2 es añadido a la cola, entonces actualizar U_{f^o} si mX_2 es factible.

e. Si la primera caja de la cola tiene ancho < ε_x, entonces ir al paso 4. En caso contrario, ir al comienzo del paso 3.

4. Fin del programa principal

Al terminar el algoritmo, $L_{f^o} \leq f^o \leq U_{f^o}$.La unión de las cajas en la lista ciertamente contendrá todos los minimizadores globales factibles de $f \in$ B.

Se actualiza U_{f^o}, reemplazándolo por cualquier U_{fmx} menor encontrado.

Para mejor entendimiento considérese una función cualquiera, se aplicará el algoritmo de optimización global a ésta de la siguiente forma:

Paso 0: Se selecciona el espacio completo (toda la caja) y se ingresa a la cola. Véase figura 3.5.

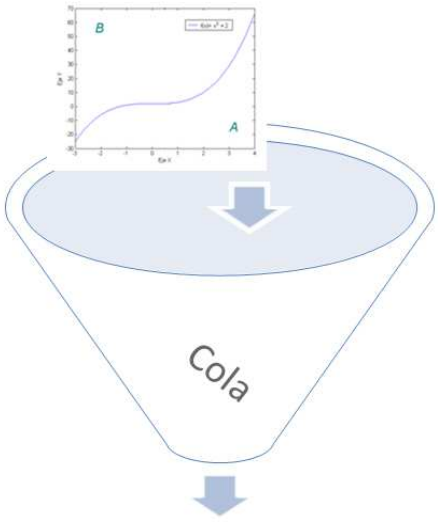


Figura 3.5 Paso 0, ingreso de toda la caja a la cola de espera.

Paso 1: Se saca la caja de la cola y se particiona en dos partes (véase figura 3.6), se ingresa el ancho de la caja, se calculan las cotas y se aplican las pruebas de rechazo, dependiendo si son rechazadas o no las partes, pueden ingresar a la cola las dos partes, una o ninguna.

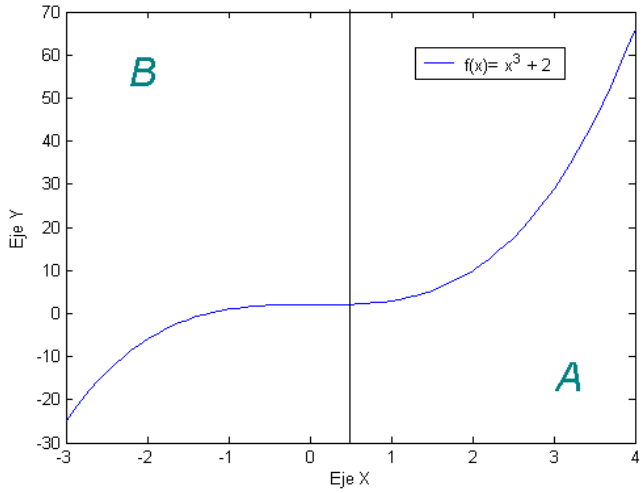


Figura 3.6 Particionamiento de una función cualquiera en 2 áreas, A y B espacios factibles respectivamente.

Paso 2: Ya definida una tolerancia dada, ε_x la cual utilizaremos como condición de término del algoritmo. Se remueve la primera caja de la cola, en este caso A, y esta es biseccionada (o dividida en 2 partes) que serán A_1 y A_2 respectivamente. Véase figura 3.7

Dadas las pruebas de rechazo, A_2 es desechado ya que la cota mayor calculada de A_1 es menor que la cota inferior calculada de A_2. A_1 es añadido al final de la Cola, y A_2 es desechado. Actualizamos U_{f^o}, obsérvese la figura 3.8 secuencia "B".

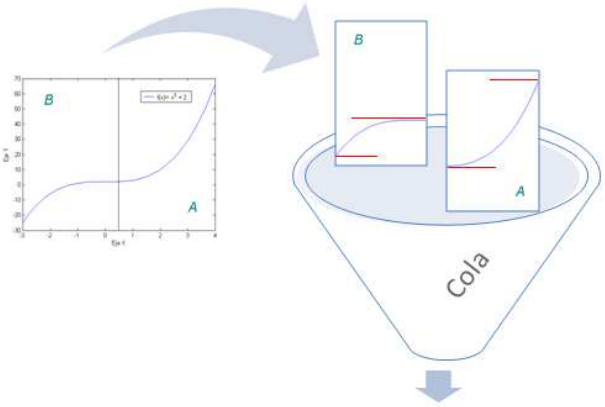


Figura 3.7 El espacio factible se divide en 2 partes, A y B ingresándose a la Cola en orden.

Parte 3: Se da comienzo el ciclo de particionamiento y acotación. Se remueve la caja del extremo de la cola, se bisecciona. Luego se extrae la siguiente partición del tope de la cola, si la caja B tiene ancho < ε_x, entonces el ciclo termina, sino se vuelve a repetir. En este caso la caja B tiene ancho mayor que ε_x, entonces se aplica nuevamente el particionamiento, dividiendo B en B_1 y B_2, para luego eliminar B_2 por la condición de que su cota inferior es mayor que la superior de B_1. Se ingresa B_1 al final de la cola y se desecha B_2. Actualizamos U_{f°, obsérvese la figura 3.8 secuencia "C".

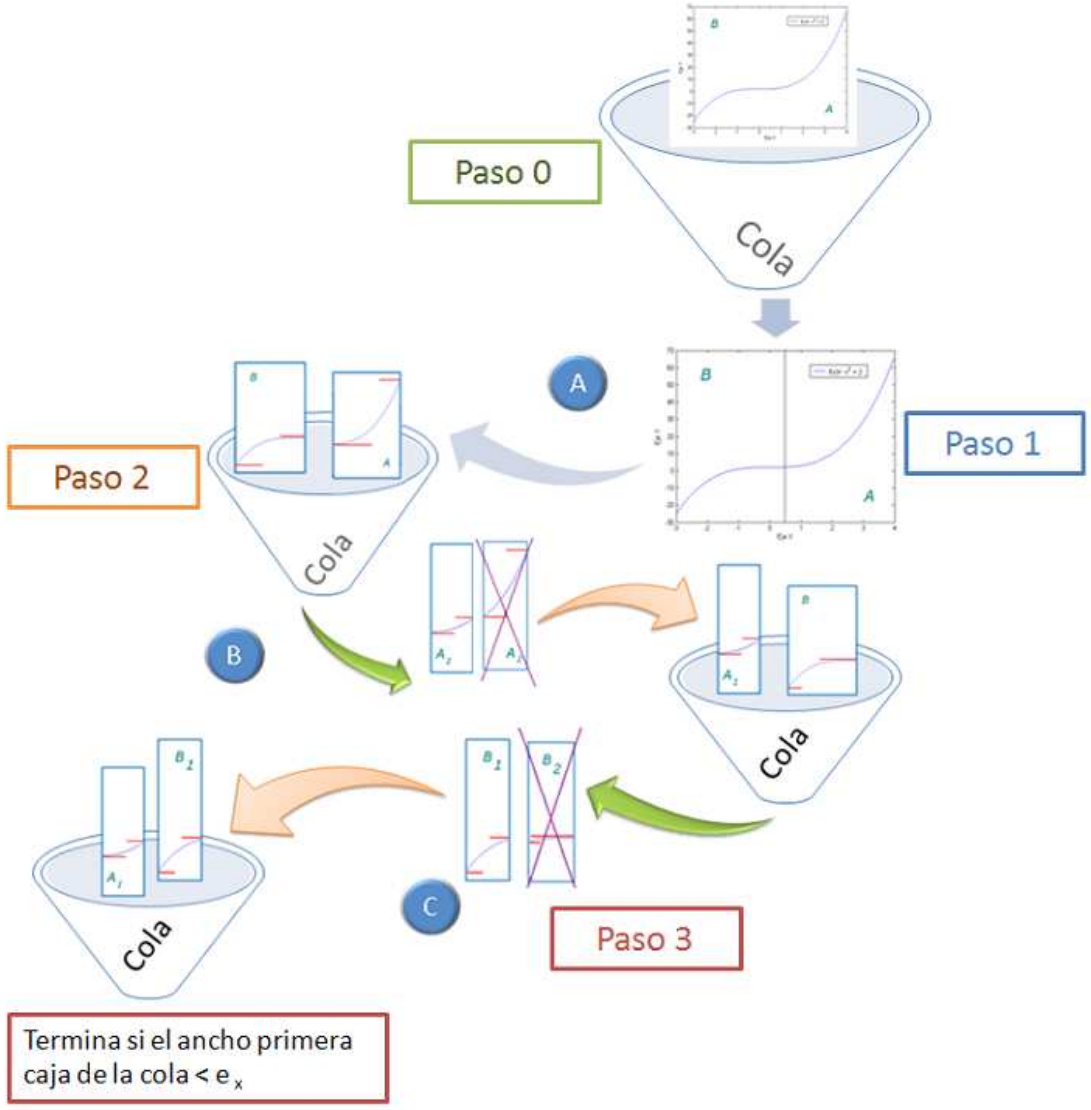


Figura 3.8 Algoritmo de Optimización Global por intervalos en tres pasos: "A": Se toma el espacio completo y este se ingresa a la cola, se extrae A del tope de la cola, este se bisecciona en A_1 y A_2. "B": A_2 es desechado

y se ingresa A1 a la cola. “C”: Se extrae B y se particiona dividiéndolo en B_1 y B_2, para luego eliminar B_2, ingresando B_1 a la cola.

El ciclo continúa hasta que la primera caja de la cola posea un ancho < ε_x, en otras palabras, hasta alcanzar la tolerancia del ancho de la caja.

Al terminar el algoritmo, la unión de las cajas en la lista contendrá todos los minimizadores globales.

CAPITULO 4. PARALELISMO, ESTUDIO DE ESTRATEGIAS Y ANÁLISIS DE PARALELIZACIÓN DE ALGORITMOS

En el tiempo actual cada vez se requiere mayor poder de procesamiento de parte de los sistemas informáticos, cada vez se usan procesos más complejos, que generalmente se hacen en serie, y los tiempos de éstos son muy relevantes, por lo que se busca la forma de disminuir estos tiempos. Para esto se necesita una computación de altas prestaciones, que se refiere a resolver problemas muy costosos computacionalmente minimizando el tiempo de respuesta. Esta es una labor imperiosa en este nuevo siglo.

En la actualidad, la computación paralela está siendo utilizada en multitud de campos para el desarrollo de aplicaciones y el estudio de problemas que requieren gran capacidad de cómputo, ya sea por el gran tamaño de los problemas que abordan o por la necesidad de trabajar con problemas en tiempo real. De esta forma, el paralelismo en la actualidad, además de constituir diversas líneas abiertas de intensa labor investigadora, puede encontrarse en infinidad de aplicaciones en campos muy variados, entre los que se destacan:

Modelado predictivo y simulación: se realiza mediante extensos experimentos de simulación por computador que con frecuencia acarrean computaciones a gran escala para obtener la precisión y el tiempo de respuesta deseado. Entre estos modelados destacamos la previsión meteorológica numérica y la oceanografía.

El desarrollo industrial para progresar en el diseño y automatización de proyectos de ingeniería, la inteligencia artificial y la detección remota de los recursos terrestres. En este campo destacamos: la inteligencia artificial y automatización (procesamiento de imágenes, reconocimiento de patrones, visión por computador, comprensión del habla, deducción automática, robótica inteligente, sistemas expertos por computador, ingeniería del conocimiento, etc.).

Investigación médica: En el área médica los computadores rápidos son necesarios en tomografía asistida, diseño de corazones artificiales, diagnóstico hepático, estimación de daños cerebrales y estudios de ingeniería genética. (Panadés, 1991).

La computación paralela se orienta a resolver rápidamente una tarea empleando múltiples procesadores simultáneamente. Esta práctica se vuelve popular a partir de fines de la década de 1980.

Se puede decir que la computación paralela es una técnica de computación que toma un problema, y éste, es dividido en subtareas que son resueltas simultáneamente, generalmente de manera fuertemente acoplada.

Los programas usualmente se ejecutan en arquitecturas homogéneas y pueden tener memoria compartida. (Ardenghi, 2007).Esto puede permitir solucionar problemas que requieran una gran cantidad de memoria, debido a que si usamos varios computadores o sistemas multiprocesadores, tendremos por lo general más memoria total disponible que la que nos puede otorgar un solo computador. (Wilkinson, 2005)

La paralelización introduce la necesidad de comunicación y sincronización, los computadores se comunican con otros a través del envió de mensajes. En las arquitecturas actuales éstas son lentas comparadas con la computación (por órdenes de magnitud). Los costos de comunicación son medidos en términos de latencia y ancho de banda. ***Latencia*** es el tiempo que se toma un mensaje para ir de una locación a otra. ***Ancho de banda*** es la cantidad de datos que pueden ser transferidos por unidad de tiempo en estado estable. Los costos de comunicación pueden ser reducidos pero no evitados. (Ardenghi, 2007)

Se conocen dos elementos importantes de la arquitectura paralela, las unidades de procesamiento y la comunicación entre estas unidades, estos dos deben ser óptimamente administrados y esa es tarea del software que es imprescindible para llevar a cabo de forma eficaz dicha tarea.

Los computadores pueden ser de diversas arquitecturas (X86, SPARC, SGl, etc) siempre y cuando este interconectados y pasando mensajes entre ellos, con un ambiente de software común para cada uno de los computadores perteneciente a la red.

La computación paralela no es una idea nueva. En el 1958 Gill ya había escrito acerca de la programación paralela. Posteriormente Holland en 1959 escribió sobre un computador capaz de realizar un numero arbitrario de cálculos simultáneamente, Conway describió el

diseño de un computador paralelo y su programación en 1963, este pensamiento venia dándose hace largo tiempo, estos autores veían en el procesamiento paralelo la forma de solucionar los grandes problemas dados en el procesamiento computacional. (Quinn, 2004)

4.1 Taxonomía clásica de Flynn

Es la más conocida clasificación de arquitectura de computación paralela. Propuesta por el profesor de ciencias de la computación Michael J. Flynn en 1972. (Flynn, 1972)(Duncan, 1990)

Un computador es clasificado según el grado de Paralelismo en su flujo de instrucción y en su flujo de datos. En un computador, su hardware puede soportar un flujo simple o múltiple de instrucciones, manipulando un flujo simple o un múltiple de datos. Por lo tanto flynn clasifica estos resultados en cuatro categorías, descripción de estas categorías se revisan en el anexoQ. (Quinn, 2004) Resumen de la Taxonomía de Flynn en la Figura 4.1

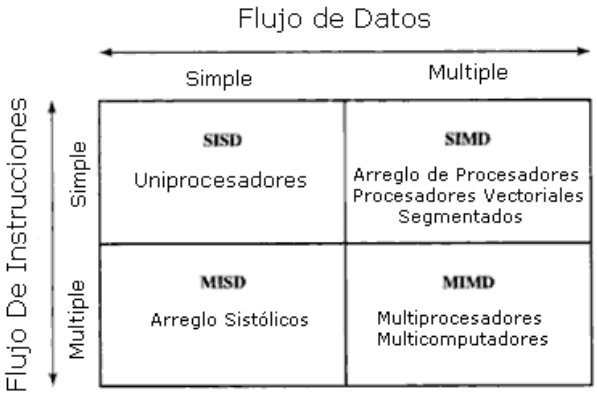


Figura 4.1 Clasificación de Arquitecturas según la Taxonomía de Flynn (Quinn, 2004)

4.2 Estrategias de Paralelización

En esta sección se presentan las dos técnicas fundamentales de programación paralela, divide y vencerás y particionamiento.

Estas técnicas están relacionadas. En el particionamiento el problema se divide en partes y cada parte se procesa separadamente. Divide y vencerás usualmente ocupa particionamiento y recursividad, de esta manera continua dividiendo el problema en partes

más y más pequeñas, después de esto resuelven las partes pequeñas y combina los resultados, para obtener el resultado final. Para más información revisar anexo D

4.2.1 Estrategia de Particionamiento

La estrategia de particionamiento divide el problema en partes. De una forma u otra esta es la base para toda la programación paralela. (Wilkinson, 2005)

La gran mayoría de las formulaciones, también requieren que los resultados de las partes sean combinadas para lograr el resultado esperado. El particionamiento puede ser aplicado de igual manera a los datos de programa (dividir los datos y operar sobre estos datos concurrentemente), esto es llamado particionamiento de datos o descomposición de dominio. El particionamiento también puede ser aplicado a las funciones de un programa, utilizando descomposición final, es decir, dividiendo el problema en funciones independientes y ejecutando éstas concurrentemente.

La idea del procesamiento por tareas es dividir una gran tarea en pequeñas tareas, entonces resolverlas por separado, cuando estén resueltas se combinan para completar la tarea principal. Esta estrategia es bien conocida y puede ser aplicada a muchas situaciones diferentes. Las pequeñas tareas operan sobre partes de datos o son separados en funciones concurrentes. Para entender mejor esto, nos referimos a un ejemplo en donde hay una secuencia de números x_0, x_1,…., x_{n-1} que son sumados entre ellos. Se podría considerar dividir esta secuencia en p partes de n/p números cada una, que se le asignara a cada uno de los p procesadores que se encargarán de las sumas parciales. La figura 4.2 muestra la forma como el procesador añade los resultados parciales para formar la suma total. (Wilkinson, 2005)

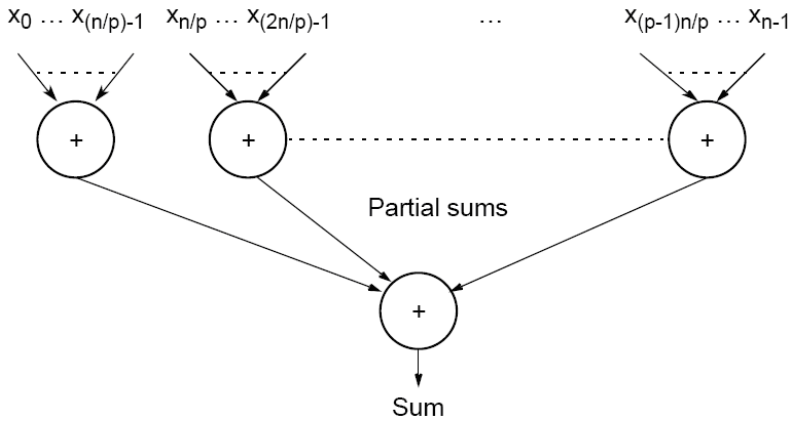


Figura 4.2 Particionamiento de una secuencia de números, sumas parciales que forman la suma final. (Wilkinson, 2005)

Nótese que cada procesador requiere acceder a los números que debe acumular, esto depende si se usa memoria compartida o paso de mensajes. En este último caso, los números necesitarían ser pasados o enviados a cada procesador individualmente. En el sistema de memoria compartida cada procesador podría acceder a los números, éstos se buscan en el espacio global de direcciones. (Wilkinson, 2005)

Este proceso revisado desde la aproximación maestro-esclavo es bastante simple. Los números son enviados desde el procesador maestro a los procesadores esclavos, estos añaden los números y operan concurrente e independientemente sobre éstos, enviando los resultados de cada procesador esclavo al procesador maestro. El procesador maestro añade todas las sumas parciales para formar el resultado final.

Lo que hay que revisar es cómo enviar los números desde el procesador maestro a procesadores esclavos. Hay dos formas, una de éstas es a través de broadcásting, la cual envía todos los números a cada procesador esclavo y obliga a que él decida la cantidad de datos correspondientes. La otra forma es que el procesador maestro elija qué datos enviará a cada uno de los procesadores esclavos según su rutina de proceso. La forma en que esto se realice la distribución de número, por parte del maestro, determinará cuanta carga va a tener cada procesador. Si envía todos los números y deja que los esclavos seleccionen se le alivianará la carga, en lo que a división y distribución de números se refiere. La carga pasará a los demás procesadores esclavos ya que éstos tienen que acceder al tipo de

asignación que se hará. En el caso de ser el maestro quien divida y distribuya los números, es él quien se llevará todo el trabajo. Análisis de esta estrategia se revisa en Anexos R. (Wilkinson, 2005)

4.2.2 Dividir para Conquistar

La aproximación dividir para conquistar se caracteriza por dividir el problema en subproblemas del mismo tipo del problema general (Figura 4.3). Este método recursivo continúa dividiendo el problema hasta que las tareas ya no pueden ser divididas, se llega a las soluciones triviales. Entonces estas tareas son ejecutadas y combinadas, así estos resultados combinados van retornando hasta llegar a la solución general y obtener el resultado global.

Para explicar este método se puede tomar como ejemplo una secuencia de números que se deben multiplicar x_0, x_1,…., x_{n-1}. Se comienza por el proceso raíz (nodo raíz), dividiendo este problema en dos y éstos en dos más hasta llegar a problemas que no se puedan dividir, esto formará un árbol de procesos llamado árbol binario.

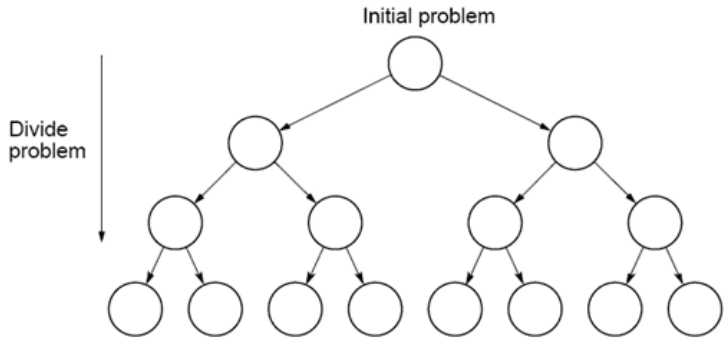


Figura 4.3 Construcción de un árbol binario por la estrategia dividir para conquistar. (Wilkinson, 2005)

Las llamadas a procesos se realizan recorriendo el árbol hacia abajo, cuando devuelve los resultados lo hace hacia arriba. En una implementación secuencial, sólo un nodo del árbol puede ser visitado al mismo tiempo. Una solución paralela ofrece la forma de atravesar varias partes del árbol simultáneamente. Una vez que la división es realizada en dos partes, ambas se puedan procesar simultáneamente. Se puede asignar simplemente un procesador a cada nodo del árbol. Esto requerirá como mínimo 2^{m+1} -1 procesadores, que dividen las

tareas en 2^m partes, en donde m representa el nivel del árbol a generar. Cada procesador solo estará activo en un nivel del árbol, dejando una muy ineficiente solución. Una muy eficiente solución es reutilizar el procesador en cada nivel del árbol, como lo muestra la figura 4.4.

La división del problema no para hasta que el número total de procesadores es utilizado. Hasta entonces, en cada etapa cada procesador mantiene la mitad de la lista pasando la otra mitad. Primero P_0 se comunica con P_4 , pasando la mitad de la lista a P_4 .entonces P_0 y P_4 pasan la mitad de su lista a P_2 y P_6 respectivamente. Finalmente P_0 , P_2 , P_4 , P_6 pasa la mitad de su lista ha P_1, P_3, P_5 y P_7 respectivamente. Cada lista al final de la última etapa tendrá n/8 números o n/p números en el caso general de p procesadores. Los niveles del árbol se darán por Log p.

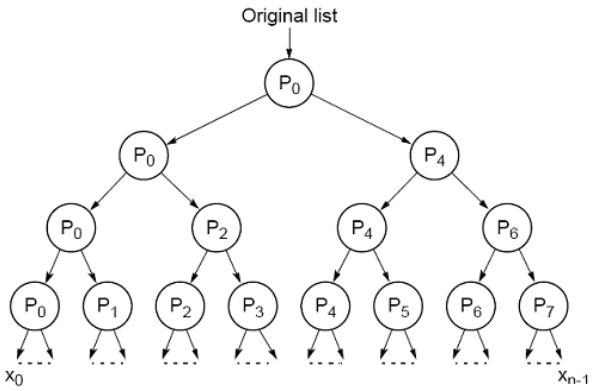


Figura 4.4 División del problema en partes y reutilización de cada procesador par, por nivel. (Wilkinson, 2005)

La combinación, acción de añadir las multiplicaciones parciales puede ser realizada como se muestra en la figura 4.5. Una vez que cada multiplicación suma parcial se haya computado, cada procesador impar pasa este resultado al procesador par; esto es P_1 pasa su multiplicación a P_0, P_3 a P_2, P_5 a P_4, P_7 a P_6. Los procesadores pares añaden las multiplicaciones parciales que recibieron del otro procesador par, a su propia multiplicación parcial traspasando el resultado al otro proceso que estaba llamando en ese momento. Esto

continúa hasta que P_0 obtiene el resultado final. El análisis de esta estrategia se presenta en el anexo S. (Wilkinson, 2005)

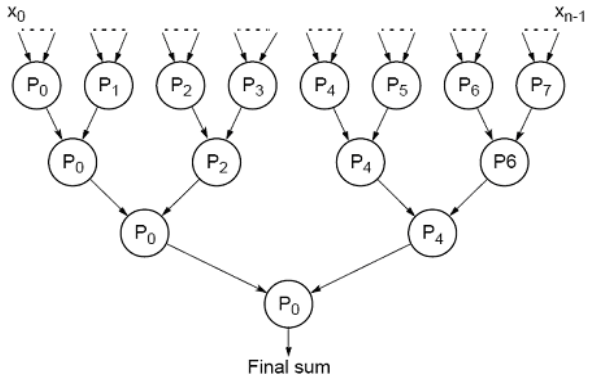


Figura 4.5 Suma de resultados parciales. (Wilkinson, 2005)

Ejemplo de estrategia de paralelización se muestra en anexos T.

4.3 Paradigmas de Programación Paralela

Cada uno de estos modelos es una abstracción de la arquitectura de hardware y memoria por lo que teóricamente estos modelos deberían poder implementarse en cualquier arquitectura (éstas se muestran en el anexo B). Ejemplos de éstos son:

Modelo de memoria compartida en una maquina con arquitectura de memoria distribuida: En la aproximación ALLCACHE de la Kendall Square Research (KSR), la memoria de la máquina está físicamente distribuida, pero aparece para el usuario como una única memoria compartida (espacio global de direcciones). Generalmente esta aproximación es conocida como Memoria Compartida Virtual.

Modelo paso de mensajes en una maquina con arquitectura de memoria compartida: La SGI Origin empleaba el tipo de arquitectura de memoria compartida CC-NUMA, donde cada tarea tiene acceso directo a la memoria global. Sin embargo la capacidad de enviar y recibir mensajes con MPI, es comúnmente provista por una red de maquinas con memoria distribuida. No solo esta implementada sino que es de uso muy general. (Barney, 2007)

Decidir cual modelo utilizar depende del la disponibilidad de recursos y de la elección del personal programador, no existe un mejor modelo, solo que algunos pueden trabajar mejor en ciertas configuraciones. (Barney, 2007)

4.3.1 Modelo de memoria compartida

En este modelo las tareas comparten un espacio de memoria común, que puede ser virtual, accediendo a ella asincrónicamente en su lectura-escritura. Surge la necesidad de usar mecanismos de control de acceso como semáforos y memory locks. Generalmente son empleados sobre SMPs. La ventaja que tiene es que el programador no necesita especificar explícitamente la comunicación entre datos y tareas, de esta forma se puede simplificar el desarrollo de programas. Una desventaja en términos de su rendimiento es que llega a ser más difícil de manejar y entender los datos locales.

4.3.2 Modelo de hebras o hilos

En el modelo de programación paralela mediante hebras o hilos, un único proceso puede tener múltiples trayectorias de ejecución concurrentes, cómo se muestra en la Figura 4.8. La analogía más simple que puede ser usada para describir las hebras, es el concepto de un único programa que incluye un número de subrutinas:

- El programa principal a.out es programado para ejecutarse en el sistema operativo nativo. a.out carga y adquiere todos los recursos de sistema y de usuario necesarios para funcionar.

- a.out ejecuta algún trabajo serial, y luego crea un número de tareas (hebras) que pueden ser despachadas y ejecutadas por el sistema operativo concurrentemente.

- Cada hebra tiene datos locales pero comparte los recursos de a.out. Esto ahorra la sobrecarga de los recursos del programa asociados por la replicación de recursos por cada hebra. Cada hebra también se beneficia de la memoria global debido a que comparte el espacio de memoria de a.out.

- Una hebra puede ser descrita como una subrutina dentro del programa principal. Cualquier hebra puede ejecutar cualquier subrutina al mismo tiempo que las otras hebras.

- Las hebras se comunican unas con las otras a través de la memoria global (actualizando posiciones de direccionamiento). Esto es asegurando que las hebras no actualicen las posiciones de memoria global al mismo tiempo.

- a.out persiste para proveer los recursos compartidos necesarios hasta que la aplicación sea completada.

Las hebras están asociadas con arquitectura de memoria compartida y sistemas operativos.(Guillen, 2006)(Barney, 2007)

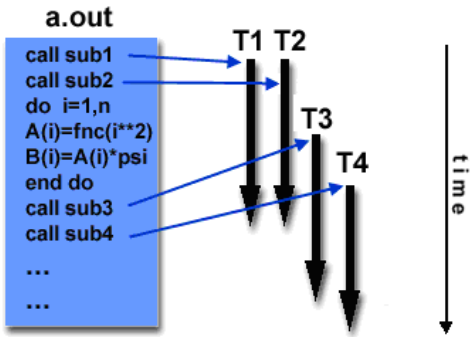


Figura 4.8 Ejecución de instrucciones según el modelo de hebras. (Guillen, 2006)(Barney, 2007)

4.3.3 Modelo de Pase de Mensajes

Son sistemas difíciles de programar, como de analizar, verificar, testear y depurar, de igual forma ocurre con las aplicaciones multithread. El costo se ve compensado con aplicaciones que realmente requieren de una gran carga computacional, como son las grandes bases de datos (ejemplo: warehousing), con algoritmos de cálculo extremadamente potentes y pesados, de búsqueda y análisis de datos con data mining, simulaciones y aplicaciones científicas que requieren procesar grandes cantidades de datos numéricos y de alta precisión. Este modelo se muestra en la Figura 4.9 (Rodríguez, 2004)

Características de este Modelo:

- Un conjunto de procesos usando solo memoria y datos locales

- Los procesos se comunican enviando y recibiendo mensajes

- La transferencia de datos requiere operaciones cooperativas para ser ejecutada por cada proceso (una operación de envío debe tener una operación de recepción)

- Manejado por el Programador

- La programación con pase de mensajes es realizada enlazando y haciendo llamadas a bibliotecas las cuales administran el intercambio de datos entre los procesadores

- Requiere gran esfuerzo de programación pero se logra alto rendimiento.

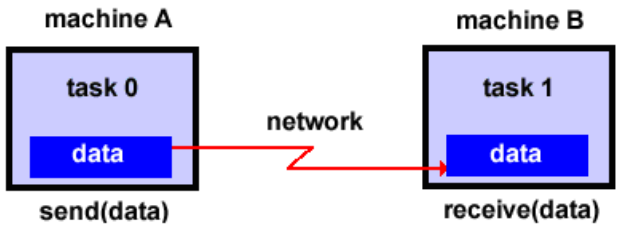


Figura 4.9 Comunicación de dos maquinas por paso de mensajes. (Blaise, 2007)

4.3.4 Modelo de Dato Paralelo

El objetivo del modelo de dato paralelo es realizar operaciones simultáneas sobre un conjunto de datos. Estos datos se organizan con estructuras típicas como arrays (arreglos, vectores).

Un conjunto de tareas trabajan colectivamente realizando la misma operación sobre la misma estructura de datos pero ocupándose cada una de una partición diferente dentro de dicha estructura. Revisar Figura 4.10

En arquitecturas de memoria compartida, todas las tareas pueden tener acceso a la estructura de datos a través de la memoria global. En la arquitectura de memoria distribuida la estructura de datos se divide y reside como “pedazos” en la memoria local de cada tarea.

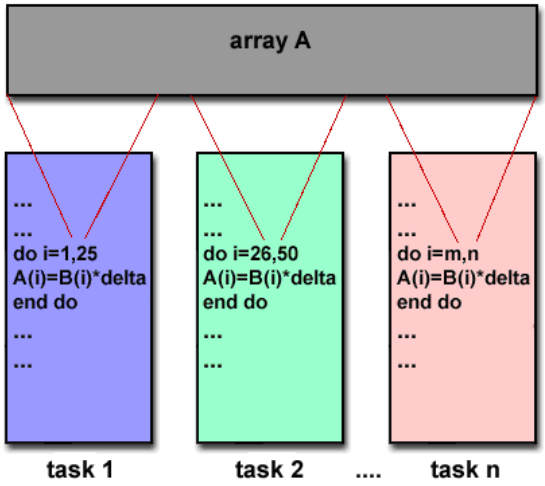


Figura 4.10 Ejecución de un programa con dato paralelo. (Blaise Barney, 2007)(Jorge, 2007)

♦ La paralelización es generalmente realizada por constructores y compiladores que soporten la característica de dato paralelo. Ejemplo: Fortran 90 o HPF.

.4.3.5 Modelo Hibrido

Se llama modelo hibrido, el cual resulta de la combinación de dos o mas paradigmas presentados anteriormente**.** Véase el ejemplo:

♦ MPI con OpenMP en un cluster de SMPs. En este caso se utiliza el paradigma para la parelización entre los procesadores de un SMP y MPI para la comunicación entre nodos.

- Dato paralelo con MPI. Este caso es comúnmente implementado sobre un cluster se SMPs similar al ejemplo anterior. (Barney, 2007)(Jorge, 2007)

Es muy importante saber elegir bien el modelo antes de comenzar a desarrollar una aplicación paralela. Para esto se necesita:

- Establecer los objetivos de rendimiento y escalabilidad.
- Determinar la arquitectura de hardware en la que se implementará.
- Determinar el tiempo y esfuerzo que se puede invertir.

- Determinar el grado de paralelismo del problema. Esto se puede calcular con la ley de Amdahl, ésta se presenta en el anexo D. (Jorge, 2007)

4.4 Diseño de Programas paralelos

Diseñar programas paralelos no es tarea fácil, se requiere de una gran creatividad y conocimiento para aplicar las mejores técnicas, logrando obtener el mejor provecho de la paralelización de algún problema complejo. Ian Foster propuso cuatro pasos para identificar el proceso de diseño de algoritmos paralelos. (Foster, 1995). Éste desea fomentar el desarrollo de algoritmos paralelos escalables, postergando para los pasos finales las consideraciones que dependen de la arquitectura de la maquina en si. Los cuatro pasos se definen como partición, comunicación, aglomeración y mapeo. El esquema de diseño de programas paralelos se muestra en la figura 4.11 (Quinn, 2004)

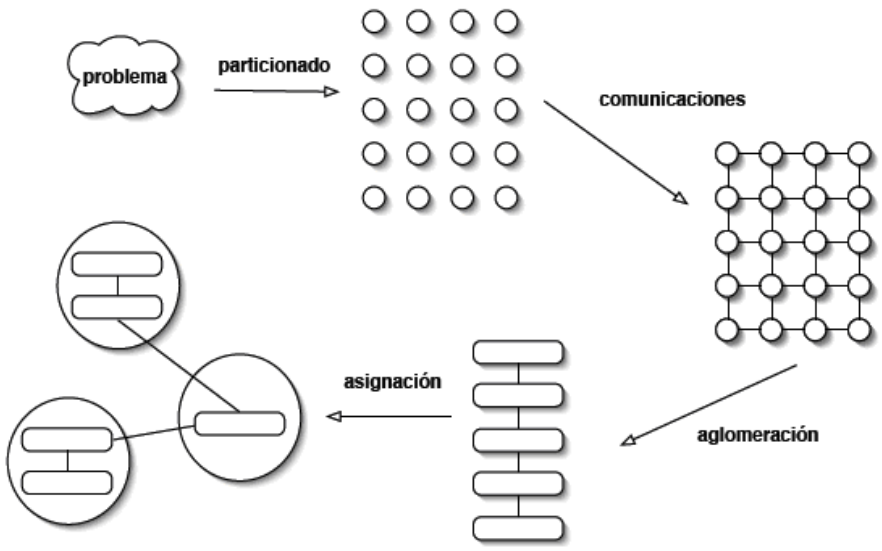


Figura 4.11 Metodología de diseño de Foster para algoritmos paralelos. (Quinn, 2004)

4.4.1 Partición

En la etapa de partición se busca oportunidades de paralelismo y se trata de subdividir el problema lo más finamente posible, para así poder distribuirlos a múltiple tareas**.** Hay dos formas de proceder con la descomposición, a pesar que esta depende de las fuentes del paralelismo, la cual es mencionada en el anexo D.

Descomposición del dominio: El centro de atención son los datos. Se determina la partición apropiada de los datos y luego se trabaja en las tareas asociados con los datos.

Descomposición funcional: Es el enfoque alternativo al anterior. Primero se descomponen las tareas y luego se ocupa de los datos.

Estas técnicas son complementarias y pueden ser aplicadas a diferentes componentes de un problema e inclusive al mismo problema para obtener algoritmos paralelos alternativos.

Se puede evaluar la calidad del proceso de partición. Un buen diseño satisface los siguientes atributos. (Quinn, 2004)

- Se ha identificado al menos un número de tareas, un orden de magnitud mayor que el número de procesadores disponibles.
- Se evitan los cálculos redundantes y las copias de datos.
- Todas las tareas tienen tamaños comparables.
- Es escalable el número de tareas con el tamaño del problema.
- Se han identificado varios particionados alternativos.

4.4.2 Comunicación

Después de haber identificado las tareas primitivas particionadas, el siguiente paso es lograr la comunicación entre ellas. Los datos deben ser transferidos o compartidos entre tareas, lo que se denomina la fase de comunicación. La comunicación requerida por un algoritmo puede ser definida en dos fases. Primero se definen los canales que conectan las tareas que requieren datos con las que los poseen. Segundo se especifica la información o mensajes que deben ser enviados y recibidos en estos canales. (García, 2003)

Entender las comunicaciones entre los procesadores es esencial.

• La comunicación de pase de mensajes es programada explícitamente. El programador debe entender y codificar la comunicación.

• Los compiladores de paralelismo de datos hacen toda la comunicación transparente. El programador no necesita entender las comunicaciones subyacentes.

Se puede evaluar la calidad del proceso de comunicación. Un buen diseño satisface las siguientes proposiciones (Quinn, 2004)

- Se ha tenido en cuenta diseños basados en una o varias tareas por procesador.
- Las operaciones de comunicación son balanceadas entre tareas.
- Cada tarea se comunica con solo un pequeño número de tareas vecinas.
- Las tareas pueden realizar su comunicación de forma concurrente.
- Las tareas pueden realizar sus cómputos de forma concurrente.
- Se ha tenido en cuenta la asignación estática y dinámica de tareas.
- Si se usa asignación dinámica, Se producen cuellos de botella.
- Si se usa asignación estática, Se asignan al menos, a razón de 10:1 tareas por procesador.

4.4.3 Aglomeración

En este proceso se agrupan las tareas dentro de una tarea más grande, para mejorar las prestaciones y simplificar la programación. A veces se quiere que el número de las tareas consolidadas sean mayores que el número de los procesadores en el cual el algoritmo será ejecutado. Sin embargo, cuando se desarrollan programas usando MPI se deja la aglomeración con una tarea por procesador. En este caso el mapeo de tareas a procesadores es trivial. (Quinn, 2004)

Los objetivos de la aglomeración se pueden resumir en:

- Minimizar las comunicaciones:
  - Incrementando la localidad.

- Combinando grupos de tareas. Menos mensajes, pero más grandes.

- Mantener la escalabilidad sin modificar el diseño.

- Reducir los costos de ingeniería del software.

Se Incrementa la localidad cuando se aglomeran tareas que se comunican con otras, eliminando la comunicación, por que el control de los valores de los datos están ahora en la memoria de las tareas consolidadas. Es ideal lograr aglomerar tareas relacionadas para que se pueda lograr la concurrencia. Agrupar tareas en enviadas y recibidas, reduciendo el tiempo de mensajes enviados. Los mensajes grandes toman menos tiempo en enviarse que varios mensajes pequeños del mismo tamaño del mensaje grande en su conjunto, debido al costo de comunicación asociado cada vez que se envía un mensaje, llamado latencia del mensaje.

La escalabilidad se refiere a que con la aglomeración se logre portar el programa, en algún futuro a un ambiente con más procesadores sin que se vea afectado el funcionamiento del algoritmo. La aglomeración podría permitir hacer uso del código secuencial existente reduciendo el tiempo y el costo de desarrollar programas paralelos, de esta forma se puede reutilizar el código.

Se puede evaluar la calidad del proceso de aglomeración. Un buen diseño satisface las siguientes proposiciones. (Quinn, 2004)

- Se ha incrementado la localidad.

- Los cálculos replicados compensan las comunicaciones que evitan.

- Si se replican datos, compromete esto la escalabilidad.

- Pueden desarrollarse los cálculos concurrentemente.

- Tienen las tareas aglomeradas un coste similar de cálculos y comunicaciones.

- Es el número de tareas una función creciente del tamaño del problema.

- Se adapta el número de tareas al número de procesadores.
- Las tareas resultantes permiten reutilizar código sin muchos cambios.

4.4.4 Mapeo

Proceso de asignar tareas a los procesadores, también es conocido como asignación. La asignación de tareas puede ser estática o dinámica. En la asignación estática, las tareas son asignadas a un procesador al comienzo de la ejecución del algoritmo paralelo y se ejecutan ahí hasta el final del proceso. La asignación estática en ciertos casos puede resultar en un tiempo de ejecución menor respecto a asignaciones dinámicas y también puede reducir el costo de creación de procesos, sincronización y terminación.

En la asignación dinámica se hacen cambios en la distribución de las tareas entre los procesadores en tiempo de ejecución, o sea, hay migración de tareas en tiempo de ejecución. Esto es con el fin de balancear la carga del sistema y reducir el tiempo de ejecución. Sin embargo, el costo de balanceo puede ser significativo y por ende incrementar el tiempo de ejecución.

Objetivos:

- Maximizar el uso de los procesadores. Equilibrar la carga.
- Minimizar las comunicaciones. Tareas conectadas a un mismo procesador.

Se puede evaluar la calidad del proceso de mapeo. Un buen diseño satisface las siguientes proposiciones. (Quinn, 2004)

- Se han tenido en cuenta diseños basados en una o varias tareas por procesador.
- Se ha tenido en cuenta la asignación estática y dinámica de tareas.
- Si se usa asignación dinámica, Se producen cuellos de botella.
- Si se usa asignación estática, Se asignan al menos, a razón de 10:1 tareas por procesador.

4.5 Balanceo de Carga

Se puede definir balanceo de carga como un esquema aplicado al procesamiento distribuido y/o al sistema de comunicación, que tendrán los distintos computadores o procesadores con el fin de que éstos no se saturen y no queden ociosos mediante el procesamientos de datos en la ejecución de un programa en particular.(Quiñónez, 2005)

El balanceo de carga para un sistema paralelo es uno de los problemas más importantes que tiene que ser resuelto a fin de permitir el uso eficiente de los sistemas de computación paralela. Varios objetivos deben ser tomados en consideración.

Todo el trabajo debe completarse lo más rápido posible.

- En la medida que los procesadores son de alto rendimiento, deben mantenerse ocupados. Si sólo hay poco trabajo que hacer, la distribución de las tareas tiene que ser muy bien planificada. Por otra parte, en otros períodos de tiempo la distribución se puede hacer en línea recta porque no hay suficiente trabajo por hacer.

- Las cargas de trabajo deben ser distribuidas equitativamente, la misma cantidad de trabajo debe asignarse a cada procesador, dependiendo de la potencia de éste.

- Hay precedencia de limitaciones entre las distintas tareas (se puede empezar a construir el techo sólo después de terminar las paredes). Así también se tiene que encontrar un orden de procesamiento inteligente de las diferentes tareas.(Decaer, 2001)

Como se mencionó en la sección anterior, la asignación de carga puede ser estática; en donde las tareas son asignadas a un procesador hasta finalizar la aplicación, como también puede ser dinámica; en donde las tareas serán distribuidas en tiempo de ejecución, las cargas de procesadores cambiará a medida que la aplicación se ejecuta. Entre los algoritmos de balanceo de carga están los siguientes:

Balanceo centralizado: Un nodo ejecuta el algoritmo y mantiene el estado global del sistema. Este método no es aplicable a problemas muy grandes ya que el nodo encargado del balanceo se convierte en un cuello de botella.

Balanceo completamente distribuido: Cada procesador mantiene su propia visión del sistema intercambiando información con sus vecinos para así hacer cambios locales. El costo de balanceo se reduce pero no es óptimo debido a que solo se dispone de información parcial.

Balanceo semi-distribuido: Divide los procesadores en regiones, cada uno con un algoritmo centralizado local. Otro algoritmo balancea la carga entre las regiones. El balanceo puede ser iniciado por envío o recibimiento. Si es **balanceo iniciando por envío,** un procesador con mucha carga envía trabajo a otros. Si es **balanceo iniciado por recibimiento,** un procesador con poca carga solicita trabajo de otros. Si la carga por procesador es baja es iniciada por recibimiento. De lo contrario, en ambos casos, se puede producir una fuerte migración innecesaria de tareas. (Quiñónez, 2005).

Un sistema muy interesante formado por tres tipos de procesos es una Granja de procesos, descrita en el anexo M.

CAPITULO 5. DESCRIPCIÓN DE C++, MPI Y CLUSTER

En el capitulo anterior se hablo de que la computación paralela busca la división del trabajo en tareas más pequeñas para trabajar sobre éstas de manera concurrente, además ésta se puede aplicar a cualquier arquitectura de computadores. También que la paralelización puede ser de datos, de tareas y de funciones, y que los modelos que se pueden aplicar son por paso de mensajes, multithread, memoria compartida y dato paralelo, además de un hibrido de éstos. En el modelo de paso de mensajes recae gran responsabilidad en el programador en su analizar, verificar, testear y depurar.El costo que tiene estas implementaciones se ve compensado con aplicaciones que requieran una gran carga computacional, tales como las grandes base de datos como los data Warehousing , con algoritmos de cálculos extremadamente potentes y pesados, de búsqueda y análisis de datos con data mining, simulaciones y aplicaciones científicas. Una agrupación de maquinas de mínima o mediana capacidad computacional, como lo es un cluster, es menos costoso comparativamente a otros computadores de mayor rendimiento, para implementar sistemas paralelos, por ende la importancia de estudiarlos. Pero éstos también requieren de ciertos factores que son importantes como la necesidad de mecanismos portables de programación para adaptar un programa a las especificaciones de un sistema distribuido, Ej. Sockets. Considerar que la velocidad de transmisión siempre es inferior que en un sistema de bus procesador-memoria, aunque se reduzca al mínimo la velocidad del bus o logrando que la red sea los mas rápida posible.

En este capitulo se concentrara en el sistema de paso de mensajes, que se trabaja generalmente en cluster y que utiliza un lenguaje secuencial como el lo es C, además de un conjunto de librerías de funciones de paso de mensajes como los son las Librerías MPI: Estos ítems son relevantes para el estudio que se realiza.

5.1 LENGUAJE C ++

El lenguaje c++ se comenzó a desarrollar en el año 1980. El autor de este lenguaje es B. Stroustrup que pertenecía a AT&T, American Telephone and Telegraph (AT&T, 2008). Este lenguaje comenzó a ser utilizado fuera de AT&T en el año 1983 como también su nombre, que hace referencia al carácter del operador incremento de C (++). Se comenzó

a estandarizar internamente en 1987 por AT&T. En 1989 se formó un comité ANSI, seguido posteriormente por un comité ISO, para estandarizarlo a nivel americano e internacional.

El mayor cambio que presenta el lenguaje C++ es la de un concepto estructurado, que se basa en estructuras de control bien definidas, bloques de código, subrutinas independientes que soportan recursividad y variables locales. Agrega el concepto orientado a objeto que presenta una nueva definición, la de objeto, que permite descomponer un problema en varios subgrupos relacionados. El objeto contiene sus propias instrucciones y datos que le relacionan con si misma. Además agrega características propias de la POO, programación orientada a objeto, que son:

Encapsulación: Es el mecanismo que agrupa el código y los datos que maneja que son propios de cada objeto. Ese código y datos pueden ser privados para el objeto o públicos para otras partes del programa.

Polimorfismo: Es la cualidad que permite que un nombre se utilice para dos o mas propósitos relacionados pero técnicamente diferente. Por ejemplo en C tenemos tres funciones distintas para devolver la suma. Sin embargo en C++ que incorpora Polimorfismo, cada función se puede llamar sum() en cualquiera de los objetos creados. El Polimorfismo se puede aplicar tanto a funciones como a operadores.

Herencia: Proceso mediante el cual un objeto puede adquirir las propiedades de otro objeto. La información se hace manejable gracias a la clasificación jerárquica.

Objeto: Conjunto de variables y funciones pertenecientes a una clase y que están encapsulados. A este encapsulamiento es al que se denomina objeto, por lo tanto la clase, define las características y funcionamiento del objeto. (Guerrero, 2004)

Para más profundización y detalle sobre la creación de clases y objetos, revisar el anexo N.

5.2 MPI, Message Passing Interface

Varios lenguajes de programación paralela se han introducido en los últimos 40 años. Muchos de ellos son de alto nivel que simplifican distintos aspectos de la gestión de

paralelismo. No solo estos tipos han ganado aceptación en la comunidad de la programación paralela, ya que la mayoría continua haciendo programas paralelos como por ejemplo en Fortran C aumentado con funciones para el paso de mensajes entre procesos. MPI es el más popular estándar de paso-mensajes de soporte a la programación paralela. Prácticamente todos los computadores paralelos comerciales traen soporte MPI, y la reunión de sus bibliotecas libres. (Quinn, 2004)

Este interfaz permite al programador facilidades para controlar localidad, concurrencia, asignación de tareas y comunicaciones. A través de esta una aplicación se puede ver el entorno paralelo como un número estático de procesos.

Un proceso MPI es creado en un rango de 0 más componentes. La colección inicial de procesos se denomina grupo mundial, a cada proceso MPI se le asigna un identificador único o rango dentro del intervalo de 0 y n-1, en donde n es igual al número de procesos del grupo mundial, cada proceso puede consultar por su rango además del tamaño del grupo. Se pueden utilizar uno de los dos modelos de programación paralela sobre un cluster, (SPMD: Simple Program Multiple Data) en donde los procesos pueden ejecutar el mismo programa sobre datos distintos, (MPMD: Multiple Program Multiple Data) en donde se ejecutan distintos programas sobre distintos datos. (Rodríguez, 2004)

Estructura General del Programa MPI, Figura 5.1.

Figura 5.1 Estructura General de un Programa MPI (Blaise2, 2007)

Información sobre rutinas de comunicación y tipos de datos que se utilizan en MPI, se especifican en los anexos E, F, G, H, I.

5.3 Clusters

Los cluster o agrupamiento de maquinas, son sistemas de procesamiento paralelo o distribuido, en donde varios PCs autónomos y heterogéneos son interconectados por una red trabajando juntos como un recurso de computación integrado. (Rodríguez, 2007)

Existen varias formas de interconexión, las cuales se mencionan en el anexo C.

En 1994, se integro el primer cluster de PCs en el centro de Vuelos Espaciales Goddard de la NASA, para resolver problemas computacionales. Los Pioneros del proyecto fueron Thomas Sterling, Donald Becker y otros Cientificos de la Nasa trabajando en CESDIS (Center of Excellence in Space Data and Information Sciences) bajo el patrocinio del Proyecto de la Tierra y Ciencias del Espacio (ESS) Construyeron un cluster de computadoras que consistía en 16 procesadores 486DX4, usando una red Ethernet de 10Mbps, con un costo de 40,000 dólares. El rendimiento del cluster era de 3.2 Gflops.

Los Investigadores de la Nasa le dieron el nombre de Beowulf a este Cluster, en honor al Héroe de la Leyendas medievales quien derroto al monstruo gigante Grendel. (Medina, 2007)(Bernal, 2005)

5.3.1 Tipos de Clusters

Estos tipos se basan en el uso que se le da al cluster y los diferentes servicios que este puede ofrecer, podemos identificar dos tipos.

5.3.1.1 Alto Rendimiento

Configuración de equipos diseñada para proporcionar una capacidad de cálculo mayor. Para tareas que requieren gran poder computacional, grandes cantidades de memoria, o ambos a la vez. Las tareas podrían comprometer los recursos por largos períodos de tiempo. En esta clasificación se encuentran los sistemas de tipo Beowulf.

5.3.1.2 Alta Disponibilidad

Grupo de dos o mas maquinas, que como característica, comparten los discos de almacenamiento de datos y están constantemente monitorizándose entre si. Máxima disponibilidad de servicios. Rendimiento sostenido. (Medina, 2007)(Bernal, 2005)

5.3.2 Arquitectura de un Cluster

Los componentes que forman un sistema Cluster son:

1. Un conjunto de ordenadores de altas prestaciones.

2. Redes de interconexión de altas prestaciones (Myrinet, Gigabit, Infiniband).

3. Tarjetas de conexión a red de alta velocidad.

4. Protocolos y servicios de comunicación de alta velocidad.

5. Middleware, compuesto de dos subniveles de software:

- La imagen de sistema único (SSI: Single System Image) proporciona al usuario una visión unificada del cluster como un único recurso o sistema de cómputo.

- Disponibilidad del sistema que permite servicios como puntos de chequeo, recuperación de fallos, soporte para tolerancia a fallos.

6. Entornos y herramientas de programación paralela, compiladores paralelos, Java, PVM, MPI.(Sánchez, 2008)**.**La figura 5.3 muestra un esquema cluster.

5.3.3 Cluster Beowulf

Nombre que recibe en general las técnicas y procedimientos que son necesarios para utilizar la distribución Linux como cluster computacional, agrupar computadores con esta distribución para formar un súper computador virtual paralelo.

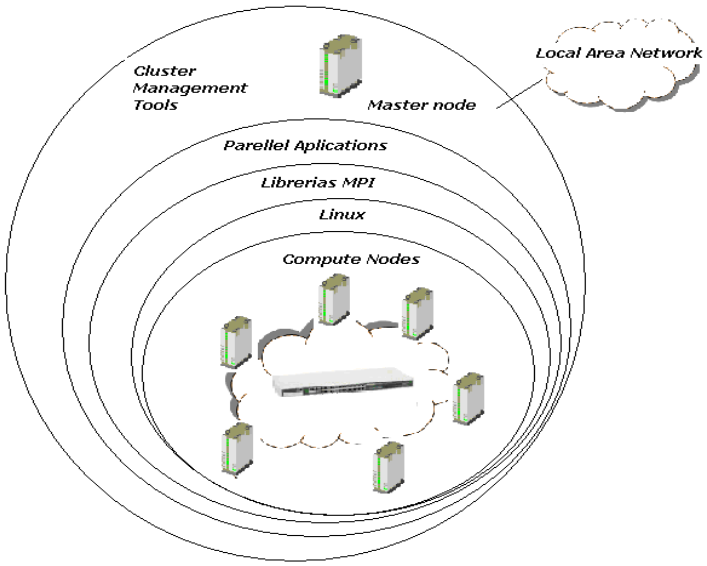


Figura 5.3 Componentes HW y SW de un Cluster (Adaptación)(Rodríguez, 2004)

5.3.3.1 Requerimientos de Hardware

Este tipo de sistemas están constituidos por componentes de hardware que son comunes encontrarlos en el mercado, como los PC de uso masivo, capaz de ejecutar Linux, adaptadores de Ethernet y switches comunes. Estos sistemas consisten en un nodo maestro y uno o más nodos esclavos que son conectados a través de una red ethernet u otra topología de red, las cuales son mencionadas en el anexo C. El nodo maestro controla el cluster entero y presta los servicios de sistemas de archivos a los nodos esclavos, además de ser la conexión hacia el mundo exterior a través de su consola. Los cluster beowulf pueden tener más de un nodo maestro, y nodos dedicados a variadas labores específicas como consolas o estaciones de supervisión. Los nodos son configurados y controlados por el nodo maestro y harán solo lo que éste les indique. En configuraciones en donde los nodos esclavos no tienen disco duro, no conocen su dirección IP hasta que el nodo maestro se lo indique.

5.3.3.2 Requerimientos de Software

Beowulf utiliza cualquier distribución Linux. Además de una biblioteca de paso de mensajes como PVM y MPI

CAPÍTULO 6. PROCESO DE IMPLEMENTACIÓN DEL CLUSTER DE ALTO RENDIMIENTO EN LA FACULTAD DE CIENCIAS EMPRESARIALES (FACE) DE LA UNIVERSIDAD DEL BIO-BÍO

La Facultad de Ciencias Empresariales (FACE) cuenta con un cluster de alto rendimiento "High Performance Computing (HPC)" cuyas instalaciones físicas se encuentran en la sala de servidores, ubicada en el segundo piso de la Facultad.

El cluster, basado en el proyecto beawulf, se encontraba en funcionamiento hasta mediados de Octubre de este año.

Para el cumplimiento de los objetivos de esta memoria se debe contar de forma necesaria con un cluster HPC a disposición, este es una herramienta fundamental para el proceso de desarrollo y pruebas de los algoritmos, por ende se vio necesario comenzar la tarea de una restructuración del cluster de forma inmediata. La nueva implementación constará de mayor cantidad de nodos y mejor seguridad.

6.1 Características Técnicas

El cluster a implementar pertenece a la categoría de HPC (High Performance Computing ó Alto Rendimiento) que es ideal para procesar grandes volúmenes de información en mucho menor tiempo que con soluciones tradicionales.

6.1.1 Topología

Un nodo maestro donde se invocarán los procesos y cuenta con salida al exterior (LAN de la facultad) y otra hacia la red del cluster (comunicación y tráfico de red).

Siete nodos esclavos que se encuentran unidos entre sí por medio de un switch, y se encargan de ejecutar las tareas enviadas por el maestro y a su vez se comunicarán en tiempo de ejecución entre ellos y hacia el maestro.

El cluster se organiza de la siguiente manera:

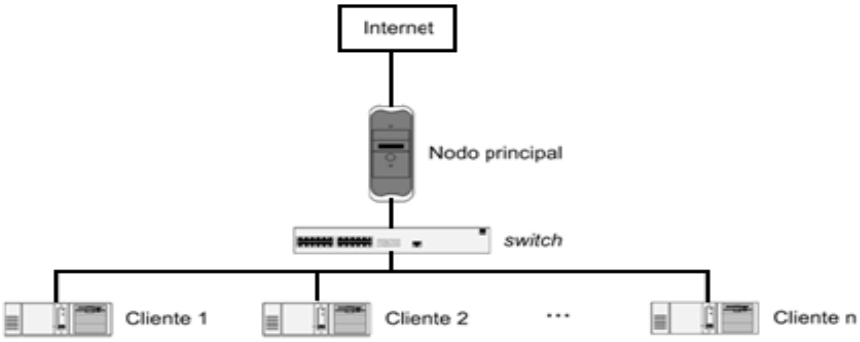


Figura 6.1 Topología Cluster ubicado en Facultad de Ciencias Empresariales de la Universidad del Bio-Bío. Sede Concepción

6.1.2 Hardware

A grandes rasgos el cluster (cluster.face.ubiobio.cl) cuenta con:

Siete Computadores homogéneos con procesador P4 y 256 MB Ram, con tarjetas ethernet 10/100 base TX, que corresponden a los nodos esclavos, Un computador maestro de similares características pero con dos tarjetas ethernet 10/100 base TX; una para la red del cluster y otra para la LAN. Para administrar la red se cuenta con un switch de 24 bocas.

6.1.3 Sistema Operativo

Se instaló un sistema operativo único para todos los equipos, el elegido fue Debian GNU/Linux 4.0. Para mayor detalle de la instalación visitar la bitácora de instalación en el anexo O.

6.1.4 Paquetes necesarios

Se instala y configura LAM/MPI en todas las máquinas.

Una de las distintas medidas que se implemento como forma de elevar considerablemente la seguridad al cluster fue cambiar el número de puerto utilizado por el servicio ssh, por otro que solo conozca el administrador del sistema. A este tipo de técnicas se les conoce como seguridad por oscuridad.

Se cambio el puerto por defecto, ya que la mayoría de los delincuentes informáticos utiliza guiones que buscan servidores que respondan a peticiones a través del puerto 22. Cambiar de puerto el servicio de ssh disminuye considerablemente la posibilidad de una intrusión a través de este servicio. El fichero de configuración se encuentra en la siguiente ruta:

/etc/ssh/sshd_config

De esta forma, se cuenta con la herramienta fundamental para proceder a someter a distintas pruebas y distintos escenarios los distintos algoritmos que se desarrollaron a lo largo de la memoria. El cluster de la Facultad de Ciencias Empresariales de la universidad del Bio-Bío se encuentra configurado y operativo, listo para ser utilizado.

CAPÍTULO 7. ANÁLISIS DE ESTRATEGIAS IMPLEMENTADAS PARA EL ALGORITMO DE OPTIMIZACIÓN GLOBAL

En el presente capítulo se analizarán las distintas estrategias de paralización para el algoritmo de optimización global por intervalos, en la primera parte del capítulo se analizará en base a la implementación descrita en (Muñoz y Peña, 2007) y en el algoritmo de optimización global por intervalos descrita en (Campos, 2004). Luego se profundizará en las nuevas estrategias de paralelización de intervalos desarrolladas en la presente habilitación profesional.

7.1 Descripción de la estrategia Muñoz y Peña

La estrategia previamente implementada se basa en la división y asignación de datos de dos maneras: por división del espacio inicial de búsqueda, y por división de cajas resultado obtenidas mediante una optimización inicial.

7.1.1 División del Espacio de Búsqueda

Se tiene un conjunto inicial de datos, dominio de búsqueda, el cual es particionado por el procesador maestro y asignadas las partes a cada procesador esclavo según el número disponible de éstos. Cada procesador esclavo ejecuta el algoritmo de optimización global de forma independiente sobre su respectiva porción de datos, retornando los resultados obtenidos al procesador maestro, el cual compara los Óptimos parciales y descarta aquellos que no corresponden a Óptimos globales. Véase figura 7.1

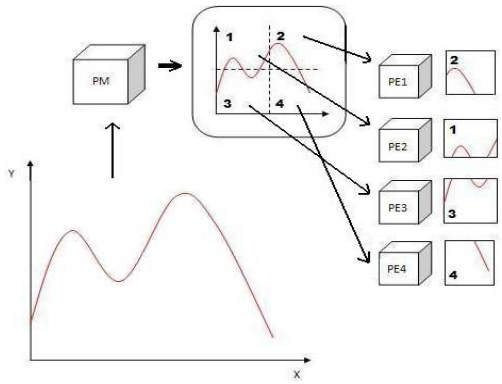


Figura 7.1 Particionamiento de datos por espacio de búsqueda. (Muñoz y Peña, 2007)

***Algoritmo 1**: División de Espacios de Búsqueda* (Figura 7.2)

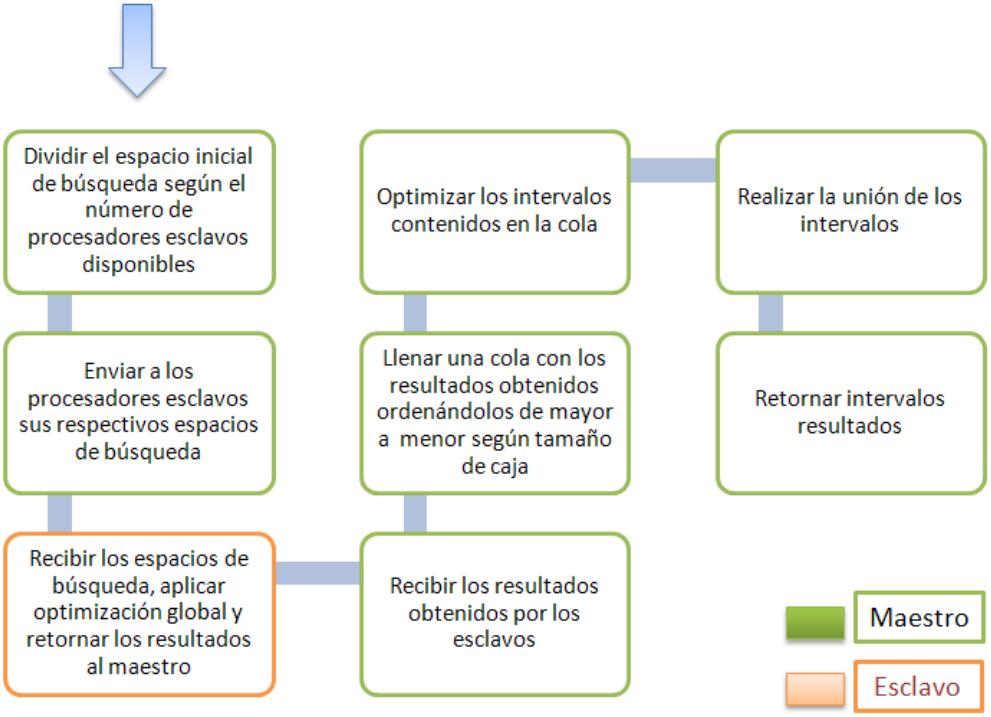


Figura 7.2 Algoritmo de división de espacios de búsqueda

7.1.2 División de Cajas Resultado

Esta estrategia se diferencia de la anterior por el hecho de realizar una optimización inicial en el procesador maestro, obteniendo una cola que contiene las cajas resultantes, asignando éstas a cada procesador esclavo según el número disponible de éstos. Véase la figura 7.3. Cada procesador esclavo ejecuta el algoritmo de optimización global de forma independiente sobre sus respectivas cajas, retornando los resultados obtenidos al procesador maestro, el cual compara los óptimos parciales y descarta aquellos que no corresponden a óptimos globales.

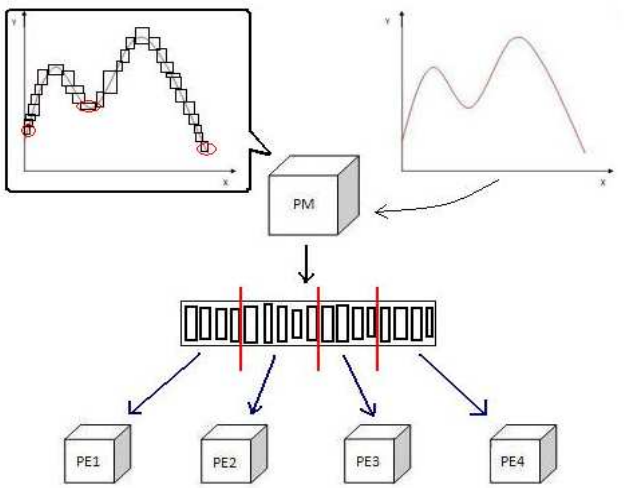


Figura 7.3 Particionamiento de datos por división de Cajas Resultado. (Muñoz y Peña, 2007)

Algoritmo 2: *División de Cajas Resultado*

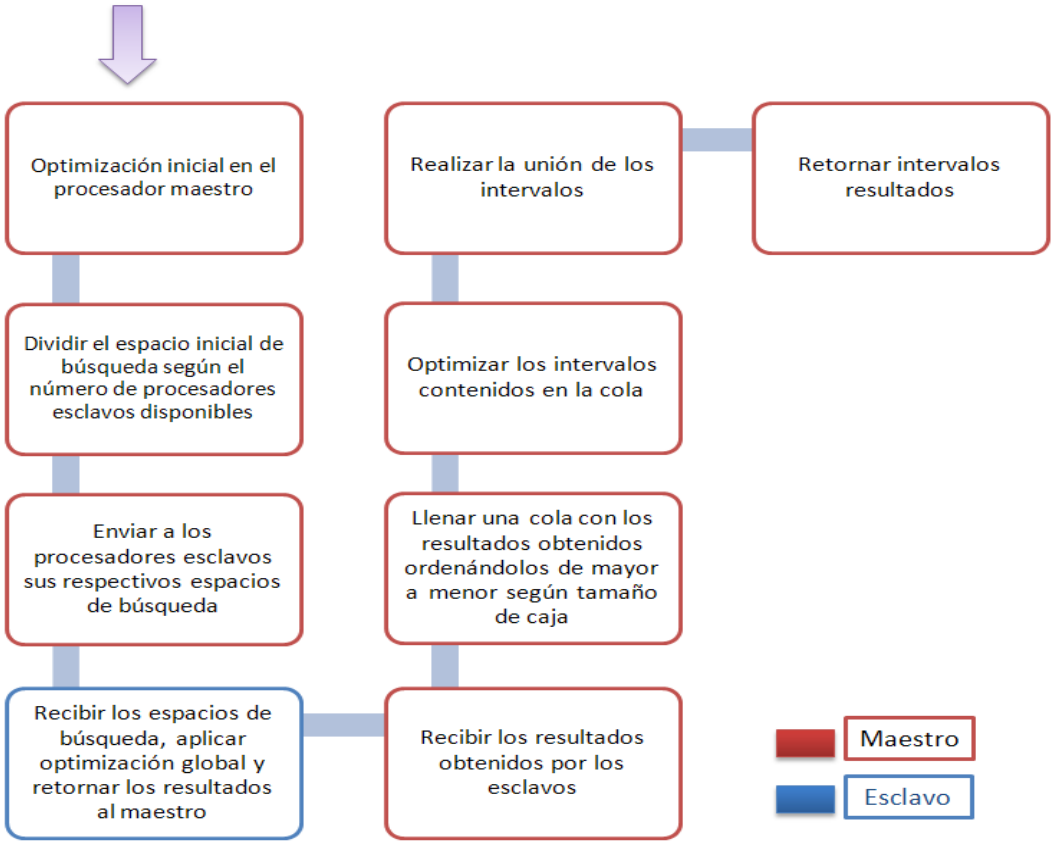


Figura 7.4 Algoritmo por división de caja resultado.

7.2 Nuevas Estrategias de Paralelización

Analizadas las estrategias vistas anteriormente se puede concluir que la estrategia de cajas resultado es más eficiente por el hecho que utiliza una optimización inicial obteniendo posibles cajas contenedores de óptimos globales, es decir, un dominio fidedigno, reduciendo así el gran problema de un amplio espacio de búsqueda inicial, en donde pueden ser enviados dominios a los procesadores esclavos que contengan regiones no óptimas, que puedan dar como resultado una mala utilización de los recursos disponibles (procesos y tiempo de ejecución).

Los distintos algoritmos que se expondrán a continuación fueron desarrollados con la estrategia de cajas resultados, teniendo aún así la opción de realizar o no una optimización inicial, dependiendo del número de iteraciones que sean ingresadas por el usuario, en caso de ser cero, el dominio será todo el espacio de búsqueda dado. Véase la figura 7.5

```
/*Cola donde guardaremos los resultados de la minimizaciÃ'n inicial y el tamano del universo*/
    IQUEUE cola;
    int total;
    total = ObtenerUniverso();

      if(iteraciones > 0) cola=globalMinMHSE(U,E,Y,lh,Ux,Jd,J,Jb,iteraciones);
       /*Si se hace una minimizacion inicial*/

      else cola.push(U);
       /*Sino ingresa directo a la cola*/
```

Figura 7.5 Extracto de código del archivo paralelismo.cpp común para todas las versiones de algoritmos.

Las nuevas estrategias de paralelización desarrolladas son las siguientes:

Caja Resultado con Distribución Equilibrada

Caja Resultado con Ranking Uf Final

Caja Resultado con Uf Compartido

Caja Resultado con Uf Compartido y Balanceo de Carga Dinámico enfocado a la Derecha

7.2.1 Caja Resultado con Distribución Equilibrada

El algoritmo CRDE es el siguiente:

Maestro:

1. Realiza una minimización global inicial
2. Calcula la cantidad de cajas para cada esclavo en forma dinámica
3. Envía las cajas a su esclavo correspondiente

Esclavo:

4. Recibe las cajas y aplica una minimización local
5. Envía las cajas con los óptimos locales al maestro

Maestro:

6. Recibe los óptimos locales y aplica una minimización global final
7. Realiza una selección final

A continuación se muestra un esquema del algoritmo CRDE en la figura 7.6.

Figura 7.6 Esquema del algoritmo 1: Caja Resultado con Distribución Equilibrada

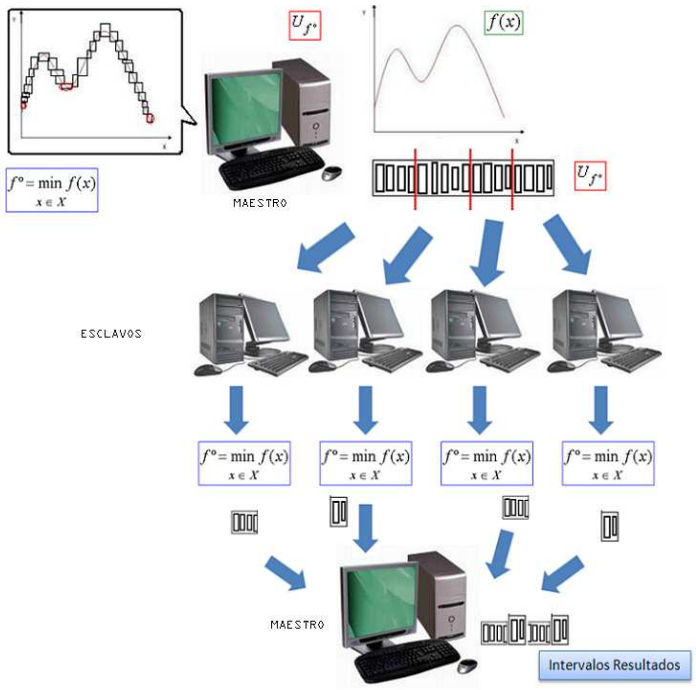


Figura 7.7 Esquema que describe el modelo utilizado de forma general en los algoritmos desarrollados: maestro - esclavo – maestro.

El algoritmo caja resultado con distribución equitativa tiene como principal característica la forma de distribuir las cajas a los esclavos. La función AsignaCantidadCajas es la encargada de esta tarea, en ella existen dos variables principales, la primera es esclavos_faltan que corresponde la cantidad de esclavos que van quedando, es decir, cada vez que se entrega la cantidad de cajas a un esclavo, esta variable disminuye. La segunda variable es el número de cajas o tamaño de la cola, el cual se obtiene directamente de la esta, entonces cada vez que se envían cajas a los esclavos el tamaño varía, esto sucede en la función armarArreglo. Cada vez que se utiliza la función AsignaCantidadCajas se tiene un número de esclavos diferente y un tamaño de cola distinto, lo que hace que la distribución de cajas por esclavos sea dinámica.

También es una característica importante del algoritmo que es el único, de los cuatro algoritmos en total, que al obtener todas las cajas resultados de parte de los esclavos aplica una optimización final en el maestro, seguido de una selección final en donde quedan solamente las cajas con los óptimos finales.

7.2.2 Caja Resultado con Ranking Uf Final

El algoritmo CRRUF es el siguiente:

Maestro:

1. Realiza una minimización global Inicial.
2. Comparte el menor límite superior encontrado (Uf) con todos los esclavos.
3. Calcula la cantidad de cajas para cada esclavo en forma dinámica.
4. Envía las cajas a su esclavo correspondiente.
5. Envía el limite (Uf) superior a cada procesador esclavo.

Esclavo:

6. Recibe el límite superior (Uf), las cajas y aplica una minimización local.
7. Envía el menor límite superior encontrado y las cajas con los óptimos locales al maestro.

Maestro:

1. Recibe los óptimos locales y los límites inferiores de cada esclavo.
2. Selecciona el menor límite superior (Uf) recibido y elige los óptimos globales utilizando el Uf seleccionado.

A continuación se muestra un esquema del algoritmo CRDE en la figura 7.8.

Figura 7.8 Esquema del algoritmo 2: Caja Resultado con Ranking Uf Final

El algoritmo resultado con ranking Uf final tiene interesantes características en relación con el algoritmo anterior, la primera se basa en que se comparte el menor límite superior encontrado (Uf) con todos los esclavos, esto significa que los esclavos ya tienen un avance significativo en relación al algoritmo caja resultado con distribución equitativa, porque no empiezan de cero. Obteniendo la ventaja de una base solida, como lo es un Uf menor, ahorrando así tiempo y esfuerzo. Al final de las optimizaciones locales en los esclavos se envía al maestro el Uf menor encontrado por cada esclavo, el maestro aplica una selección o ranking para encontrar el Uf menor y utiliza este en la selección final de las cajas.

Otra característica es la eliminación de la optimización final presente en el algoritmo caja resultado con distribución equitativa. Al momento de llegar a la optimización final, todas las cajas cumplen con la condición de término, por ende la optimización final no es necesaria, en vez de eso se modifico la función SeleccionFinalCajas.

```
void SeleccionFinalCajas(IQUEUE& cola,double Uf, double (*Jb)(interval_vector&,double[],int))
{

IQUEUE aux = cola;
double Ufi,F,e[1];
int p;
interval_vector Caja,FinalBox;

   /*Comenzamos a recorrer la cola guardando los UF */
     while(!aux.empty())
    {
      Caja = aux.front();
      aux.pop();
/*Evaluamos en la prueba del punto medio*/
      F=Jb(Caja,e,p);
      if(Ufi>F) Ufi = F;
    }
              double tres = Uf-Ufi;

    while(!cola.empty())
    {
    FinalBox = cola.front();
    cola.pop();
/*Evaluamos en la prueba del punto medio*/
    F=Jb(FinalBox,e,p);
    if(Uf-F==tres)
       aux.push(FinalBox);
    }
 cola = aux;
}
```

Figura 7.9 Función SeleccionarFinalCajas que se encuentra en el archivo funciones_utiles.cpp

La función recorre la cola y se extrae la primera caja o frontis, se evalúa en la prueba del punto medio, si el punto medio es menor que el Uf existente (que nace del ranking de Uf recibidos), pasa a ser el nuevo Uf, de esta forma encontramos el menor Uf existente. El siguiente paso es realizar una resta entre el Uf inicial y el valor resultado de la evaluación

con la prueba del punto medio, ahora el valor obtenido lo utilizaremos para encontrar las cajas de la forma que al momento de recorrer la cola se evalúa nuevamente con la prueba del punto medio, si el valor dado de esta prueba menos el Uf inicial dan el mismo resultado de la sustracción, se ingresa a la cola de las cajas resultado, siendo seleccionada como parte de la solución final. Véase figura 7.9.

7.2.3 Caja Resultado con Uf Compartido

El algoritmo CRRUF es el siguiente:

Maestro:

1. Realiza una Minimización Global Inicial.
2. Comparte el menor límite superior encontrado (Uf) con todos los Esclavos.
3. Calcula la cantidad de Cajas para cada Esclavo en forma dinámica.
4. Envía las Cajas a su Esclavo correspondiente y el límite (Uf) superior a cada procesador esclavo.

Ciclo: *El maestro espera Uf(s) de todos los esclavos y envía el Uf actualizado a todos los procesos esclavos.*

Esclavo:

5. Recibe las Cajas y el límite superior (Uf).
6. Aplica una Minimización Local con las cajas y el Uf integrados.

Ciclo: *Cada vez que se actualice Uf se envía al maestro para su procesamiento correspondiente.*

Esclavo:

7. Envía el Uf final, encontrado en la optimización local, al Maestro.
8. Envía las Cajas con los Óptimos Locales al Maestro.

Maestro:

9. Recibe los Óptimos Locales y los límites inferiores de cada Esclavo.

10. Selecciona el menor límite superior (Uf) recibido y elige los Óptimos Globales utilizando el Uf seleccionado.

11. Retorna intervalos resultados.

El algoritmo de caja resultado con Uf compartido toma la base del algoritmo resultado con ranking Uf final, esto quiere decir que el maestro envía un Uf inicial y además existe una selección final, ahora se suma a eso el proceso de actualización constante del límite superior. El procesador maestro se encuentra atento para recibir el nuevo Uf que ha sido enviado por un procesador esclavo, cada vez que el esclavo encuentra un nuevo Uf en su globalización local lo comparte con el maestro, en ese momento el maestro comparte el Uf recibido con todos los procesos esclavos. Al hacer esto se actualizan todos los Uf y cada esclavo continúa con su optimización local pero con el nuevo Uf recibido, este procedimiento es continuo hasta que los esclavos terminan de hacer las optimizaciones locales. Véase figura 7.10.

```
void ActualizarUf()
{
int j,total = ObtenerUniverso();
double aux,Uf;
MPI::Status status;

       if(MPI::COMM_WORLD.Iprobe(MPI_ANY_SOURCE,3,status))
      {
            int source = status.Get_source();
            MPI::Request recvRequest= MPI::COMM_WORLD.Irecv(&Uf, 1, MPI_DOUBLE,source,3);
            recvRequest.Wait();

            for(j=1;j<total;j++)
                MPI::COMM_WORLD.Isend(&Uf,1,MPI_DOUBLE,j,2);
      }
}
```

Figura 7.10 ActualizarUf se ubica en el archivo paralelismo.cpp del algoritmo caja resultado con Uf compartido, se aprecia el proceso de recibir un nuevo Uf que luego es enviado a los demás esclavos.

Figura 7.11 Esquema del algoritmo 3: Cajas Resultado con Uf Compartido

7.2.4 Caja Resultado con Uf Compartido y Balanceo de Carga Dinámico Enfocado a la Derecha

El algoritmo de caja resultado con Uf compartido y balanceo de carga dinámico enfocado a la derecha corresponde a la unión conceptual del algoritmo con Uf compartido y el de balanceo de carga dinámico de forma distribuida. Se mantiene una actualización constante del límite superior entre el maestro y los esclavos, en donde las tareas serán distribuidas en tiempo de ejecución, las cargas de procesadores cambiarán a medida que el algoritmo se ejecuta.

Debido a las características homogéneas del cluster, (similares características de los equipos y sistemas operativos en ellos), se puede distribuir las cargas de trabajo de forma equitativa entre los procesos esclavos, es decir, similar cantidad de trabajo se asigna a cada procesador de manera inicial. La forma de distribuir la carga de trabajo al momento que un esclavo se encuentre ocioso es por balanceo iniciado por recibimiento, un proceso con poca carga solicita trabajo a su vecino de la derecha.

El proceso de petición de carga en el algoritmo se explica a continuación:

1. ***Esclavo Ocioso*****:**

- Como no le quedan cajas le indica al maestro que termino su proceso de optimización global y se encuentra ocioso

2. **Maestro:**

- Pregunta si el vecino derecho del esclavo ocioso está muerto (refiérase al cese o término definitivo del proceso de optimización del esclavo).

  Si: el vecino derecho está muerto:

  2. a:

  - Avisa (mediante un mensaje al esclavo ocioso) que muera.

  Si: El vecino de la derecha del esclavo ocioso está vivo:

2. b:

- Envía un mensaje con petición de carga al esclavo derecho del esclavo ocioso.

3. Esclavo Derecho:

- Recibe el mensaje de petición de carga.
- Revisa si cumple con la restricción mínima de cajas para poder compartir la carga

Si cumple con la restricción:

3. a:

- Avisa mediante un mensaje al esclavo que se encuentra a su izquierda, indicándole que se prepare para recibir un vector con cajas.
- Divide sus cajas a la mitad y las envía al esclavo.

Si no cumple con la restricción:

3. b:

- Envía un mensaje a su vecino izquierdo que no puede compartir.

4. Esclavo Ocioso:

- Recibe el mensaje enviado por el Esclavo Derecho.

Si el mensaje es positivo:

4. a:

- Espera recibir el vector con cajas
- Transforma el vector en IQUEUE o cola.

- Extrae el ancho de la primera caja en la cola formada y lo utiliza como épsilon para seguir con el ciclo de optimización.

Si el mensaje es negativo:

4. b:

- El esclavo ocioso muere.

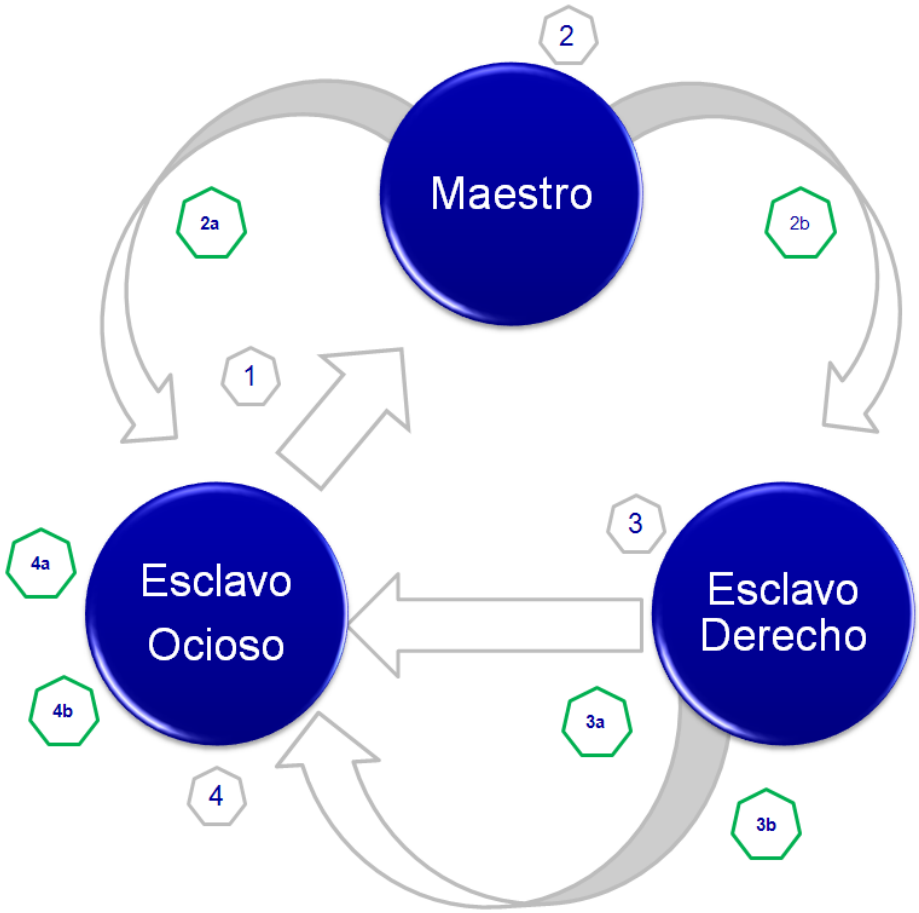


Figura 7.12 Esquematización del proceso de distribución de carga de trabajo en forma dinámica implementado en el Algoritmo de Caja Resultado con Uf Compartido y Balanceo de Carga Dinámico enfocado a la Derecha.

Figura 7.13 Esquema del algoritmo 4: Caja Resultado con Uf Compartido y Balanceo de Carga Dinámico enfocado a la Derecha

CAPÍTULO 8. RESULTADOS Y CONCLUSIONES OBTENIDAS AL PROCESO DE DEPURACIÓN Y PRUEBAS DE LOS ALGORITMOS EN PROBLEMAS CLÁSICOS.

Para evaluar el desempeño de los cuatro algoritmos desarrollados se utilizó el cluster de la universidad y como patrón de comparación se eligió a un conjunto de problemas clásicos.

8.1 Problemas Clásicos de Prueba

Problema: Beale 1

$$f(x) = (1.5 + x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$$

Óptimo Conocido

$$x^* = (3, 0.5)$$

Dominio : [-4.4,4.5][-4.5,4.5]

Problema: Booth

$$f(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$$

Óptimo Conocido

$$x^* = (1, 3)$$

Dominio : [-10,10][-10,10]

Problema: Matyas

$$f(x) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$$

Óptimo Conocido

$$x^* = (0, 0)$$

Dominio : [-10,10][-10,10]

Problema: Three Hump Camel Back

$$f(x) = 2x_1^2 - 1.05x_1^4 + \frac{1}{6}x_1^6 + x_1x_2 + x_2^2$$

Óptimo Conocido

$$x^* = (0,0)$$

Dominio : [-5,5][-5.5]

Problema: Six Hump Camel Back

$$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{x_1^6}{3} + x_1x_2 - 4x_2^2 + 4x_2^4$$

Óptimo Conocido

$$x_1^* = \begin{pmatrix} 0.0898400 \\ -0.712659 \end{pmatrix}, \qquad x_2^* = \begin{pmatrix} -0.898400 \\ 0.712659 \end{pmatrix}$$

Dominio : [-5,5][-5.5]

Problema: Treccani

$$f(x) = x_1^4 + 4x_1^3 + 4x_1^2 + x_2^2$$

Óptimo Conocido

$$x_1^* = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \qquad x_2^* = \begin{pmatrix} -2 \\ 0 \end{pmatrix}$$

Dominio : [-5,5][-5.5]

Problema: Goldstein-Price

$$f(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 - 6x_1x_2 + 3x_2^2)] \times [20 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$$

Óptimo Conocido

$$x^* = \begin{pmatrix} 0 \\ -1 \end{pmatrix}$$

Dominio : [-2,2][-2.2]

8.2 Pruebas

Los problemas clásicos, que se obtienen de la literatura, fueron usados para evaluar los siguientes algoritmos implementados:

- Caja Resultado con Distribución Equitativa
- Caja Resultado con Ranking Uf Final
- Caja Resultado con Uf Compartido
- Caja Resultado con Uf Compartido y Balanceo de Carga Dinámico enfocado a la Derecha

La base de las pruebas esta en un universo total superior a 3000. La estructura de las pruebas se basa en la consideración de diferentes anchos de caja (épsilon), que sirven de condición de término para la optimización global. Estos épsilon son 0.01 ,0.0001 y 0.000001.

Se considero distintas iteraciones que corresponden a la cantidad de veces que el algoritmo de optimización global se ejecutará en el procesador maestro, antes de ser enviado a sus correspondientes procesadores esclavos, estas iteraciones son 10, 50, 100, 500. Las diferentes combinación formadas entre las iteraciones y los épsilon se toman de base para evaluar las distintos algoritmos implementados, sobre el cluster de alto rendimiento (HPC) disponible en la facultad de ciencias empresariales (FACE) de la universidad del Bio-Bío. Este cluster cuenta con 6 nodos activos lo cual permito realizar pruebas sobre dos hasta seis procesos de forma paralela.

Adicionalmente a las pruebas mencionadas anteriormente, se realizaron evaluaciones sobre el algoritmo caja resultado con Uf compartido y el algoritmo caja resultado con Uf compartido y balanceo de carga dinámico enfocado a la derecha, modificando la cantidad de veces que el límite superior Uf es actualizado en tiempo de proceso.

Los resultados obtenidos son en base al promedio de cinco intentos por proceso.

Los resultados obtenidos son representados gráficamente, mostrando la interacción de todos los algoritmos antes mencionados. Para mayor detalle consultar Anexo U.

8.2.1 Pruebas función Six_Hump_Camel_Back

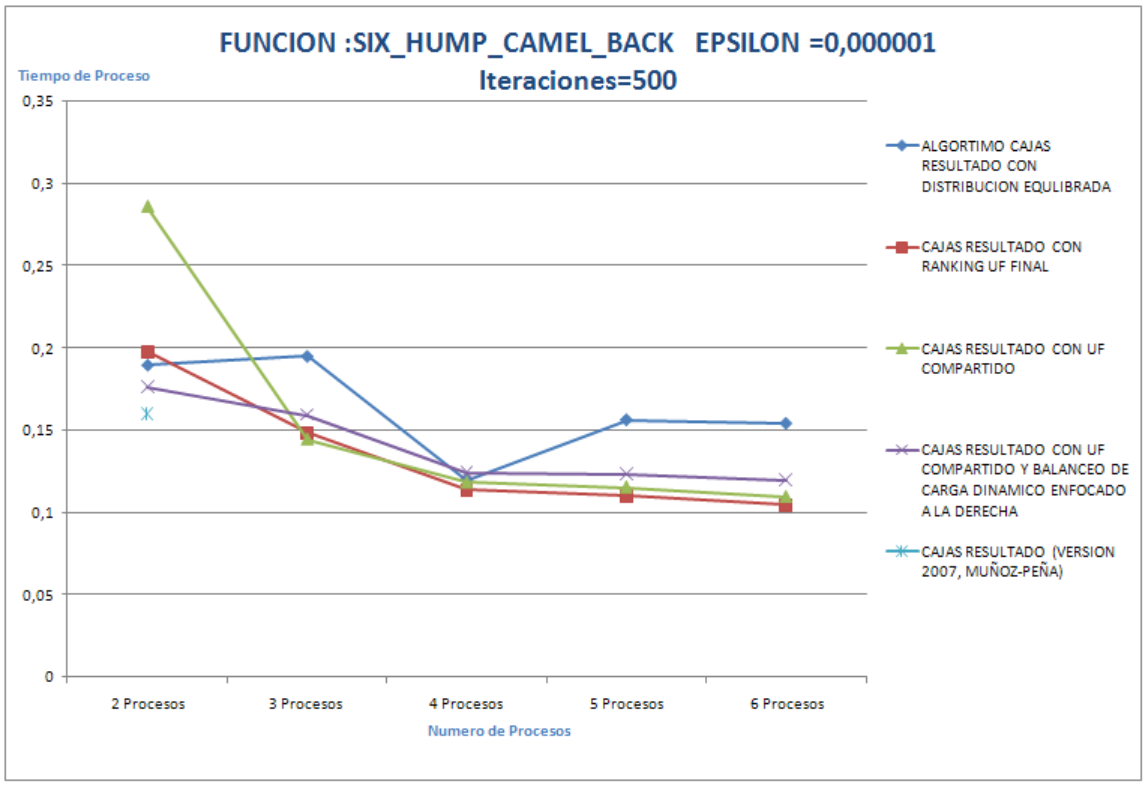


Figura 8.1 Funcion Six_Hump_Camel_Back con épsilon=0,000001 y 500 iteraciones.

Se puede observar en la figura 8.1 un mejor desempeño del algoritmo cajas resultado con ranking Uf final no siendo muy estrecha la diferencia en relación al algoritmo cajas resultado con Uf compartido y cajas resultado con uf compartido y balanceo de carga enfocado a la derecha. Cabe señalar que el algoritmo cajas resultado (2007, Muñoz-Peña) solo logro correr con dos procesos.

Es interesante señalar que al evaluar la función con cuatro procesos los resultados obtenidos por los distintos algoritmos son muy similares.

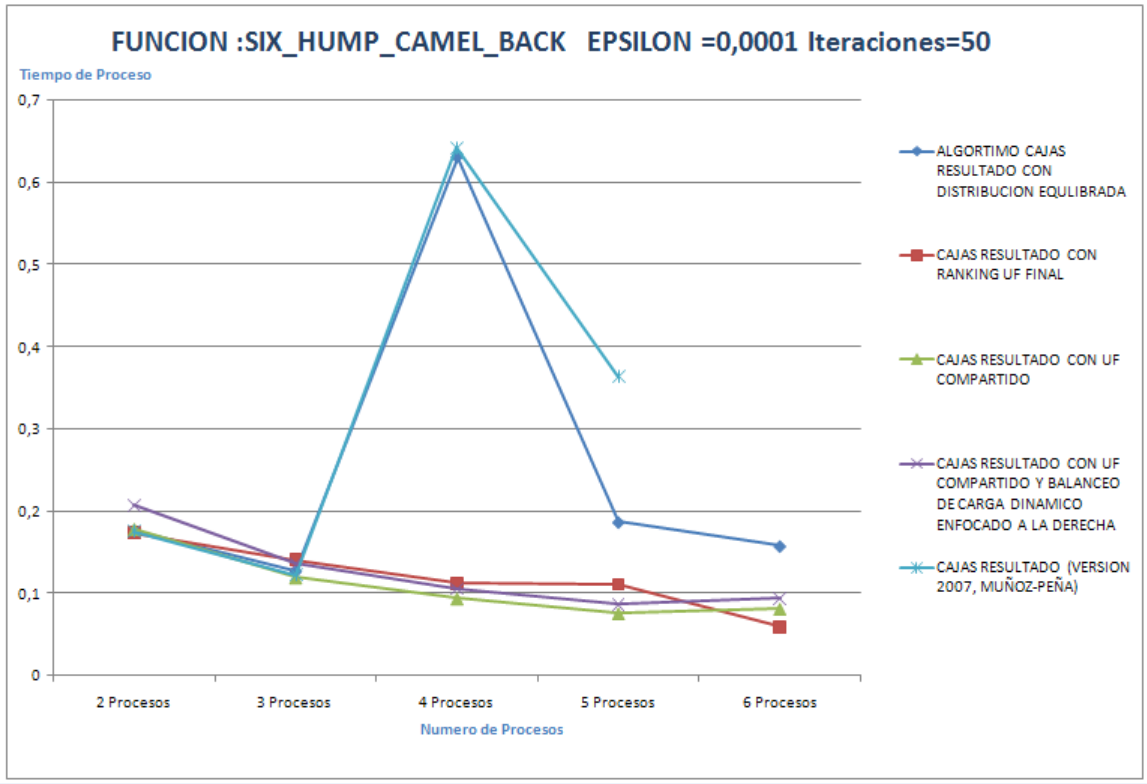


Figura 8.2 Funcion Six_Hump_Camel_Back con épsilon=0,0001 y 50 iteraciones.

La figura 8.2 muestra que en la evaluación en la función permanece similar para todos los algoritmos hasta el proceso tres, pero al evaluarla con 4 procesos se ve un notable incremento en los tiempos para los algoritmos cajas resultado con distribución equilibrada y cajas resultado (2007.Muñoz-Peña), reduciendo de forma significativa su tiempo en el proceso cinco, no alcanzando el rendimiento similar que obtienen los algoritmos restantes. Cabe señalar que el algoritmo cajas resultado (2007, Muñoz-Peña), no funciona con 6 procesos.

Como conclusión general de la función se puede observar que las dos figuras analizadas los algoritmos de menor rendimiento son el de cajas resultado con distribución equilibrada y cajas resultado (2007.Muñoz-Peña). Con los datos obtenidos no se puede concluir de forma certera que algoritmo de los restantes tienen mejor desempeño, ya que las variaciones presentadas son demasiado pequeñas y poco concluyentes.

8.2.2 Pruebas función Beale 1

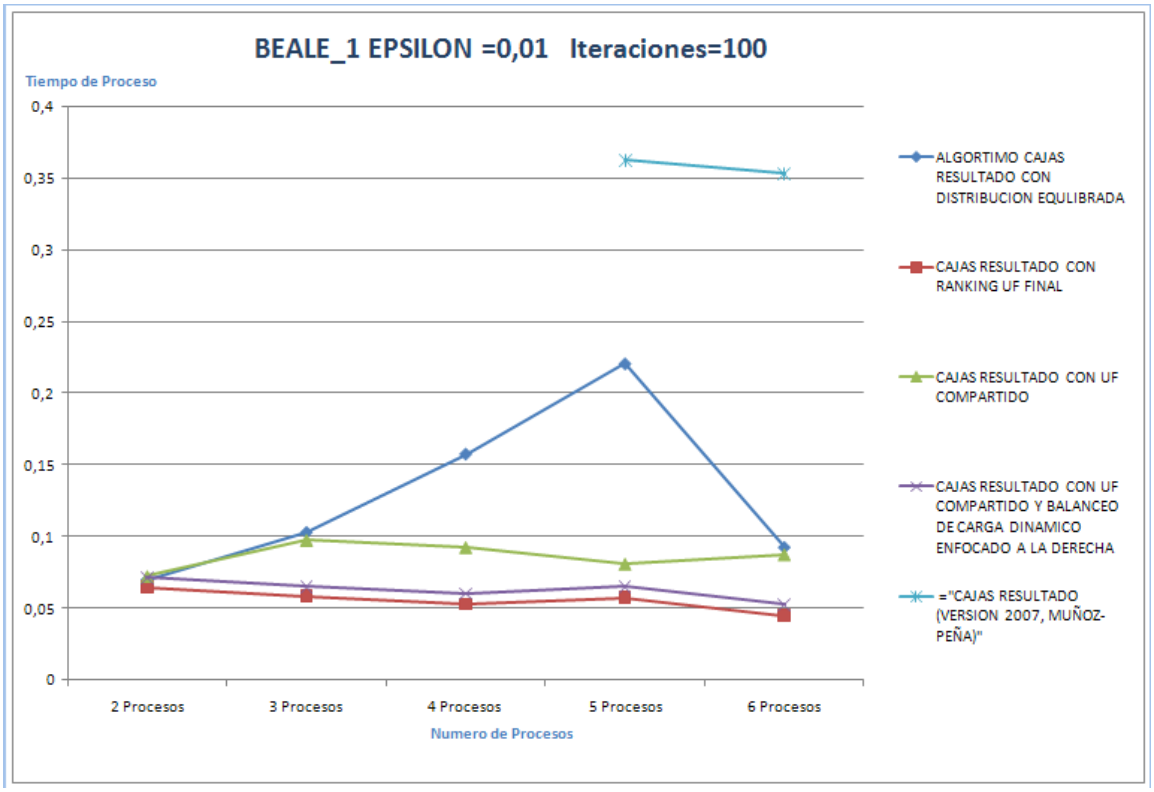


Figura 8.3 Función Beale_1 con épsilon=0,01 y 100 iteraciones.

La figura 8.3 muestra un rendimiento similar en las funciones cajas resultado con ranking Uf final, también en cajas resultado con Uf compartido y balanceo de carga enfocado a la derecha, siendo estos los mejores resultados evaluados con distintos procesos en ejecución. Se observa que el algoritmo cajas resultado (2007, Muñoz-Peña) tiene un desempeño muy por debajo en comparación con los demás algoritmos. El algoritmo cajas resultado con distribución equilibrada incrementa su tiempo a medida que se aumenta el numero de procesos, siendo el quinto proceso causante de su peor promedio.

Observamos de igual manera un comportamiento singular de parte del algoritmo cajas resultado con Uf compartido en relación a los demás algoritmos, por que en el quinto proceso su promedio de tiempo disminuye al contrario de los otros, en donde los tiempos incrementan.

Se puede concluir que los mejores algoritmos para esta función son los de cajas resultado con ranking Uf final, junto con el de cajas resultado con Uf compartido y balanceo de carga enfocado a la derecha, por su comportamiento regular y sobresaliente durante las pruebas a esta función.

8.2.3 Pruebas función Booth

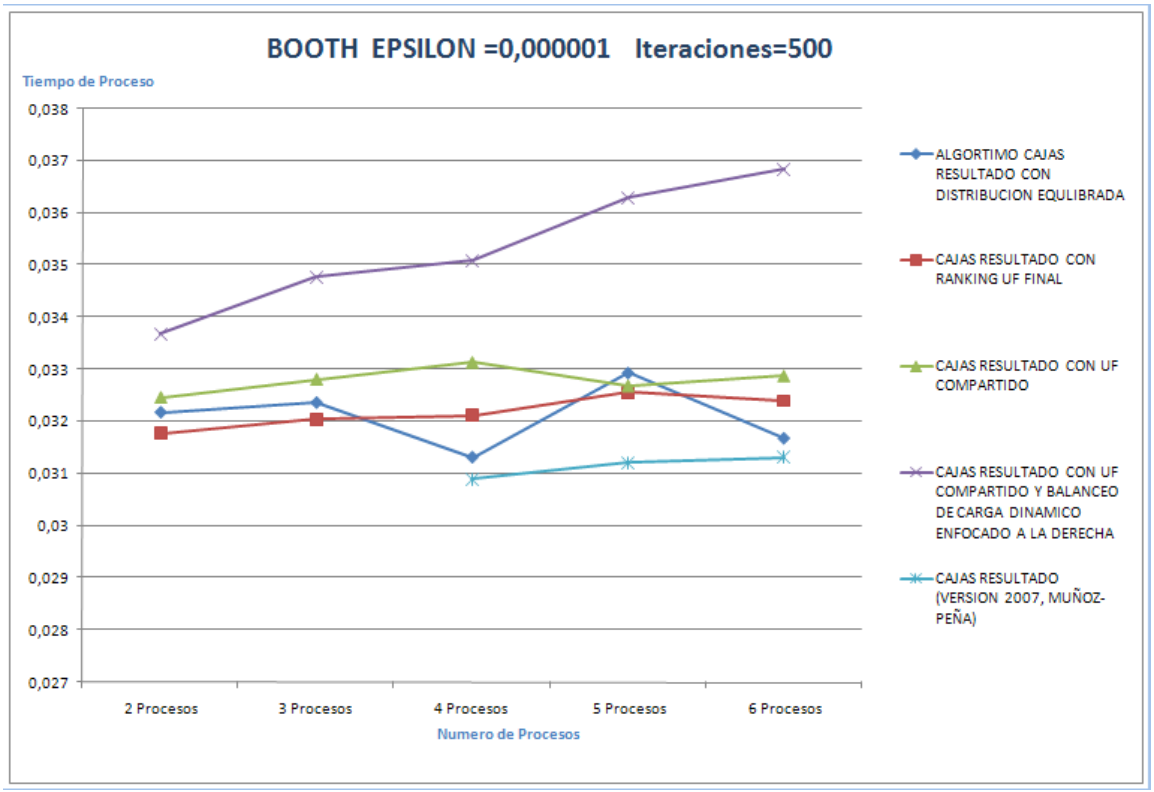


Figura 8.4 Función Booth con épsilon=0,000001 y 500 iteraciones

Obsérvese la figura 8.4 se observa un pésimo desempeño del algoritmo cajas resultado con Uf compartido y balanceo de carga enfocado a la derecha que se muestra con una notable diferencia en relación a los tiempos de ejecución de los procesos restantes. El algoritmo cajas resultado con ranking Uf final muestra un comportamiento lineal, con muy pocas variaciones, lo que lo hace el mas estable. Por otro lado la sorpresa observada es el algoritmo caja resultado con distribución equilibrada, porque obtiene tiempos envidiables. Cabe señalar que el algoritmo cajas resultado (2007, Muñoz-Peña) solo comenzó a trabajar

en el proceso cuatro hacia delante, mostrando excelentes tiempos en el corto segmento del gráfico que aparece.

Como conclusión final se puede señalar que la función Booth no necesita mayor poder de procesamiento paralelo, en relación ha las funciones vistas anteriormente, los algoritmos mas sencillos son los que se comportan e mejor forma al evaluar la función, siendo los de mejor comportamiento el algoritmo de cajas compartida con ranking Uf final y el de cajas resultado con distribución equilibrada, al contrario del algoritmo de cajas resultado con Uf compartido y balanceo de carga dinámico enfocado a la derecha. En este caso particular la función no aprovecha de buena forma el balanceo de carga y la actualización del Uf, por ende el tiempo de ejecución obtenido es mayor.

8.2.4 Pruebas función Matyas

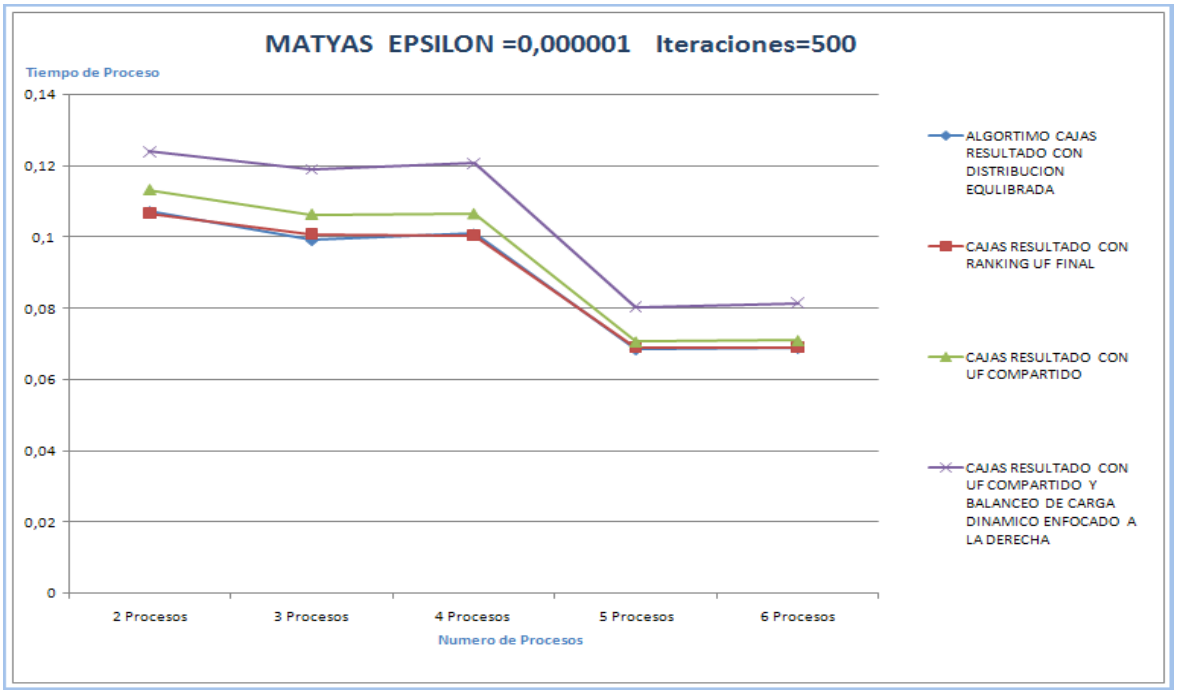


Figura 8.5 Función Maytas con épsilon=0,000001 y 500 iteraciones

En la figura 8.5 solo se utilizan cuatro algoritmos para las pruebas, debido que el algoritmo caja resultado (2007, Muñoz-Peña) no fue posible ejecutarlo para esta función en particular.

Se observa el similar comportamiento de todos los algoritmos en esta función, especialmente en los tiempos de los algoritmos de cajas resultado con distribución equilibrada y caja de resultado con ranking de Uf final, siendo estos dos los que poseen el mejor desempeño en el escenario dado.

8.2.5 Pruebas función Three Hump Camel Back

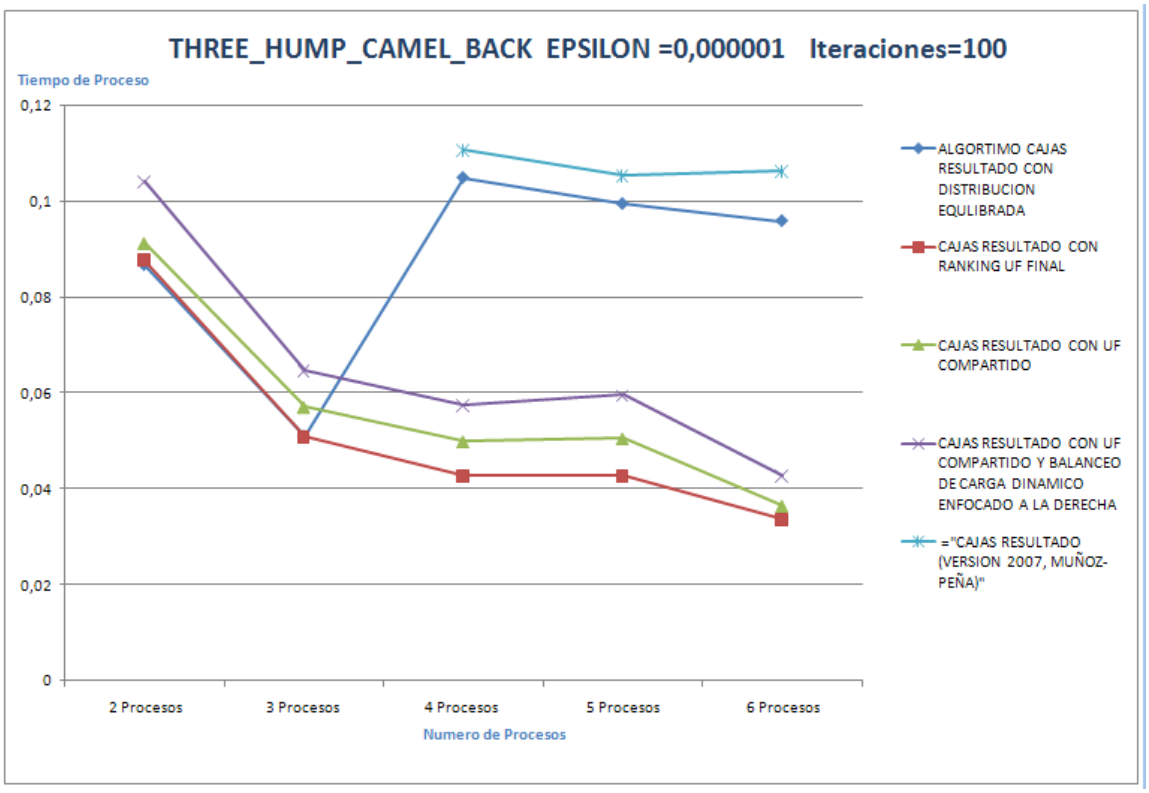


Figura 8.6 Función Three Hump Camel Back con épsilon = 0.000001 y iteraciones = 100.

Como se observa en la figura 8.6 cuatro de los cinco algoritmos inician de manera similar, produciéndose en el proceso tres un gran aumento en el tiempo del algoritmo cajas resultado con distribución equilibrada que sigue manteniendo un elevado tiempo junto con el algoritmo caja resultado (2007, Muñoz - Peña) que registra el peor tiempo. En esta prueba podemos concluir sin lugar a dudas el que posee mejor desempeño es el algoritmo caja resultado con ranking Uf final.

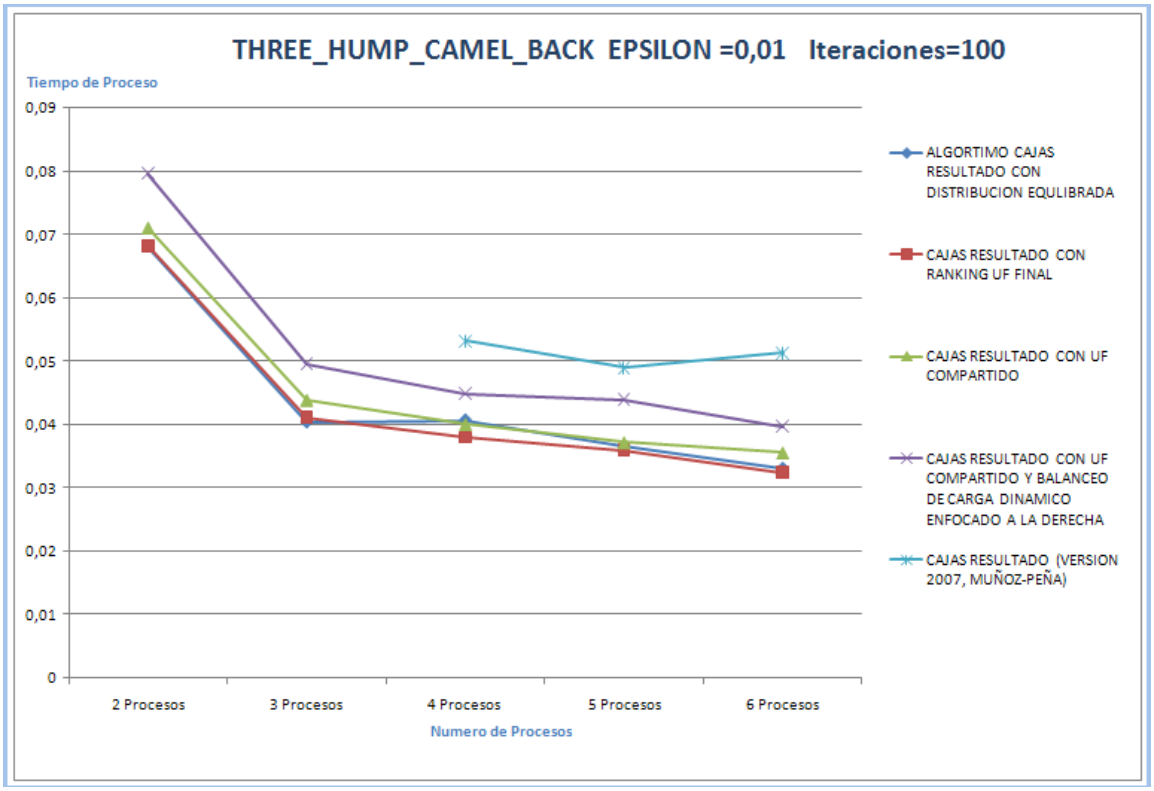


Figura 8.7 Función Three Hump Camel Back con epsilon = 0,01 y iteraciones = 100.

La misma ventaja se observa en la figura 8.7 de parte del algoritmo cajas resultado con ranking Uf final, pero la gran diferencia observada es el buen comportamiento que muestra el algoritmo cajas resultado con distribución equilibrada, esto se atribuye a la baja precisión (0,01) de la prueba vs la gran precisión observada en la figura anterior (0.000001). Nótese que el algoritmo cajas resultado (2007, Muñoz-Peña), no es capaz de ejecutar la prueba hasta los tres procesos simultáneos.

Obedeciendo a lo que muestra la figura y el comportamiento constante del algoritmo durante las pruebas en la función Three Hump Camel Back, se puede concluir que el algoritmo cajas resultado con ranking Uf final presenta el mejor comportamiento y el mejor desempeño total a lo largo de las pruebas realizadas.

8.2.6 Pruebas función Treccani

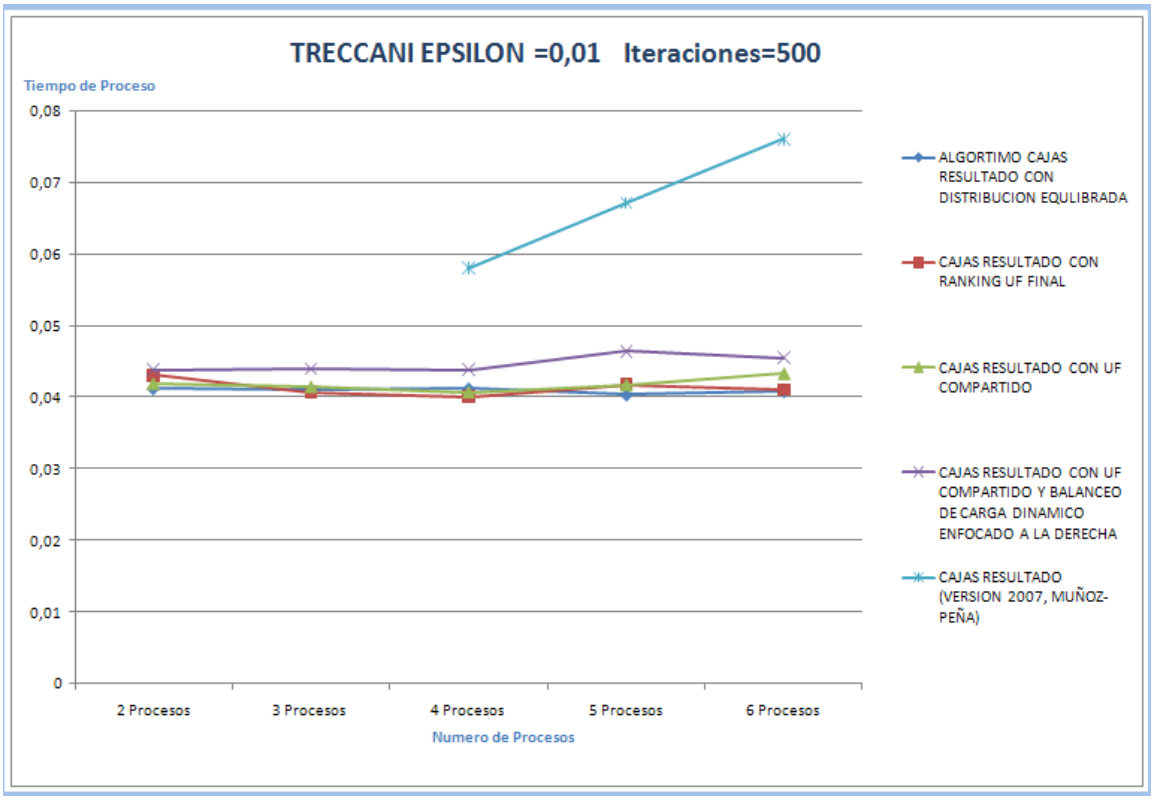


Figura 8.8 Función Treccani con epsilon = 0,01 y iteraciones = 500.

Obsérvese la figura 8.8 en donde se aprecia una gran diferencia entre el algoritmo cajas resultado (2007, Muñoz-Peña), que presenta los peores tiempos, y los demás algoritmos que presentan un desempeño muy similar con distintos procesos en ejecución.

En esta función no se puede concluir con certeza cual de los algoritmos es mejor, debido al similar comportamiento que presentan y la poca diferencia es sus tiempos de respuesta con distintos procesos corriendo.

8.2.7 Pruebas función Goldstein Price

Se puede observar en la figura 8.9 un similar comportamiento del algoritmo cajas resultado con ranking Uf final y cajas resultado con Uf compartido, también se identifica que con tres procesos los distintos algoritmos tienen aproximadamente el mismo resultado.

Existe una diferencia significativa desde cuatro procesos en adelante entre el algoritmo cajas resultado con distribución equilibrada y los demás algoritmos, no siendo relevante la participación del algoritmo cajas resultado (2007, Muñoz-Peña) debido a su única aparición con cuatro procesos.

Claramente en la figura 8.9 se distingue el predominio del algoritmo cajas resultado con Uf compartido y balanceo de carga enfocado a la derecha, observándose una diferencia cercana a los dos segundos en relación al algoritmo cajas resultado ranking Uf final y cajas resultado Uf compartido.

Al momento de evaluar algoritmo cajas resultado con Uf compartido y balanceo de carga enfocado a la derecha con cinco procesos se obtiene una diferencia que asciende a 33 segundos aproximadamente en relación al resultado obtenido por el algoritmo cajas resultado con distribución equilibrada.

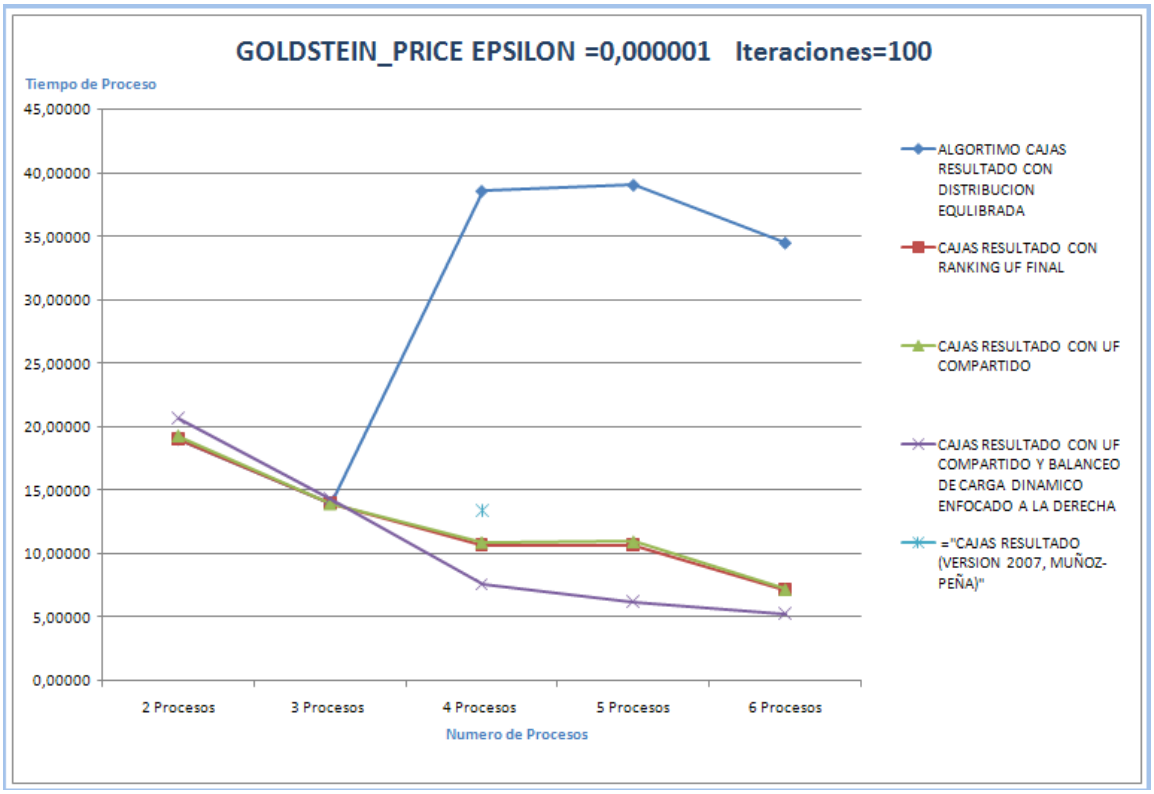


Figura 8.9 Función Goldstein_Price con epsilon = 0,000001 y iteraciones = 100.

Se puede concluir que mientras más carga computacional tenga la evaluación de la función mejor será la respuesta del algoritmo cajas resultado con Uf compartido y balanceo de carga enfocado a la derecha, debido a que esta función presenta las condiciones mas apropiadas para la utilización del balanceo de carga.

8.2.8 Pruebas de actualización de límite superior (Uf)

8.2.8.1 Pruebas de frecuencias de actualización del límite superior (Uf) para algoritmo cajas resultado Uf compartido.

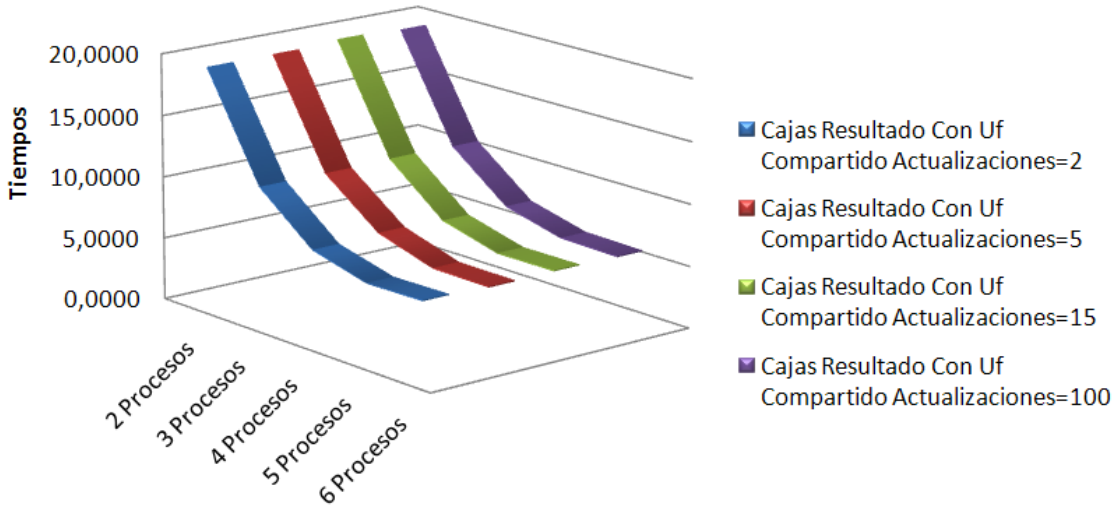


Figura 8.10 Algoritmo cajas resultado con Uf compartido evaluado con distintas actualización del límite superior (Uf) en la función Goldstein Price.

Al variar la frecuencia en que es actualizado el valor del límite superior, la variación en los tiempos de ejecución es mínima. Véase la figura 8.10.

Los tiempos son muy similares ya sea con 2, 5, 15, 100 veces entre actualización, por ende se puede concluir que el cambio de frecuencia en la actualización del Uf es poco relevante en lo que a tiempos de ejecución se refiere.

8.2.8.2 Pruebas de frecuencias de actualización del límite superior (Uf) para algoritmo cajas resultado Uf compartido y balanceo de carga dinámico enfocado a la derecha.

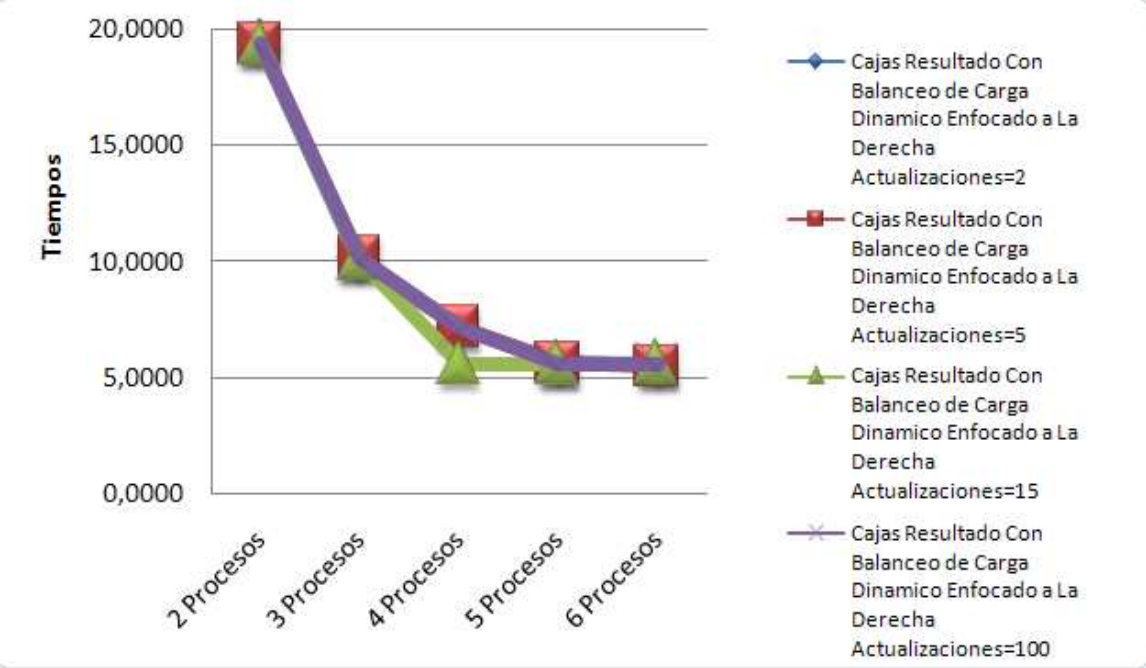


Figura 8.11 Algoritmo cajas resultado Uf compartido y balanceo de carga dinámico enfocado a la derecha con distintas actualización del límite superior (Uf).

Al variar la frecuencia en que es actualizado el valor del límite superior, no se presentan variaciones significativas en su tiempo ejecución, a excepción de cuando se actualiza el limite superior (Uf) cada quince veces sobre cuatro procesos en ejecución. De forma extraordinaria algoritmo cajas resultado Uf compartido y balanceo de carga dinámico enfocado a la derecha disminuye los tiempos considerablemente.

8.3 Conclusiones Pruebas Realizadas

Tras haber realizado una cantidad un sinnúmero de pruebas de los distintos algoritmos implementados con los distintos problemas se llega a las siguientes conclusiones:

1- Cada Algoritmo tiene sus ventajas y desventajas y no se puede afirmar con certeza que algoritmo es mejor o peor, ya que depende de cual es el problema

paralelo a tratar. Dependiendo de los distintos factores los algoritmos funcionaran de distintas maneras.

2- Podemos afirmar que para problemas que requieren de mayor poder de procesamiento, el algoritmo cajas resultado con Uf compartido y balanceo de carga dinámico enfocado a la derecha, se desempeña de mejor forma que los distintos algoritmos restantes, superando con creces los otros resultados. Desempeñándose de mejor forma mientras mayor sea el número de procesos, ya que compensa el gasto de comunicación invertida por este algoritmo, por el beneficio que da una distribución dinámica de tareas.

3- Se puede afirmar conforme a las pruebas realizadas que para problemas de menor procesamiento paralelo es más recomendable el algoritmo de cajas resultado con ranking Uf final, el cual tiene la ventaja de eliminar la optimización final presente en el algoritmo de cajas resultado con distribución equilibrada.

CONCLUSIONES

Hemos vistos conceptos y teorías que nos permitieron lograr obtener y formular una estrategia para mejorar el funcionamiento de una plataforma de optimización global que incluye una librería de aritmética de intervalos. Se mostraron conceptos nuevos, tales como el manejo de una aritmética intervalar, que sirve para poder eliminar el error de redondeo y truncamiento que hacen los distintos procesadores. Revisamos conceptos de optimización comprendiendo sus pasos fundamentales y distintas pruebas de descarte para comprender mejor su funcionamiento, se analizaron distintas estrategias de paralelización, arquitecturas de programación paralela, y todos los conceptos relativos a ellas revisando conceptos nuevos para nosotros. Se revisaron también conceptos como cluster, que es un grupo de computadores trabajando como uno solo, que nos permite, usando programación paralela con memoria distribuida, alcanzar performance cercanas a un supercomputador con menos costos económicamente hablando, ya que estos supercomputadores no están al alcance de todas las personas. También se concluye que el paso de mensajes es la mejor opción en el caso de usar la arquitectura de programación paralela antes mencionada, siendo MPI la herramienta estudiada y las que nos permitió una gestión óptima de los mensajes que intercambien los distintos computadores, además de considerar la arquitectura MIMD de computadores definida por Flynn. Se puede concluir que hay que tener bien claro como es el algoritmo, en cómo se manejan las tareas, cómo se comunican y cómo se pueden agrupar.

Como el cálculo de óptimos globales consume muchos recursos de procesamiento y es de alta importancia para distinta áreas, es relevante buscar estrategias para disminuir el recurso tiempo sin utilizar un computador de mayor prestación, que signifique un mayor valor económico.

El proceso de desarrollo y implementación del algoritmo permitió ampliar los conocimientos de las herramientas y técnicas ocupadas para resolver los distintos problemas dejando un marco teórico y practico muy amplio.

La reconfiguración del cluster de la Universidad del Bio-Bío fue una experiencia educadora y enriquecedora que nos permitió aprender en forma practica el concepto de cluster de computadores.

El Proceso de pruebas aporto con valiosa información sobre la variabilidad de los distintos algoritmos y el comportamiento de cada uno comprendiendo mejor su funcionamiento y sus tendencias en cuanto se iban cambiando los valores de éstas.

Cada Algoritmo tiene sus ventajas y desventajas y no se puede afirmar con certeza que algoritmo es mejor o peor, ya que depende de cual es el problema paralelo a tratar. Dependiendo de los distintos factores los algoritmos funcionaran de distintas maneras.

Un buen esquema de comunicación es fundamental para asegurar un óptimo funcionamiento al momento de desarrollar e implementar aplicaciones de forma paralela en general mas aun si utilizamos carga de trabajo.

Todo lo mencionado anteriormente nos hace cumplir todos los objetivos, tanto general como específicos adquiridos al inicio de la habilitación profesional. Los conocimientos que se adquirieron en el transcurso de esta investigación nos han servido para poder decir que una combinación de aplicaciones de altas prestaciones, cómo es el caso de una biblioteca de optimización global por intervalos, con una adecuada estrategia de paralelización y un adecuado esquema de comunicación, nos permite abrir nuevas expectativas para estudios posteriores con otras aplicaciones de altas prestaciones, además de aplicar nuevas técnicas de distribución paralela, que es un marco muy interesante de investigación.

ANEXOS

Anexo A: Terminologías Generales de Paralelismo

Algunos de los términos más comúnmente usados en computación paralela:

Tarea (Task)

Una sección lógicamente discreta de trabajo computacional. Una tarea es un programa o un conjunto de instrucciones que es ejecutada por un procesador.

Tarea Paralela (Parallel Task)

Una tarea que puede ser ejecutada por Multiples procesadores.

Ejecución Serial (Serial Execution)

La ejecución de un programa secuencialmente. Sin embargo, las tareas paralelas tendrán secciones de un programa paralelo que deben ser ejecutadas serialmente.

Ejecución Paralela (Parallel Execution)

La ejecución de un programa por diferentes tareas, donde cada tarea es capaz de ejecutar la misma o diferente instrucción al mismo instante de tiempo.

Memoria Compartida (Shared Memory)

Describe una arquitectura computacional donde todos los procesadores tienen acceso directo a una memoria física común. Desde el punto de vista de programación, describe un modelo donde las tareas paralelas pueden directamente direccionar y acceder a las mismas posiciones lógicas de memoria donde la memoria física reside.

Memoria Distribuida (Distributed Memory)

Se refiere a acceder a través de un canal de comunicación a memoria física que no es común entre diferentes procesadores. Desde el punto de vista de programación, las tareas pueden lógicamente ver las memorias locales de otras máquinas y deben usar comunicaciones para acceder a la memoria de otras máquinas donde las tareas se están ejecutando.

Comunicaciones (Communications)

Las tareas paralelas necesitan intercambiar datos y existen diferentes maneras para llevar a cabo esto, bien sea a través de un bus de memoria compartido sobre una red, sin embargo el evento verdadero de intercambio de datos es comúnmente referenciado como consideración de las comunicaciones del método empleado.

Sincronización (Synchronization)

La coordinación de las tareas paralelas está asociada con la comunicación. A menudo se establecen puntos de sincronización dentro de una aplicación donde una tarea puede no proceder hasta que otra(s) tarea(s) alcanzan el mismo lugar (punto lógicamente equivalente).

Usualmente la sincronización involucra la espera de al menos una tarea, y puede implicar un aumento del tiempo de ejecución de la aplicación paralela.

Granularidad (Granularity)

En computación paralela, la granularidad es una medida equivalente de la razón cómputo-comunicación.

- **Gruesa (Coarse):** relativamente grandes cantidades de trabajo computacional son hechas entre eventos de comunicación.

- **Fina (Fine):** relativamente pequeñas cantidades de trabajo computacional son hechas entre eventos de comunicación.

Carga Paralela (Parallel Overhead)

La cantidad de tiempo requerido para coordinar las tareas paralelas. La carga puede incluir factores tales como:

• Tiempo para arrancar las tareas

• Sincronizaciones

• Comunicación de datos

• Compiladores paralelos, bibliotecas, SO, etc.

• Tiempo de finalización de las tareas

Masivamente Paralelo (Massively Parallel)

Hace referencia al hardware que compone un sistema paralelo dado el cual posee muchos procesadores (más de 1000).

Scalability

Hace referencia a la capacidad de un sistema paralelo (hardware y/o software) el cual demuestra un aumento proporcionado en la eficiencia paralela con la inclusión de más procesadores. Los factores que contribuyen a la escalabilidad:

• Hardware – particularmente ancho de banda y red de comunicación

• El algoritmo de aplicación

• La codificación paralela

• Características de la aplicación específica (Guillén, 2006)

Anexo B: Arquitectura de Memoria Para Computadores Paralelos

Para hablar de computador paralelo se debe tener algún tipo de diseño o arquitectura. Se puede decir que existen dos tipos, los Multiprocesadores con Memoria Compartida y los Multicomputadores con memoria distribuida. También se puede mencionar un tercer tipo, que es el caso de un hibrido de los las dos arquitecturas mencionadas anteriormente.

Sistemas Multiprocesador Con Memoria Compartida

Son sistemas altamente acoplados: cuando se envía un mensaje el retardo es corto y la tasa es alta. Comunes en sistemas paralelos. Es un área de almacenamiento de datos accesible por todos los procesadores, es decir que Multiples procesadores trabajan de forma independiente, pero accediendo al mismo espacio físico de memoria, por lo que algún cambio realizado será visto por todos los procesadores. En general, esta técnica utiliza una gran cantidad de hardware y no es efectiva para un número grande de procesadores. Las ventajas son la facilidad de programación, la existencia de muchas librerías para su programación y la gran potencia que presentan normalmente los procesadores individuales. Se ve en la figura A1. (Rodríguez, 2004)

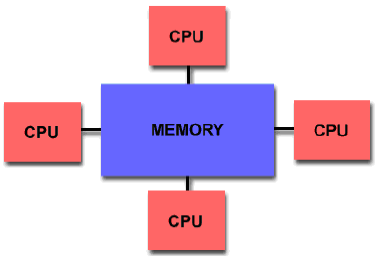


Figura A1 Esquema para un sistema con memoria compartida (Barney, 2007)

Dependiendo de la forma en que los procesadores comparten la memoria, basados en los tiempos de acceso a ésta, se puede hacer una subdivisión de los multiprocesadores. En las maquinas UMA (Uniform Memory Access, Acceso Uniforme a la Memoria) la memoria física esta uniformemente compartida por todos los procesadores, como se ve en la figura A2. Éste quiere decir que todos los procesadores tienen el mismo tiempo de acceso a todas

las palabras de memoria. Cada procesador puede tener su cache privada, y los periféricos son también compartidos de alguna manera.

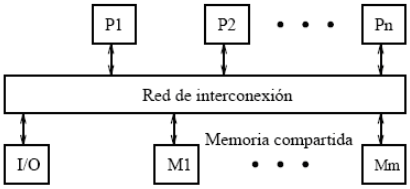


Figura A2 Modelo UMA de multiprocesador (Flores, 2004)

Los que no tienen los mismos tiempos de acceso a memoria se llaman NUMA (Not Uniform Memory Access, Acceso no uniforme a la memoria). Es un sistema de memoria compartida donde el tiempo de acceso varia según el lugar donde se encuentre localizado el acceso. Lo muestra la figura A3.

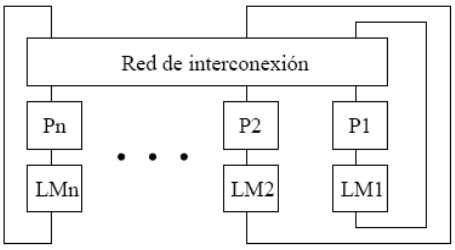


Figura A3 Modelo NUMA de multiprocesador (Flores, 2004)

También se puede nombrar al COMA (Cache Only Memory Access). Un multiprocesador que solo usa caché como memoria es considerado de tipo COMA. La figura A.4 muestra el modelo COMA de multiprocesador. En realidad, el modelo COMA es un caso especial del NUMA donde las memorias distribuidas se convierten en caché. No hay jerarquía de memoria en cada módulo procesador. Todas las cachés forman un mismo espacio global de direcciones. El acceso a las cachés remotas se realiza a través de los directorios distribuidos de las cachés. Dependiendo de la red de interconexión empleada, se pueden utilizar jerarquías en los directorios para ayudar en la localización de copias de bloques de caché. El emplazamiento inicial de datos no es crítico puesto que el dato acabará estando en el lugar en que se use más. (Flores, 2004)

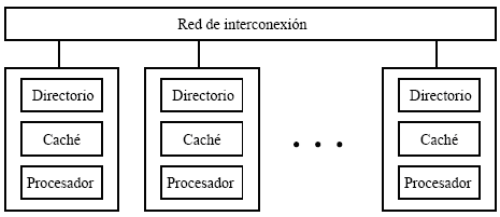


Figura A4 Modelo COMA de multiprocesador (Flores, 2004)

La utilización del sistema Multiprocesador trae consigo ventajas como las que el espacio global de direcciones provee una amigable programación, respecto de la memoria. Además de que el compartimiento de datos entre tareas es rápido y uniforme debido a la proximidad de la memoria con la CPU. La principal desventaja que presenta es la carencia de estabilidad entre memoria y CPU. Al añadir más CPU puede incrementar geométricamente su tráfico, sobre el canal de comunicación, compartición, Memoria-CPU. Por otra parte la responsabilidad del programador es mayor, debiendo asegurar el correcto acceso por parte de cada unidad de procesamiento a la memoria global, cada vez es más dificultoso y caro el diseñar y producir maquinas con memoria compartida con números cada vez mayores de procesadores. (Barney, 2007)

Sistema Multi-computadores con Memoria Distribuida

Este sistema cambia radicalmente del modelo que comparte una memoria para todos los procesadores, teniendo como característica común la utilización de la red de comunicación de los inter-procesadores. Un multicomputador se puede ver como un computador paralelo en el cual cada procesador tiene su propia memoria local. La memoria del sistema se encuentra distribuida entre todos los procesadores y cada procesador sólo puede direccionar su memoria local. Para acceder a las memorias de los demás procesadores debe hacerlo por paso de mensajes. Esto significa que un procesador tiene acceso directo sólo a su memoria local, siendo indirecto el acceso al resto de memorias de los otros procesadores. Este acceso local y privado a la memoria es lo que diferencia los Multicomputadores de los Multiprocesadores. Este sistema se muestra en la figura A5.

Cuando un proceso requiere datos de otro proceso, que utiliza un espacio de memoria que no le pertenece, Es usualmente el programador quien se lleva la responsabilidad de definir explícitamente como y cuando será la comunicación de datos. La sincronización entre tareas es la responsabilidad que se le asigna al programador. (Barney, 2007)

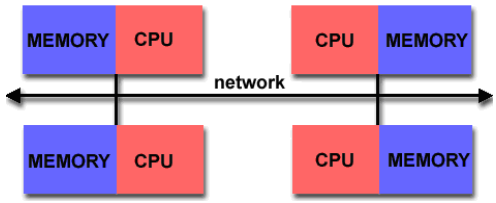


Figura A5 Esquema para un sistema con memoria distribuida (Barney, 2007)

Con esta arquitectura la memoria es escalable al número de procesadores. Cada procesador puede acceder rápidamente a su propia memoria fuera de interferencia y fuera de sobrecargas incurridas en el tratar de mantener la coherencia de la cache. Las desventajas que presenta esta arquitectura, es que el programador se lleva la mayor carga asociada a los detalles de comunicación de datos entre los procesadores y es éste el que garantiza el éxito de este sistema. También puede ser difícil el mapeo de la estructura de datos existente, basada en memoria global, para este tipo de organización de memoria. (Barney, 2007)

Estos tipos de Arquitecturas presentan un bajo acoplamiento por lo que el paso de mensajes entre procesadores debería incrementarse, siendo este el punto donde hay que prestar la mayor atención por parte del diseñador y programador de estas arquitecturas.

Sistemas Híbridos

Se habla de un tercer tipo de sistema paralelo, este utilizan las ventajas de memoria compartida y del sistema distribuido, se muestra en la figura A6. Los componentes de memoria compartida son usualmente maquinas SMP (Symmetric multiprocessing). El componente distribuido lo proporciona una interconexión de red entre las máquinas SMP. Las tendencias indican que esta arquitectura será la que se desarrollara más rápidamente y prevalecerá. (Jorge, 2007)

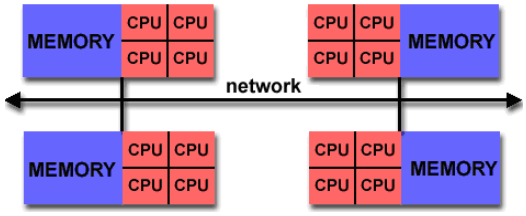


Figura A6 Esquema para un sistema hibrido de memoria (Barney, 2007)

Se resumen algunas de las características dominantes de las máquinas compartida y distribuida en lo que a memoria se refiere, se revisan en la figura A7.

Arquitectura	UMA	NUMA	Distribuida
Ejemplos	SMPs Sun Vexx DEC/Compaq SGI Challenge IBM POWER3	SGI Origin Sequent HP Exemplar DEC/Compaq IBM POWER4 (MCM)	Cray T3E Maspar IBM SP2
Comunicación	MPI Threads OpenMP shmem	MPI Threads OpenMP shmem	MPI
Escalabilidad	Hasta 10 procesadores	Hasta 100 procesadores	Hasta 1000 procesadores
Desventajas	Ancho de banda entre memoria y CPUs	-Ancho de banda entre memoria y CPUs -Tiempos de acceso no uniformes	- Administracion del Sistema - Programación difícil de desarrollar y mantener

Figura A7 Cuadro, características de memoria compartida y distribuida (Barney, 2007)

Anexo C: Organización de Procesadores

La organización de procesadores o red se refiere a como están entrelazados o conectados los procesadores o nodos en un computador paralelo.

Se revisara algunos tipos de organización de procesadores, dichas organizaciones también son aplicables a módulos de memoria

Bus y Ethernet

En una red basada en un bus, los procesadores comparten el mismo recurso de comunicación. La arquitectura **bus** es fácil y económica de implementar, pero no es escalable ya que sólo un procesador puede usar el bus en un momento dado. A medida que se incrementa el número de procesadores, el bus se convierte en un cuello de botella debido

a la congestión. Esta topología es muy popular en multiprocesadores de memoria compartida como el Encore Multimax y el Sequent Symetry, en donde el **bus** (Figura A8) es usado para leer y escribir en la memoria global (compartida).

En principio, la memoria global simplifica la programación paralela ya que no hay que tomar en cuenta la localidad**.** Sin embargo, la mayoría de las máquinas paralelas de memoria compartida usan memorias caché para reducir el tráfico en el bus, por lo tanto, la localidad continua siendo importante ya que el acceso al caché es mucho más rápido que a la memoria compartida.

Figura A8 Topología bus (García, 2003)

Un conjunto de máquinas conectadas por **Ethernet** es otro ejemplo de una interconexión por bus. Es muy popular para interconectar estaciones de trabajo y computadores personales. Todos los computadores conectados vía Ethernet comparten el mismo canal de comunicación (Figura A9). Una máquina que requiere enviar un mensaje tiene que esperar hasta que el canal este libre, si detecta una colisión, espera cierto tiempo y retransmite.

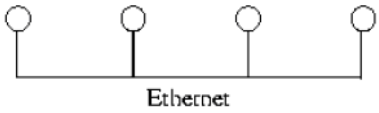


Figura A9 Topología ethernet (García, 2003)

Al igual que la topología bus, las mallas son fáciles y económicas de implementar, sin embargo el diámetro se incrementa al añadir nodos. La topología de malla es de dimensión uno, si los dos nodos extremos también son conectados, entonces se tiene un anillo (Figura A10).

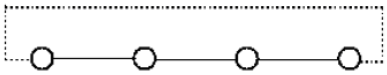


Figura A10 Topología de malla de dimensión 1, lineal o anillo (García, 2003)

Mallas de 2 dimensiones (Figura A11) y de 3 dimensiones son comunes en computación paralela y tiene la ventaja de que pueden ser construidas sin conexiones largas. El diámetro de las mallas puede ser reducido a la mitad si se extiende la malla con conexiones toroidales (La ventaja de los toroides es que por su geometría y su material, confinan muy bien el campo magnético, limitando así las pérdidas (Balum, 2008), de forma que los procesadores en los bordes también estén conectados vecinos (Figura A11). Esto sin embargo presenta dos desventajas, conexiones más largas son requeridas y los beneficios de esta interconexión se pierden si la máquina es particionada entre varios usuarios.

Los siguientes arreglos de procesadores usan mallas bidimensionales: Goodyear Aerospace MPP**,** AMT, DAP y MasPar MP-1. El multicomputador Intel Paragon XP/S también conecta sus procesadores mediante mallas bidimensionales.

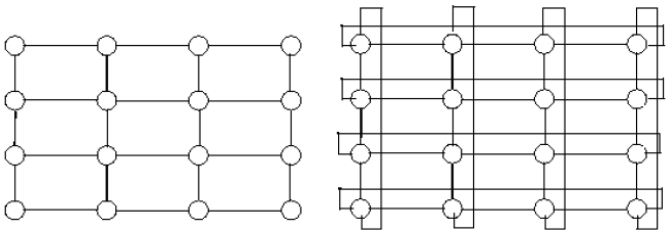


Figura A11 Topología de malla bidimensional y torus bidimensional (García, 2003)

Mariposa (Butterfly)

Comparada con las mallas, esta topología presenta menor diámetro. La máquina **BBN TC2000** usa una red en mariposa para enrutar datos entre procesadores(Figura A12).

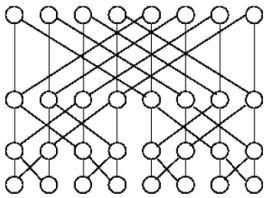


Figura A12 Topología de mariposa (García, 2003)

Árboles Binarios

Estas organizaciones son particularmente útiles en problemas de ordenamiento, multiplicación de matrices, y algunos problemas en los que su tiempo solución crece exponencialmente con el tamaño del problema (NP-complejos). El esquema básico consiste en técnicas de división y recolección (Figura A13).

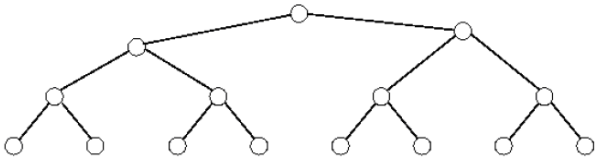


Figura A13 Árboles Binarios (García, 2003)

Pirámides

Esta topología intenta combinar las ventajas de la topología malla y de árboles. Con esta organización se incrementa la tolerancia fallas y el número de vías de comunicación sustancialmente (Figura A14).

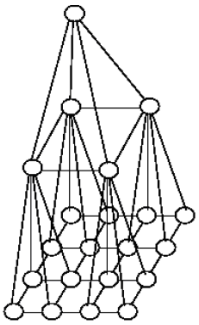


Figura A14 Pirámide (García, 2003)

Hipercubo

Un hipercubo puede ser considerado como una malla con conexiones largas adicionales, las cuales reducen el diámetro e incrementan el ancho de bisección. Un hipercubo puede ser definido recursivamente. Un hipercubo de dimensión-cero es un único procesador (Figura A15) y un hipercubo de dimensión-uno conecta dos hipercubos de dimensión cero. En general, un hipercubo de dimensión d+1 con 2^{d+1} nodos, se construye conectando los procesadores respectivos de dos hipercubos de dimensión *d*. Esta organización se aprecia en máquinas construidas por **CUBE** y los grupos de elementos de procesamiento del arreglo de procesadores Connection Machine **CM-200** están conectados por un hipercubo.

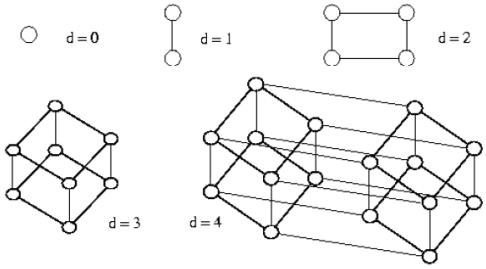


Figura A15 Hipercubos de dimensión del 0 al 4 (García, 2003)

Omega

La Figura A16 muestra una red omega de 3 etapas que conectan 8 procesadores). La red está formada por crossbar switches 2x2. Los switches tienen cuatro estados posibles: recto, cruzado, broadcast superior y broadcast inferior; que son configurados dependiendo de la conexión deseada (Figura. A17).

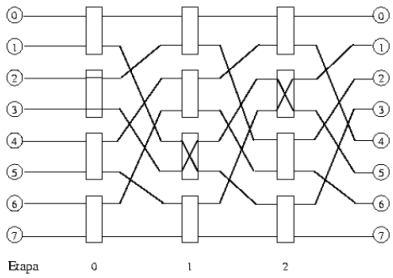


Figura A16 Red omega de 3 etapas que conecta 8 procesadores (García, 2003)

Debido a esto, estas topologías se llaman dinámicas o reconfigurables. Los switches son unidireccionales y debemos ver la red plegada en donde los procesadores de la izquierda y la derecha son los mismos. Estas redes reducen considerablemente la competencia por ancho de banda, pero son altamente no escalables y costosas. El performance-switch de la SP2 (se usa para aplicaciones paralelas, tiene memoria distribuida y usa el paso de mensajes entre cada procesador paralelo (IBM , 1994)) es una red omega.

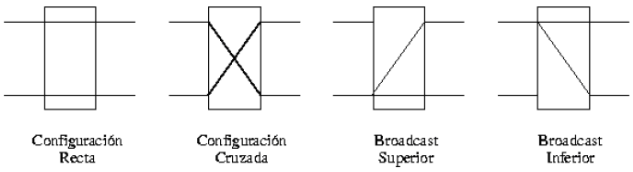


Figura A17 Configuraciones posibles de los crossbar switches 2x2 (García, 2003)

Fibras de Interconexión

La fibra de interconexión es un conjunto de switches, llamados routers, que están enlazados por distintas configuraciones o topologías. En la Figura A18 se muestran dos fibras de interconexión: una con 8 routers y la otra con 16 routers. Los procesadores están conectados a los routers, los cuales se reconfiguran de acuerdo a la interconexión deseada. Los procesadores de la **SGI Origin 2000** están conectados usando una fibra de interconexión en donde los routers están compuestos por crossbar switches 6x6. (García, 2003)

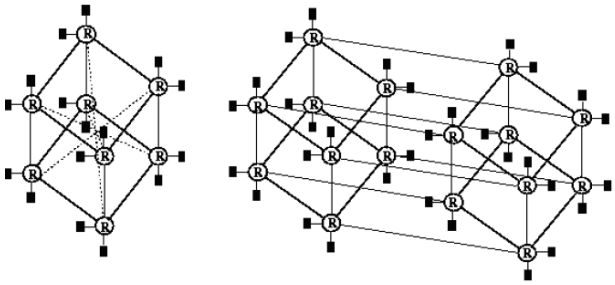


Figura A18 Fibras de interconexión (García, 2003)

La siguiente tabla (Figura A19) muestra las características de cada organización revisada. Nótese que sólo las mallas tienen la longitud de los enlaces constante y que en el hipercubo el número de enlaces es una función creciente del tamaño de la red.

Organización	Nodos	Diámetro	Ancho de bisección	Número de Enlaces Constante	Longitud de Enlaces Constante	¿Dinámica?
Bus o Ethernet	k	1	1	Si	No	No
Malla Dimensión 1	k	$k-1$	1	Si	Si	No
Malla Dimensión 2	k^2	$2(k-1)$	k	Si	Si	No
Malla Dimensión 3	k^3	$3(k-1)$	k^2	Si	Si	No
Mariposa	$(k+1)2^k$	$2k$	2^k	Si	No	No
Arboles Binarios	2^k-1	$2(k-1)$	1	Si	No	No
Pirámide	$(4k^2-1)/3$	$2\log k$	$2k$	Si	No	No
Hipercubo	2^k	k	2^{k-1}	No	No	No
Omega	2^k	1	2	Si	No	Si
Fibra de Interconexión	2^{k+1}	$<k$	$>2^{k-1}$	3	No	Si

Figura A19 Tabla resumen, características de las organizaciones de procesadores (García, 2003)

Anexo D: Fuentes del Paralelismo

El procesamiento paralelo tiene como principal objetivo explotar el paralelismo inherente a las aplicaciones informáticas. Todas las aplicaciones no presentan el mismo perfil cara al paralelismo, algunas se pueden paralelizar mucho y otras muy poco. Además de este factor cuantitativo evidente, es necesario considerar también un factor cualitativo, como la manera a través de la cual se explota el paralelismo. Cada técnica de explotación del paralelismo se denomina fuente. Se distinguen tres fuentes principales. (Flores, 2004)

El paralelismo de tareas - El paralelismo de control - El paralelismo de datos

Paralelismo de Tareas o Trivial

En esta Fuente la aplicación se encuentra con una serie de tareas que se pueden realizar simultáneamente y de manera independiente. Cada tarea se podría hacer en un procesador diferente, cada tarea tiene su propio conjunto de datos de entrada.

Ventajas:

- Se obtienen buenos rendimientos.

- No hay parte secuencial

- No hay sobrecarga de comunicaciones

- No hay desigualdad de carga

- El tiempo invertido en la paralelización es mínimo.

- Se pueden usar algoritmos secuenciales

- No hace falta usar un lenguaje de programación paralelo

Inconvenientes:

- Realmente no se puede considerar que el problema haya sido paralelizado sino replicado.

- Si el problema no es hacer muchas simulaciones sino una más rápida o precisa, se debe paralelizar.

Paralelismo de Control / Funcional

Segmentado:

La aplicación realiza una única tarea que se puede segmentar en etapas

• El paralelismo se introduce solapando el procesamiento de varios conjuntos de datos de entrada

• Los resultados parciales se transmiten entre las etapas del cauce o pipe

Características: • La aplicación se debe de poder dividir en etapas

Ventajas: • Paralelización sencilla y natural

Inconvenientes:

• Las ineficiencias son debidas:

- Al tiempo consumido en el llenado del cauce => tiempo de inicialización

- A la *desigualdad* de la duración de las diferentes etapas

Posibles soluciones*:*

*Usar CPUs más rápidas para las etapas más pesadas

*Paralelizar la etapa más costosa (combinación de Paralelismos)

• **Escalabilidad**: El número de procesadores que se puede usar viene limitado por el número de etapas de la aplicación

Posible solución:

Combinar este paralelismo con otro asignando varios procesadores a cada etapa.

Simultaneo

La aplicación realiza una única tarea que se compone de varias funciones o fragmentos de código que se pueden ejecutar en paralelo

Características:

• La aplicación se debe de poder dividir en funciones independientes

Paralelismo de Datos

Síncrono o Balanceado

Una operación se realiza síncronamente sobre todos los datos, los nuevos datos dependen solo de los antiguos

• Los datos se reparten entre los procesadores uniformemente

Las ineficiencias son debidas:

• A que el número de datos no es múltiplo del número de procesadores

• Tiempo consumido en la comunicación

Requiere más trabajo de programación.

Débilmente Asíncrono o Desbalanceado

Cada procesador realiza una parte de un problema heterogéneo, se produce una sincronización y se vuelve a tratar el problema.

Cuando termina un paso temporal algunos procesadores se mantienen en espera hasta que terminen todos, se intercambia información y se continúa con el siguiente paso temporal.

Las ineficiencias son debidas a:

• La desigualdad dinámica del trabajo en cada procesador

• Al tiempo consumido en la interacción entre los procesadores

Es la más difícil de programar porque combina las dificultades de los paralelismos segmentado y totalmente síncrono (en este caso la interacción entre los procesadores no es tan simple)

Ley de Amdahl

Una operación o tarea tiene solo una fracción f que es paralelizable mientras que el resto es serial. Aumentando el número de procesadores NP, el tiempo de ejecución T decrece por la fórmula:

$$T = (1-f) + f / NP$$

Nota: Esta fórmula no contempla la sobrecarga de paralelización que incluye entre otras cosas los costos de comunicación y la sincronización entre procesadores.

Anexo E: Tipos de datos en MPI

Los tipos de datos de C++ se muestran el la figura A21.

C Data Types	
MPI_CHAR	signed char
MPI_SHORT	signed short int
MPI_INT	signed int
MPI_LONG	signed long int
MPI_UNSIGNED_CHAR	unsigned char
MPI_UNSIGNED_SHORT	unsigned short int
MPI_UNSIGNED	unsigned int
MPI_UNSIGNED_LONG	unsigned long int
MPI_FLOAT	float
MPI_DOUBLE	double
MPI_LONG_DOUBLE	long double

Figura A21 Tabla de tipos de datos en C para MPI (Blaise2, 2007)

Anexo F: Comunicación punto a punto con MPI

Comunicación síncrona, con bloqueo

Estas no retornan de la llamada a subrutina hasta que el proceso de copia desde el buffer del usuario al buffer de sistema se completa.

Parámetros de Entrada/Salida:

buffer: Dirección de inicio del buffer

count: Número de entradas en el buffer

datatype: Tipo de dato del buffer

dest: Proceso destino (id del proceso)

tag: "Marca" del mensaje (integer)

comm: Communicador

source: Id del proceso emisor

status: Estructura con información del estado de comunicación.

Request: Estado de la comunicación.

MPI_Send(): Para enviar un mensaje de un proceso a otro .

int **MPI_Send**(void *buffer, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm);

MPI_Recv(): Recibe el mensaje .Bloquea hasta que se recibe el dato solicitado por el proceso correspondiente.

int **MPI_Recv**(void *buffer, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Status *status);

-Suponga 2 procesos

-Proceso 1 recibe la copia del proceso 0, como lo muestra la Figura A22.

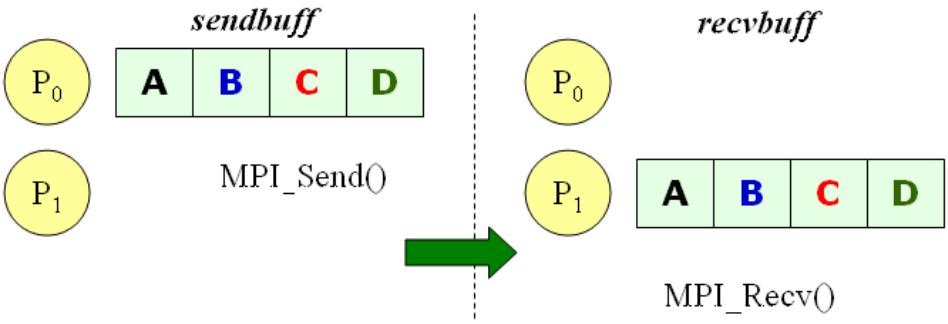


Figura A22 Interpretación de MPI_Send() y MPI_Recv() (Rodríguez, 2007)

Comunicación Asíncrona, sin bloqueo

El programa inmediatamente después de invocar las funciones retorna el punto en donde fue invocada y asegura solamente que el proceso de copiado de buffer se inicia, no asegurando la terminación de este. Para verificar que se haya completado dicho proceso existe la rutina **MPI_WAIT**(..).

MPI_Isend():

int **MPI_Isend**(void *buffer, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm, MPI_Request *request);

MPI_Irecv():

int **MPI_Irecv**(void *buffer, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Request *request);

MPI_Wait(): Esta función Bloquea el proceso hasta que una operación de envío o recibo se haya completado.

int **MPI_Wait** (MPI_Request *request, MPI_Status *status);

Utilizando la misma Interpretación anterior, como se ve en la Figura A23.

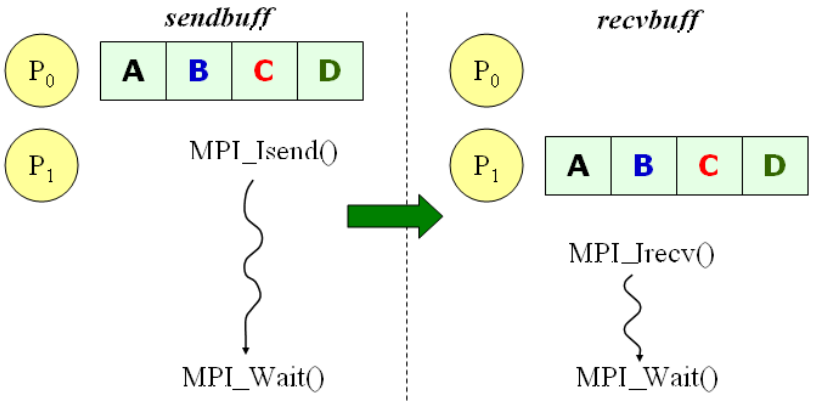


Figura A23 Interpretación de MPI_Isend() y MPI_Irecv(), con MPI_Wait() (Rodríguez, 2007)

Anexo G: Comunicaciones colectivas con MPI

Estas son comunicaciones que involucran más de dos procesos, generalmente todos. La ventaja de las comunicaciones colectivas en MPI, es que asegura que el costo de la comunicación sea de orden de LOG (size), donde size es la cantidad de procesos involucrados.

Parámetros de Entrada/Salida:

buffer: Dirección de inicio del buffer

count: Número de entradas en el buffer

datatype: Tipo de dato del buffer

root: d del proceso root (integer)

comm: Communicador

recvbuf: Dirección del buffer de recepción.

op: Operación de reducción

sendbuf: Dirección del buffer para envío.

recvbuf: Dirección del buffer de recepción.

sendcnt: Numero de elementos a enviar a cada proceso

sendtype: Tipo de dato del buffer de entrada (sendbuf())

recvcnt: Numero de elementos en el buffer de recepción.

Recvtype: Tipo de dato del buffer de recepción.

MPI_Bcast (): Es para enviar un dato a todos los procesos root que es el origen. Esto lo muestra la Figura A24.

int **MPI_Bcast** (void *buffer, int count, **MPI_Datatype** datatype, int root, **MPI_Comm** comm);

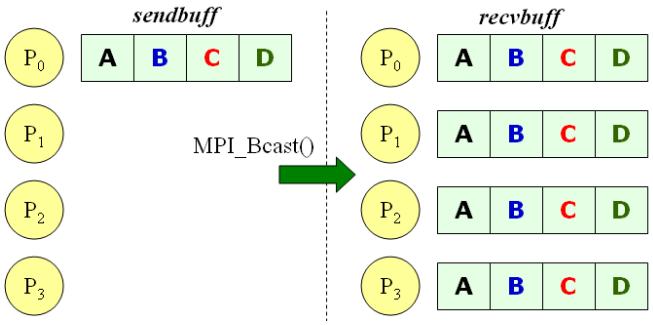


Figura A24 Interpretación de la función MPI_Bcast(). (Rodríguez, 2007)

MPI_Reduce(): Aplica una operación de reducción sobre todos los procesos del grupo y lo retorna a un proceso.

int **MPI_Reduce** (void *sendbuf, void *recvbuf, int count, **MPI_Datatype** datatype, **MPI_Op** op, int root, **MPI_Comm** comm);

-Estado inicial donde cada proceso tiene o produce un valor

-El proceso maestro recibe la suma total de los datos de todos los procesos, incluyendo el suyo.

-En este caso la operación (op) usada fue la suma. Esto se ve en la Figura A25

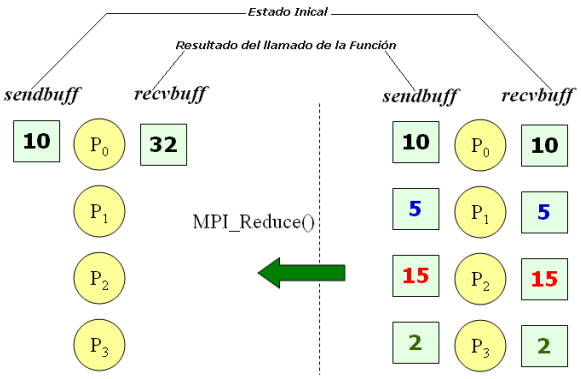


Figura A25 Interpretación de la función MPI_Reduce(). (Rodríguez, 2007)

MPI_Allreduce(): Aplica una operación de reducción sobre todos los procesos del grupo y lo retorna a todos los procesos. Esto equivale a una función **MPI_Reduce**() seguida de un **MPI_Bcast**().

int **MPI_Allreduce** (void *sendbuf, void *recvbuf, int count, **MPI_Datatype** datatype, **MPI_Op** op, **MPI_Comm** comm);

-Tomando la misma operación que se aplico en la interpretación de MPI_Reduce()

-Todos todos los procesos reciben el mismo resultado, que corresponde a la sumatoria de los valores individuales de cada proceso. Examinar Figura A26

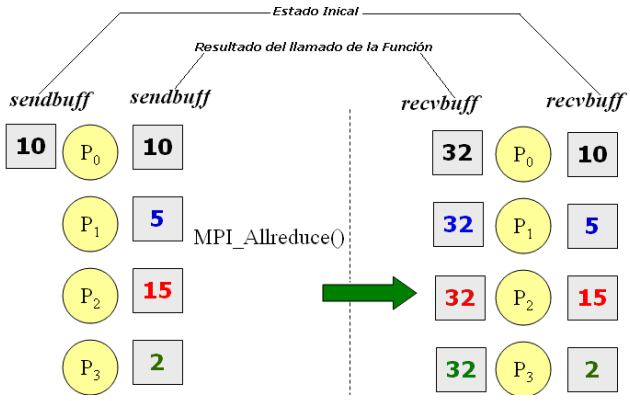


Figura A26 Interpretación de la función MPI_AllReduce(). (Rodríguez, 2007)

MPI_Scatter():Cada proceso recibe una parte del proceso fuente. Figura A27

int **MPI_Scatter**(void *sendbuf, int sendcnt, **MPI_Datatype** sendtype, void *recvbuf,int recvcnt,**MPI_Datatype** recvtype, int root, **MPI_Comm** comm);

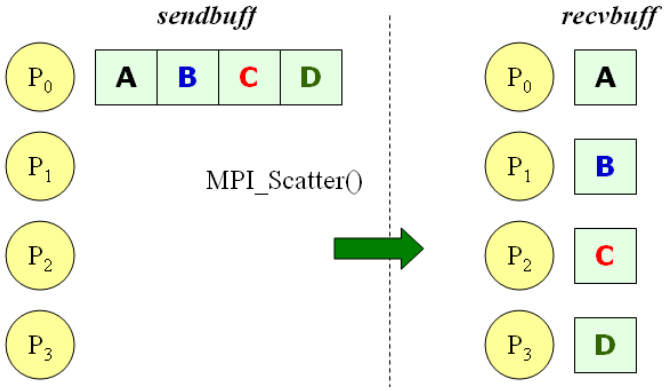


Figura A27 Interpretación de la función MPI_Scatter(). (Rodríguez, 2007)

MPI_Gather(): Reúne datos de distintos procesos y los retorna a un proceso determinado . Es la operación inversa ha MPI_Scatter(). Se muestra en la Figura A28

int **MPI_Gather** (void *sendbuf, int sendcnt, **MPI_Datatype** sendtype, void *recvbuf, int recvcnt,**MPI_Datatype** recvtype,int root,**MPI_Comm** comm);

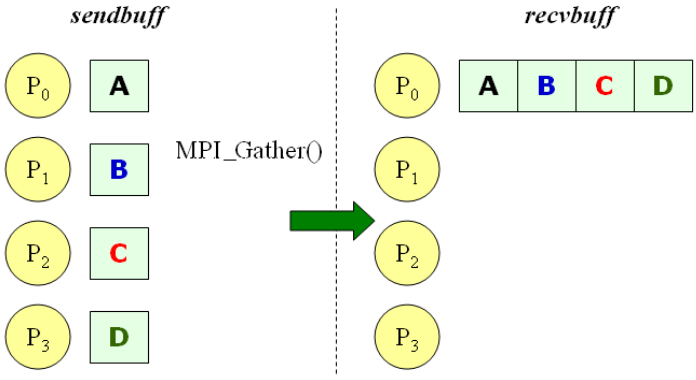


Figura A28 Interpretación de la función MPI_Gather(). (Rodríguez, 2007)

MPI_Allgather(): realiza el mismo procedimiento de Gather(), retornando su resultado a todos los procesos. Esta funcion se examina en la Figura A29

int **MPI_Allgather**(void *sendbuf, int sendcnt, **MPI_Datatype** sendtype, void *recvbuf, int recvcnt, **MPI_Datatype** recvtype, **MPI_Comm** comm);

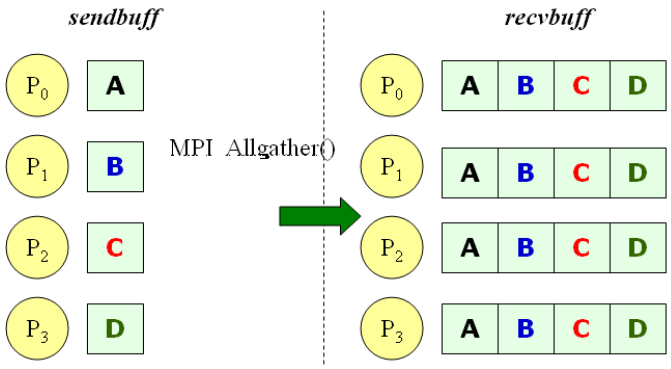


Figura A29 Interpretación de la función MPI_Allgather(). (Rodríguez, 2007)

MPI_Alltoall():Cada proceso realiza operación de dispersión enviando datos a todos los procesos ordenados por un índice. La función es mostrada en la Figura A30

int **MPI_Alltoall**(void *sendbuf, int sendcnt, **MPI_Datatype** sendtype, void *recvbuf, int recvcnt, **MPI_Datatype** recvtype, **MPI_Comm** comm);

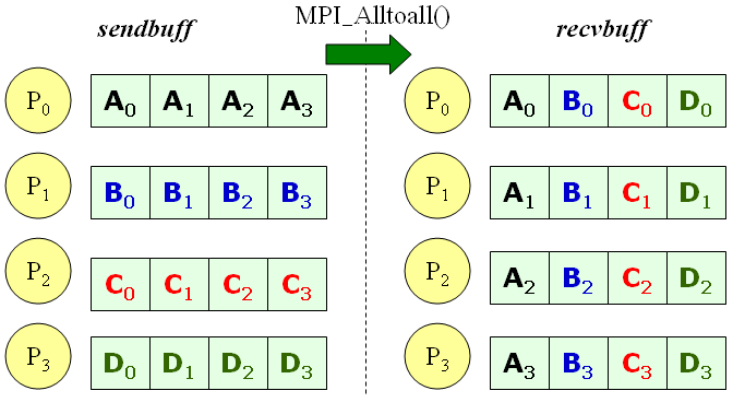


Figura A30 Interpretación de la función MPI_Alltoall(). (Rodríguez, 2007)

Anexo H: Operaciones Colectivas

- **MPI_MAX** retorna valor máximo
- **MPI_MIN** retorna valor míximo
- **MPI_SUM** retorna la suma
- **MPI_PROD** retorna el producto
- **MPI_LAND** retorna AND lógico
- **MPI_BAND** retorna el AND bitwise
- **MPI_LOR** retorna OR lógico
- **MPI_BOR** retorna el OR bitwise
- **MPI_LXOR** retorna XOR lógico
- **MPI_BXOR** retorna XOR bitwise
- **MPI_MINLOC** retorna el valor mínimo y la localización (valor del segundo elemento de la estructura).
- **MPI_MAXLOC** retorna el máximo y la localización.

Anexo J: Funciones de C++

Funciones InLine y Automáticas

La ventaja de la función inline es que se puede ejecutar más rápidamente que las funciones normales. Si el compilador no es capaz de cumplir la petición, la función se compila como una función normal y la solicitud inline se ignora. Las restricciones son cuatro, no puede

contener variables de tipo static, una sentencia de bucle, un switch o un goto. Se tiene que definir antes de ser usarla.

En las funciones automáticas su definición es lo suficientemente corta y puede incluirse dentro de la declaración de la clase. La palabra inline no es necesaria. Se aplican las mismas restricciones de la función inline. El uso más común de las funciones automáticas es para funciones constructoras. (Guerrero, 2004)

Funciones Amigas

Estas se utilizan cuando una función no es miembro de una clase, y ésta pueda tener acceso a los miembros privados de dicha clase. El prototipo de esta función viene precedido por la palabra friend.

This, New, Delete

This es una variable por defecto para todas las funciones miembro de una clase. Este puntero contiene la dirección del objeto de la clase ala que se está aplicando la función miembro. (García, 1998)

Para asignar memoria dinámica en C, se hacía con malloc y para liberar se usaba free. En C++ se puede asignar memoria utilizando new y liberarse mediante delete. Debe llamarse a delete solo con un puntero obtenido mediante new. Se le puede pasar un valor inicial a los objetos con la sentencia new. (Guerrero, 2004)

También se pueden crear arreglos asignados dinámicamente, estos pueden utilizar la sentencia new.

Declaración de un Arreglo

Puntero =new tipo[tamaño];

Anexo K: Propiedades básicas de la teoría de conjuntos aplicadas a intervalos

Obsérvese el cuadro siguiente con las principales propiedades básicas:

Asociatividad y Conmutatividad	Identidades	Inclusión monótona
Si I, J y K son intervalos:	Si 0 y 1 son números reales:	Si $I \subset K$ y $J \subset L$, entonces:
$I+(J+K)=(I+J)+K$ $I*(J*K)=(I*J)*K$ $I+J=J+I$ $I*J=J*I$	$0+I=I+0=I$ $1*I=I*1=I$	$I+J \subset K+L$ $I-J \subset K-L$ $I*J \subset K*L$ $\left[\frac{I}{J} \subset \frac{K}{L}\right]$ ssi $0 \notin J$ y $0 \notin L$

Tabla 2 propiedades básicas de la teoría de conjuntos aplicadas a intervalos

La propiedad distributiva no es aplicable a la aritmética de intervalos, pero si se aplica a la aritmética de números reales, porque en la aritmética de intervalos I*(J + K) = I*J + I*K no se verifica, pero la sub-distributividad I *(J+K) $\subseteq$I*J + I*K si se cumple, obsérvese el siguiente ejemplo:

Considérese las funciones *f*1(I,j,k)= i * (j + k) y *f*2(i,j,k)= i * j + i * k que son equivalentes en la aritmética real, si se extienden a aritmética intervalar se obtienen las siguientes funciones:

F1(I,J,K)= I*(J + K) y F2(I,J,K)= I*J + I*K

Si se tiene que I= [1, 3], J= [2, 5] y K= [3, 4], la evaluación de F1 es:

$$F1(I, J, K)= [1, 3]*([2, 5] – [3, 4])=[1, 3]*[-2, 2]=[-2, 6]$$

Mientras que la evaluación de F2 es:

$$F2(I, J, K)= [1, 3]*[2, 5]-[1, 3]*[3, 4]=[2, 15]-[3, 12]=[-10, 12]$$

Se observa que los resultados de F1 y F2 son diferentes, destacándose que $[-2, 6] \subset [-10, 12]$, la consecuencia de esto es que el resultado de una función depende del orden en que las operaciones son efectuadas.

Anexo R: Principales métodos estocásticos y probabilísticos

Random Search	Simulated Annealing	Clustering Methods	Genetic Algorithms
Es el más básico de los algoritmos, el cual explora aleatoriamente el valor Óptimo, partiendo desde un punto inicial (aleatorio), y buscando puntos (al azar) en los que disminuye el valor de la función objetivo	**Este método se basa en la idea del enfriamiento de los métales para encontrar su nivel más bajo de energía.**	**Este método comienza con una muestra uniforme de puntos de la región de búsqueda y entonces crea grupos o clusters de puntos cercanos que corresponden a una región común de atracción**	**Seleccionan aleatoriamente una población inicial de soluciones, la que va evolucionando mediante operadores genéticos hasta una generación final que contiene un conjunto de soluciones mejores que el inicial.**

Tabla 3 Principales métodos estocásticos y probabilísticos

A) Este método ha sido bastante estudiado, y se demostró que el valor de la función más pequeño encontrado converge al mínimo global en infinitos pasos, si los puntos en los que se evalúa la función tienen una distribución uniforme sobre la región de búsqueda (Andersen, Jennings y Ryan, 1972).

B) Este método posee diversas variantes, como Nonlinear Simulated Annealing y Adaptive Simulated annealing.

C) Cada región de atracción tiene como propiedades que contiene exactamente un mínimo local, y que cualquier algoritmo que comienza en un punto cualquiera de la región converge al mínimo local de dicha región (Dekkers y Aarts, 1982),(Rinnooy Kan y Timmer, 1984), (Van Iwaarden, 1996).

D) Este algoritmo está basado en la teoría de las especies de Darwin.

Anexo M: Granja de procesos

Una granja de procesos es un sistema formado por tres tipos de procesos:

Raíz:

- Encargado de dividir las tareas entre los trabajadores
- Envía una nueva tarea a un trabajador cuando el recopilador le informa de que este ha terminado

Trabajador:

- Recibe la tarea del raíz
- Procesa la tarea
- Envía los resultados al recopilador

Recopilador:

- Recibe los resultados de los trabajadores
- Informa al raíz de que un trabajador se ha quedado ocioso

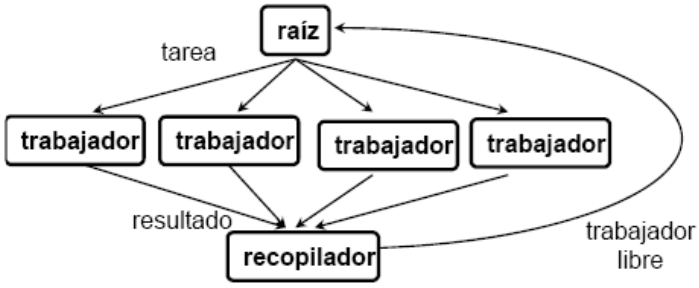


Figura 4.13 Granja de procesos (Llorente, 2002)

Se suele emplear cuando se quieren realizar varias tareas independientes que requieren diferente cantidad de computación, esto corresponde a balanceo de carga dinámico.

Limitaciones y mejoras al modelo básico

- Para evitar que los trabajadores se queden ociosos mientras esperan otra tarea, se coloca un buffer en el trabajador
- Un trabajador mantiene dos tareas: sobre la que trabaja y la siguiente
- Aumento de la productividad
- Reparto de tareas para que todos los trabajadores terminen a la vez
- Una posible solución es repartir primero las tareas más pesadas
- Los trabajadores se comunican entre sí
- Gestión muy complicada porque no se conoce a priori que trabajador desarrollará cada tarea

Para una mejor comprensión, se describen diversos términos usados en paralelismo en el anexo A.

Anexo N: Creación de Clases y Objetos

La base del encapsulamiento es la clase, que le da las características y comportamiento a los objetos. Lo primero se crea la clase y después en la función main se crean los objetos de cada una de las clases. Las variables y funciones de una clase pueden ser **publicas**, **privadas** o **protegidas**. Si no se indica nada éstas serán privadas. Estos modificadores nos indican en que partes de un programa podemos utilizar las funciones y variables.

Private: Solo tendrán acceso los de la clase en donde se definen.

Public: Se pude hacer referencia desde cualquier parte del programa.

Protected: Se puede hacer referencia desde la misma clase y las subclases. (Guerrero, 2004)

Creación de una clase

```
class nombre_clase {
funciones y variables privadas;
public:
funciones y variables publicas;
}
```

Creación del objeto:

```
nombre_clase nombre_objeto1;
nombre_clase nombre_objeto2;
```

Llamadas a las funciones de una clase:

```
nombre_objeto.nombre_funcion (parámetros);
```

***Desarrollo de funciones miembro*:**

```
valor_devuelto nombre_clase:nombre_funcion (parámetros) {
  cuerpo;
```

….

}

Nuevos recursos de C++

En C++ se pueden seguir utilizando las mismas sentencias de C para mostrar información por pantalla o pedirla mediante teclado. Pero a estas antiguas se añaden 2 nuevas de la misma potencia y mayor facilidad de uso. La cabecera que utilizan estas dos sentencias es iostream.h.

Mostrar por pantalla:
cout << expresión;

Pedir por teclado:
cin >> variable; La variable pude ser de cualquier tipo.

En este tipo de programas hay partes que requieren inicialización, para esto se hace a través de funciones constructoras, lo contrario ha éstas son las funciones destructoras.

Constructoras

```
class clase_ejemplo {
int atributo;
public:
void funcion1(int parametros);
clase_ejemplo ();
}

clase_ejemplo::clase_ejemplo(){

atributo=10;

}
```

Destructoras

El nombre de las funciones destructoras debe ser el mismo que el de la clase a la que pertenece precedido del carácter ~.Para ver mas funciones de C++ revisar anexo J

Anexo O: Bitácora de Instalación

Al ingresar por primera vez al cluster se constató que estaban inoperables tanto el maestro como los esclavos, por ende se procedió a formatear cada equipo por separado, instalando Debian GNU/Linux 4.0, por medio de la instalación por red con un CD mínimo. Un CD de instalación por red o netinst es un único CD que posibilita que instale el sistema completo. Este único CD contiene sólo la mínima cantidad de software para comenzar la instalación y obtener el resto de paquetes a través de Internet.

Uno a uno se instalo el sistema operativo Debian en los nodos esclavos y en el nodo maestro, utilizando la LAN de la facultad para acceder a internet, siendo descargados los paquetes necesarios del repositorio debían.ubiobio.cl (Mirror de Debian Linux en la Universidad del Bío-Bío).

Instalación de Paquetes

Se configuraron las tarjetas ethernet de los esclavos de forma momentánea con una ip fija para tener salida a la LAN y de esta forma descargar los paquetes necesarios para el buen funcionamiento del cluster.

Con la herramienta apt-get se descargan e instalan los siguientes paquetes y sus dependencias:

ssh; lam4-dev; lam-runtime; gcc; build-essentials

Red del Cluster

Para levantar el cluster es necesario dejar todos los nodos conectados de forma que exista comunicación entre el maestro y los esclavos.

Se comenzó con el maestro utilizando una ip fija para la tarjeta con salida exterior (LAN) y otra configuración para la tarjeta que se conecta al switch, que se encargara de administrar la red del cluster (véase la figura 6.3).

Primero se creará un script que utilizaremos para cargar la configuración de red en el maestro, este archivo se llama interfaces y encuentra en la carpeta /etc/network/ como se muestra en la figura 6.2:

```
rllanos@maestro:/etc/network$ ls
if-down.d  if-post-down.d  if-pre-up.d  if-up.d  interfaces  run
rllanos@maestro:/etc/network$  vim interfaces
```

Figura 6.2 Carpeta /etc/network/

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug eth0
iface eth0 inet static
        address 146.83.194.169
        netmask 255.255.255.128
        network 146.83.194.128
        broadcast 146.83.194.255
        gateway 146.83.194.129
        # dns-* options are implemented by the resolvconf package, if installed
        dns-nameservers 146.83.195.14
        dns-search cluster.ubiobio.cl

allow-hotplug eth2
iface eth2 inet static
        address 192.168.1.1
        netmask 255.255.255.0
        network 192.168.1.0
        broadcast 192.168.1.255
```

Figura 6.3 Archivo interfaces que contiene las direcciones para la adecuada configuración de las tarjetas ethernet del maestro.

De la misma forma se debe configurar los esclavos, claro que como ellos poseen una sola tarjeta de red, la configuración debe ser similar como se muestra en al siguiente figura 6.4.

```
# The primary network interface
allow-hotplug eth0
iface eth0 inet static
        address 192.168.1.4
        netmask 255.255.255.0
        network 192.168.1.0
        broadcast 192.168.1.255
        gateway 192.168.1.1
        # dns-* options are implemented by the resolvconf package, if installed
        dns-nameservers 146.83.195.14
        dns-search cluster.ubiobio.cl
```

Figura 6.4 Configuración del archivo interfaces en un nodo esclavo

Se debe cambiar el address de cada nodo esclavo; dependiendo del número de este, por ejemplo, (192.168.1.x), es decir, asignando una dirección ip única en la red, para cada esclavo del cluster. La gateway debe coincidir con la dirección ip que el maestro tiene asignada en la red, en este caso 192.168.1.1

Configuración de LAM

Una vez que se encuentre LAM/MPI instalado en todas las máquinas, modificaremos el archivo .bashrc del usuario, que encontraremos en la home de este */home/usuario/.bashrc*, al final del archivo se añaden las siguientes líneas:

PATH=/usr/local/lam/bin:$PATH

export PATH

PATH=/usr/local/lam/bin:$PATH

export PATH

Se modificará el archivo */etc/hosts* (ver figura 6.5) donde se indica donde están todos los nodos, y luego se debe copiar el archivo a los demás nodos.

```
127.0.0.1       localhost
146.83.194.169  maestro.cluster.ubiobio.cl      maestro

#configuracion para el cluster
192.168.1.1     maestro
192.168.1.2     esclavo2
192.168.1.3     esclavo3
192.168.1.4     esclavo4
192.168.1.5     esclavo5
192.168.1.6     esclavo6
```

Figura 6.5 Archivo hosts, con las direcciones de los nodos

Una vez localizados los nodos, se debe acceder usando ssh a todos los nodos sin necesidad de password, para ello en el nodo maestro se debe ejecutar:

ssh-keygen -t rsa

Lo cual genera una clave se almacena en /home/usuario/.ssh/id_rsa.pub que debe ser copiada a todos los nodos esclavos, para ello utilizamos una sentencia similar a esta:

scp /home/usuario/.ssh/id_rsa.pub usuario@192.168.1.2:/home/usuario/

Luego se accede al nodo y se guarda la clave:

cd /home/user/

mkdir .ssh

cat id_rsa.pub >> /home/user/.ssh/authorized_keys

rm id_rsa.pub

Luego desde el maestro se comprueba accediendo a los nodos sin necesidad de ingresar password después de la segunda vez.

rsh esclavo2 -n 'echo $SHELL'

La instrucción debería devolver /bin/bash.

Luego creamos un archivo en el directorio home, que llamaremos boot_schema, en donde indicaremos a lam, los esclavos que puede utilizar. Ver figura 6.6

```
#los nodos del cluster son los siguientes

maestro
esclavo2
esclavo3
esclavo4
esclavo5
esclavo6
```

Figura 6.6 Archivo boot_schema en donde se encuentran los nodos del cluster.

Se arranca el cluster, indicando a lamboot que utilice nuestro archivo boot_schema. Ver figura 6.7

lamboot -v -ssi boot rsh boot_schema

```
rllanos@maestro:~$ lamboot -v -ssi boot rsh boot_schema

LAM 7.1.2/MPI 2 C++/ROMIO - Indiana University

n-1<3742> ssi:boot:base:linear: booting n0 (maestro)
n-1<3742> ssi:boot:base:linear: booting n1 (esclavo2)
n-1<3742> ssi:boot:base:linear: booting n2 (esclavo3)
n-1<3742> ssi:boot:base:linear: booting n3 (esclavo4)
n-1<3742> ssi:boot:base:linear: booting n4 (esclavo5)
n-1<3742> ssi:boot:base:linear: booting n5 (esclavo6)
n-1<3742> ssi:boot:base:linear: finished
```

Figura 6.7 lamboot corriendo con un maestro y 5 esclavos.

Anexo Q: Taxonomía de Flynn

SISD (Single Instruction - Single Data). Computadores **SISD** representan la mayoría de las máquinas seriales. Tienen un **CPU** que ejecuta una instrucción en un momento dado y busca o guarda un dato en un momento dado (García, 2003). Corresponde a un, computador convencional de Von Neumann, Monoprocesador (Rodríguez, 2004) .Los computadores Uniprocesadores modernos pueden exhibir alguna concurrencia. Por ejemplo las

arquitecturas superescalares soportan la identificación dinámica y selección de múltiples operaciones independientes, éstas se pueden ejecutar simultáneamente. (Quinn, 2004)

SIMD (Single Instruction – Múltiple Data). La categoría **SIMD** se refiere a los computadores con un flujo simple de instrucciones pero con flujos múltiples de datos. En esta categoría están los Arreglos de Procesadores y los Procesos Vectoriales Segmentados. Un Arreglo de Procesadores es un computador paralelo con una sola unidad de control ejecutando un flujo de instrucción, así como múltiples procesos subordinados capaces de llevar a cabo en forma simultánea la misma operación en diferentes elementos de datos. Un Procesador Vectorial Segmentado está construido sobre un reloj de proceso muy rápido y una o más unidades funcionales de segmentado para ejecutar la misma operación en los elementos de un conjunto de datos. (Quinn, 2004)

Las Aplicaciones de esta arquitectura son aquellas que requieren operaciones aritméticas de baja precisión sobre estructuras de datos regulares como en el procesamiento de imágenes y los gráficos. (Rodríguez, 2004)

MISD (Múltiple Instruction - Single Data). La categoría **MISD** es para computadores donde hay n procesadores cada uno recibiendo una instrucción diferente y operando sobre el mismo flujo de datos.

Un arreglo sistólico es un ejemplo de un computador **MISD**. El nombre proviene de la palabra “sístole”, la cual hace referencia a la contracción del corazón durante el latido. Un arreglo sistólico es una red de elementos de procesamiento primitivo que “bombea” datos. (Quinn, 2004)

MIMD (Múltiple Instruction-Múltiple Data). En esta categoría es para computadores con un múltiple flujo de instrucciones sobre un múltiple flujo de datos. La mayoría de los multiprocesadores y multicomputadores pueden ser clasificados bajo esta categoría. Tienen más de un procesador independiente y cada uno puede ejecutar un programa diferente sobre sus propios datos. Podemos hacer otra subdivisión de los multiprocesadores dependiendo de cómo esté organizada su memoria, memoria compartida o memoria distribuida.

Anexo R: Estrategias de Particionamiento

En la computación secuencial se requiere n-1 adiciones con un tiempo de complejidad de O(n). En una implementación paralela, se cuenta con p esclavos.

Separando la comunicación y la computación en si, podemos visualizar que se puede separar esta secuencia de acciones en distintas fases. Con muchos problemas, hay una fase de comunicación seguida por una fase de computación, y estas fases son repetidas en el transcurso del proceso.

Fase 1 –Comunicación: primero se debe considerar el aspecto de comunicación de los p procesos esclavos leyendo sus n/p números. Usando envíos y recepciones individuales de rutinas de comunicación requeridas se necesita un tiempo de:

$$t_{comm\ 1} = p\ (t_{startup} + (n/p)\ t_{data})$$

Donde $t_{startup}$ es la porción de tiempo constante de transacción y t_{data} es el tiempo de transmisión de una palabra de datos.

Fase 2 –Computación: después**,** se necesita estimar el número de pasos de cómputo. Cada proceso esclavo agrega n/p números en el mismo momento, requiriendo n/ (p -1) adicionales. Como todos los procesos esclavos están operando juntos, se puede considerar todas las sumas parciales obtenidas en los n/ (p -1) pasos. Por ende el tiempo de computación paralela en esta fase es.

$$t_{comp\ 1} = n/\ (p\ -1)$$

Fase 3 – Comunicación: cada procesador esclavo debe retornar sus resultados parciales obtenidos de sus correspondientes operaciones, usando rutinas de envio y recepción individuales. Esto tiene un tiempo de comunicación de:

$$t_{comm\ 2} = p\,(t_{startup} + t_{data})$$

Fase 4 –Computación: Para la acumulación final, el procesador maestro tiene que añadir las p sumas parciales, las cuales requieren de p-1 pasos.

$$t_{comp\ 2} = p\text{-}1$$

Tiempo de ejecución total: El tiempo total de ejecución para el problema planteado (considerando envió y recepción) es:

$$t_p = (t_{comm\ 1} + t_{comm\ 2}) + (t_{comp\ 1} + t_{comp\ 2})$$

$$= (p\,(t_{startup} + (n/p)\,t_{data}) + p\,(t_{startup} + t_{data}))$$

$$= 2pt_{startup} + (n+p)\,t_{data} + p + n/p - 2$$

$$t_p = O(n)$$

Se puede observar que la complejidad obtenida es la misma que la de una ejecución secuencial, pero sólo considerando los aspectos de computación. Por lo tanto la formulación en paralelo es mucho mejor que la formulación secuencial. (Wilkinson, 2005)

Anexo S: Dividir para Conquistar

Para el análisis se asume que n es una potencia de 2. El tiempo de inicio de comunicación $t_{startup}$ no es incluido directamente para una mayor comprensión. La fase de división consiste solo en comunicación. Si se asume esta división de la lista en dos partes requerirá de una computación mínima. La fase de combinación requiere de ambas partes tanto de comunicación como de computación para añadir las sumas parciales recibidas y enviar su resultado.

Fase-Comunicación: Esta fase de división presenta un número de pasos del orden de un logaritmo, esto es, Log p pasos con p procesadores. El tiempo de comunicación de esta fase esta dado por: $t_{comm\ 1} = \frac{n}{2} t_{data} + \frac{n}{4} t_{data} + \frac{n}{8} t_{data} + \ldots \frac{n}{p} t_{data} = \frac{n(p-1)}{p} t_{data}$

Donde t_{data} es el tiempo de transmisión de una palabra de datos. El tiempo $t_{comm\ 1}$ es marginalmente mejor que el de simple broadcast, que fue el usado en el método anterior. La fase de combinación es similar, excepto que en esta solo un ítem de datos es enviado en cada mensaje (las sumas parciales); esto es,

$$t_{comm\ 2} = (\text{Log p})\ t_{data}$$

Para un tiempo total de comunicación de:

$$t_{comm} = t_{comm\ 1} + t_{comm\ 2} = \frac{n(p-1)}{p} t_{data} + (\text{Log p})\ t_{data}$$

O un tiempo de complejidad de O(n) para un numero fijo de procesadores.

Fase-Computación: al final de la fase de división, los n/p números son añadidos juntos, entonces una adición ocurre en cada etapa durante la fase de combinación, lo que conduce:

$$t_{comp} = \frac{n}{p} + \text{Log p}$$

De nuevo la complejidad de tiempo da O(n), para un numero fijo de procesadores. Para un n mas grande y un numero de procesadores p variables, obtenemos O(n/p).

El tiempo de ejecución total

$$t_p = (\frac{n(p-1)}{p} + \text{Log p})\ t_{data} + \frac{n}{p} + \text{Log p}$$

Anexo T: Estrategias de Paralelización

Se vera un algoritmo de ordenamiento llamado bucket sort (ordenamiento por casilleros). Este no está basado sobre la comparación e intercambio, pero es naturalmente un método de particionamiento, Este algoritmo divide en m regiones iguales las cuales se les asigna una cantidad de números correspondientes a cada región, según algún tipo de clasificación (condición de entrada al casillero), las condiciones son excluyentes. Bucket sort se muestra en la Figura 4.6

La complejidad temporal es similar al a de quicksort o mergesort con O(n Log n) para ordenar n números. Se asume que estos algoritmos secuenciales de ordenamiento necesitan nLogn comparaciones, considerando una comparación como un paso computacional. Así tomará (n/m Log (n/m)) pasos para n/m números en cada bucket usado en este algoritmo de ordenamiento. La clasificación de números debe ser concatenada dentro de la lista final ordenada. Se asume que esta concatenación no requiere de pasos adicionales. Combinando todas estas acciones, el tiempo secuencial es de: (Wilkinson, 2005)

$$t_s = n+m((n/m)\ \text{Log}\ (n/m))=n+n\ \text{Log}\ (n/m)=O(n\ \text{Log}\ (n/m))$$

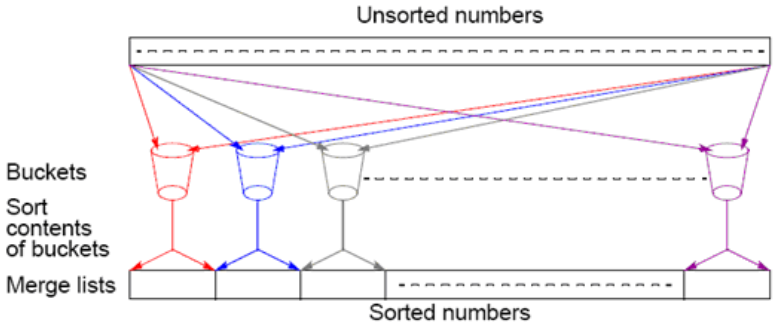


Figura 4.6 Bucket Sort. (Wilkinson, 2005)

Forma paralela de este algoritmo: Para paralizar el bucket sort, se asigna un procesador por bucket logrando, que la ecuación de complejidad sea (n/p) Log (n/p) para p procesadores, donde p=m. Otra estrategia de Paralelización es dividir esta secuencia en m

regiones, una región por cada procesador. Cada procesador mantiene p pequeños bucket, se clasifican los números dentro de los pequeños bucket. Después se envían los pequeños bucket a cada uno de los otros procesadores o bucket más grandes (bucket i para procesador i). Por ultimo se clasifican los números en el bucket más grande. Esta estrategia se muestra en la Figura 4.7

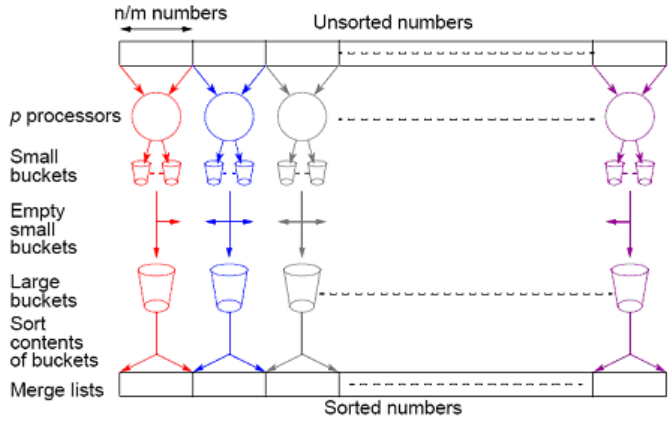


Figura 4.7 versión para paralelizar el Bucket Sort. (Wilkinson, 2005)

Fase 1 - Computación y Comunicación.

El primer paso es enviar grupo de números a cada procesador. Marcando un grupo de números dentro de las particiones se puede hacer en un tiempo constante. Este tiempo es ignorado en el tiempo total de comparación. Esta realiza la partición y la envía a cada procesador. Mejor forma es la de realizar un simple broadcast de todos los números para cada procesador y dejar que éstos hagan las respectivas particiones. Usando el broadcast o rutina de dispersión, el tiempo de comunicación es: $t_{comm1} = t_{startup} + nt_{data}$. Ésta incluye el tiempo inicial de comunicación $t_{startup}$

Fase 2 - Computación

Separa cada partición en n/p números dentro de los p pequeños bucket. El tempo de computación es: $t_{comp2} = n/p$

Fase 3 – Comunicación

Los pequeños bucket son distribuidos. Cada bucket tiene alrededor de n/p^2 números, asumiendo una distribución uniforme, cada procesador necesita enviar el contenido de p-1 pequeños bucket a otros procesos. Entonces cada procesador de los p existentes necesita esta comunicación, se tiene:

$$t_{comm\ 3} = p\ (p\text{-}1)\ (t_{startup} + (n/p^2)\ t_{data})$$

Este es el límite superior en la fase de comunicación. El limite mas bajo ocurriría si todas las comunicaciones pudieran solaparse, dejando

$$t_{comm\ 3} = (p\text{-}1)\ (t_{startup} + (n/p^2)\ t_{data}))$$

Fase 4-Computación

Los bucket más grandes son clasificados simultáneamente. Cada bucket, grande contiene alrededor de n/p números.

$$t_{comp4} = (n/p)\ \mathrm{Log}\ (n/p)$$

Tiempo de ejecución

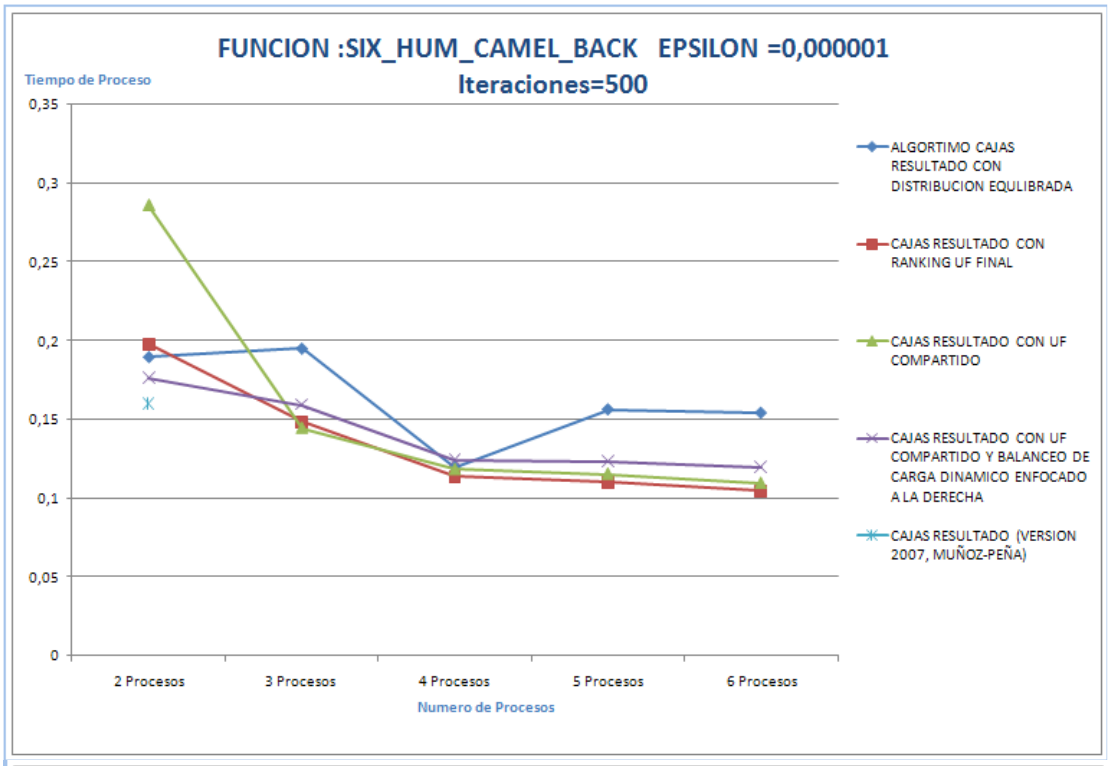
El tiempo total de ejecución incluyendo a la comunicación es:

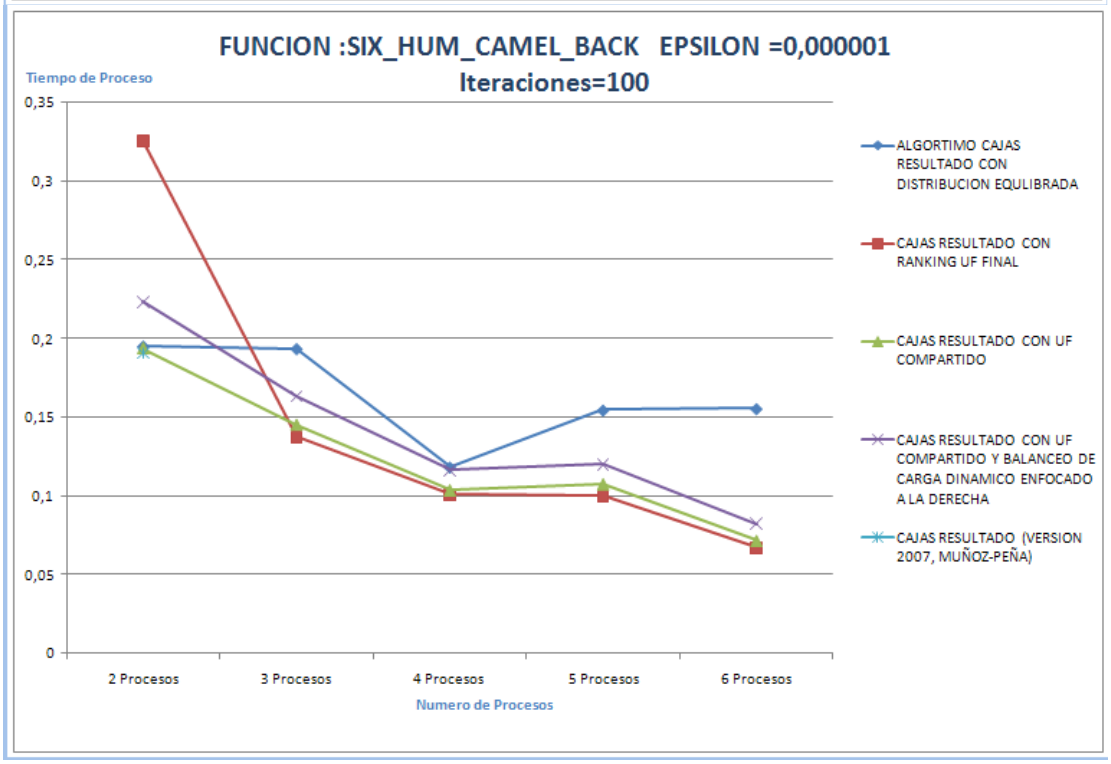
$$t_p = t_{comm1} + t_{comp2} + t_{comm\ 3} + t_{comp4}$$

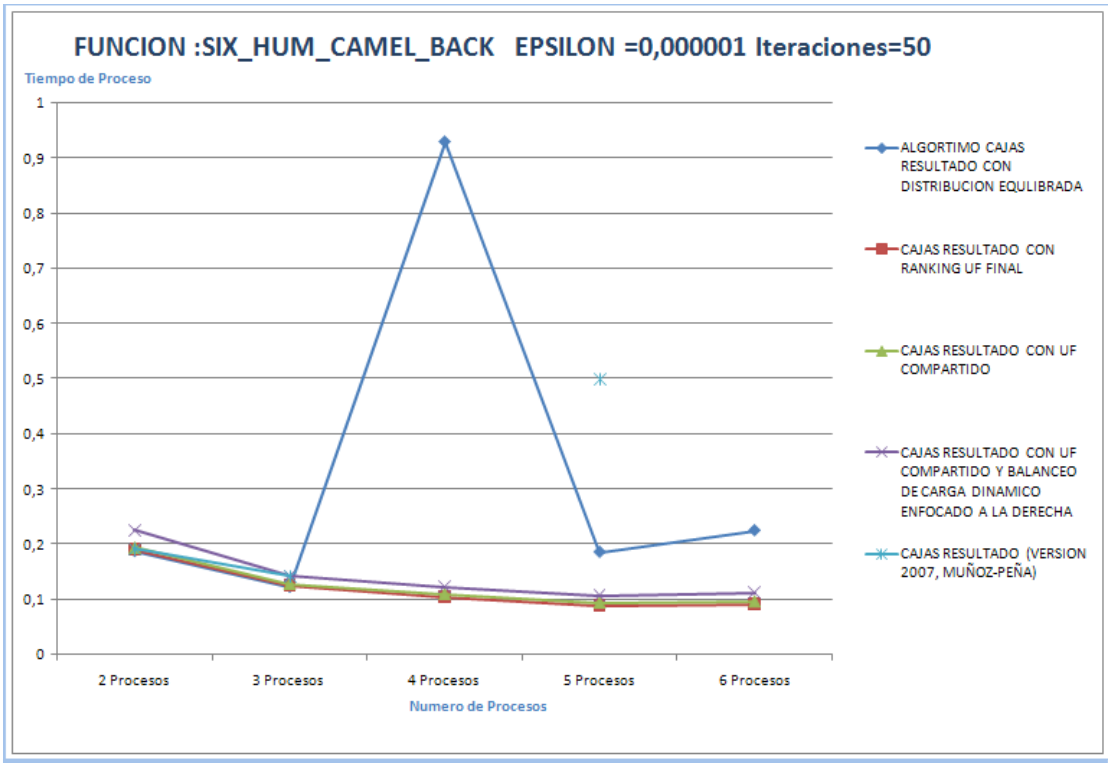
$$t_p = t_{startup} + n t_{data} + n/p + (p\text{-}1)\ (t_{startup} + (n/p^2)\ t_{data})) + (n/p)\ \mathrm{Log}\ (n/p)$$

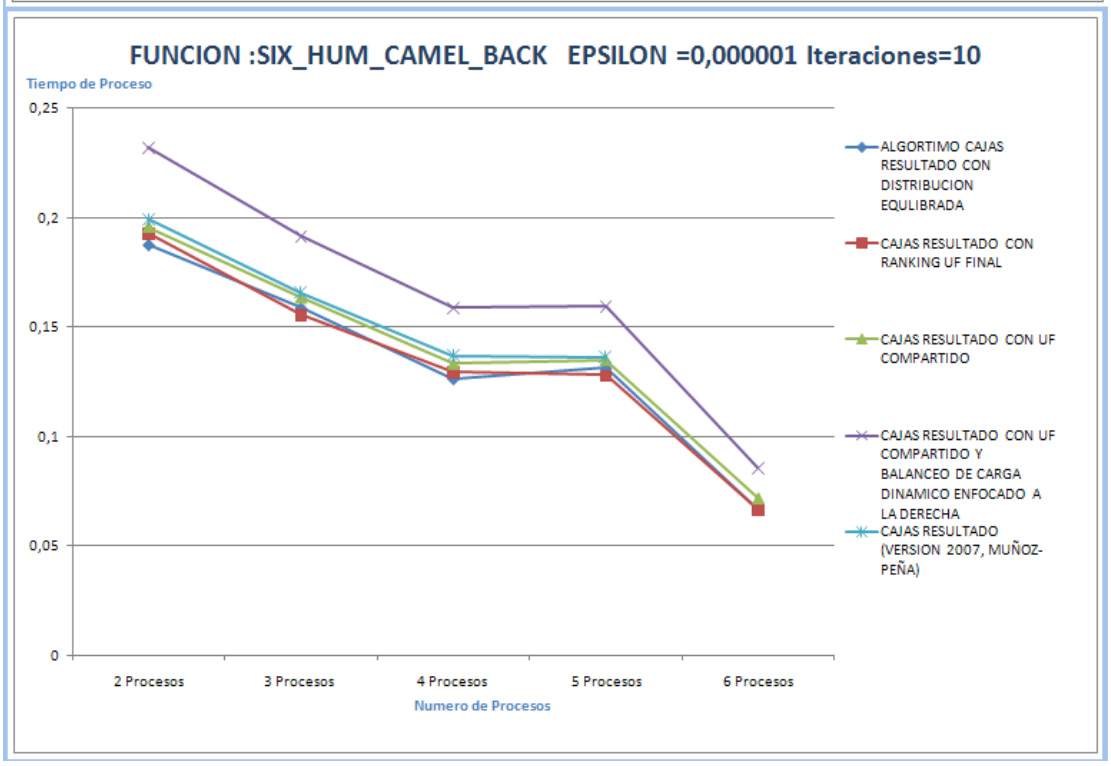
$$t_p = (n/p)\ (1+\mathrm{Log}(n/p)) + p t_{startup} + (n + (p\text{-}1)\ (n/p^2))\ t_{data}$$

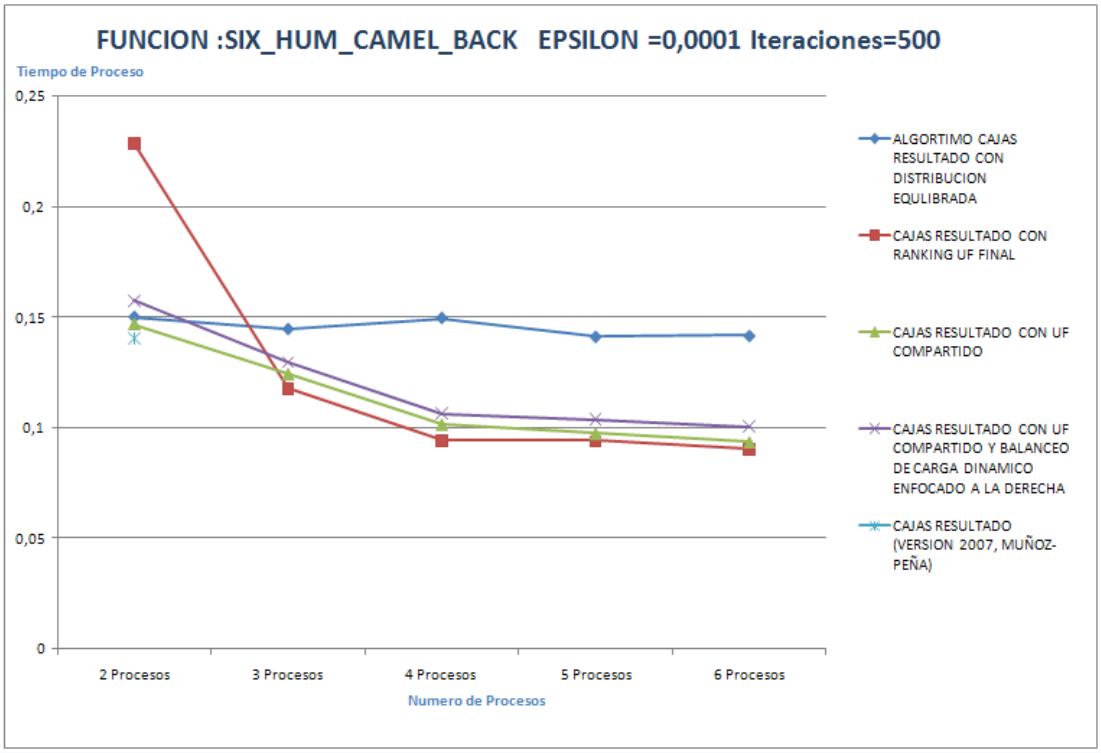
ANEXO U: GRAFICOS DE ANALIZIS DE ALGORTIMOS

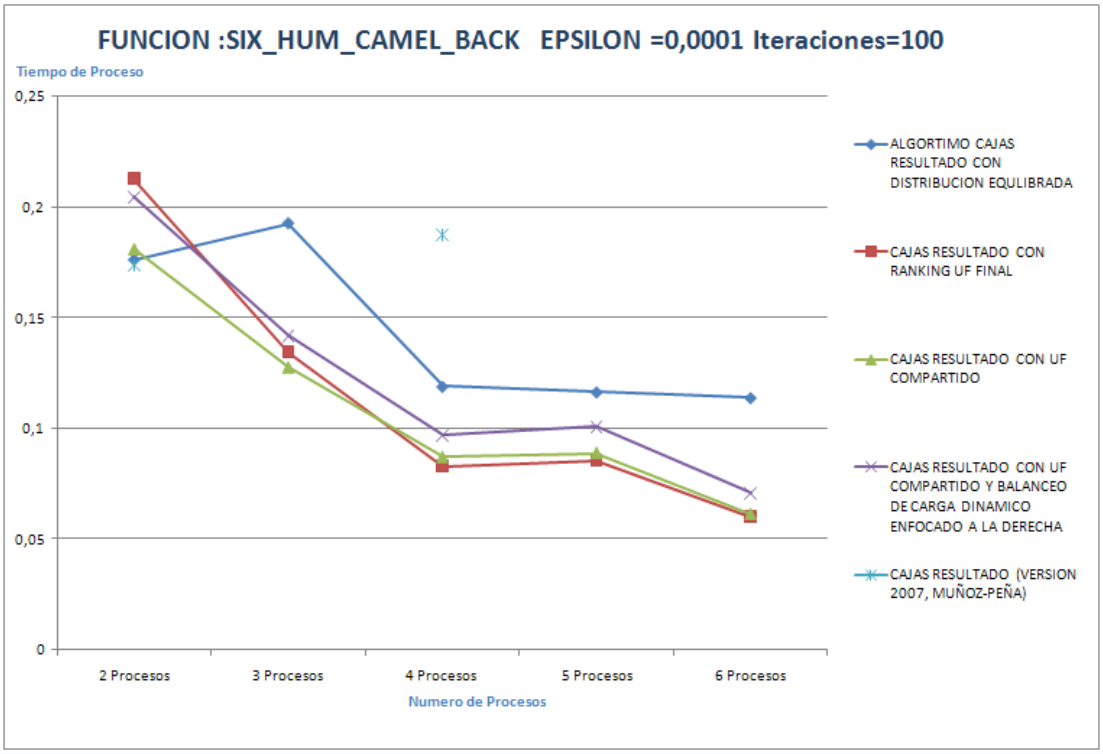


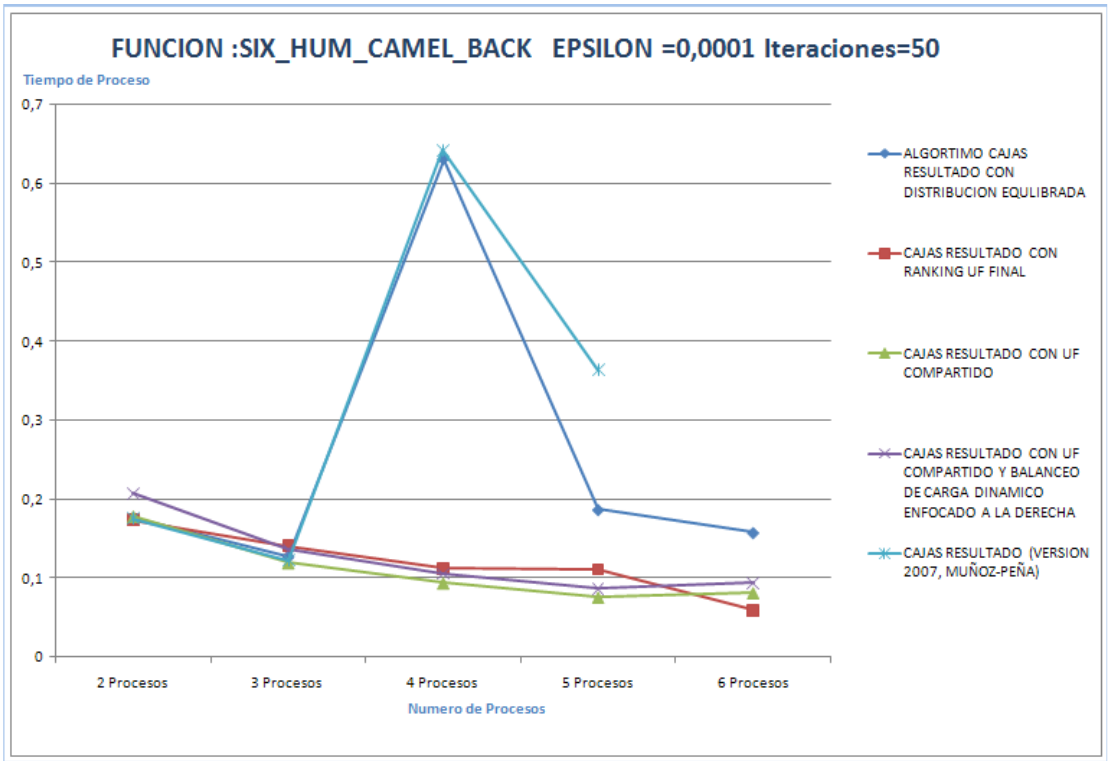


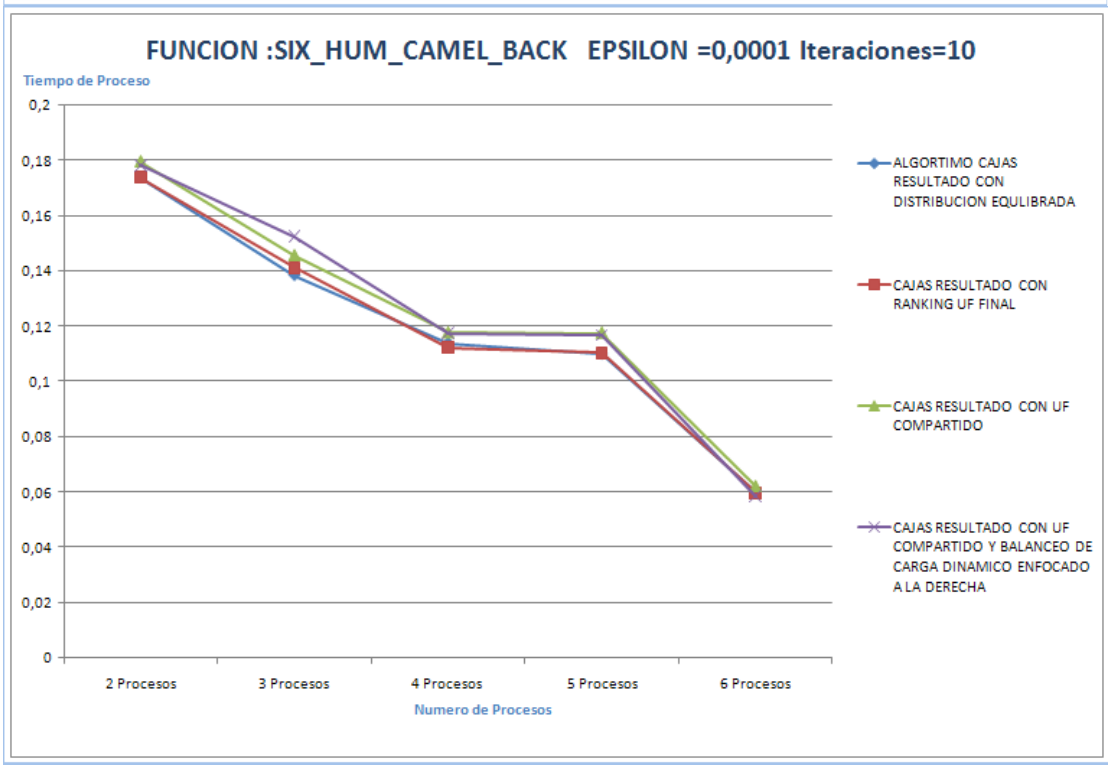


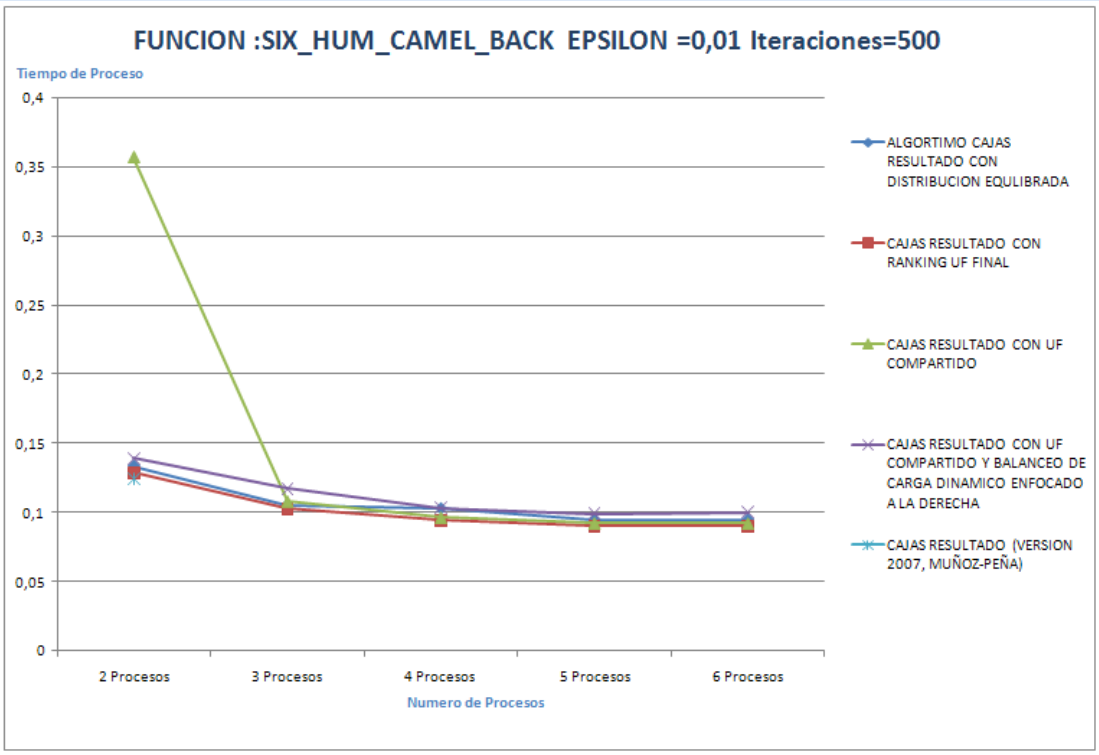


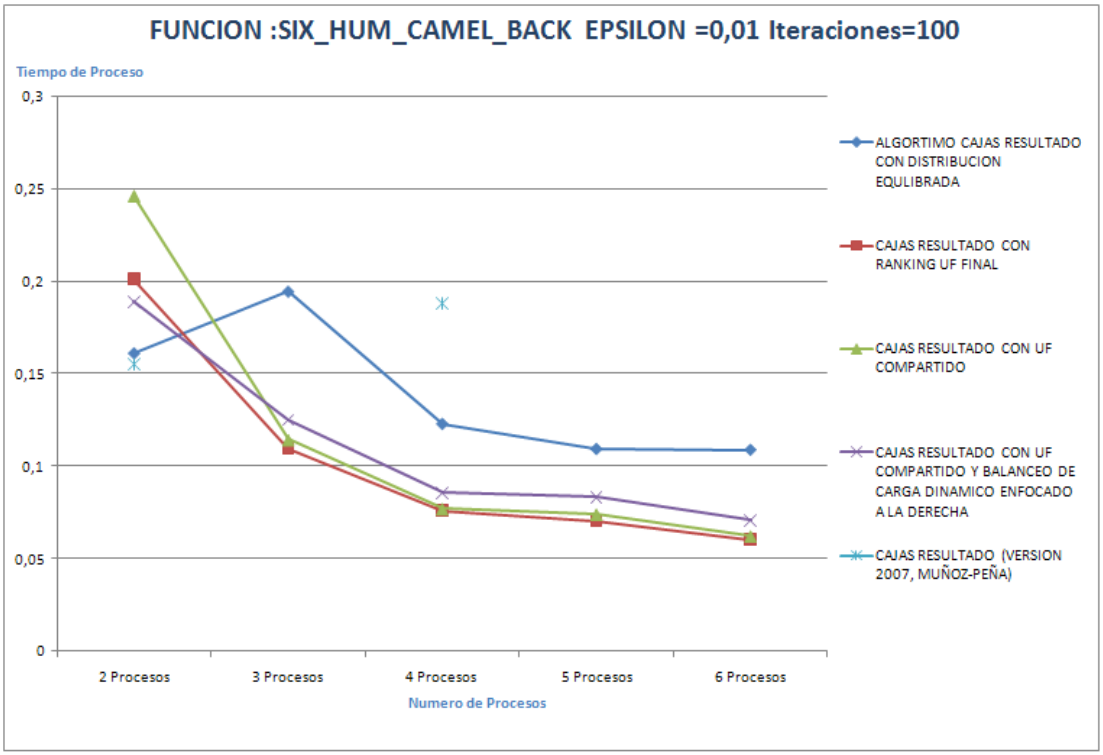


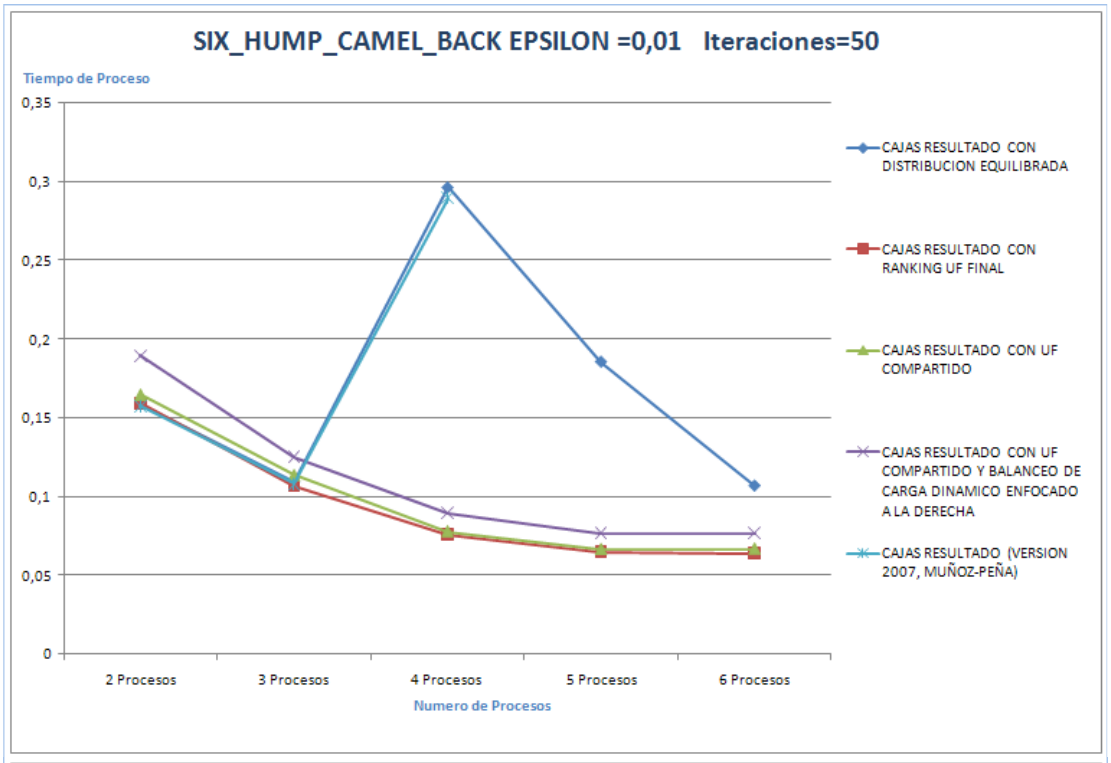


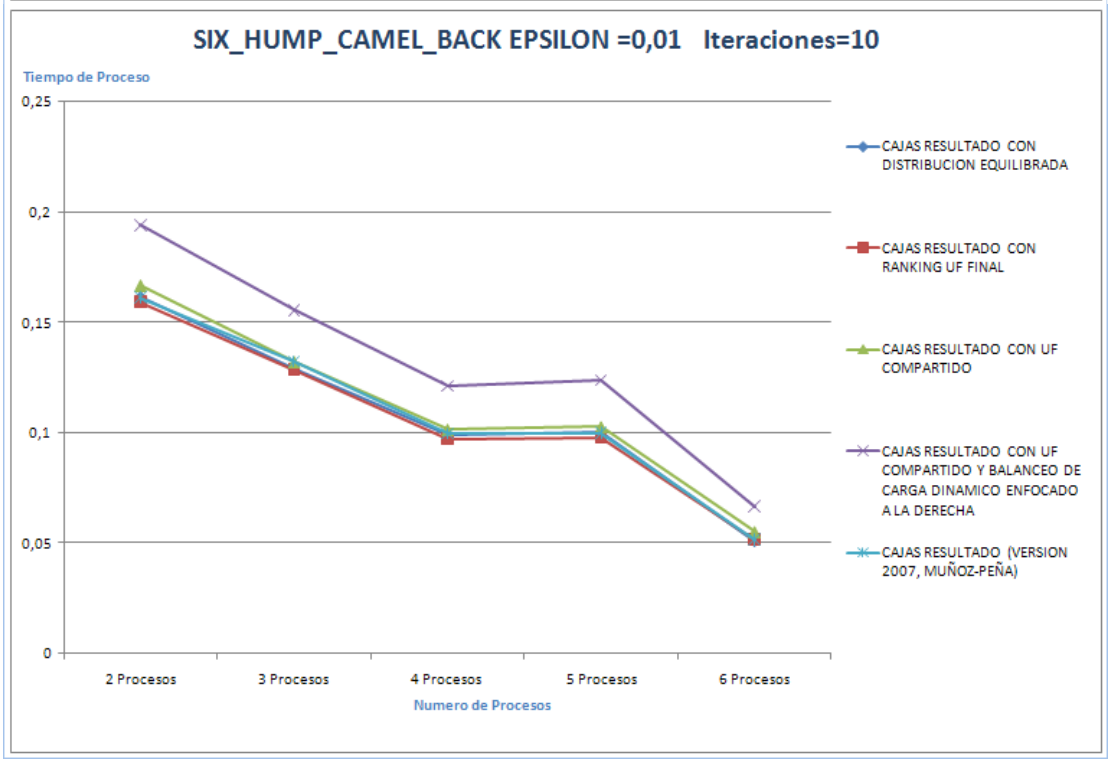


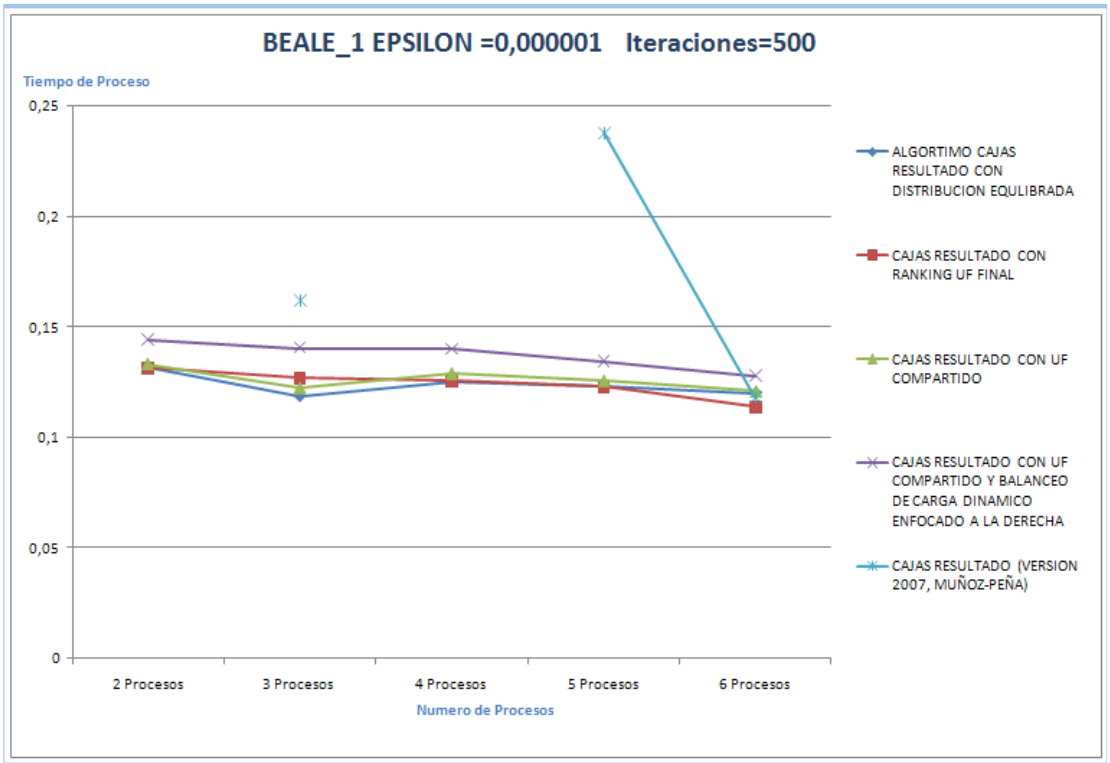


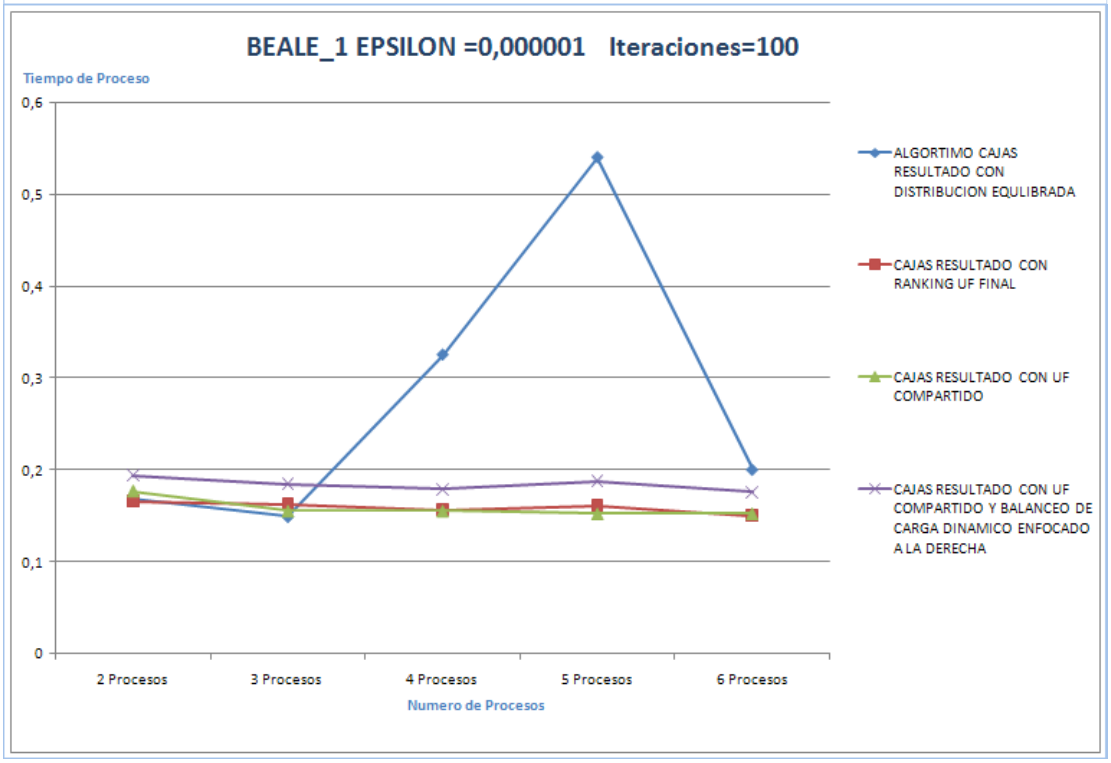


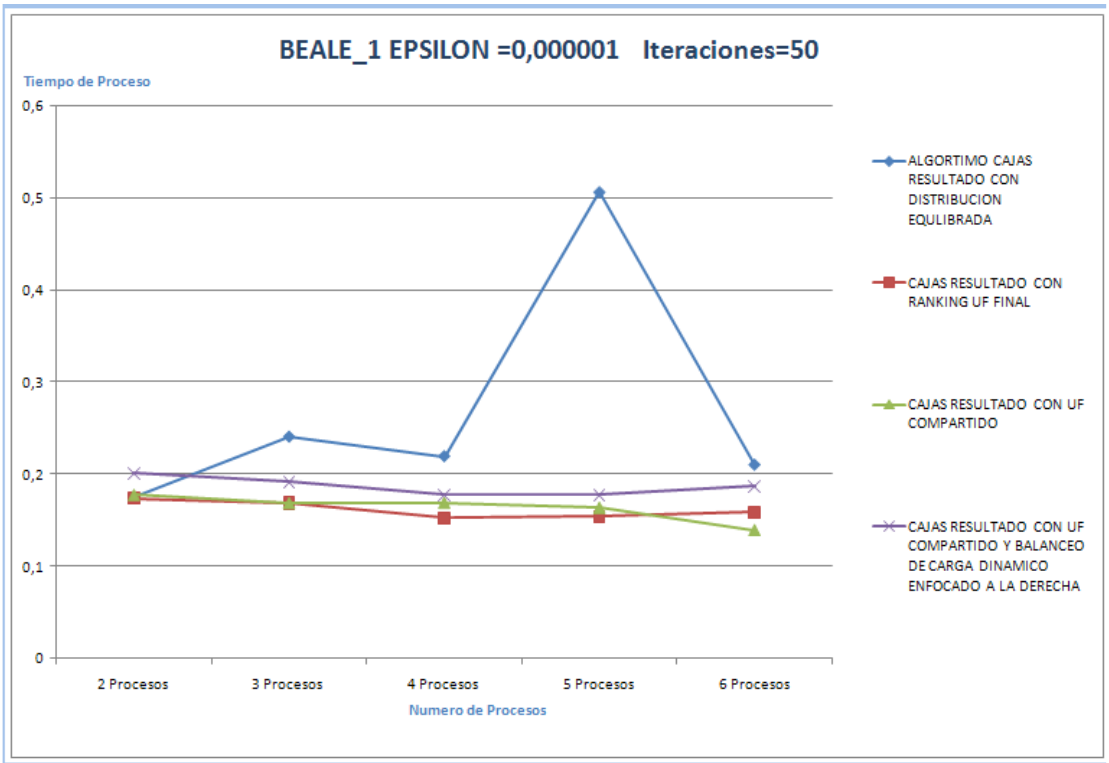


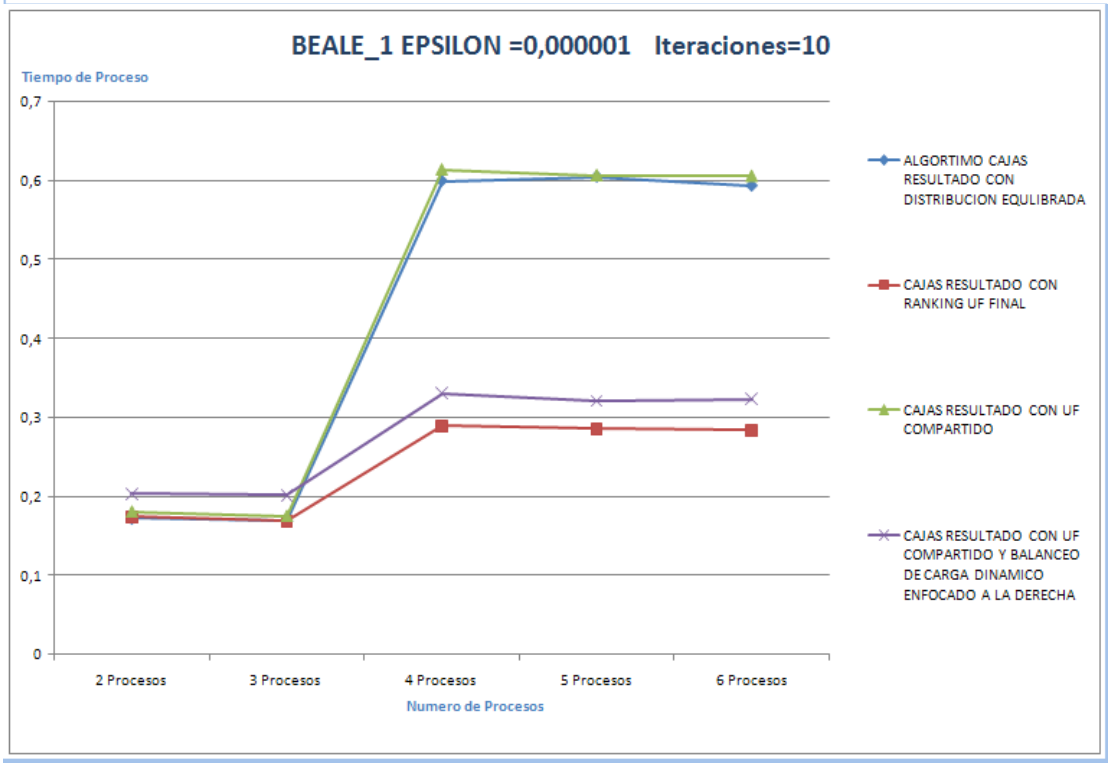


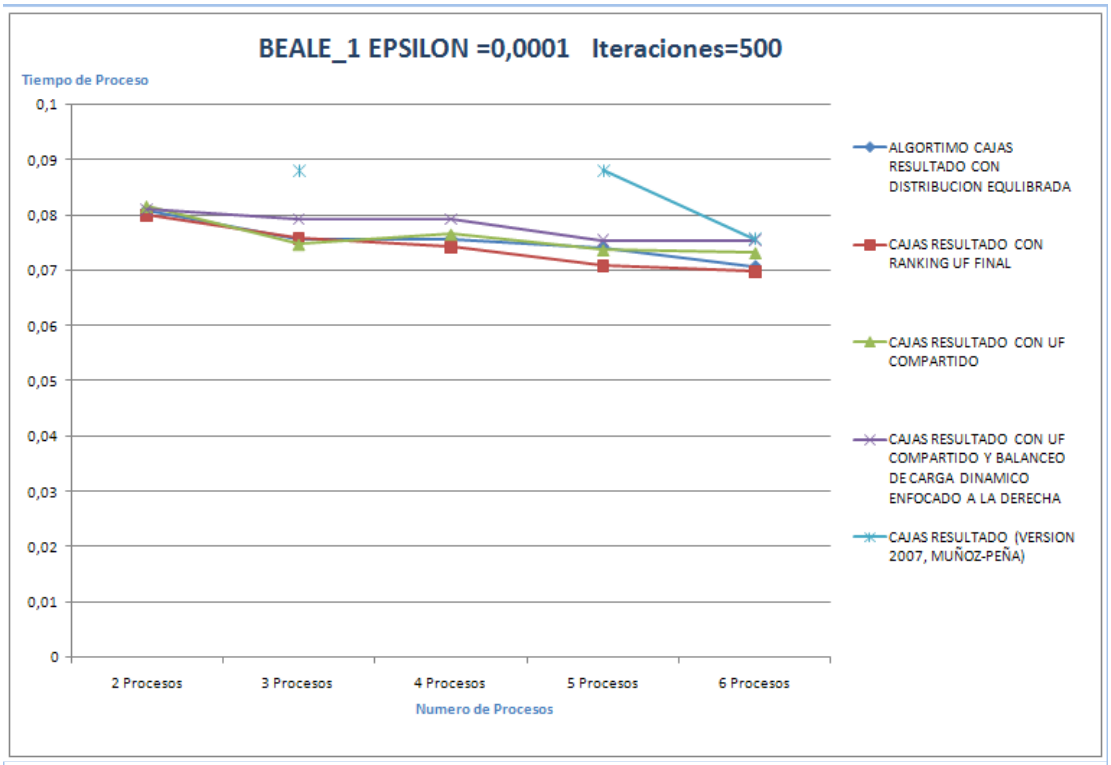


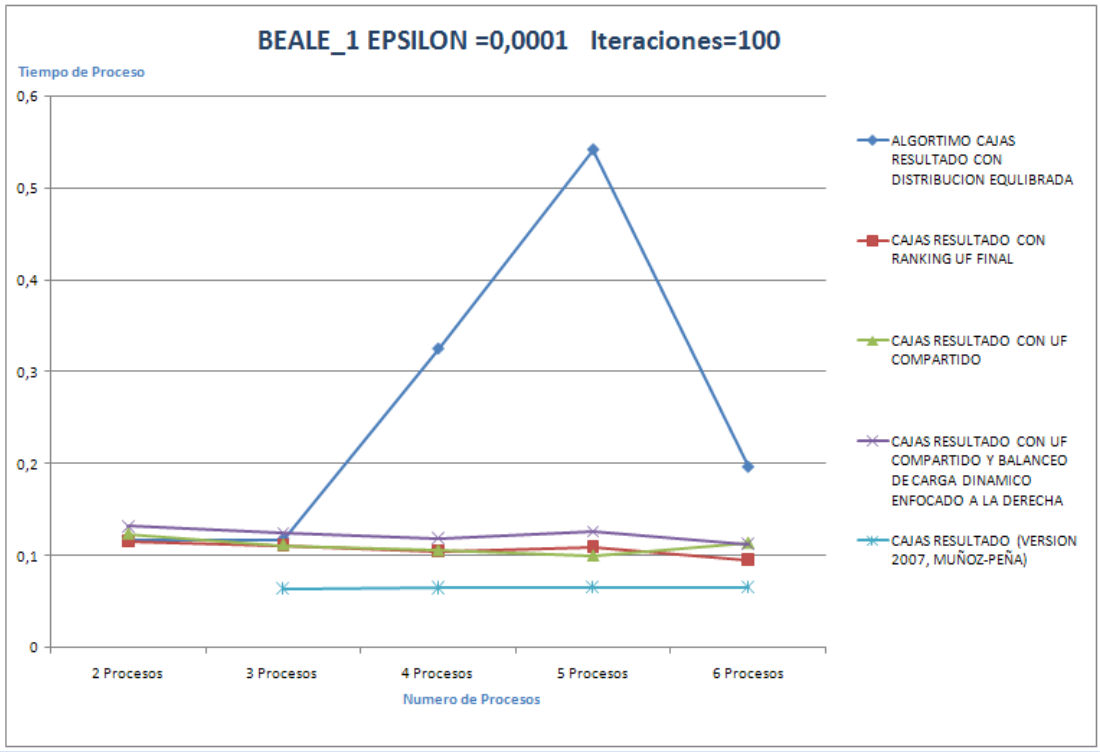


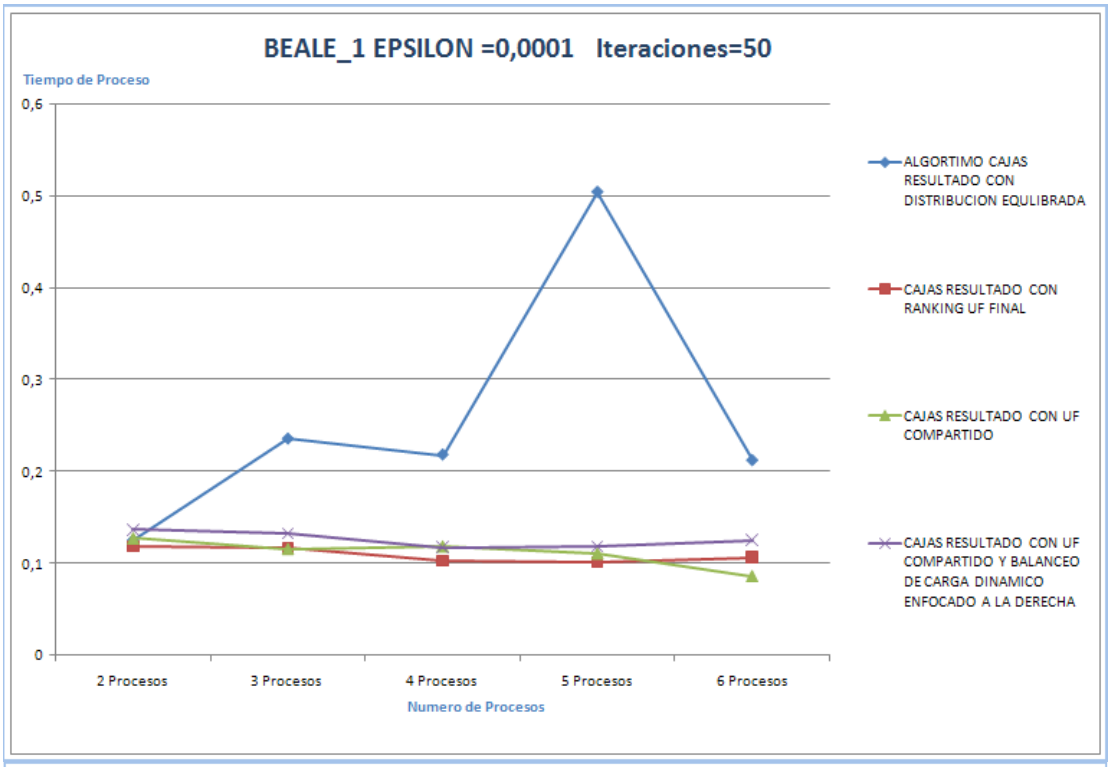


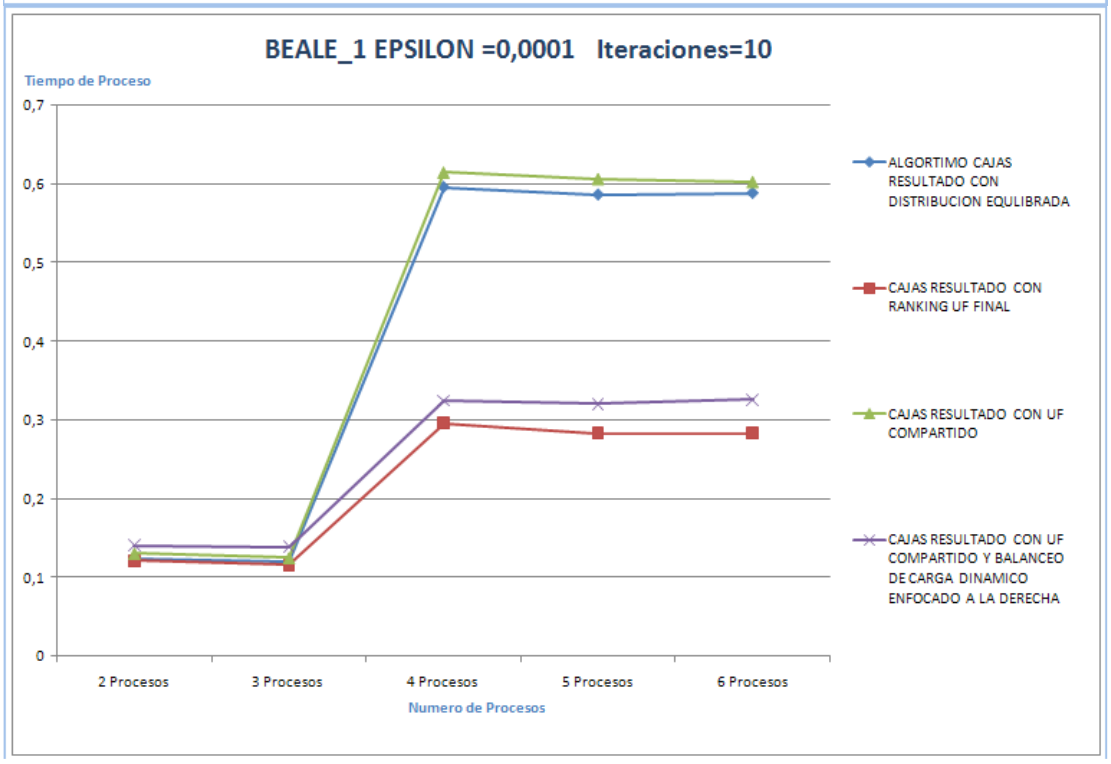


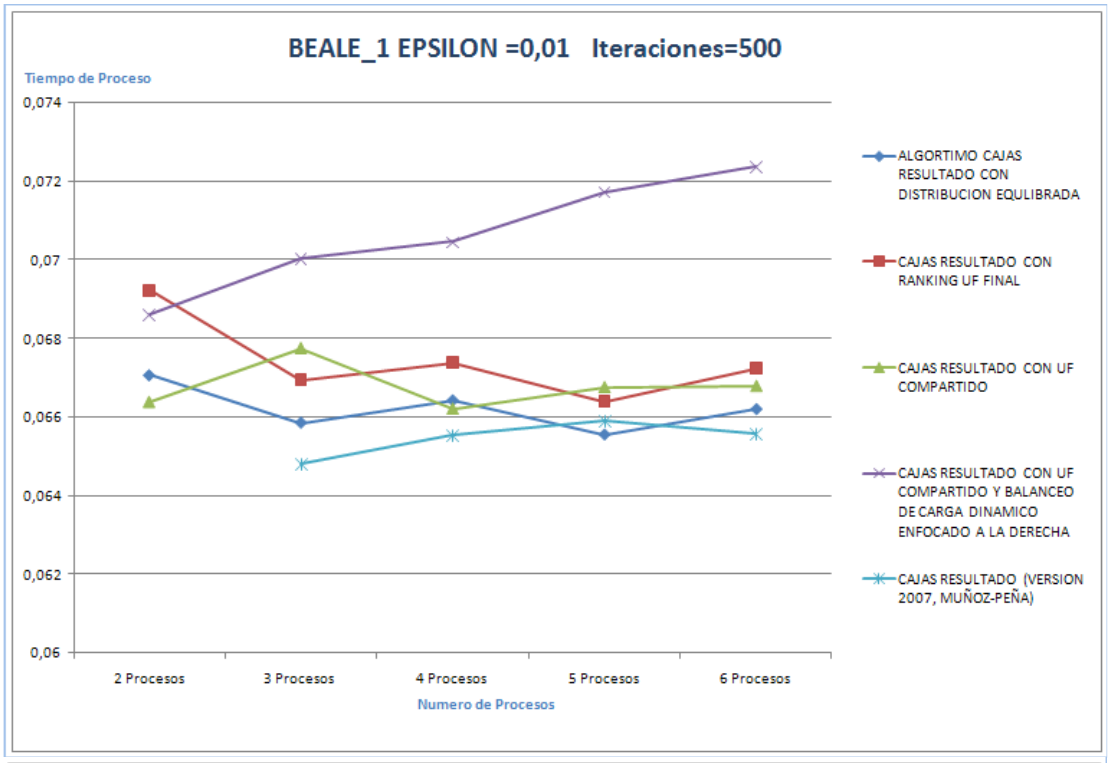


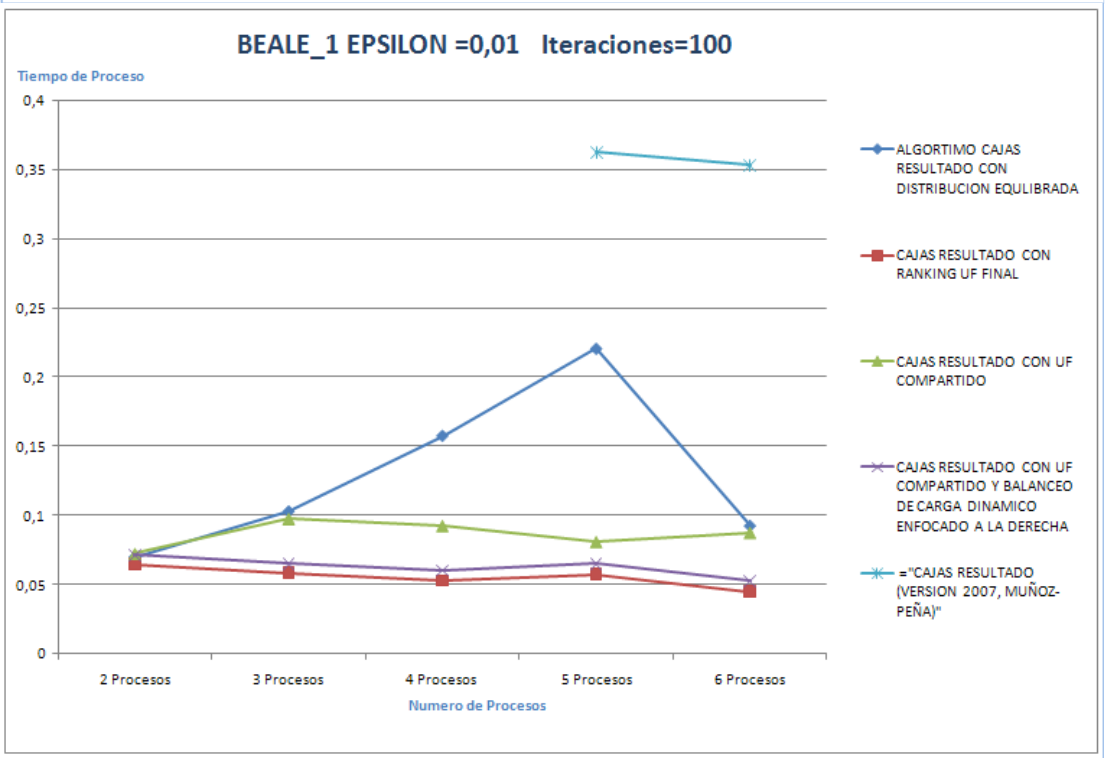


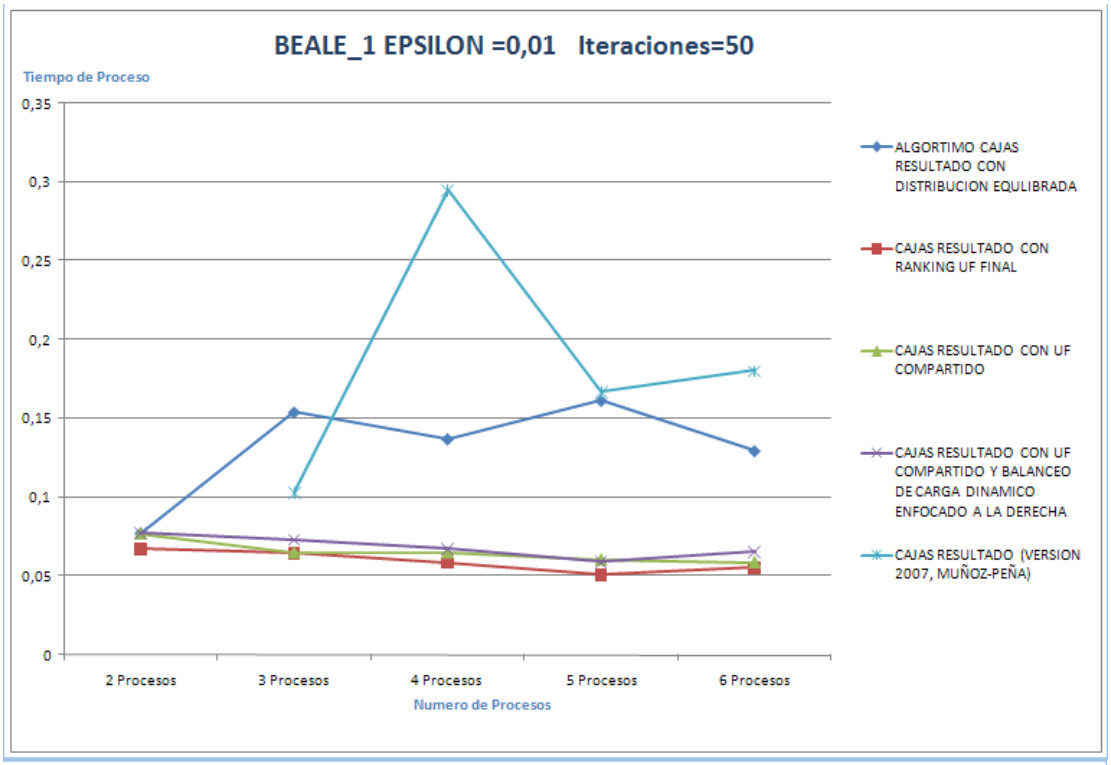


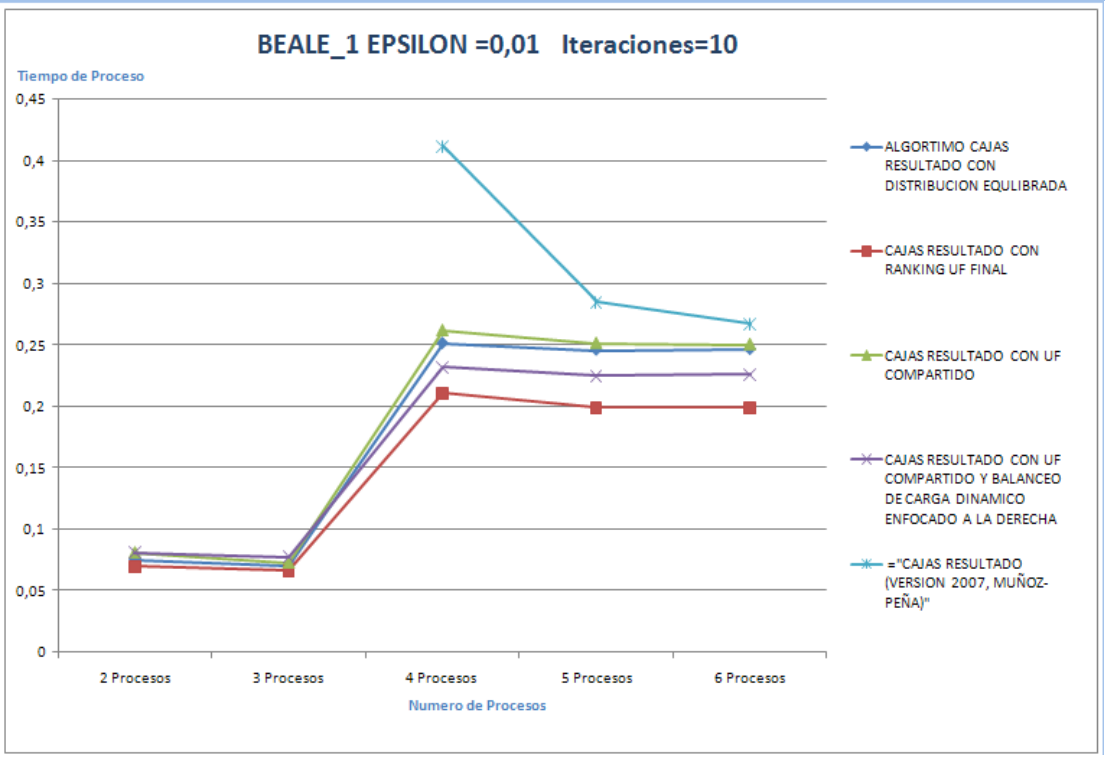


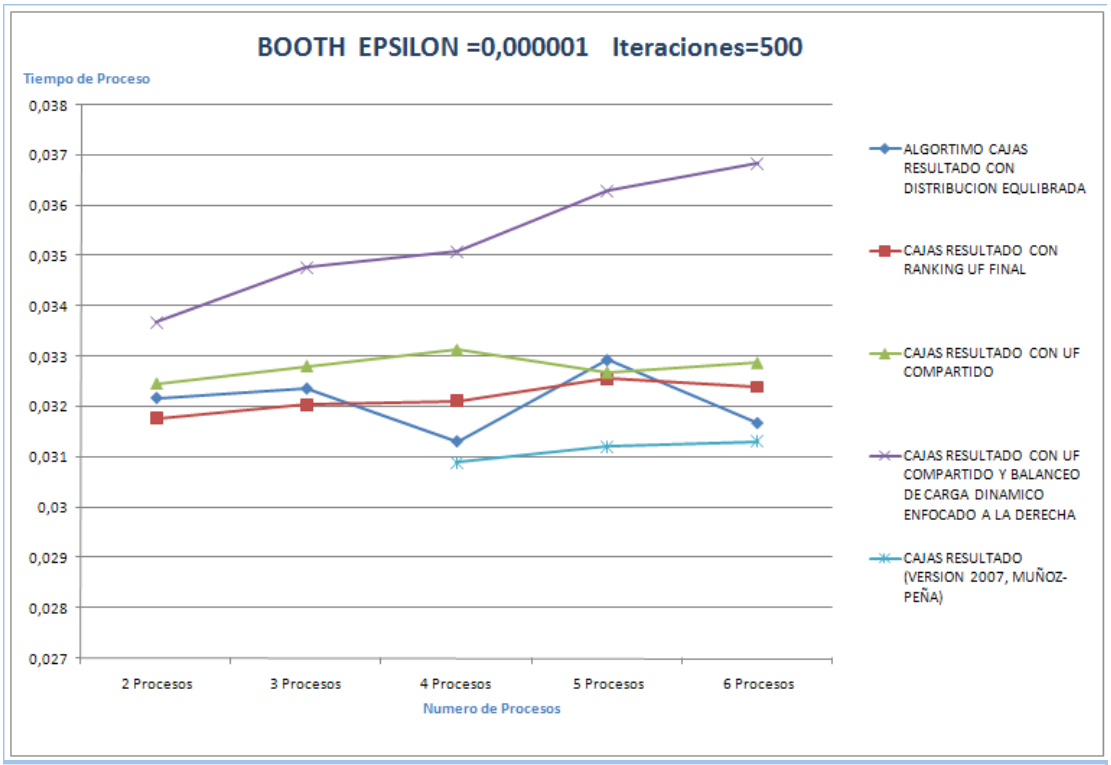


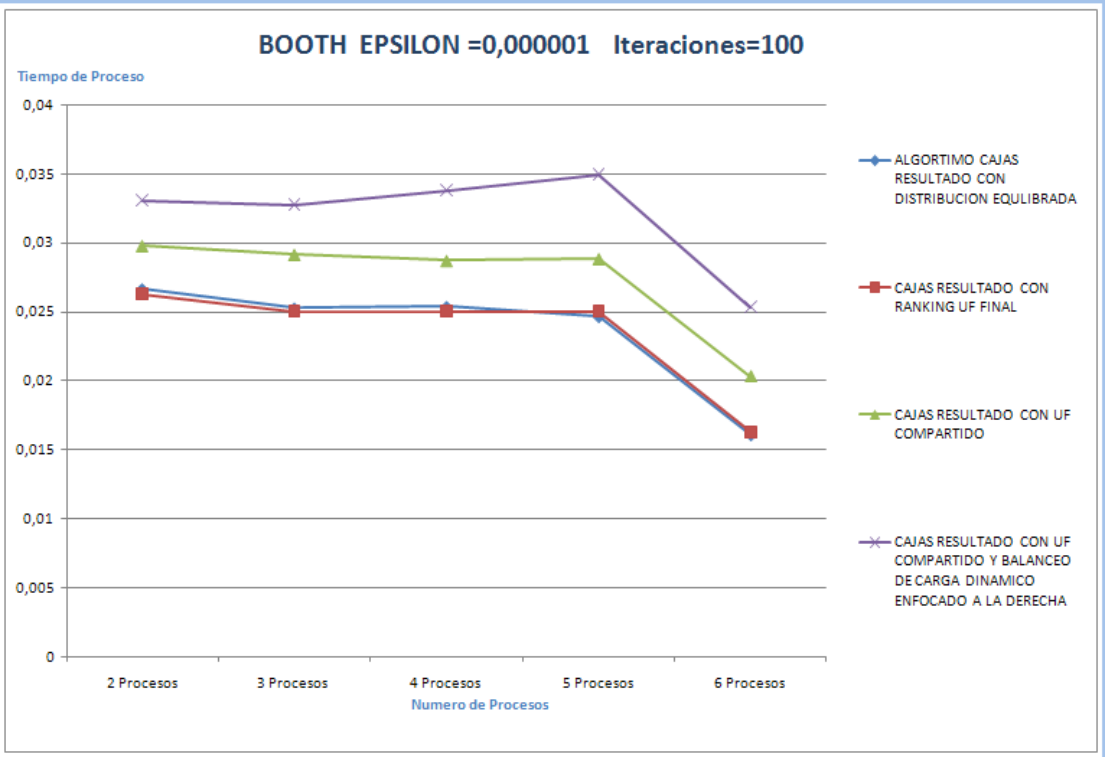


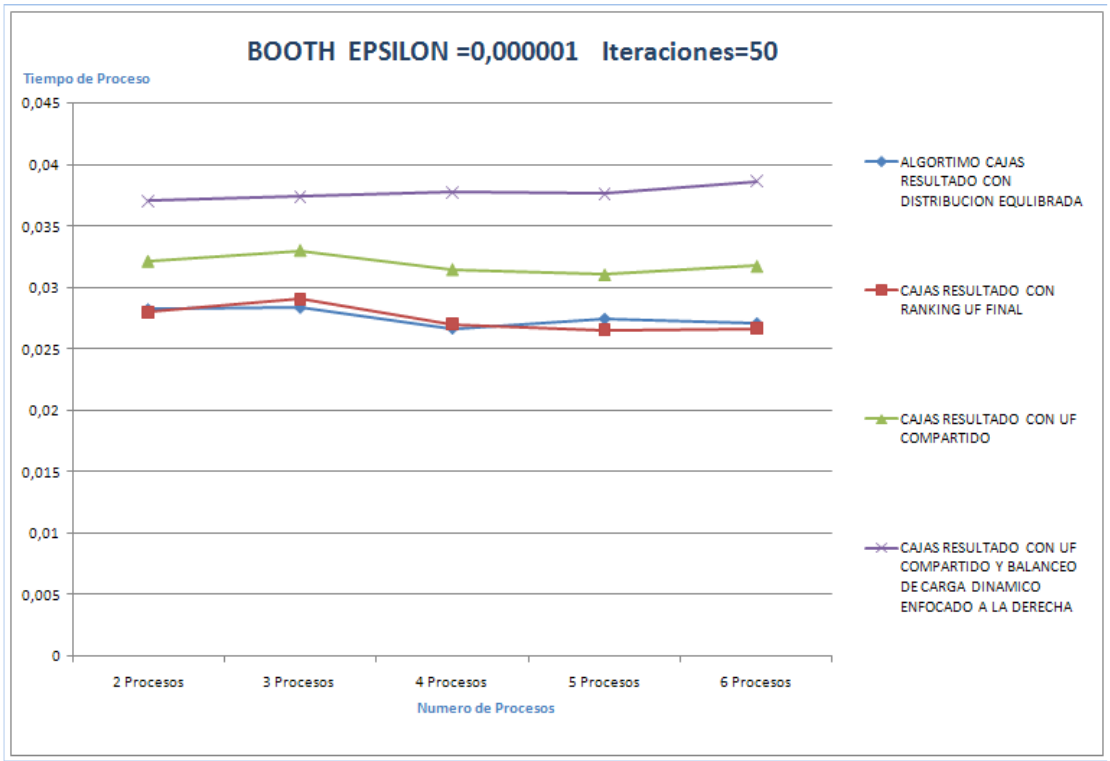


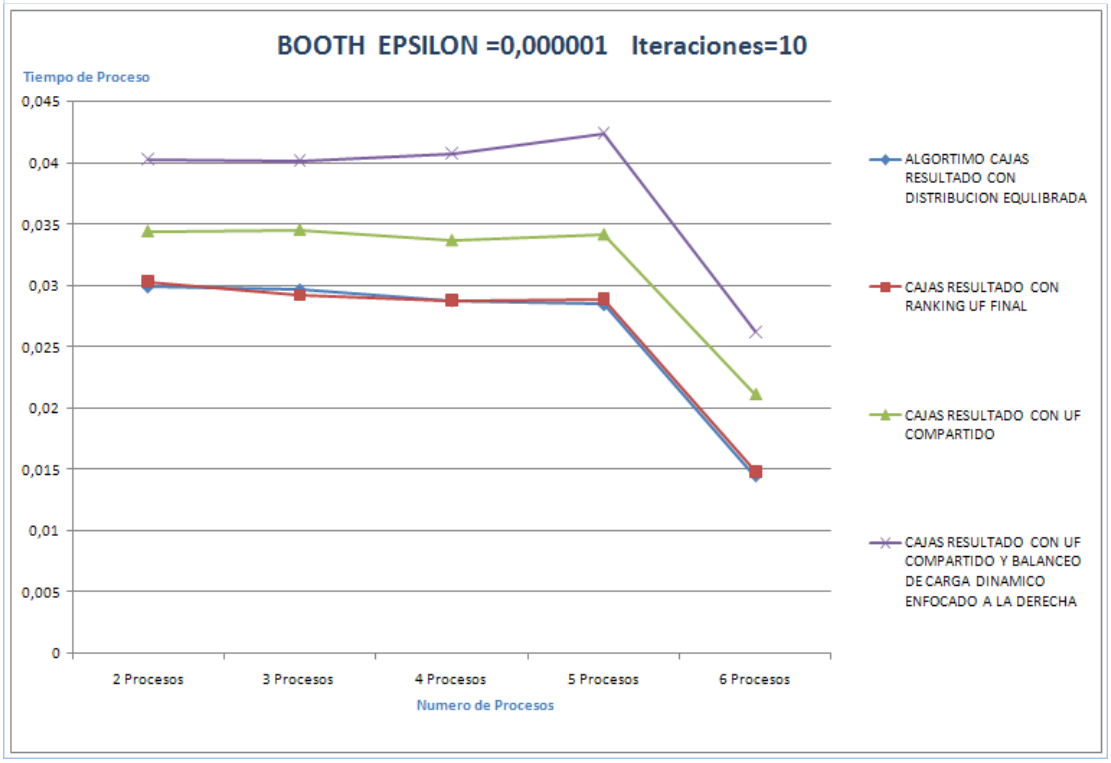


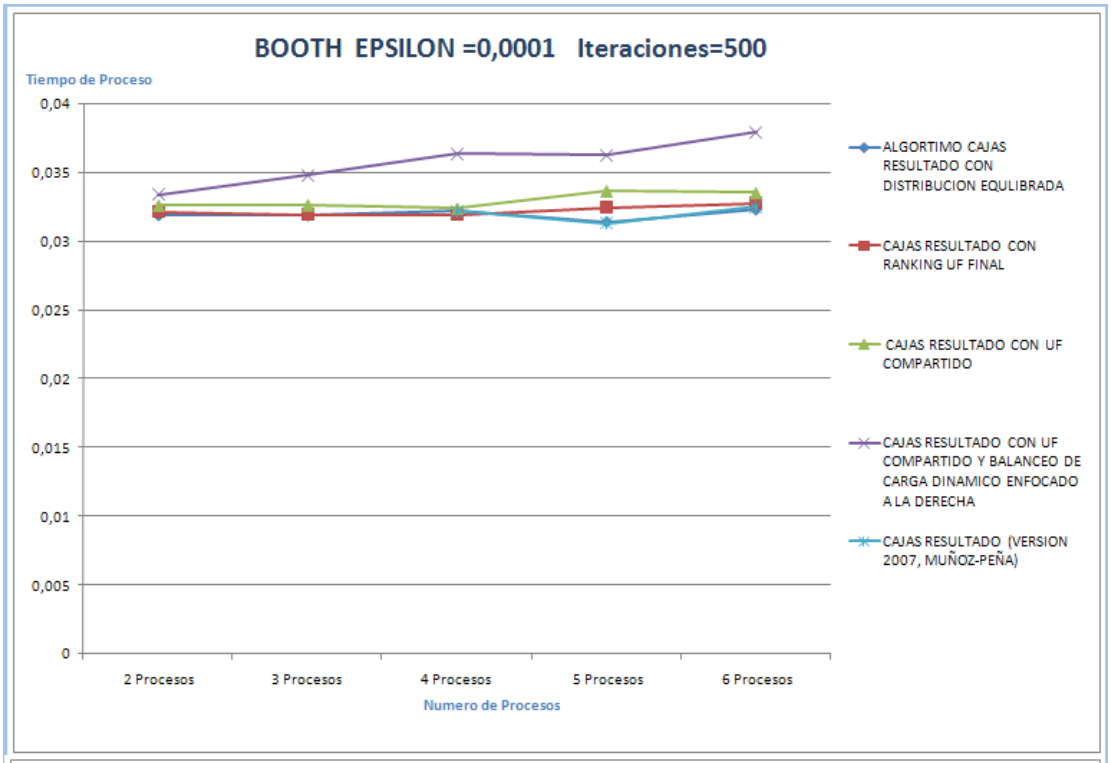


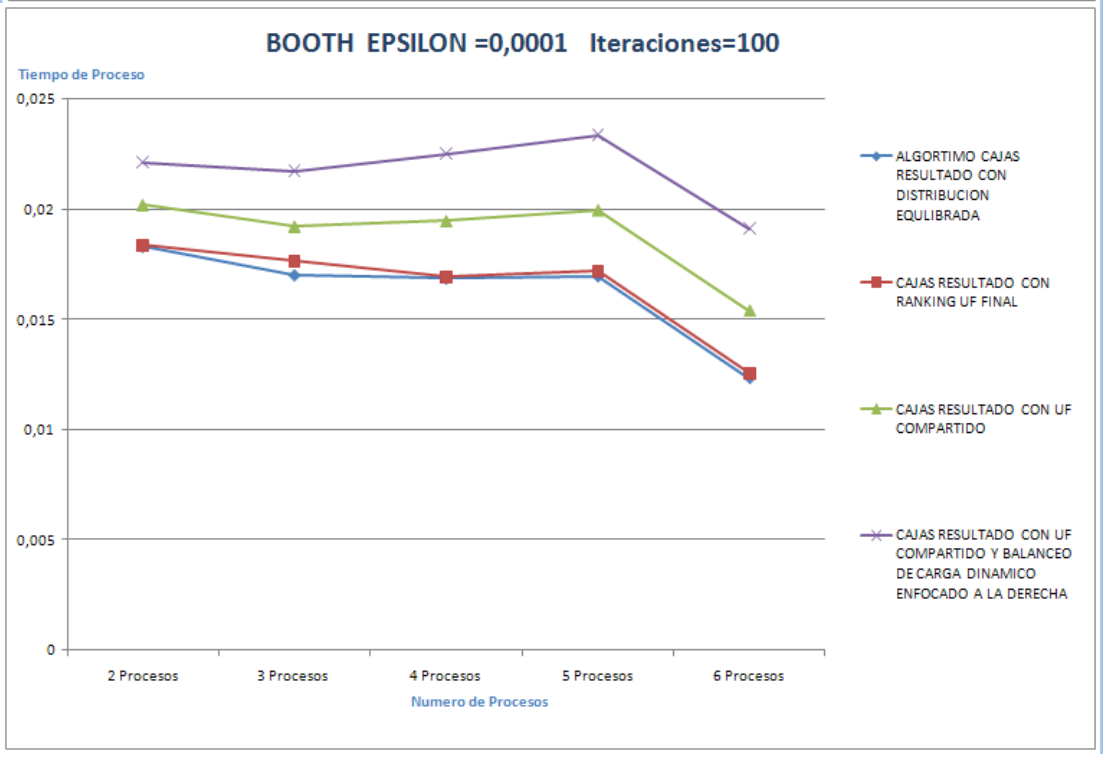


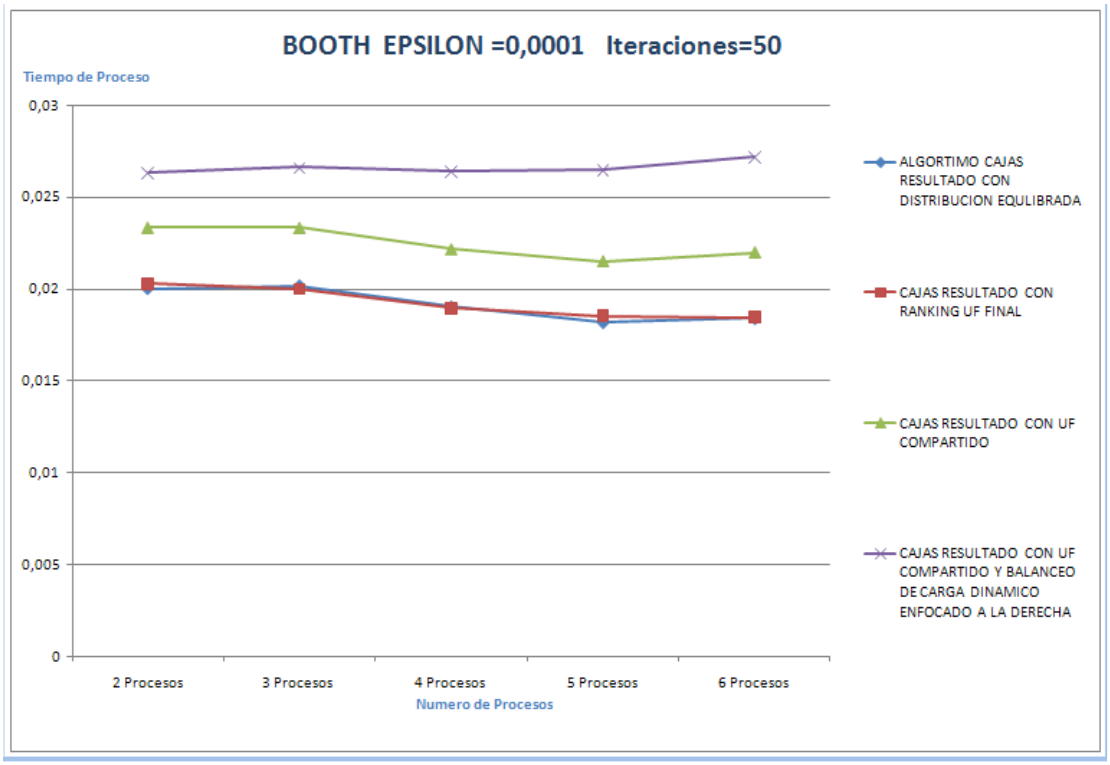


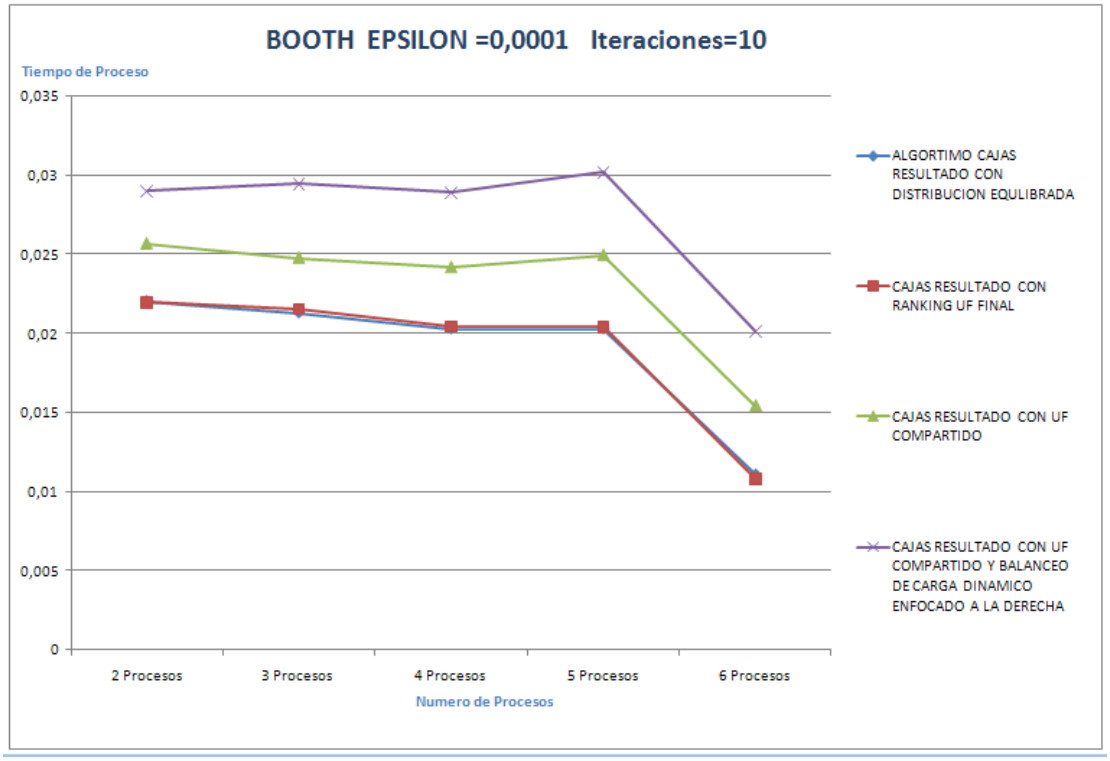


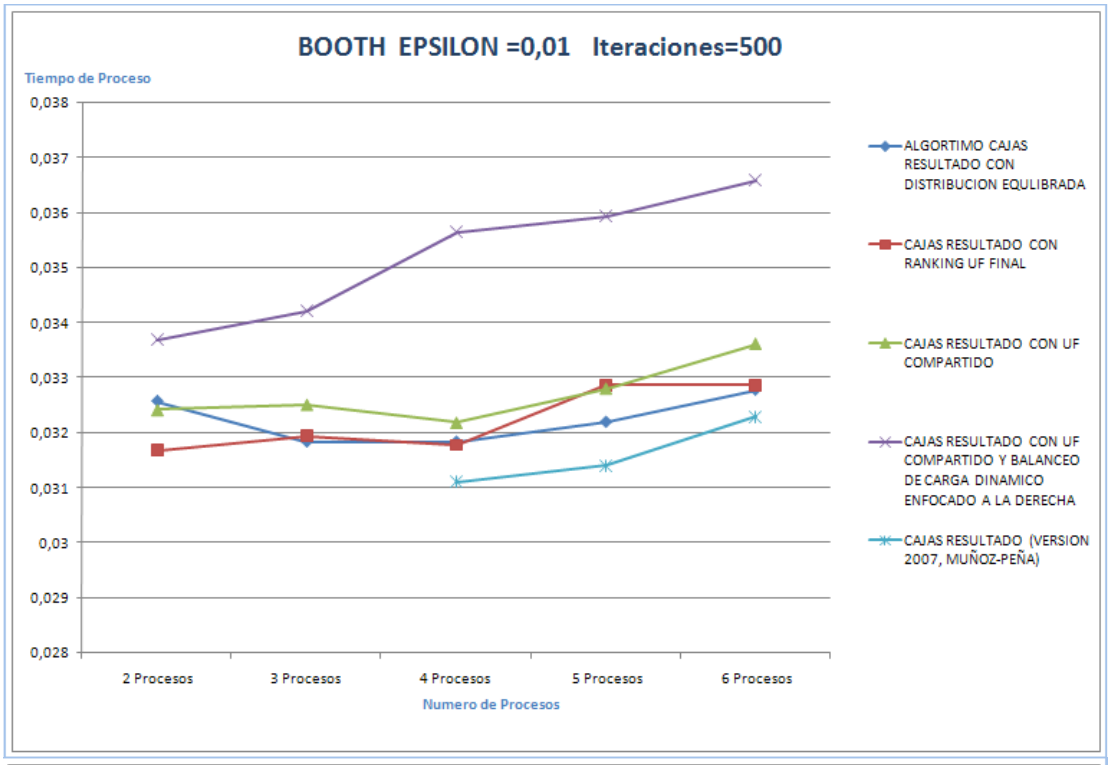


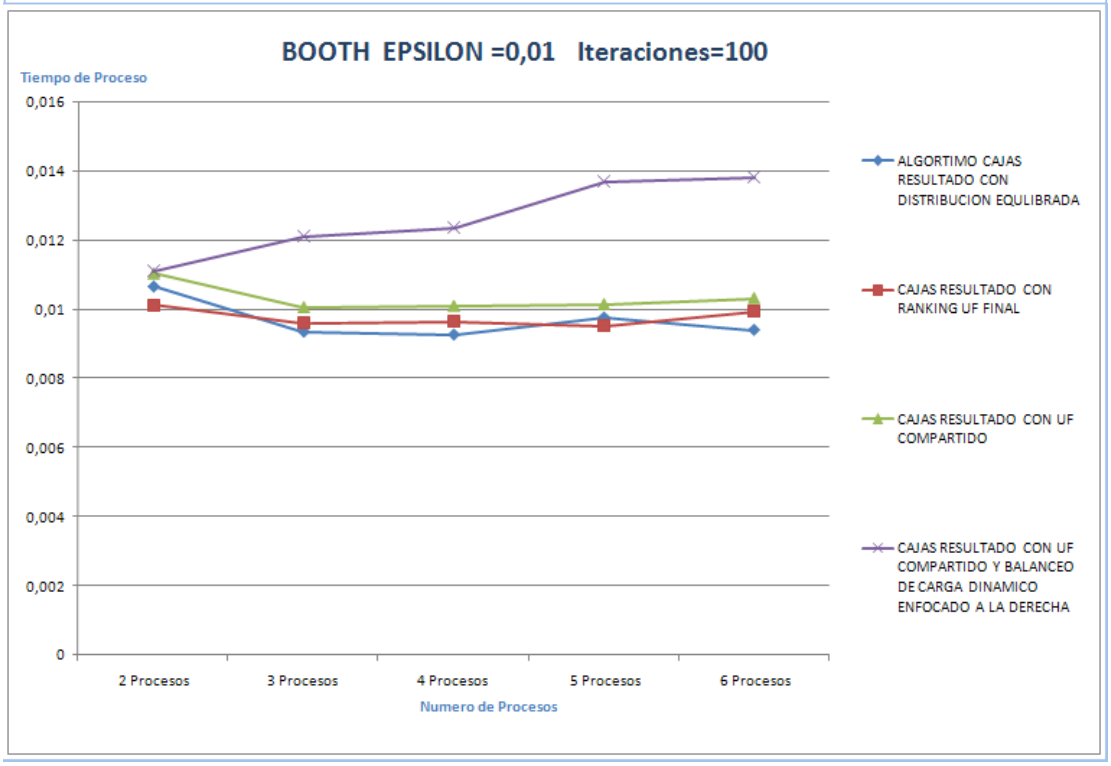


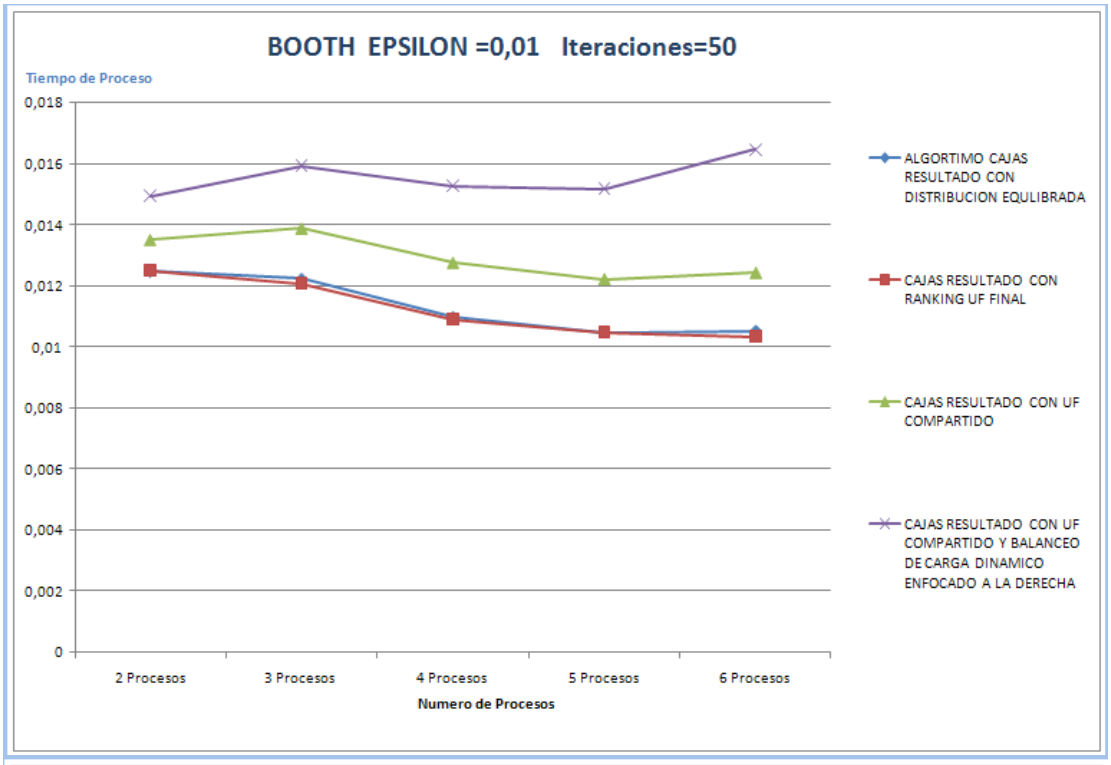


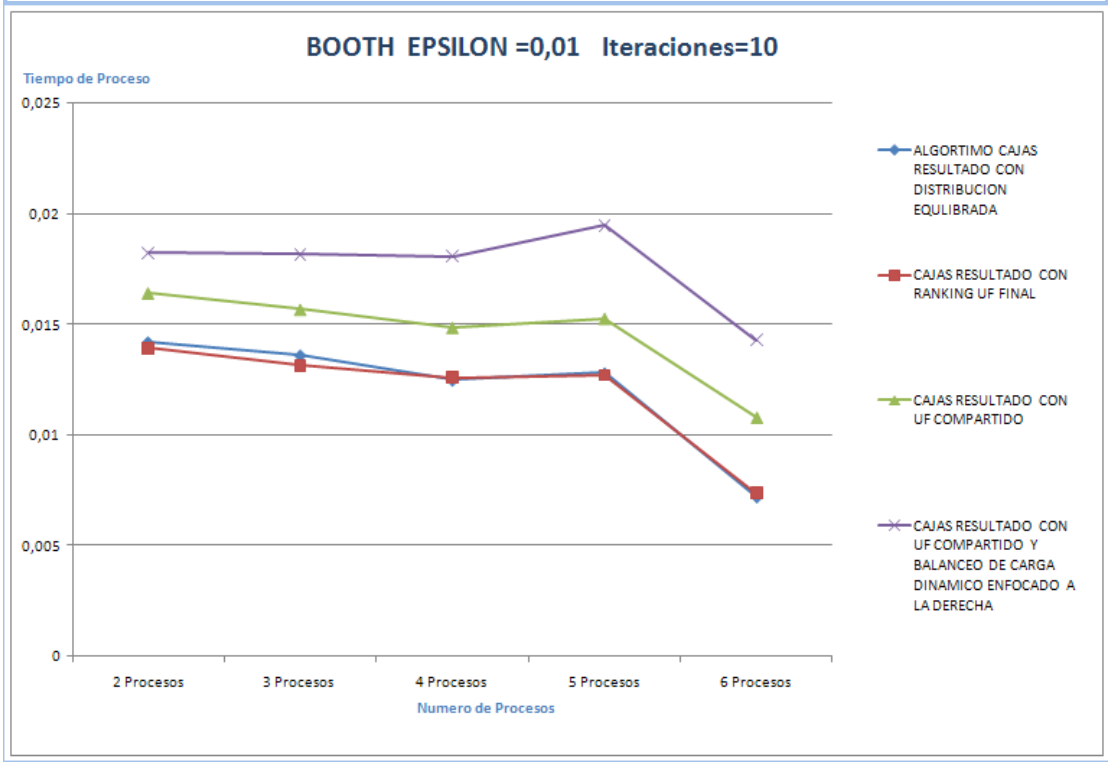


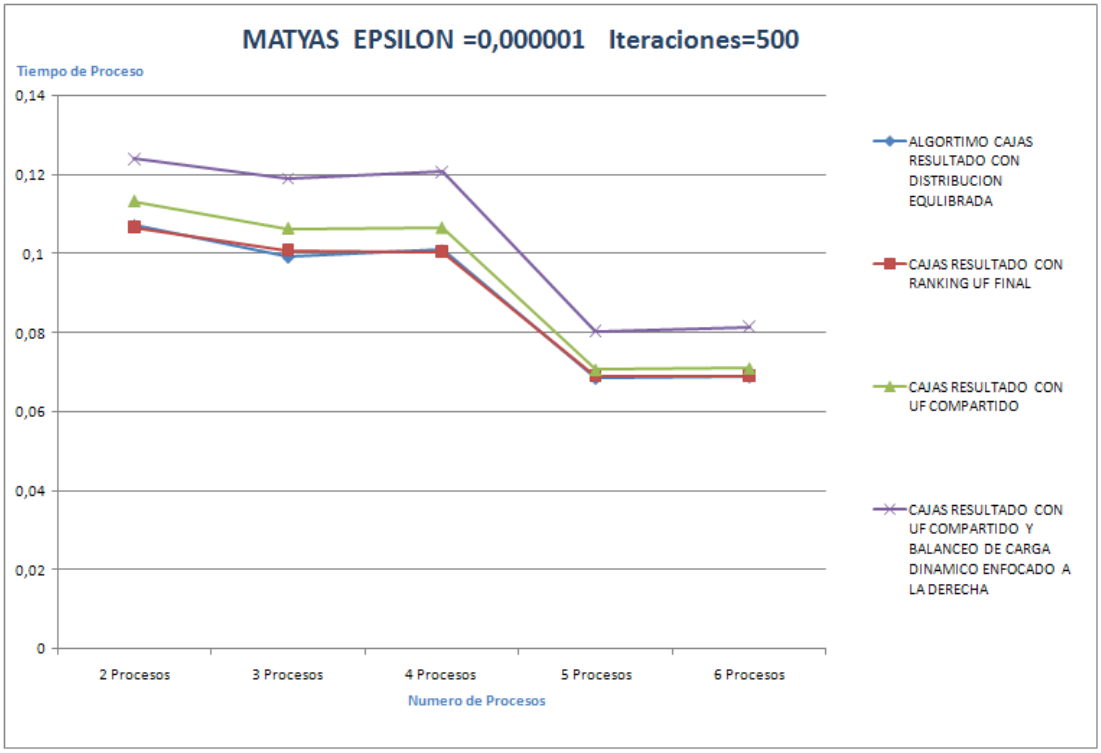


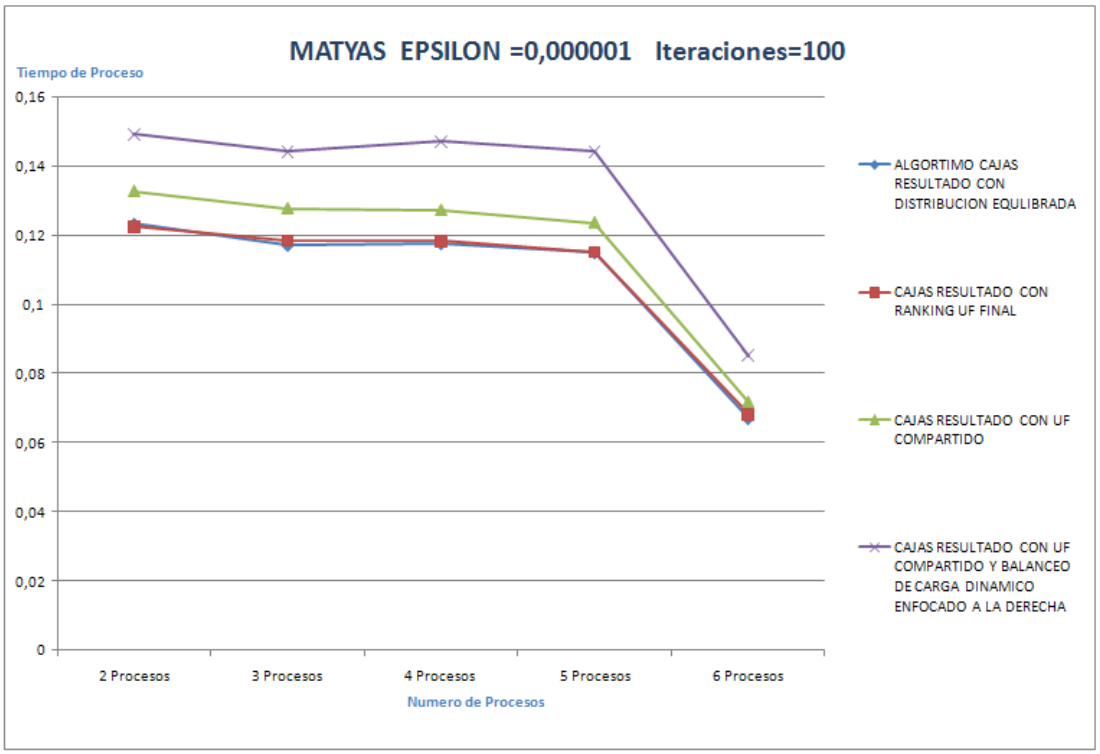


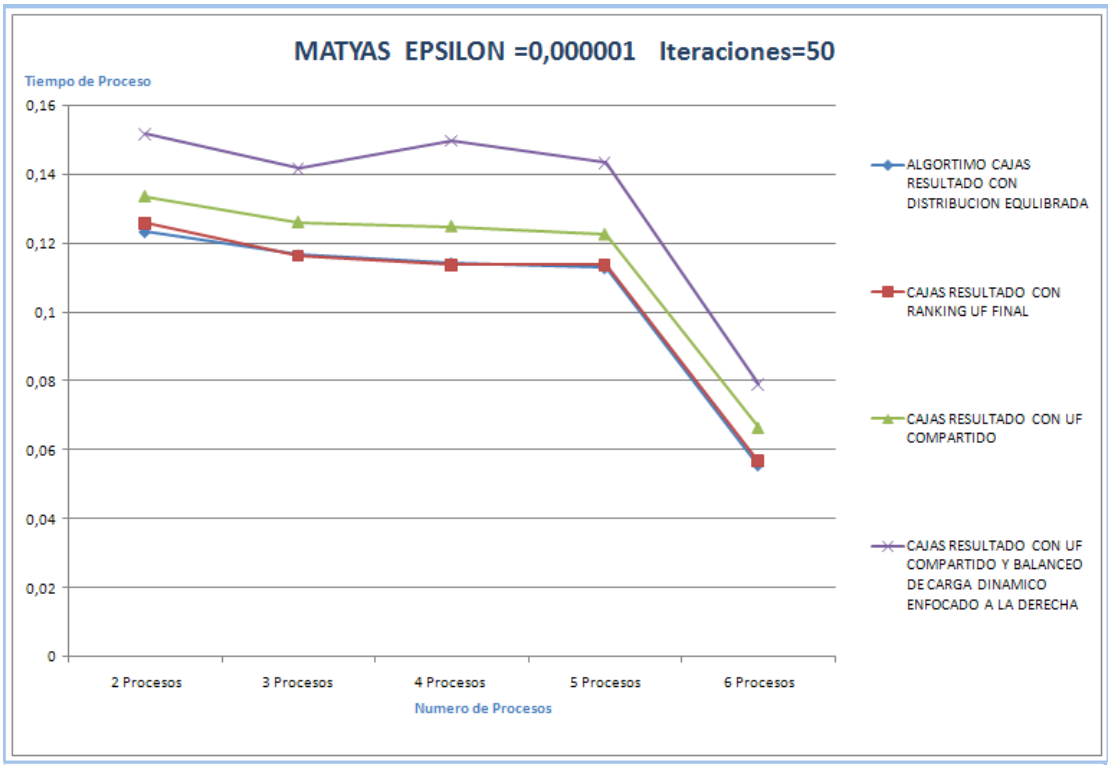


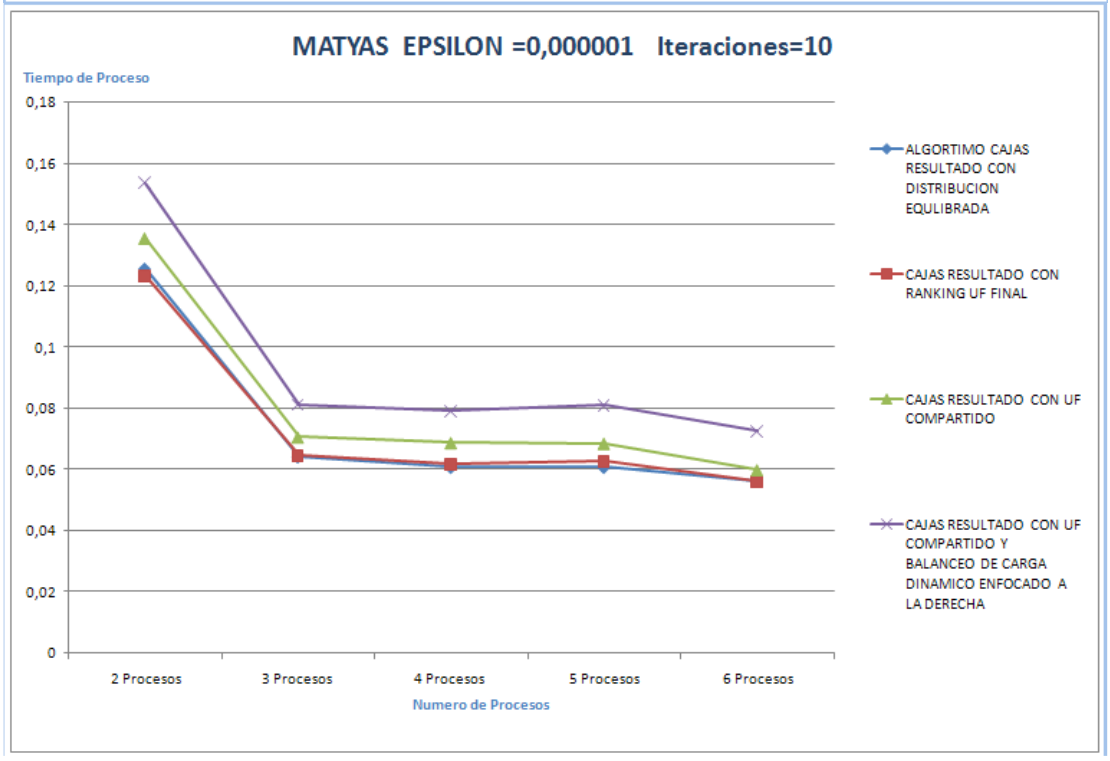


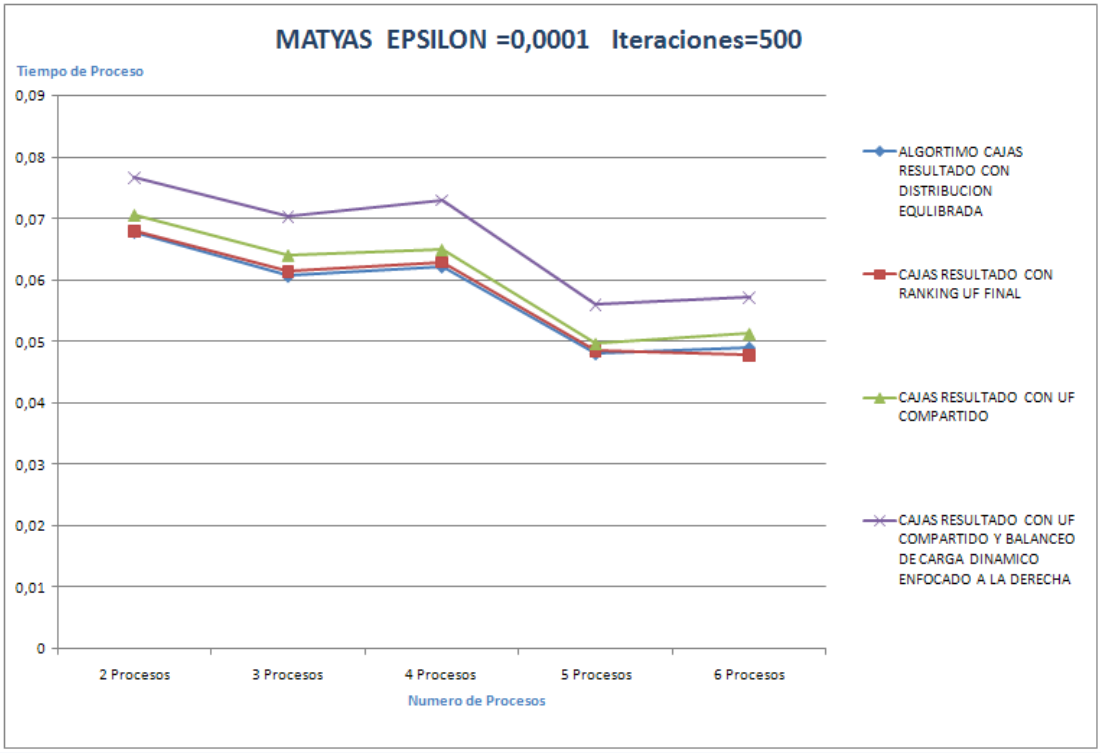


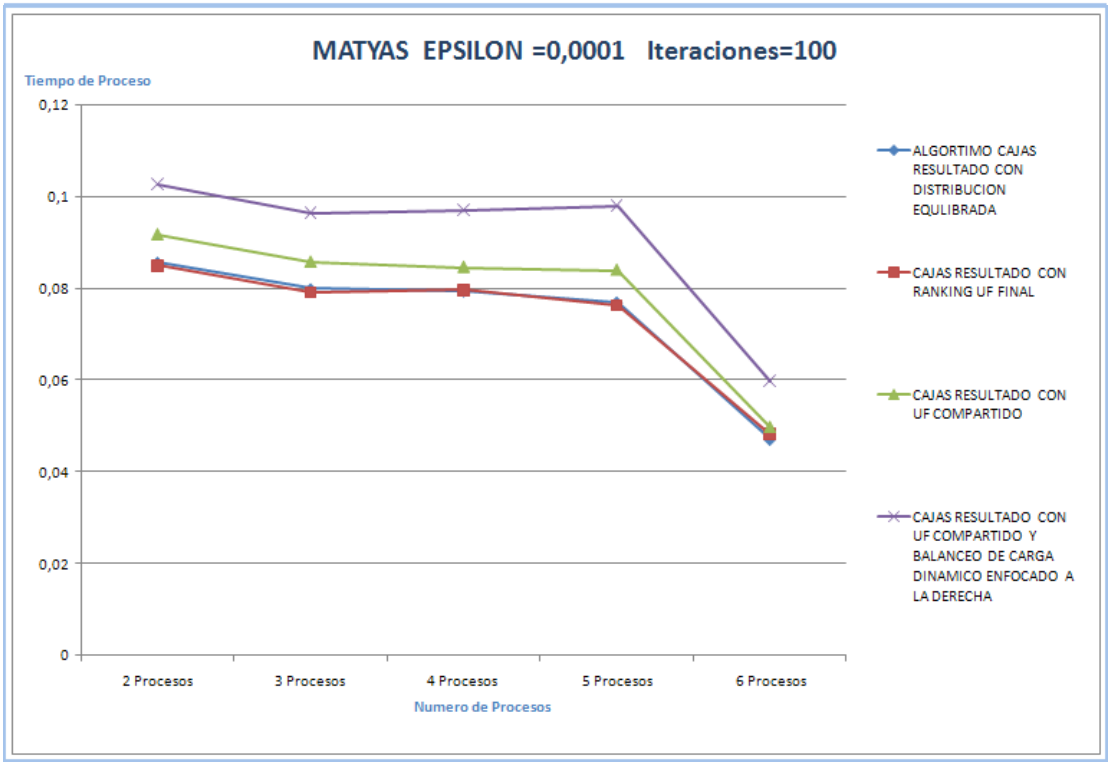


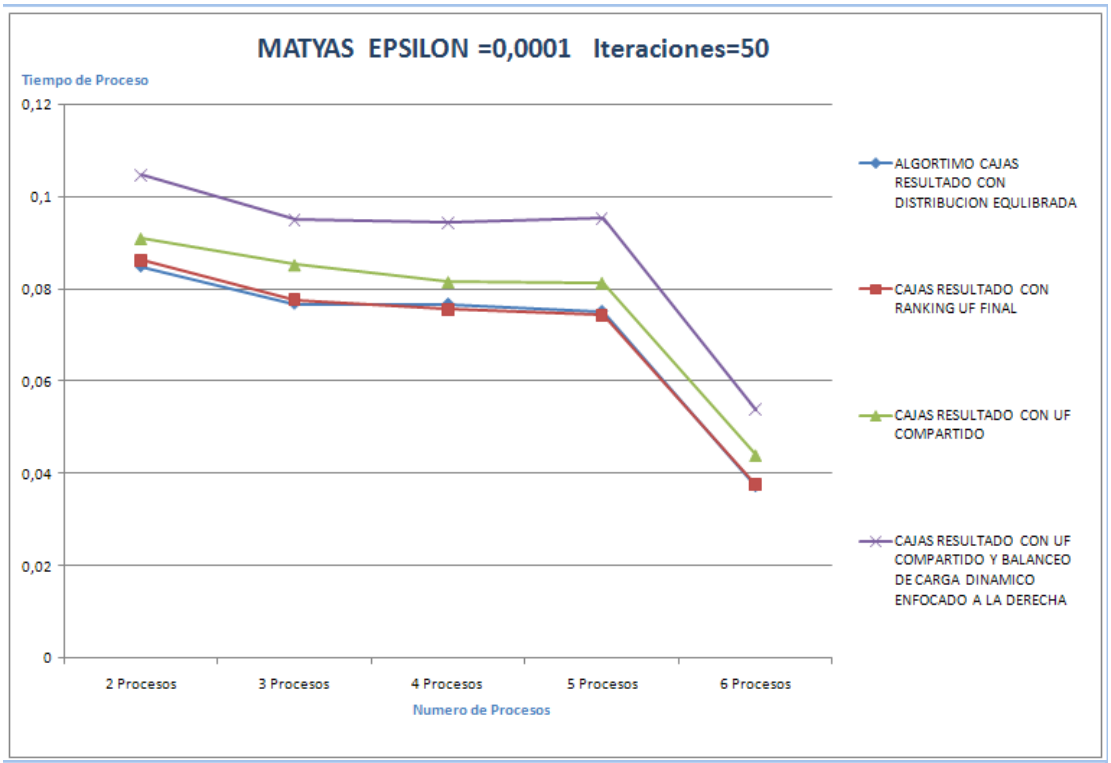


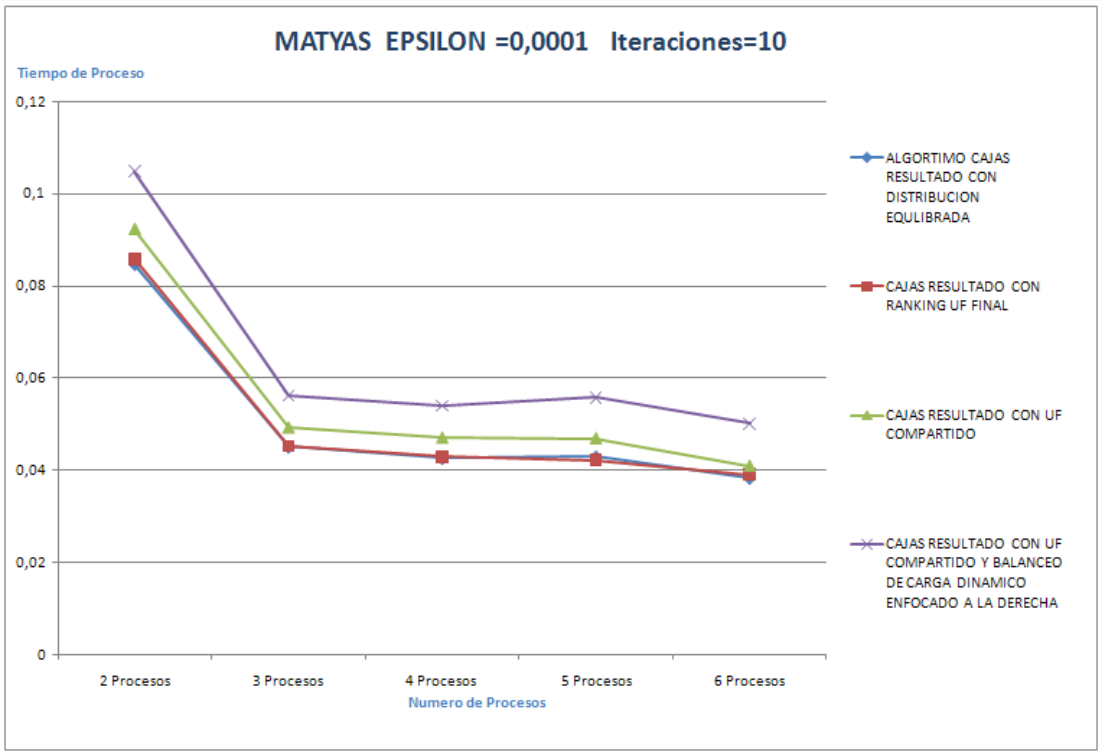


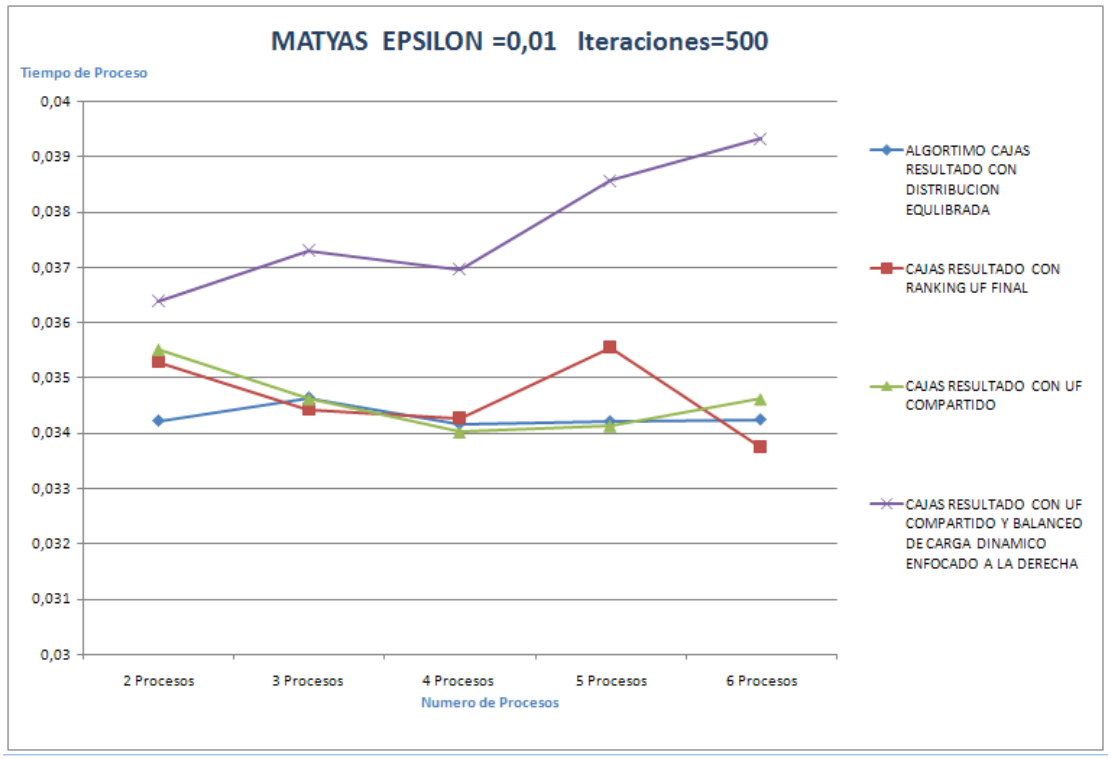


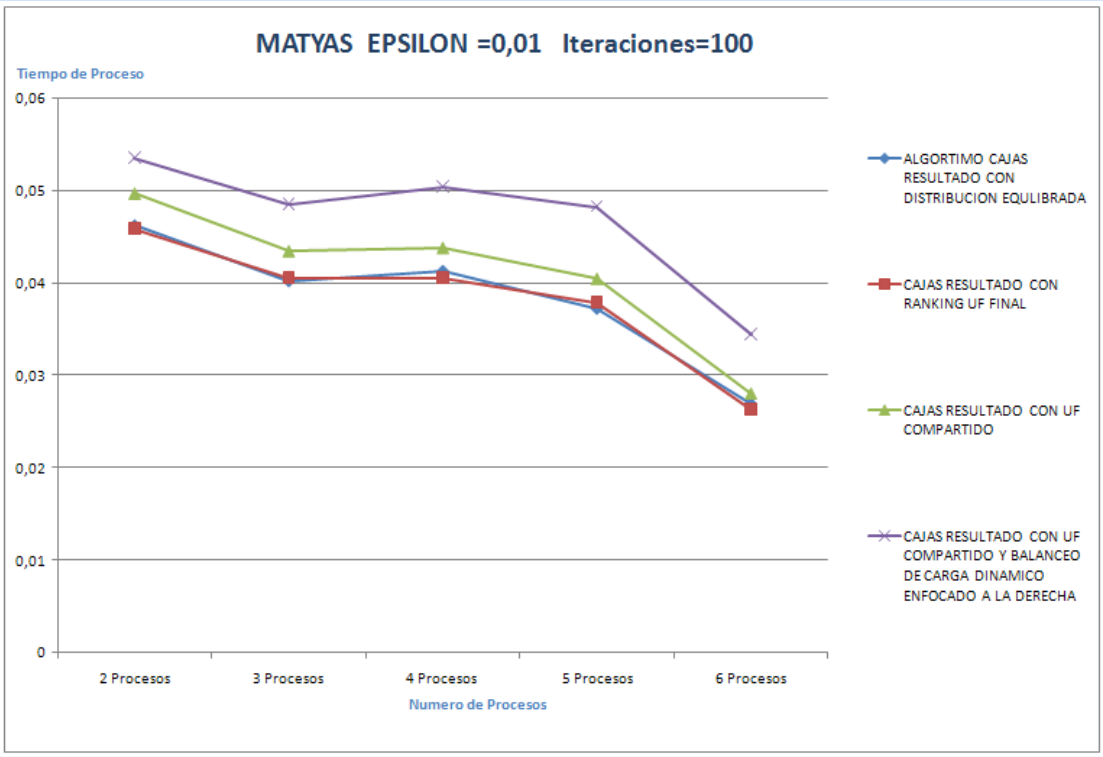


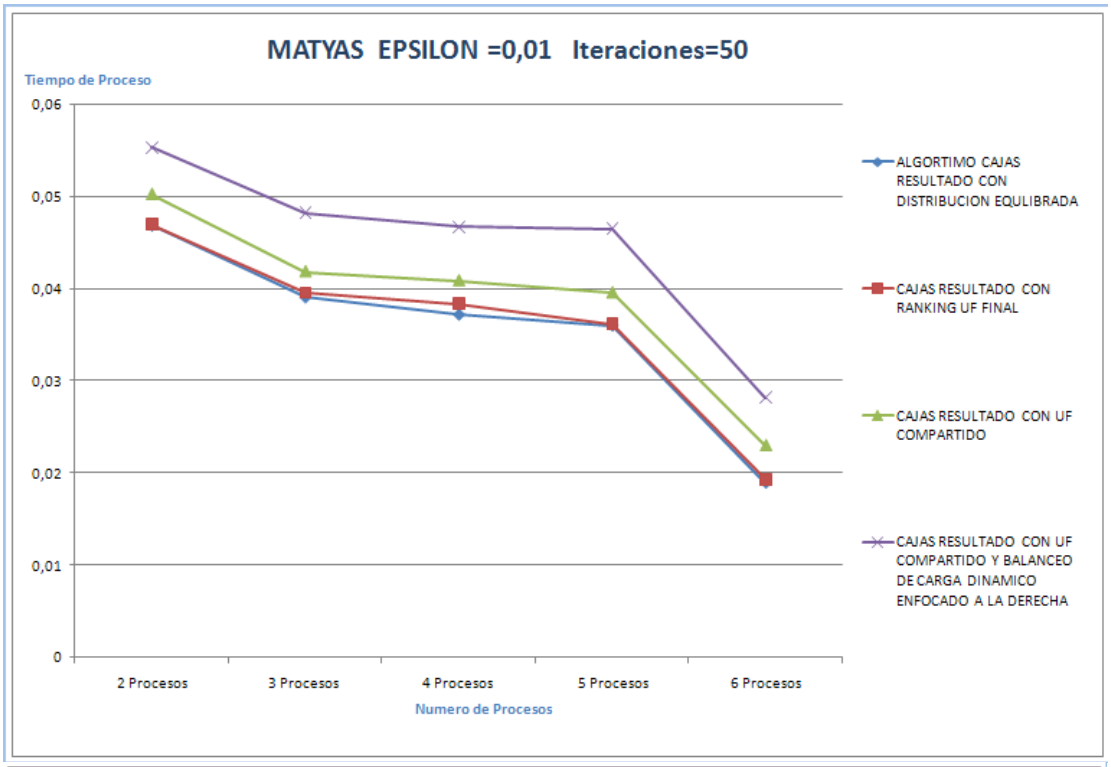


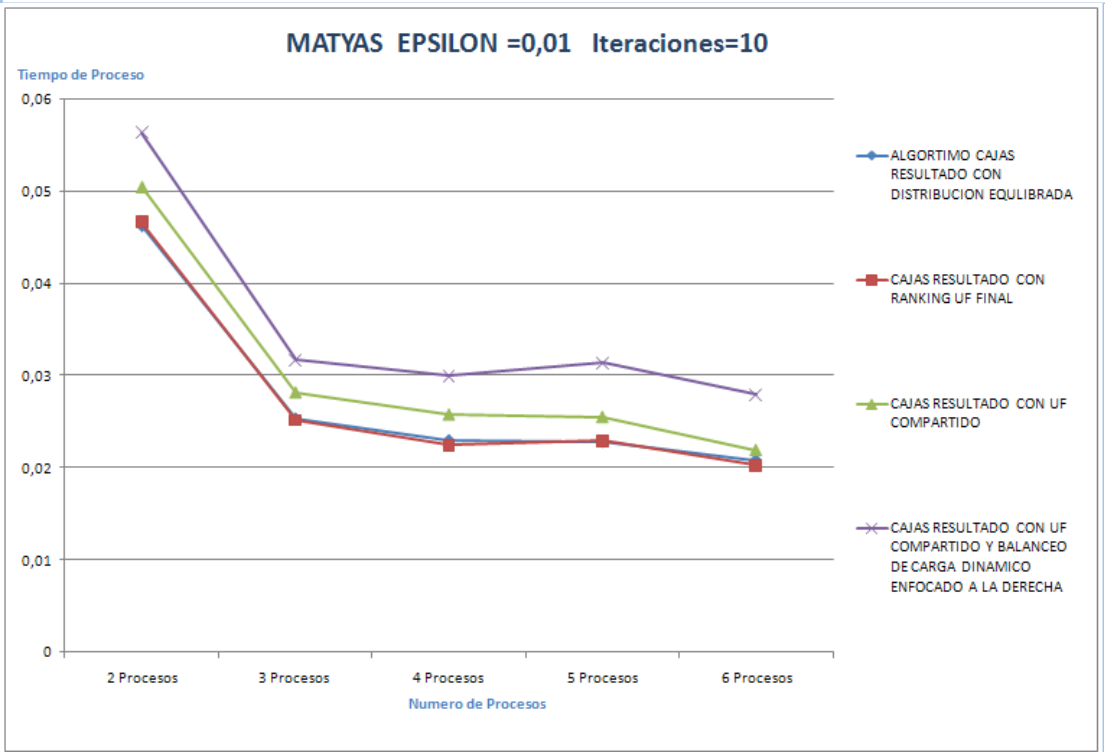


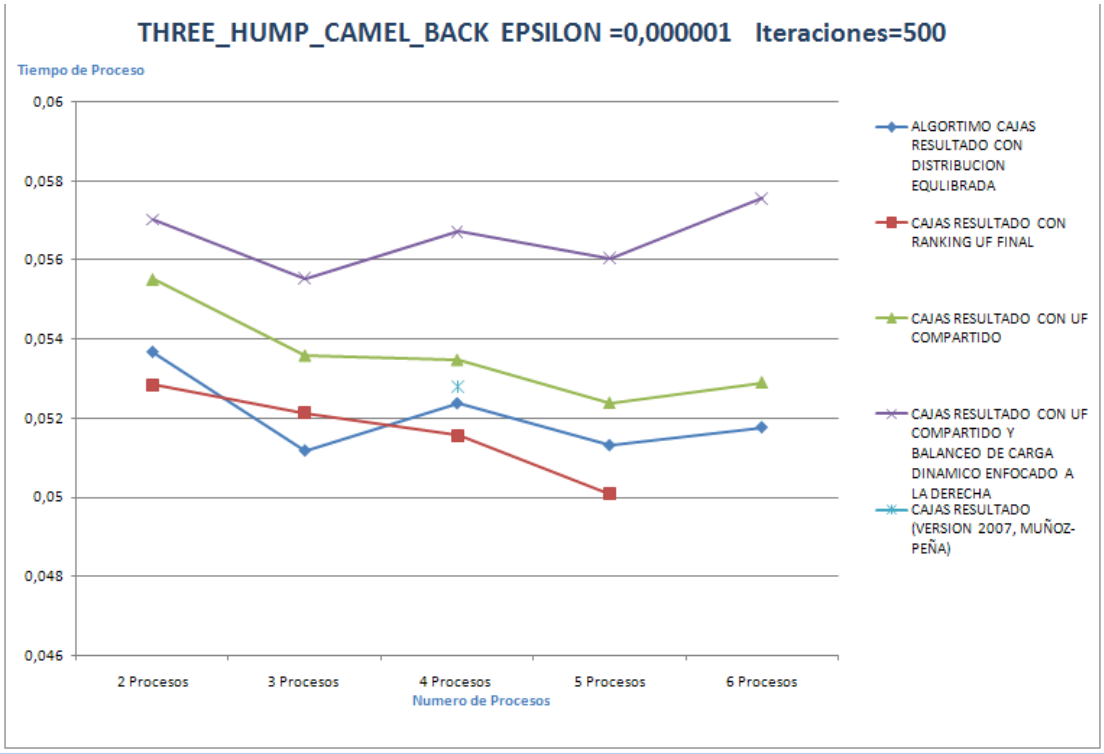


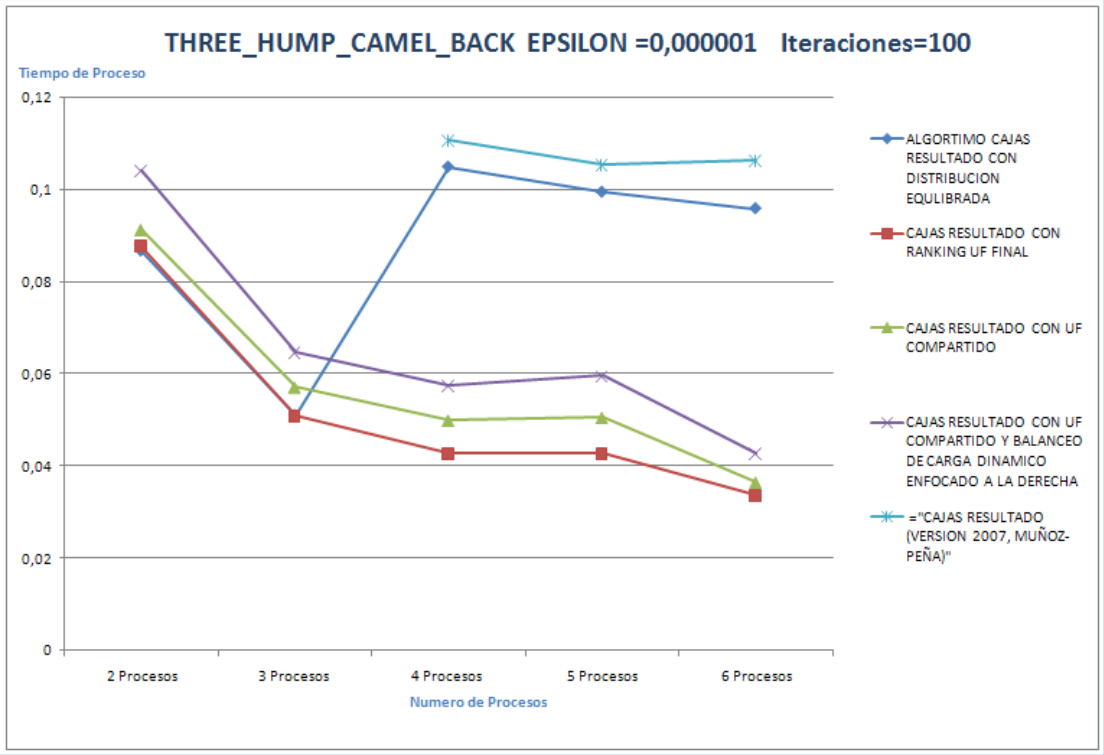


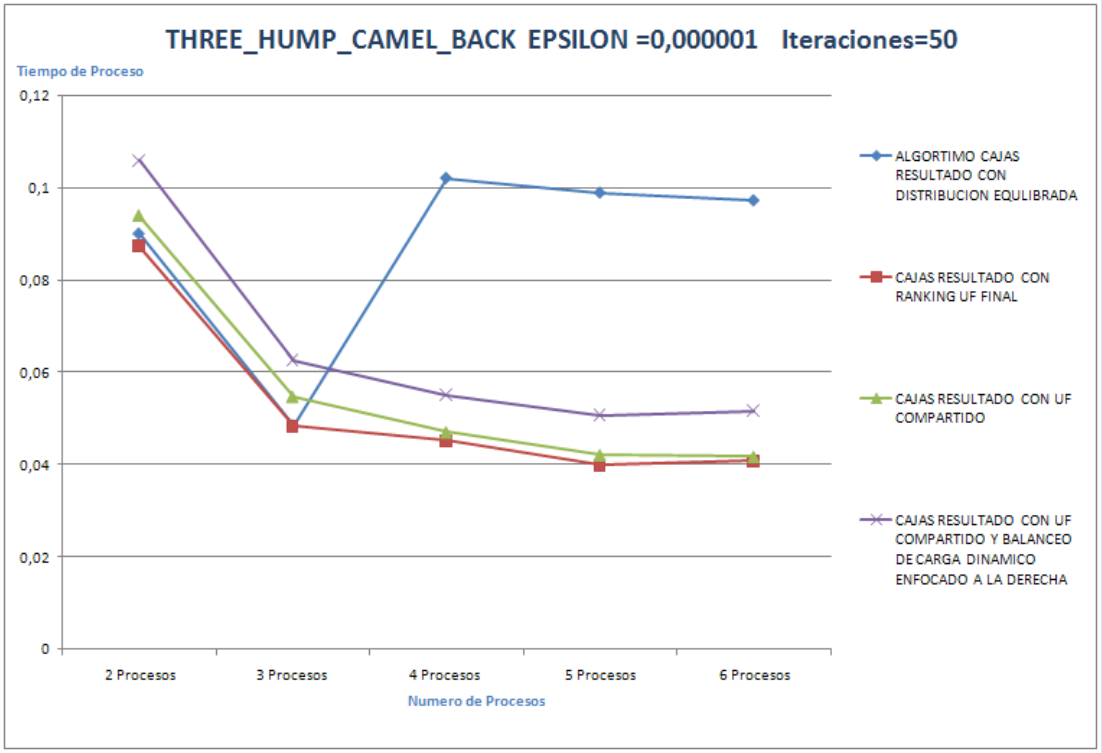


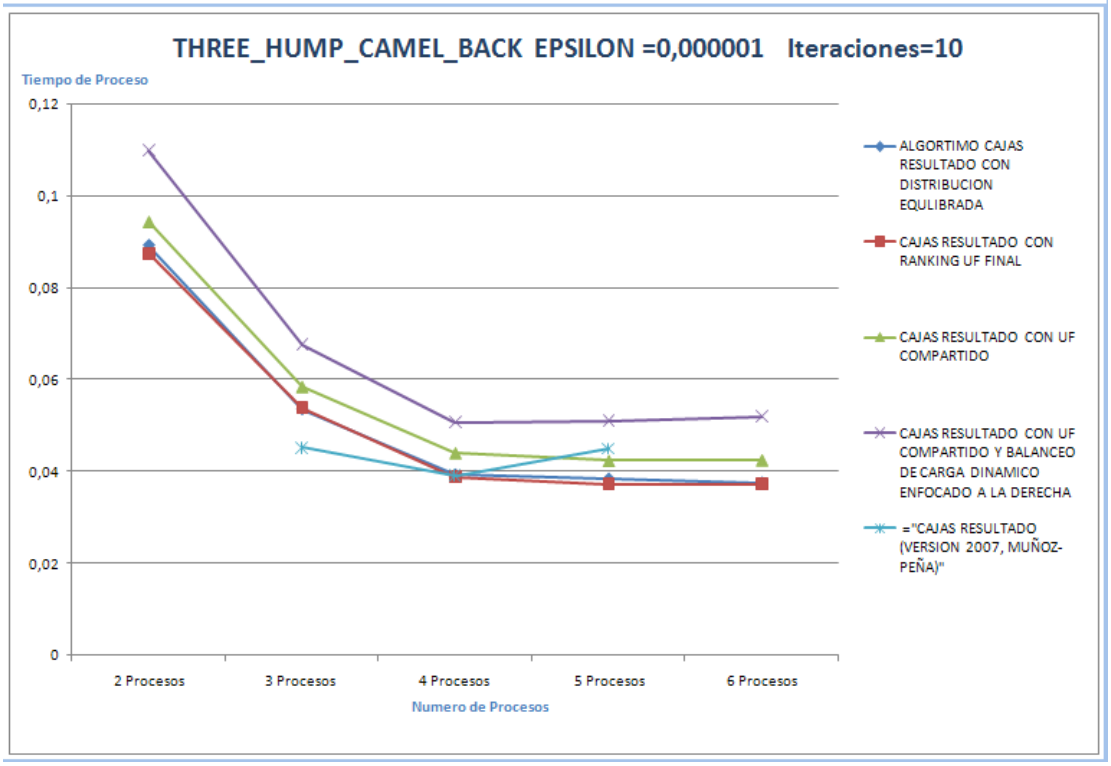


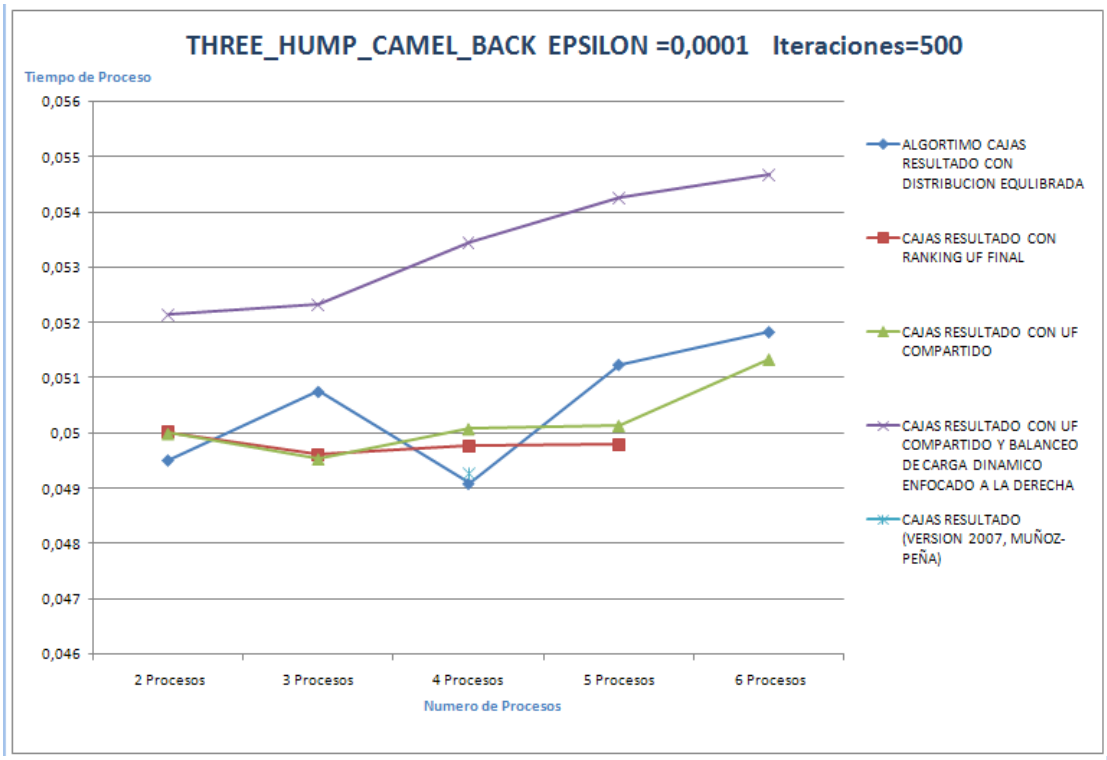


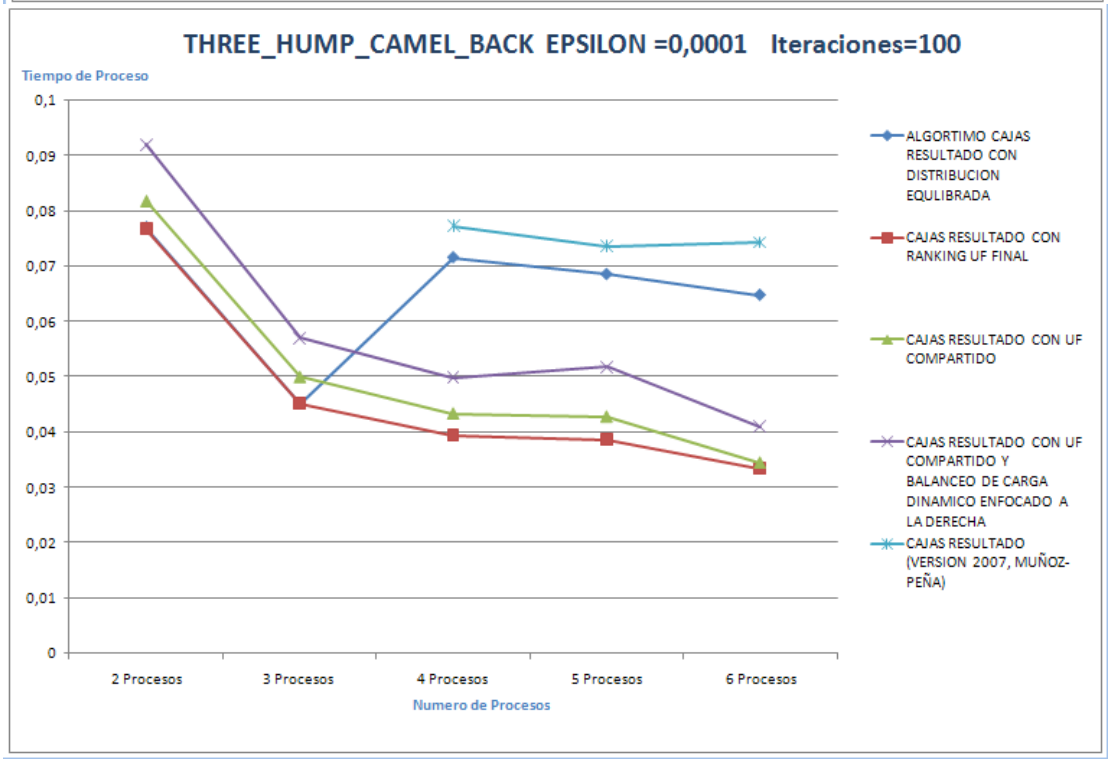


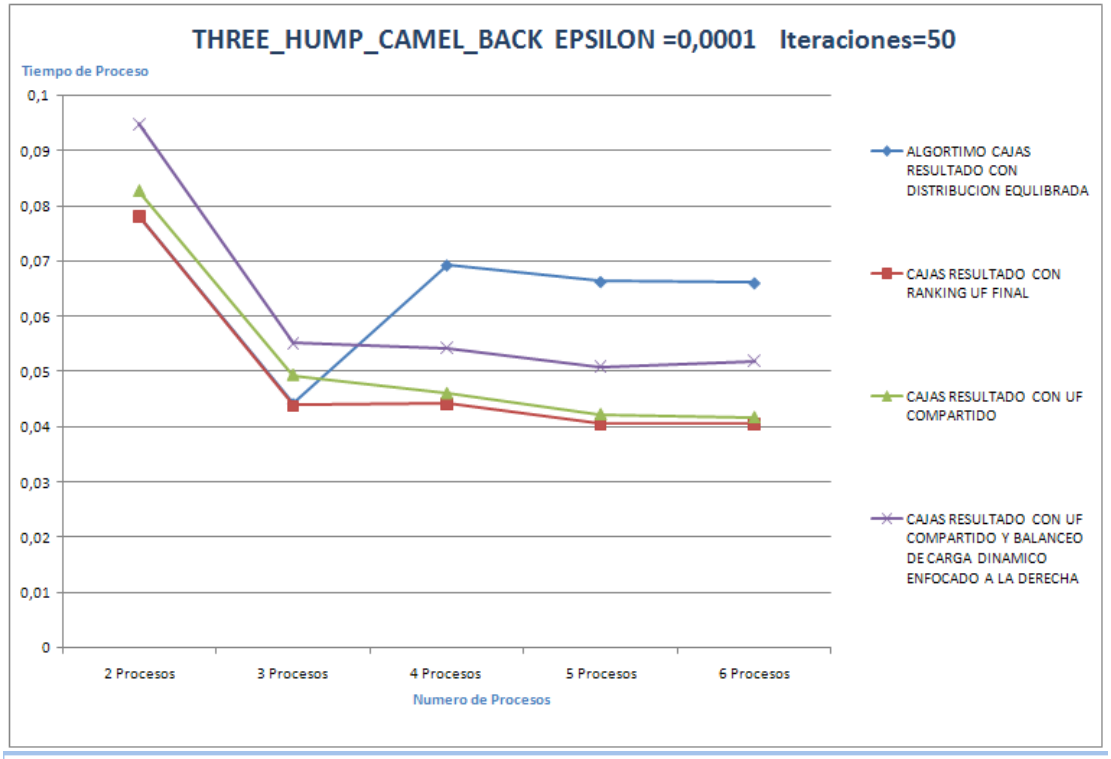


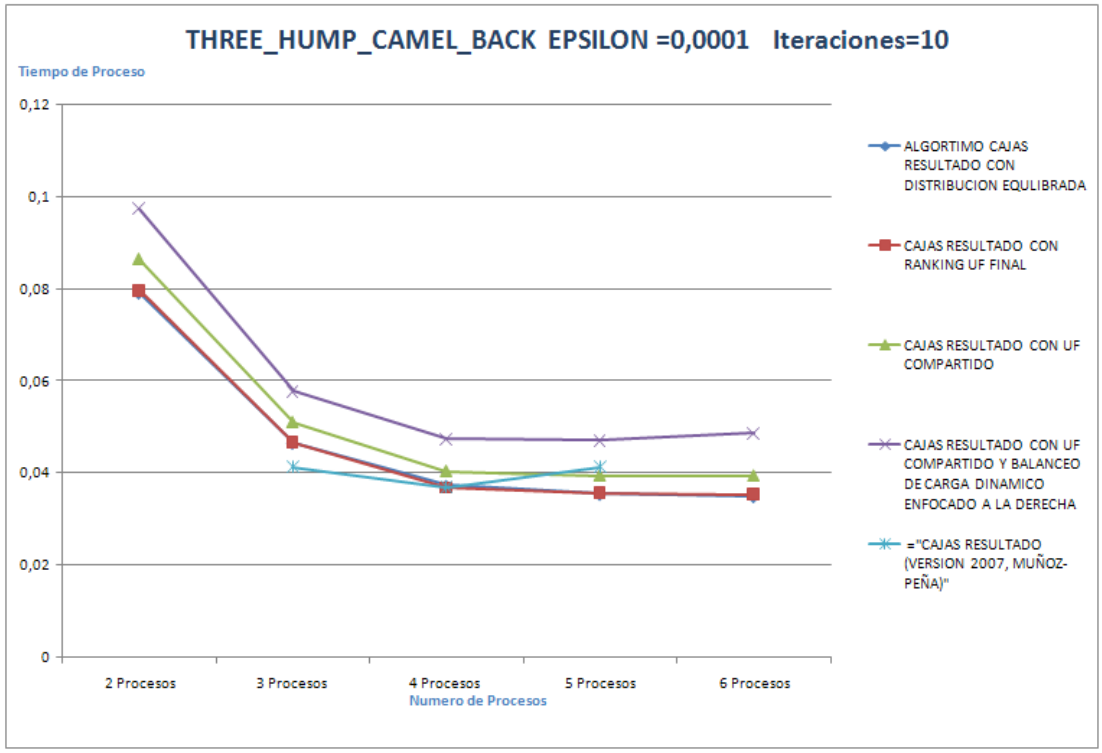


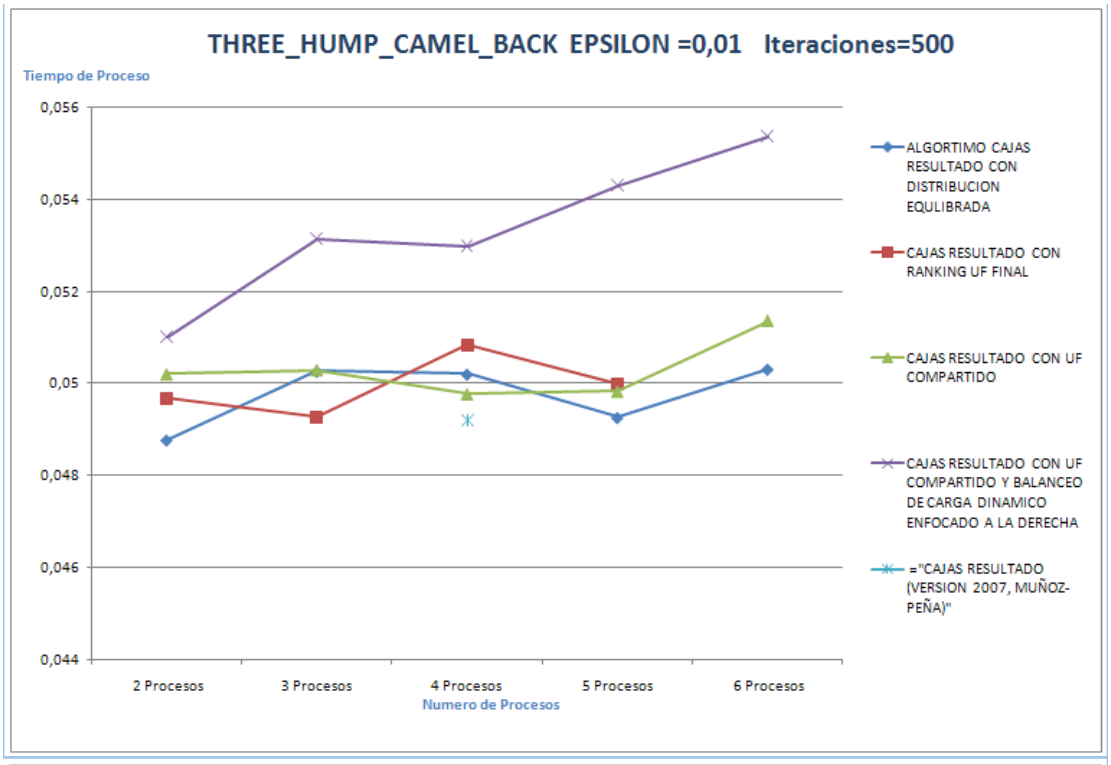


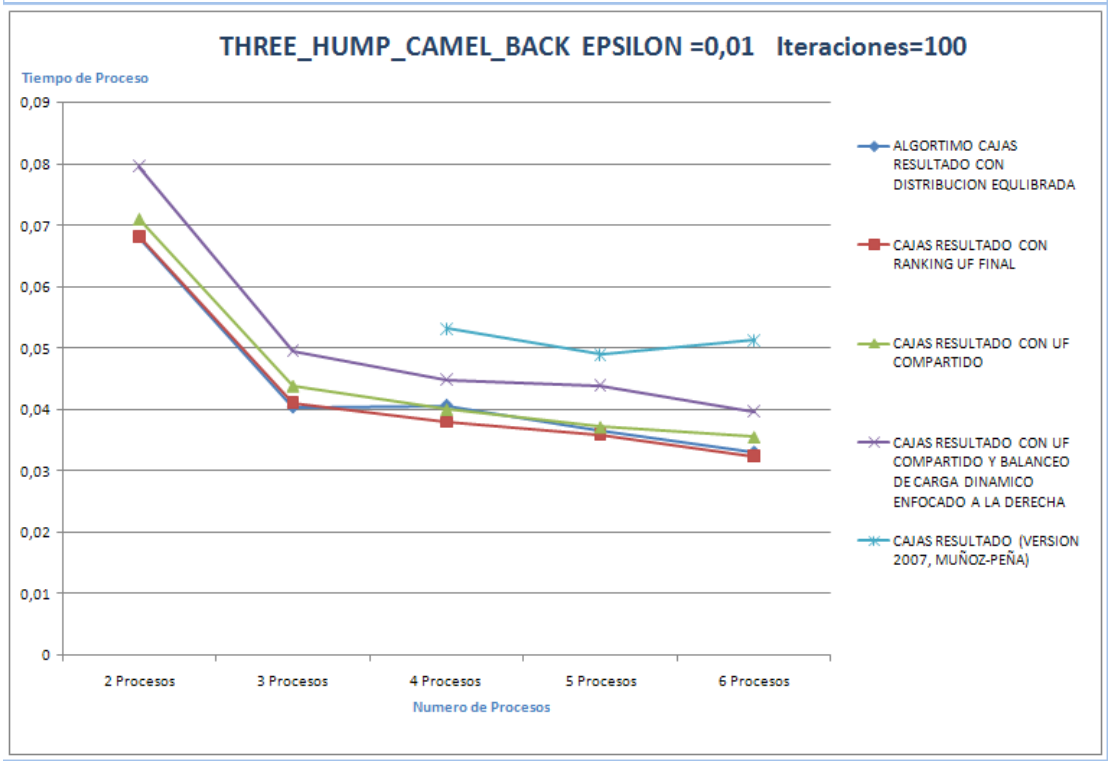


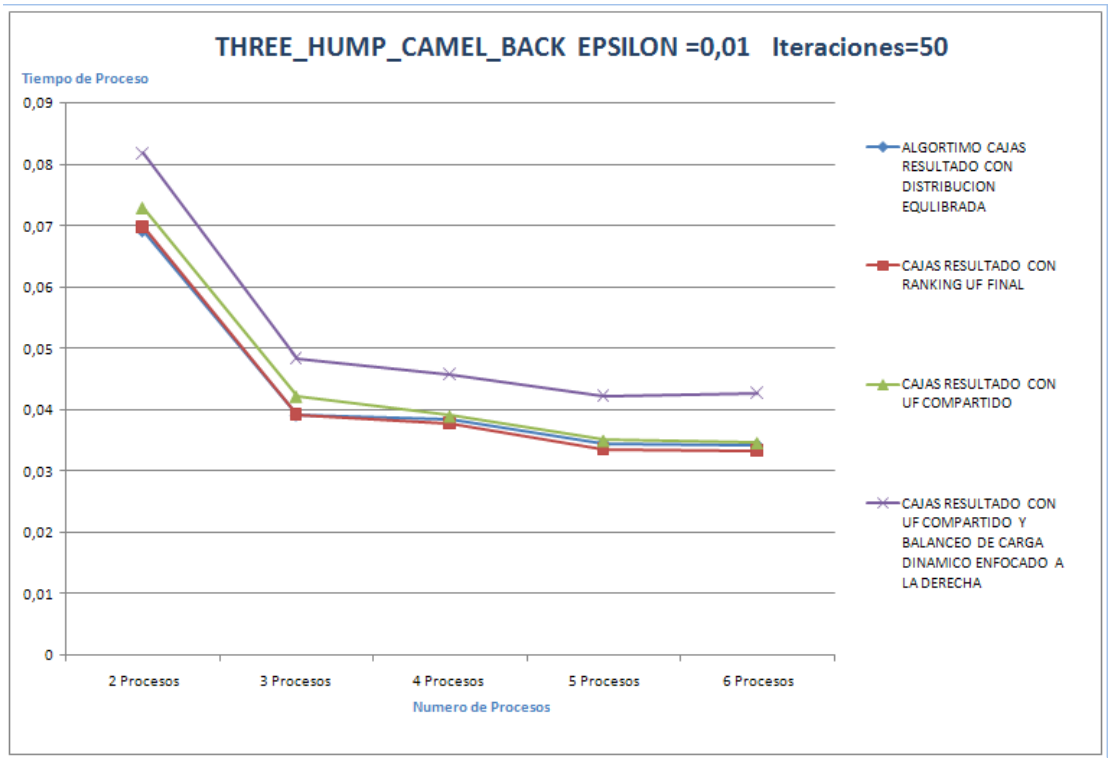


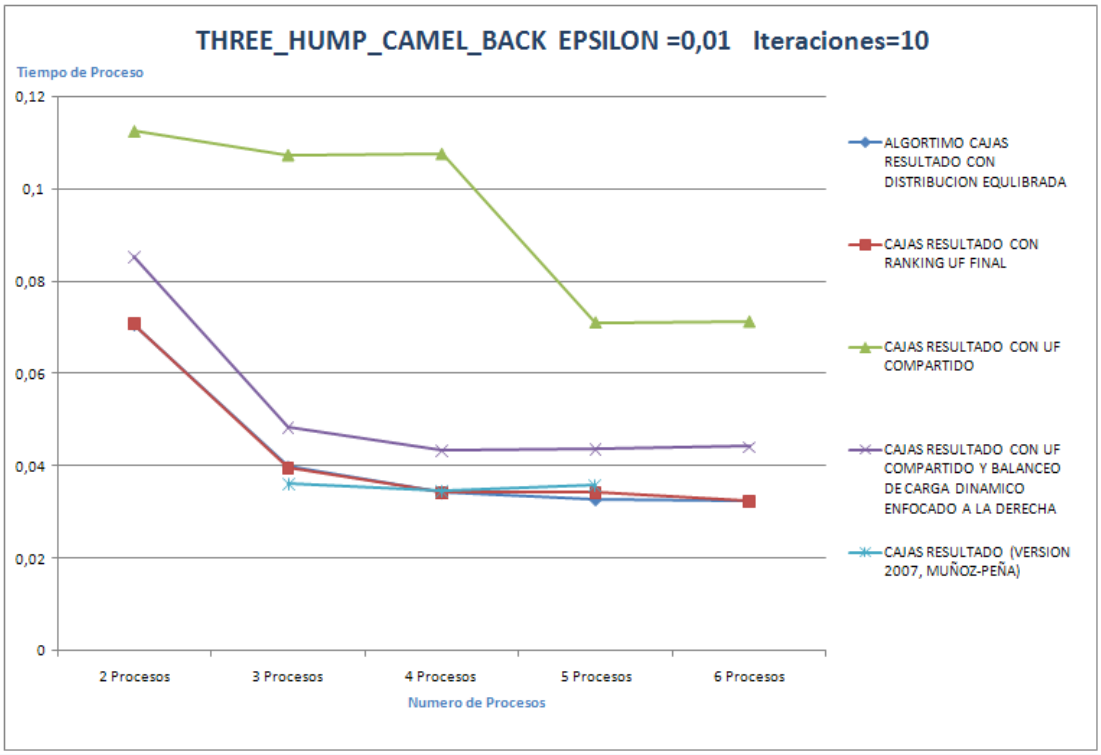


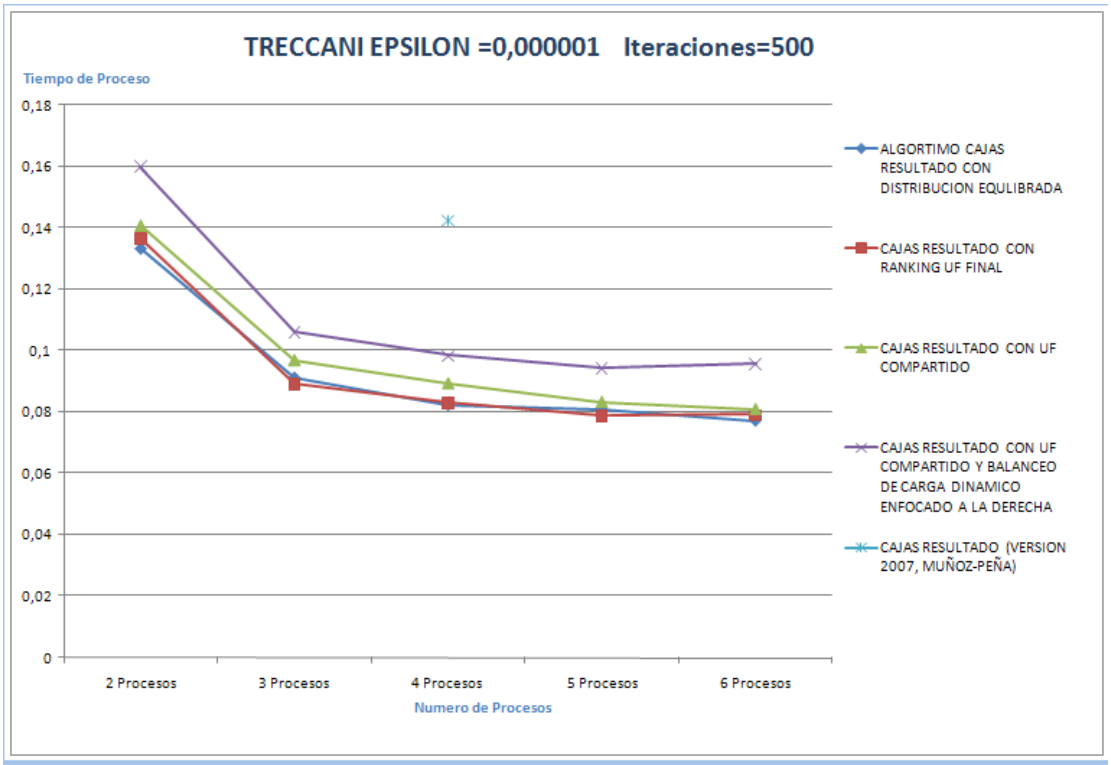


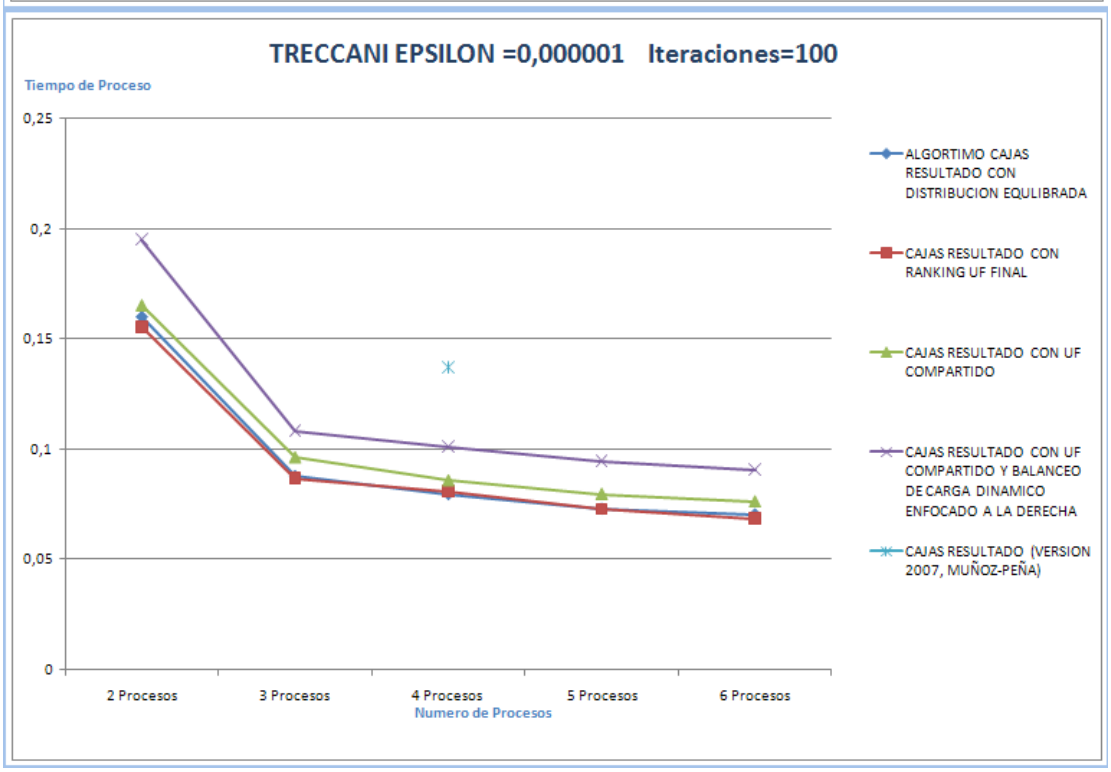


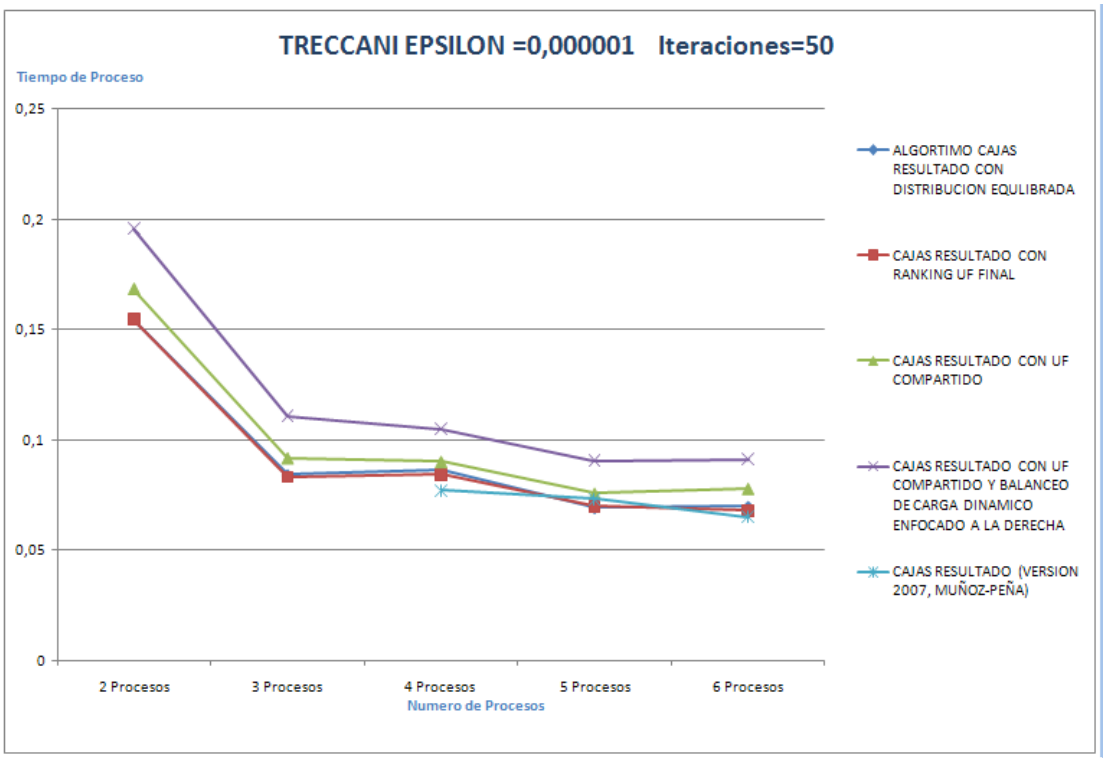


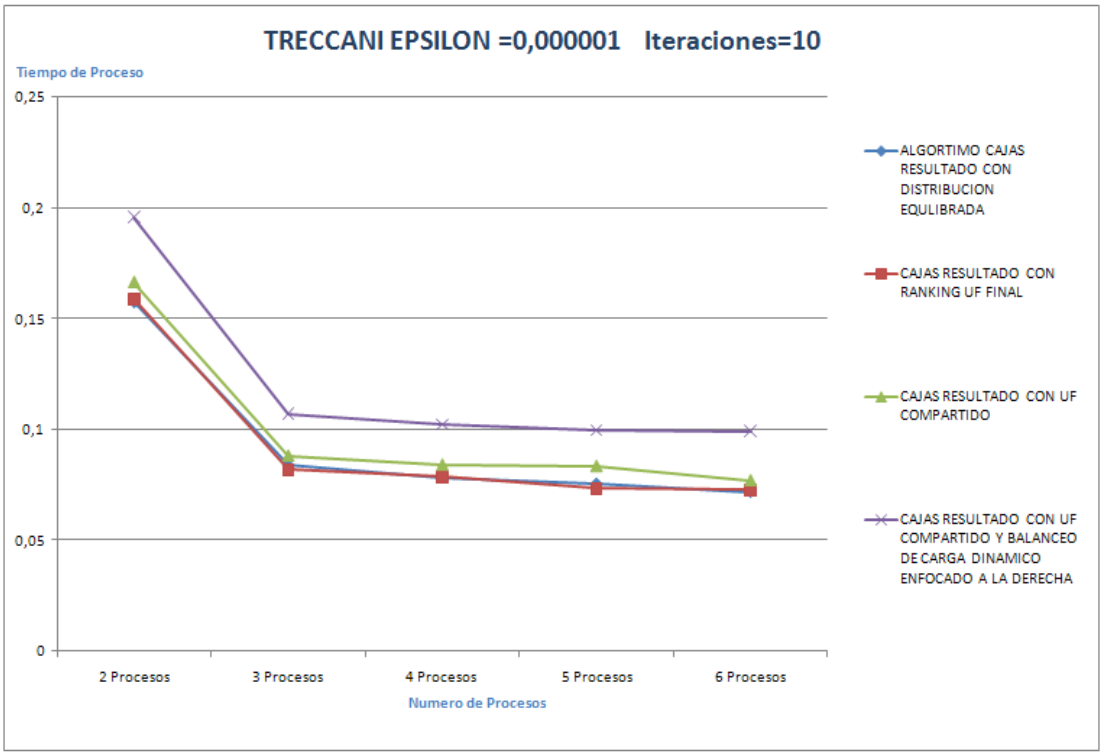


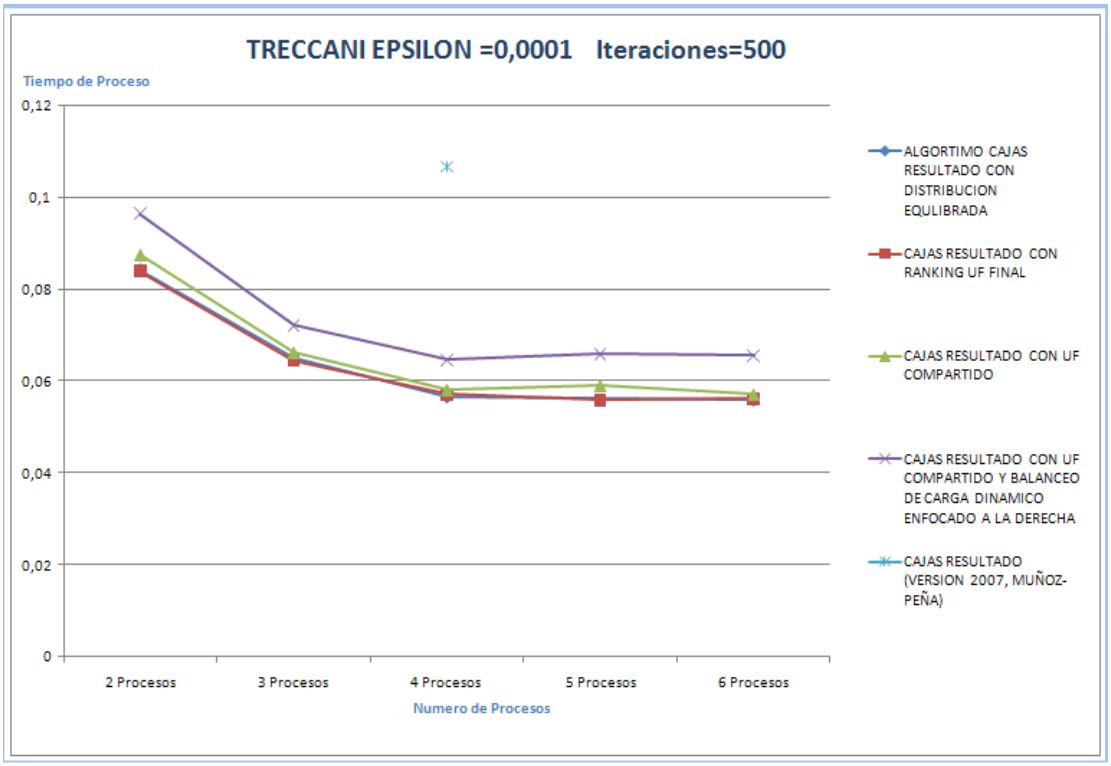


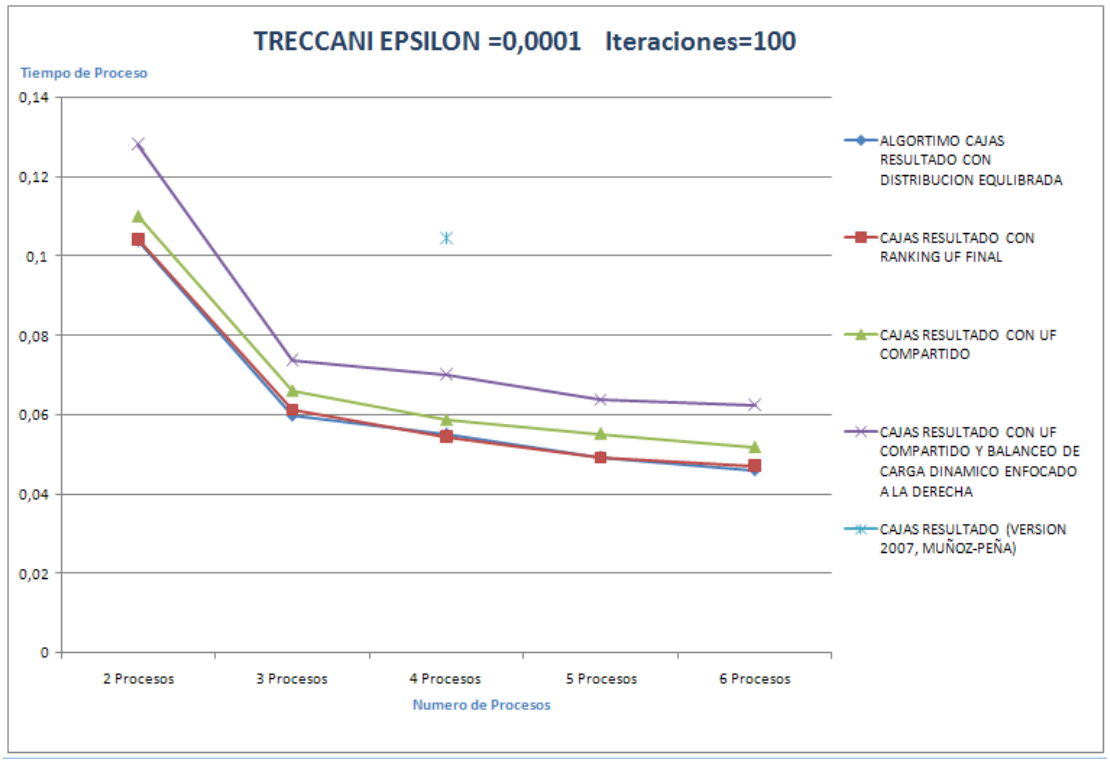


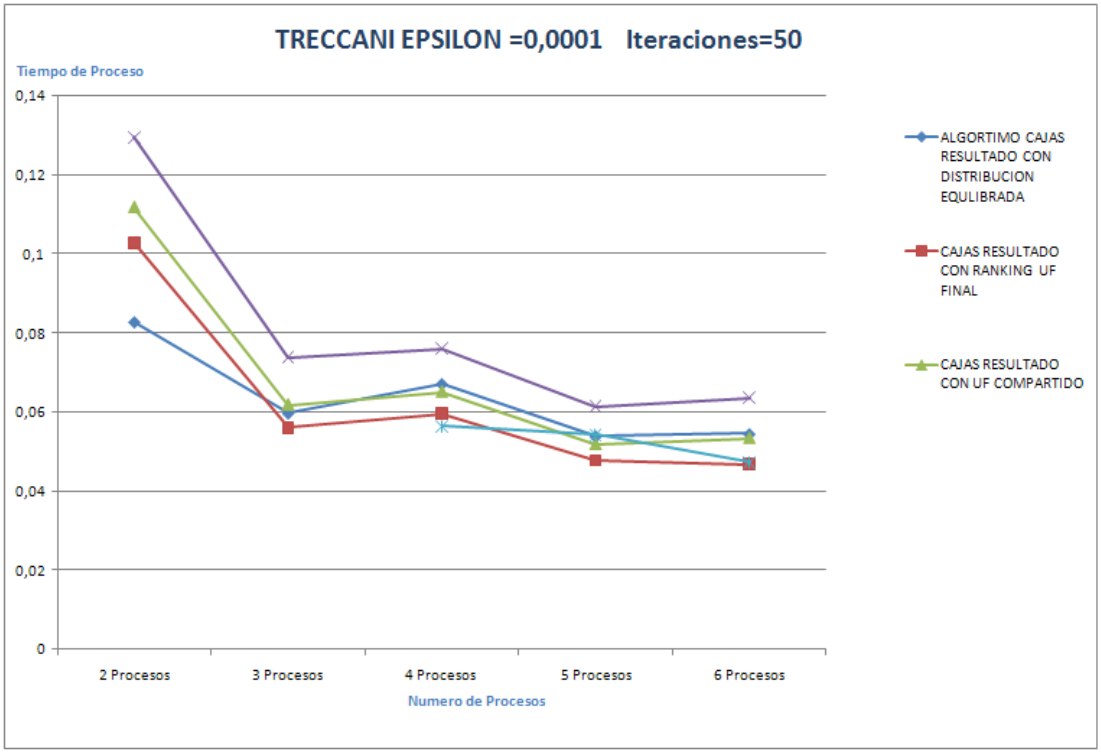


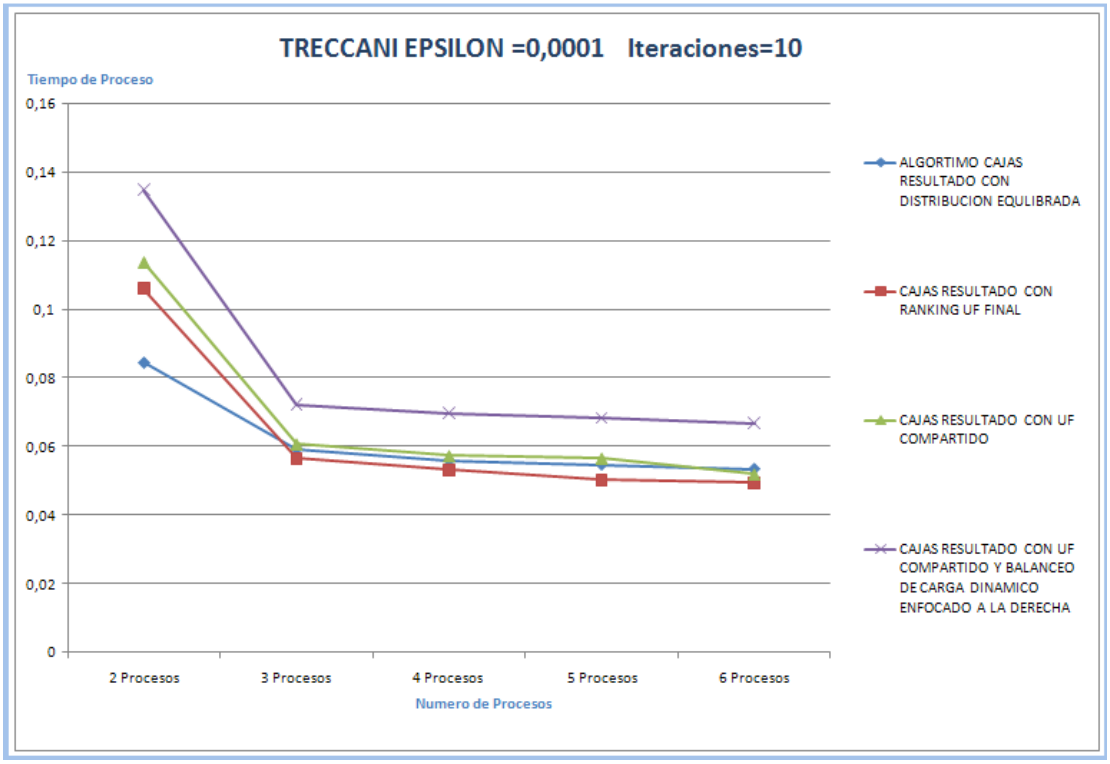


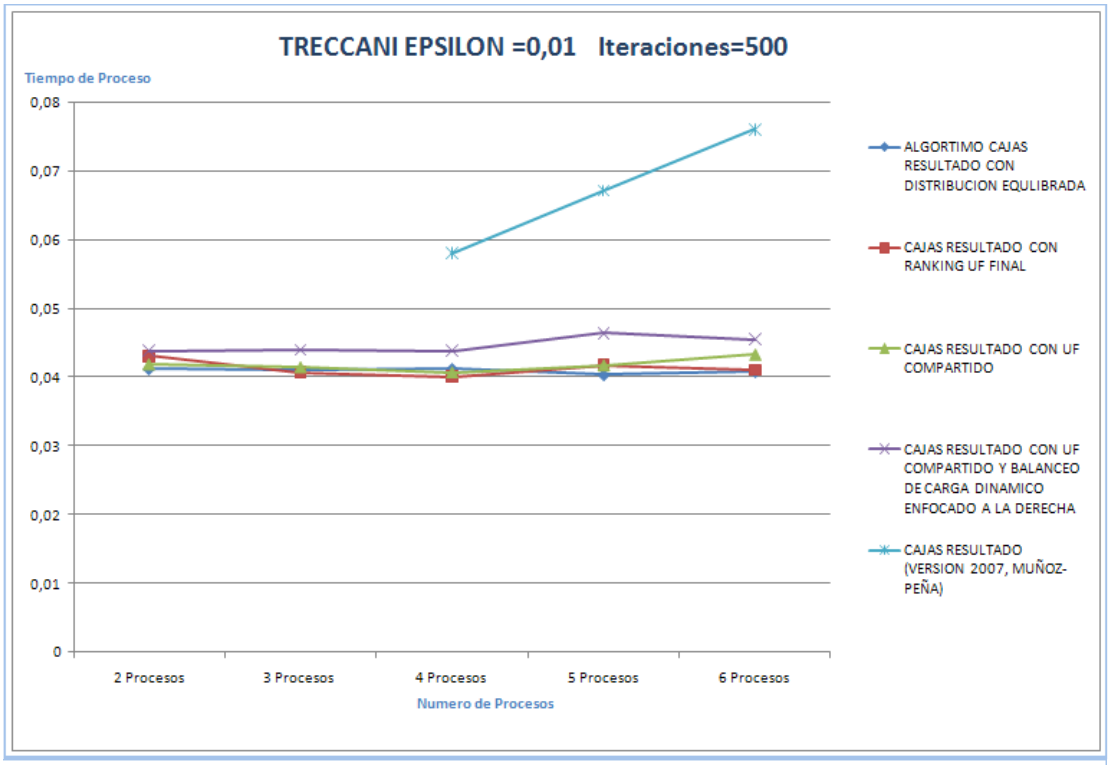


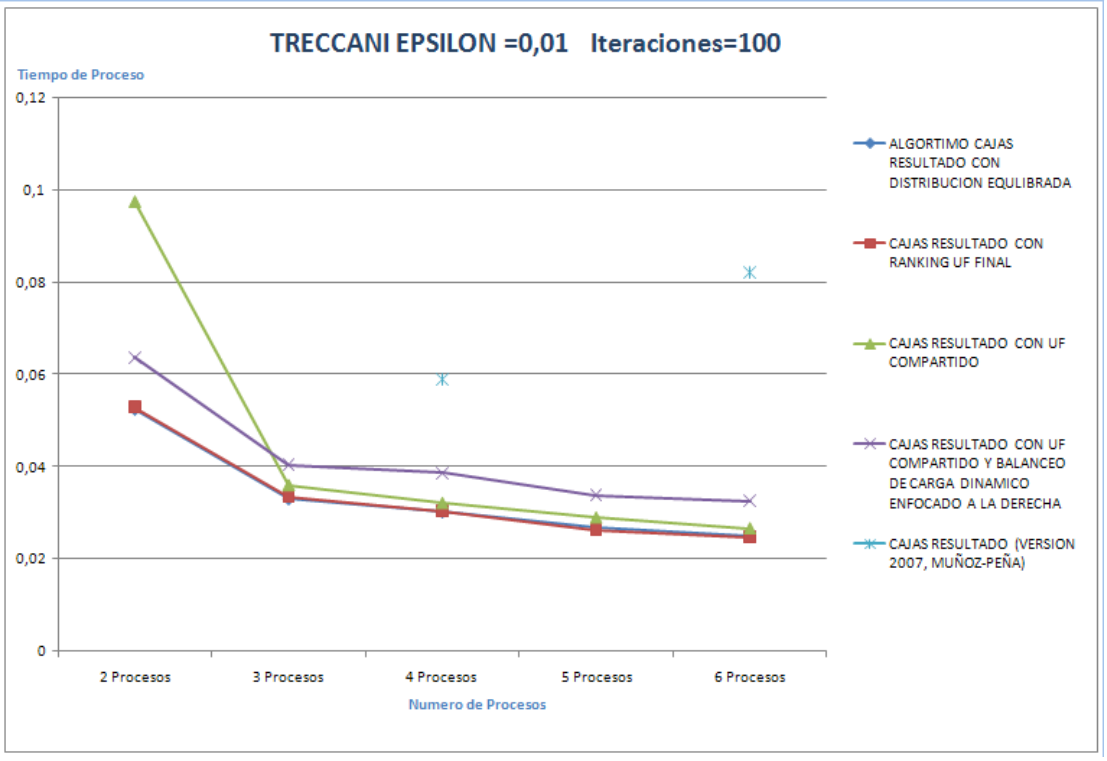


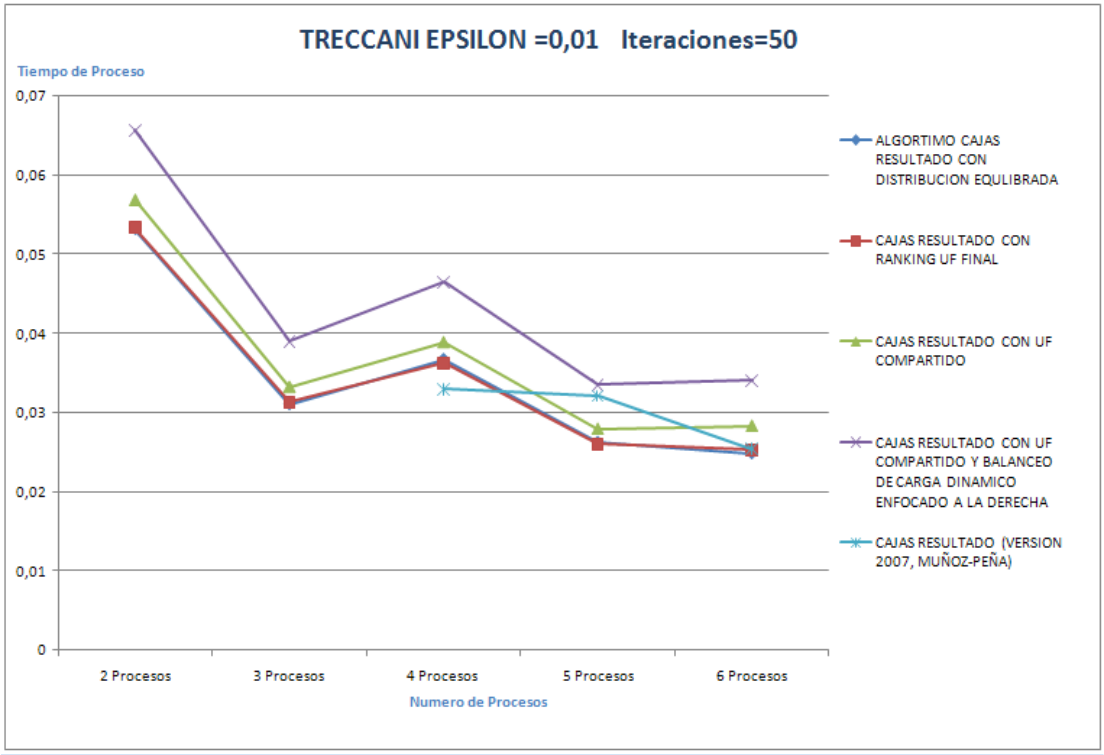


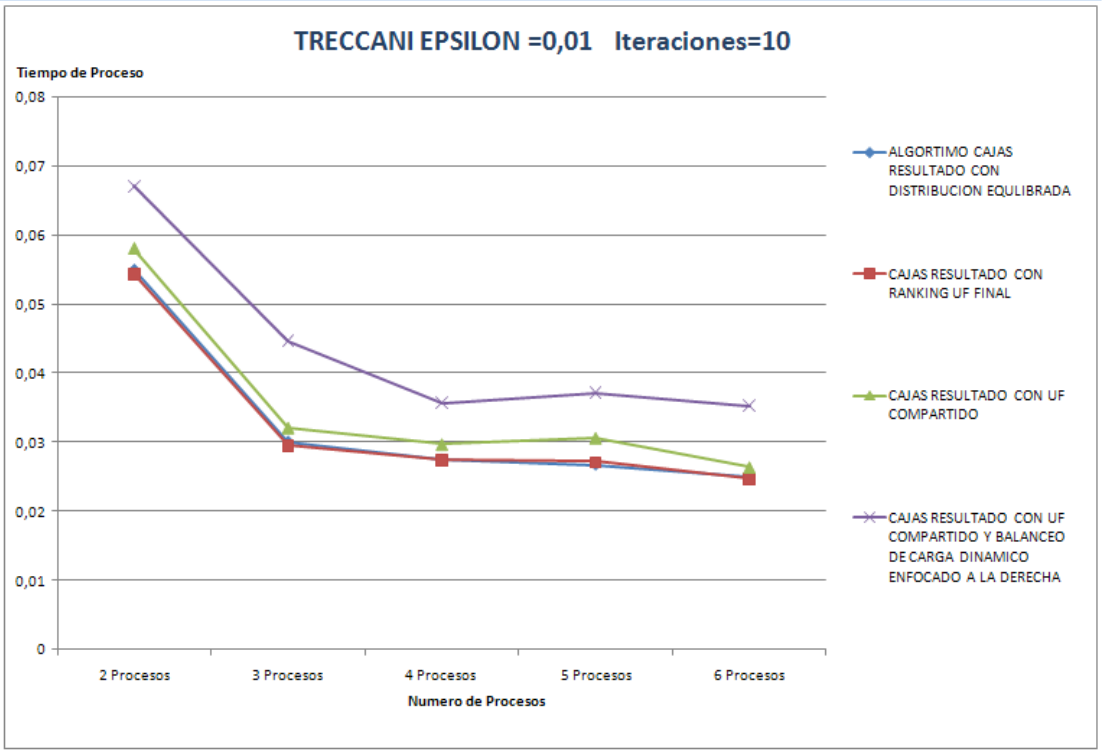


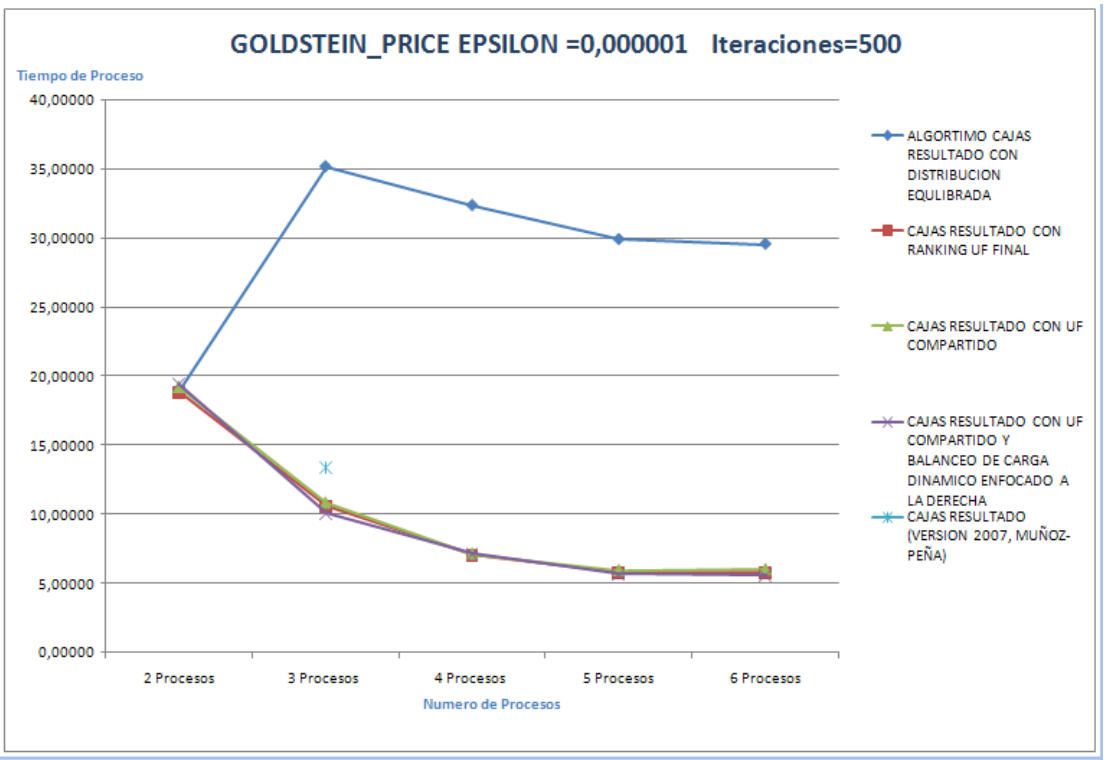


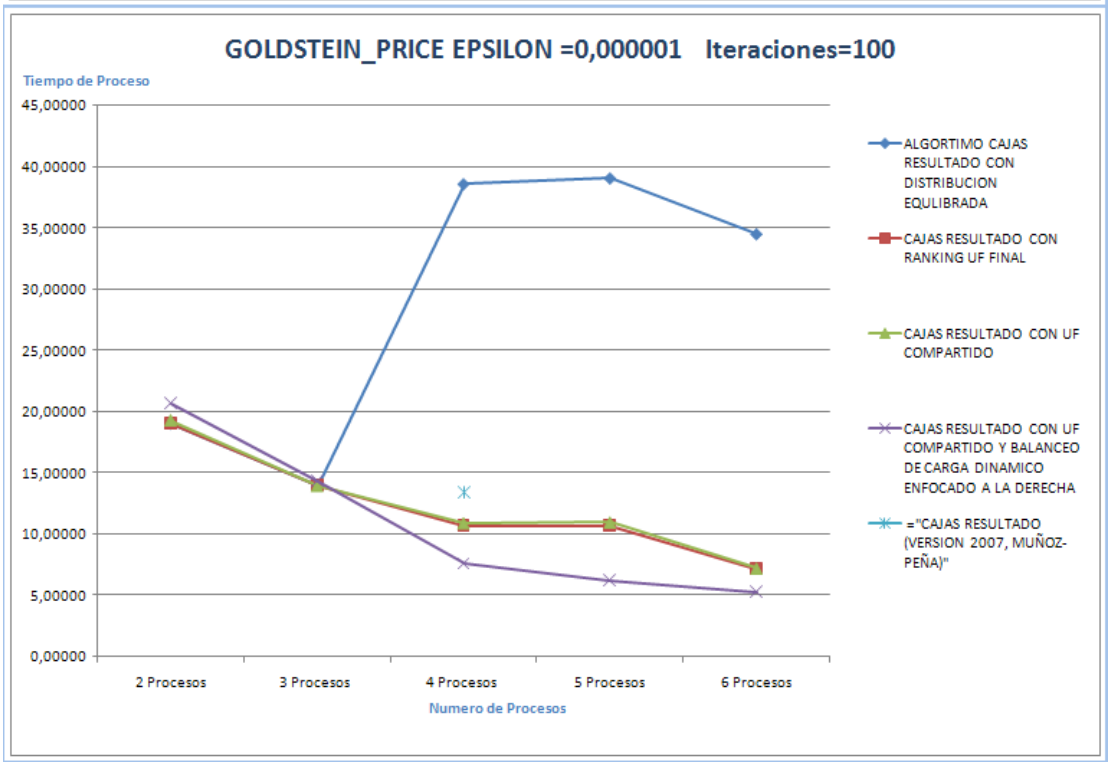


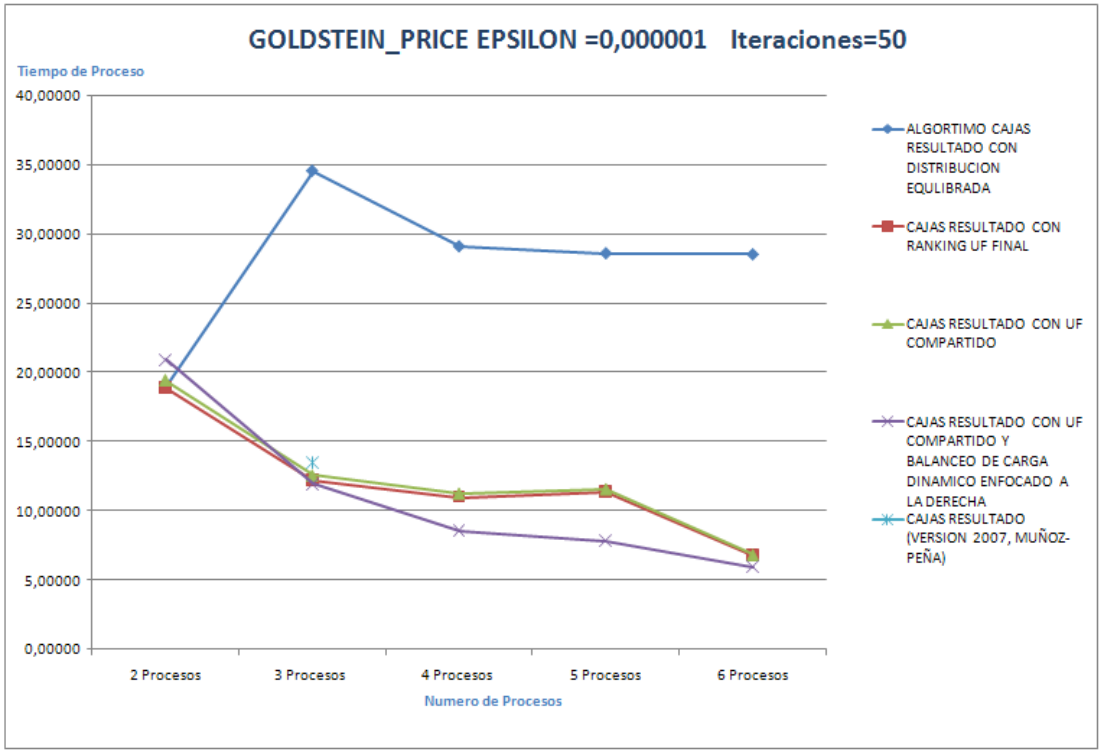


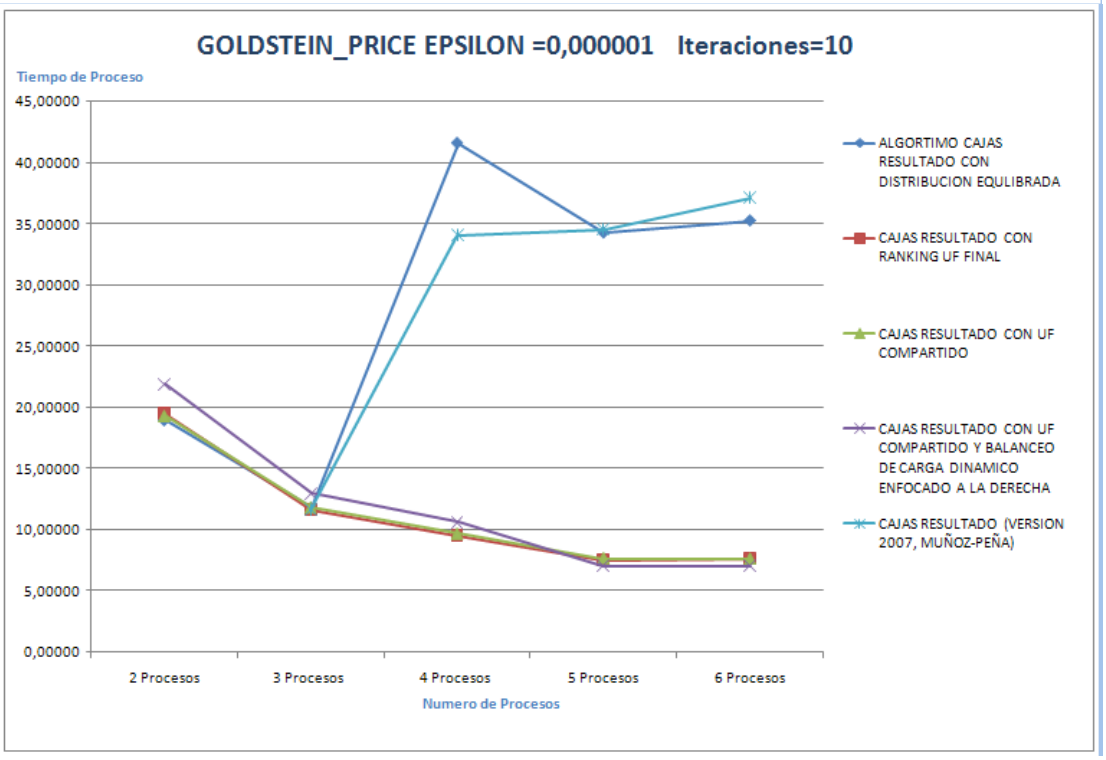


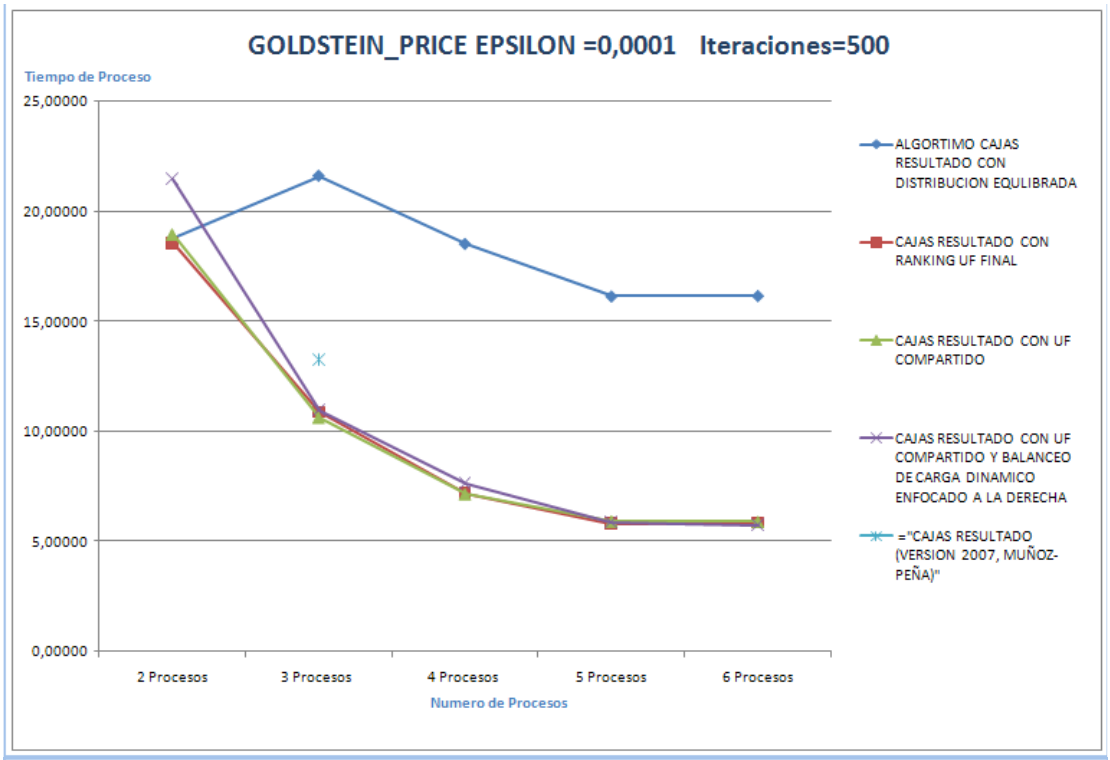


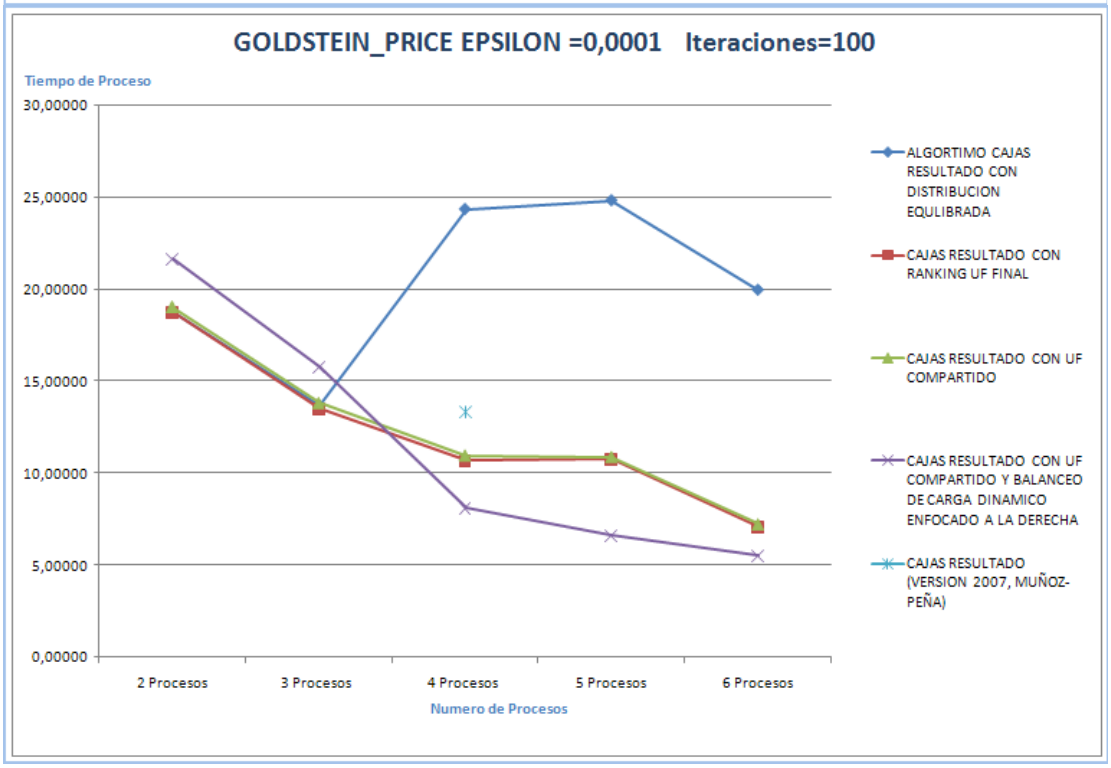


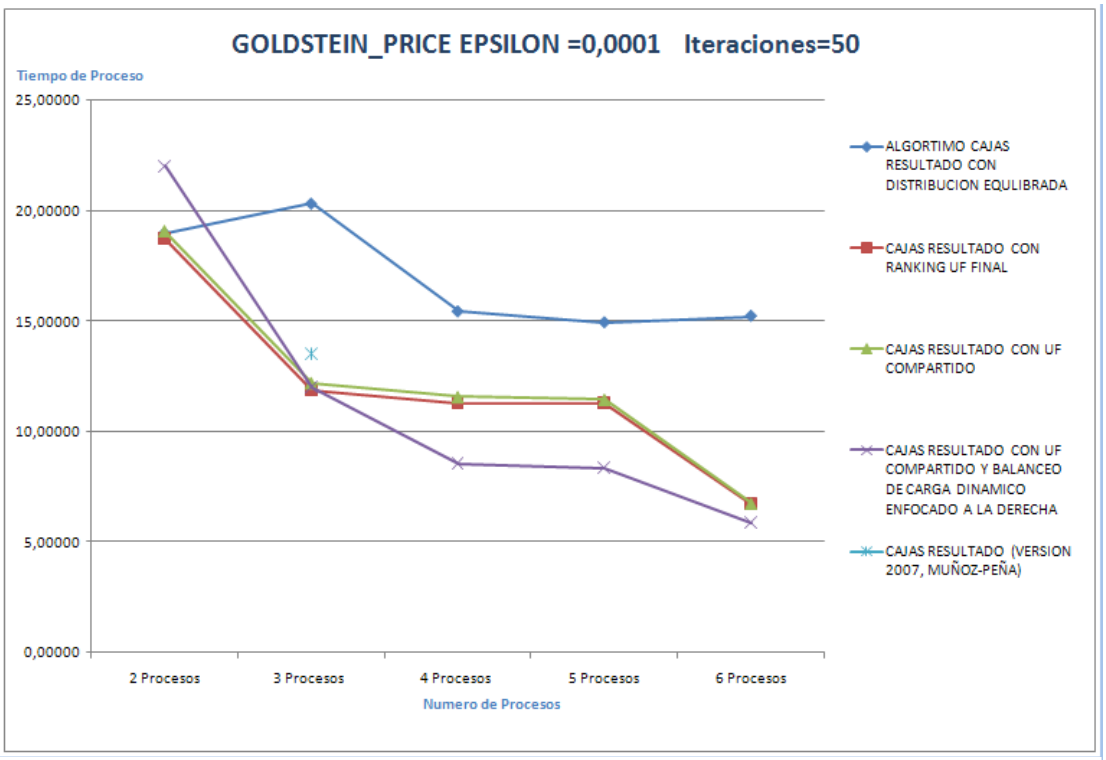


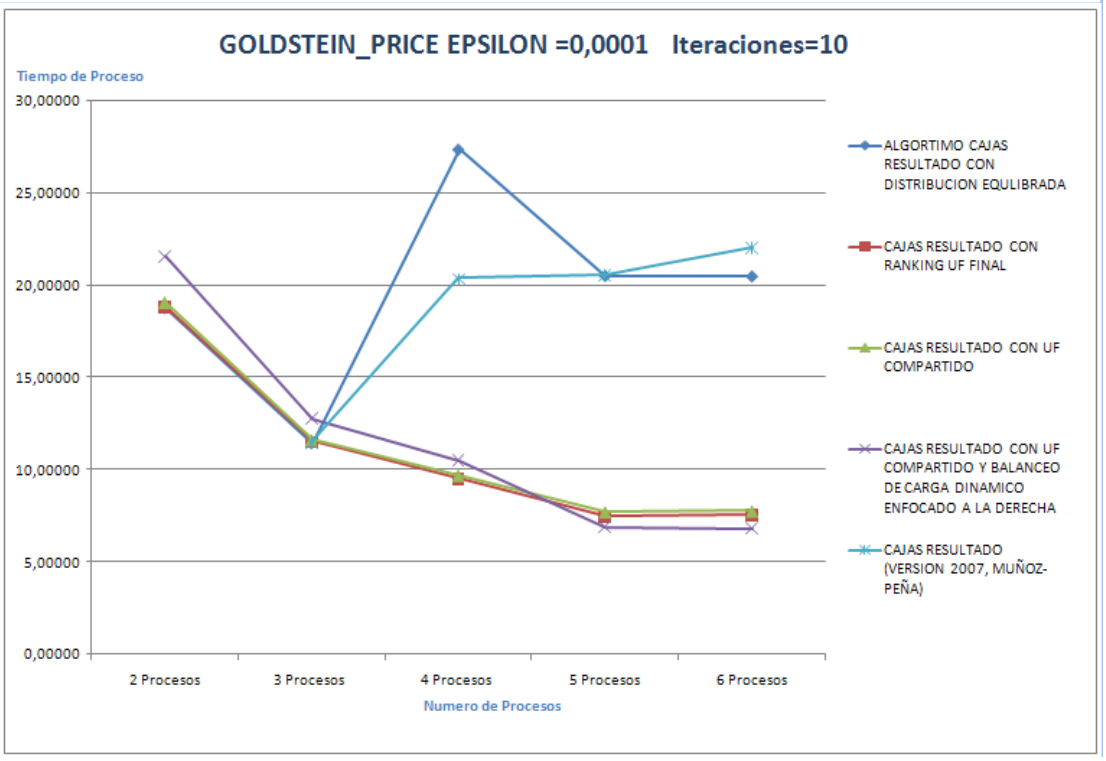


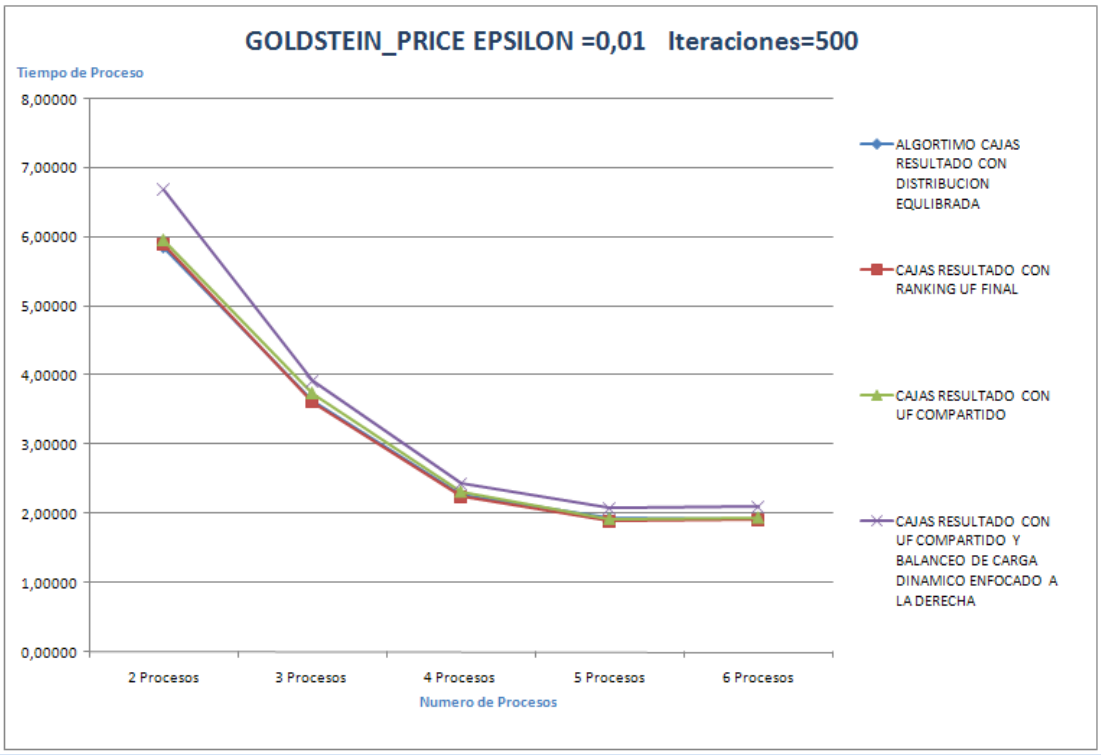


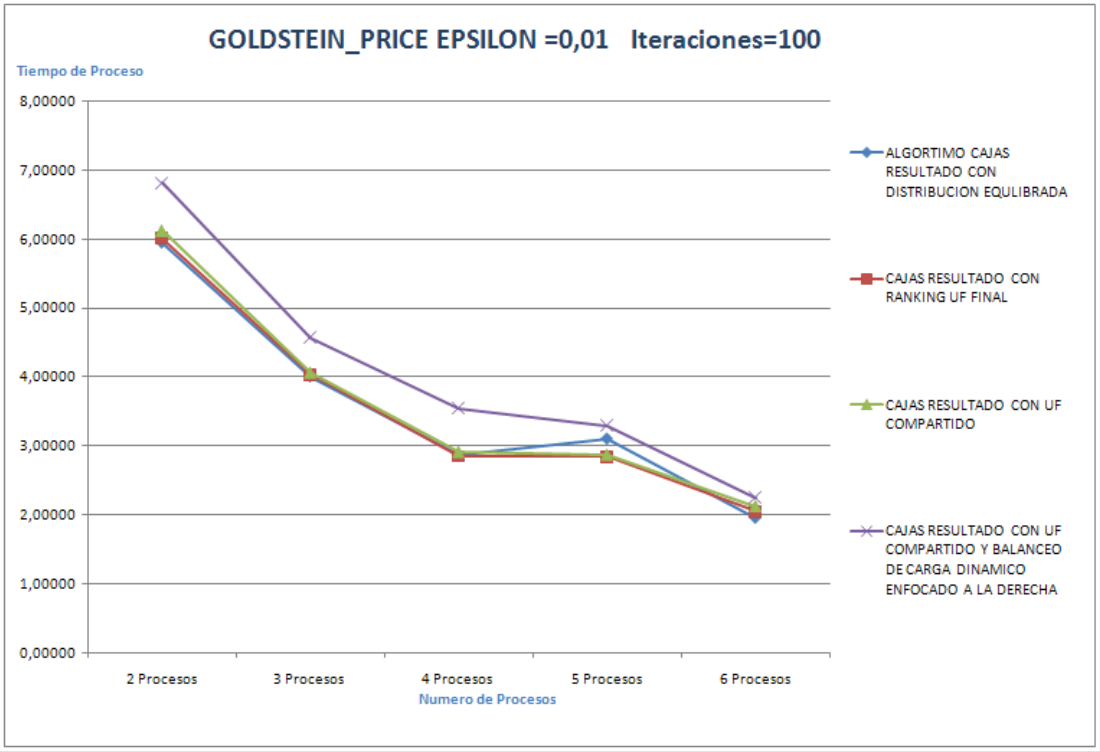


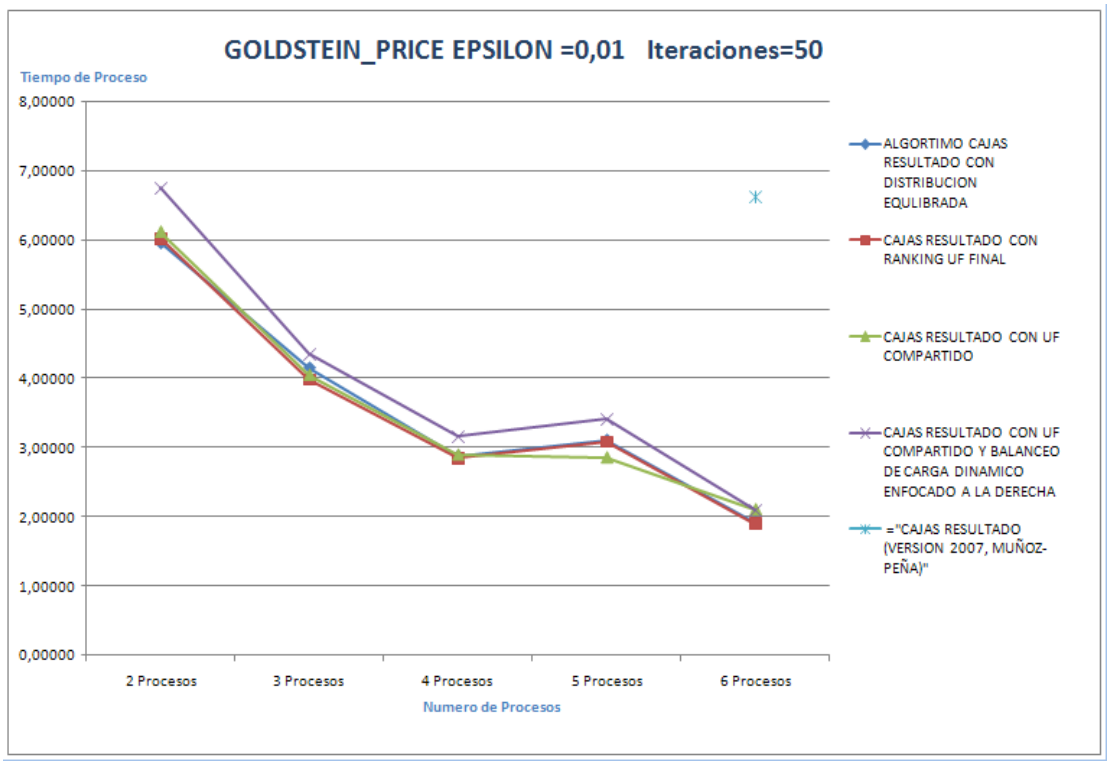












DATOS ALGORTIMO CAJAS RESULTADO CON DISTRIBUCION EQUILIBRADA

EPSILON =0,01										Iteraciones 10	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]											
N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		
1	0,167459	1	0,126907	1	0,0956969	1	0,097955	1	0,0494242		
2	0,157971	2	0,130659	2	0,097157	2	0,097178	2	0,0530539		
3	0,1637	3	0,130311	3	0,101416	3	0,101213	3	0,0517781		
4	0,158884	4	0,126344	4	0,100298	4	0,10045	4	0,0496571		
5	0,160051	5	0,1304	5	0,100695	5	0,104274	5	0,0513549		
6		6		6		6		6			
7	0,161613	7	0,1289242	7	0,09905258	7	0,100214	7	0,05105364		
EPSILON =0,01										Iteraciones 50	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]											
N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		
1	0,162838	1	0,106753	1	0,302956	1	0,189822	1	0,105636		
2	0,165623	2	0,105577	2	0,303651	2	0,182898	2	0,104284		
3	0,154071	3	0,110865	3	0,291283	3	0,182508	3	0,112178		
4	0,157305	4	0,112702	4	0,291106	4	0,193391	4	0,107606		
5	0,154684	5	0,110504	5	0,295117	5	0,180456	5	0,104351		
6		6		6		6		6			
7	0,1589042	7	0,1092802	7	0,2968226	7	0,185815	7	0,106811		
EPSILON =0,01										Iteraciones 100	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]											
N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		
1	0,165772	1	0,193949	1	0,127464	1	0,111751	1	0,106972		
2	0,163693	2	0,191605	2	0,120943	2	0,108137	2	0,107945		
3	0,153709	3	0,196132	3	0,126325	3	0,112153	3	0,108839		
4	0,159409	4	0,189952	4	0,121261	4	0,109231	4	0,108362		
5	0,163839	5	0,202537	5	0,11827	5	0,104727	5	0,111562		
6		6		6		6		6			
7	0,1612844	7	0,194835	7	0,1228526	7	0,1091998	7	0,108736		
EPSILON =0,01										Iteraciones 500	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]											
N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		
1	0,133784	1	0,106191	1	0,104359	1	0,094434	1	0,094043		
2	0,132582	2	0,106642	2	0,101681	2	0,0949888	2	0,0928419		
3	0,135757	3	0,104569	3	0,101148	3	0,0973229	3	0,0939839		
4	0,133769	4	0,10461	4	0,109671	4	0,0958431	4	0,0951781		
5	0,13264	5	0,106814	5	0,0987248	5	0,0938849	5	0,0999169		
6		6		6		6		6			
7	0,1337064	7	0,1057652	7	0,10311676	7	0,09529474	7	0,09519276		

EPSILON =0,0001									Iteraciones 10	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]										
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,177743	1	0,140049	1	0,114311	1	0,10882	1	0,0598941	
2	0,169416	2	0,139509	2	0,116152	2	0,110864	2	0,057574	
3	0,169276	3	0,13895	3	0,117936	3	0,111016	3	0,0615981	
4	0,18154	4	0,137214	4	0,110618	4	0,108397	4	0,0637629	
5	0,172226	5	0,137276	5	0,110851	5	0,111371	5	0,0575659	
6		6		6		6		6		
7	0,1740402	7	0,1385996	7	0,1139736	7	0,1100936	7	0,060079	
EPSILON =0,0001									Iteraciones 50	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]										
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,17289	1	0,113248	1	0,625455	1	0,189967	1	0,157084	
2	0,171326	2	0,112299	2	0,634115	2	0,184122	2	0,165287	
3	0,188464	3	0,121411	3	0,626362	3	0,180514	3	0,153781	
4	0,177252	4	0,11498	4	0,63516	4	0,188689	4	0,156585	
5	0,172993	5	0,172453	5	0,634194	5	0,189899	5	0,154642	
6		6		6		6		6		
7	0,176585	7	0,1268782	7	0,6310572	7	0,1866382	7	0,1574758	
EPSILON =0,0001									Iteraciones 100	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]										
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,170149	1	0,192903	1	0,118373	1	0,115847	1	0,116218	
2	0,169842	2	0,192331	2	0,121086	2	0,115759	2	0,110831	
3	0,170334	3	0,198261	3	0,117301	3	0,121599	3	0,112206	
4	0,17125	4	0,190629	4	0,121115	4	0,114057	4	0,114215	
5	0,199936	5	0,189731	5	0,117428	5	0,115714	5	0,116321	
6		6		6		6		6		
7	0,1763022	7	0,192771	7	0,1190606	7	0,1165952	7	0,1139582	
EPSILON =0,0001									Iteraciones 500	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]										
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,160573	1	0,146888	1	0,14985	1	0,140317	1	0,142439	
2	0,14623	2	0,146133	2	0,146951	2	0,14332	2	0,143924	
3	0,148949	3	0,143438	3	0,147331	3	0,140574	3	0,143279	
4	0,148153	4	0,143732	4	0,147772	4	0,140809	4	0,139327	
5	0,146731	5	0,144228	5	0,15664	5	0,142379	5	0,14024	
6		6		6		6		6		
7	0,1501272	7	0,1448838	7	0,1497088	7	0,1414798	7	0,1418418	
EPSILON =0,000001									Iteraciones 10	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]										
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,187441	1	0,158907	1	0,130574	1	0,130204	1	0,0670769	
2	0,184818	2	0,155648	2	0,127138	2	0,12507	2	0,067605	
3	0,187744	3	0,160988	3	0,123791	3	0,130012	3	0,0677531	
4	0,184549	4	0,152466	4	0,127268	4	0,129671	4	0,0653031	
5	0,193669	5	0,166011	5	0,123482	5	0,141242	5	0,0657458	
6		6		6		6		6		
7	0,1876442	7	0,158804	7	0,1264506	7	0,1312398	7	0,06669678	
EPSILON =0,000001									Iteraciones 50	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]										
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,192555	1	0,121002	1	0,946936	1	0,183178	1	0,234539	
2	0,186575	2	0,121707	2	0,978893	2	0,183409	2	0,222486	
3	0,184575	3	0,126686	3	0,900987	3	0,185179	3	0,223105	
4	0,186858	4	0,121906	4	0,904331	4	0,192746	4	0,223167	
5	0,186224	5	0,120898	5	0,917152	5	0,184616	5	0,222131	
6		6		6		6		6		
7	0,1873574	7	0,1224398	7	0,9296598	7	0,1858256	7	0,2250856	
EPSILON =0,000001									Iteraciones 100	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]										
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,208264	1	0,193428	1	0,12147	1	0,155475	1	0,155355	
2	0,183461	2	0,198798	2	0,117914	2	0,156193	2	0,156212	
3	0,206201	3	0,192062	3	0,117086	3	0,155391	3	0,156117	
4	0,189665	4	0,192568	4	0,11738	4	0,154879	4	0,154336	
5	0,189375	5	0,191934	5	0,11769	5	0,151001	5	0,155752	
6		6		6		6		6		
7	0,1953932	7	0,193758	7	0,118308	7	0,1545878	7	0,1555544	
EPSILON =0,000001									Iteraciones 500	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]										
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,188697	1	0,193149	1	0,118825	1	0,155427	1	0,155965	
2	0,194221	2	0,192067	2	0,116236	2	0,154893	2	0,153379	
3	0,187528	3	0,200354	3	0,117577	3	0,160149	3	0,155585	
4	0,18512	4	0,197057	4	0,124459	4	0,158258	4	0,154941	
5	0,194518	5	0,19417	5	0,121525	5	0,154112	5	0,152357	
6		6		6		6		6		
7	0,1900168	7	0,1953594	7	0,1197244	7	0,1565678	7	0,1544454	

EPSILON =0,01										Iteraciones 10	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,075402		1	0,071739	1	0,24974	1	0,243563	1	0,246757	
2	0,0747421		2	0,0696681	2	0,252016	2	0,24954	2	0,24294	
3	0,0749409		3	0,069345	3	0,250145	3	0,243455	3	0,244182	
4	0,0763309		4	0,0693769	4	0,25176	4	0,25033	4	0,248976	
5	0,074228		5	0,0711651	5	0,252843	5	0,242295	5	0,249478	
6			6		6		6		6		
7	0,07512878		7	0,07025882	7	0,2513008	7	0,2458366	7	0,2464666	

EPSILON =0,01										Iteraciones 50	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0758269		1	0,155748	1	0,135244	1	0,161903	1	0,13782	
2	0,077054		2	0,152469	2	0,136126	2	0,160359	2	0,128655	
3	0,076704		3	0,153337	3	0,138004	3	0,161563	3	0,123793	
4	0,0794389		4	0,152746	4	0,1352	4	0,163235	4	0,128295	
5	0,0755839		5	0,156568	5	0,140469	5	0,16127	5	0,128628	
6			6		6		6		6		
7	0,07692154		7	0,1541736	7	0,1370086	7	0,161666	7	0,1294382	

EPSILON =0,01										Iteraciones 100	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,069592		1	0,102035	1	0,158605	1	0,211796	1	0,0922651	
2	0,07128		2	0,102339	2	0,157247	2	0,221349	2	0,0918171	
3	0,0691199		3	0,106729	3	0,156894	3	0,228492	3	0,091361	
4	0,070884		4	0,099673	4	0,15832	4	0,216808	4	0,0953641	
5	0,0681221		5	0,103773	5	0,157782	5	0,226856	5	0,0912111	
6			6		6		6		6		
7	0,0697996		7	0,1029098	7	0,1577696	7	0,2210602	7	0,09240368	

EPSILON =0,01										Iteraciones 500	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0683589		1	0,065733	1	0,0662031	1	0,0675142	1	0,0671921	
2	0,066417		2	0,066376	2	0,0651431	2	0,0645871	2	0,0680101	
3	0,0662339		3	0,0654659	3	0,0674429	3	0,065464	3	0,0651691	
4	0,06529		4	0,0643201	4	0,067647	4	0,0656109	4	0,0649941	
5	0,0690632		5	0,0674081	5	0,0657098	5	0,0646029	5	0,0656722	
6			6		6		6		6		
7	0,0670726		7	0,06586062	7	0,06642918	7	0,06555582	7	0,06620752	

EPSILON =0,0001										Iteraciones 10	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,129802		1	0,118709	1	0,623085	1	0,584359	1	0,588316	
2	0,122788		2	0,124179	2	0,590254	2	0,58298	2	0,5816	
3	0,122613		3	0,119134	3	0,58984	3	0,583896	3	0,582727	
4	0,123201		4	0,122421	4	0,588828	4	0,600498	4	0,599434	
5	0,122764		5	0,120848	5	0,590107	5	0,583186	5	0,592837	
6			6		6		6		6		
7	0,1242336		7	0,1210582	7	0,5964228	7	0,5869838	7	0,5889828	

EPSILON =0,0001										Iteraciones 50	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,124477		1	0,235806	1	0,21838	1	0,499489	1	0,213507	
2	0,125476		2	0,235228	2	0,21795	2	0,498026	2	0,218018	
3	0,123587		3	0,238706	3	0,219686	3	0,49743	3	0,212419	
4	0,125705		4	0,235098	4	0,220183	4	0,526058	4	0,206316	
5	0,128276		5	0,234803	5	0,216968	5	0,501193	5	0,21229	
6			6		6		6		6		
7	0,1255042		7	0,2359282	7	0,2186334	7	0,5044392	7	0,21251	

EPSILON =0,0001										Iteraciones 100	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,117181		1	0,115406	1	0,332574	1	0,537093	1	0,197358	
2	0,119122		2	0,118258	2	0,324903	2	0,540623	2	0,196954	
3	0,119771		3	0,118406	3	0,323874	3	0,551928	3	0,19588	
4	0,117203		4	0,116358	4	0,323726	4	0,535883	4	0,196847	
5	0,118857		5	0,119836	5	0,322679	5	0,545551	5	0,197859	
6			6		6		6		6		
7	0,1184268		7	0,1176528	7	0,3255512	7	0,5422156	7	0,1969796	

EPSILON =0,0001										Iteraciones 500	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0813069		1	0,0771191	1	0,0755961	1	0,073535	1	0,0710001	
2	0,0805779		2	0,0762072	2	0,0771279	2	0,0752721	2	0,0710402	
3	0,0809989		3	0,0753791	3	0,0755501	3	0,075022	3	0,0705051	
4	0,0793581		4	0,0752559	4	0,0758729	4	0,072757	4	0,0707381	
5	0,0825679		5	0,0747809	5	0,075084	5	0,07447	5	0,070401	
6			6		6		6		6		
7	0,08096194		7	0,07574844	7	0,0758462	7	0,07421122	7	0,0707369	

EPSILON =0,000001										Iteraciones 10	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,171564		1	0,171483		1	0,608826		1	0,600365		1	0,59884	
2	0,172247		2	0,171977		2	0,593737		2	0,584578		2	0,586529	
3	0,177187		3	0,167364		3	0,608813		3	0,592813		3	0,600271	
4	0,171948		4	0,168106		4	0,593218		4	0,661948		4	0,583225	
5	0,171386		5	0,167566		5	0,593676		5	0,584641		5	0,598911	
6			6			6			6			6		
7	0,1728664		7	0,1692992		7	0,599654		7	0,604869		7	0,5935552	
EPSILON =0,000001										Iteraciones 50	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,175279		1	0,241773		1	0,217863		1	0,510268		1	0,215307	
2	0,174379		2	0,238863		2	0,228663		2	0,513317		2	0,213406	
3	0,178618		3	0,235855		3	0,216061		3	0,501761		3	0,206518	
4	0,17589		4	0,24867		4	0,217563		4	0,499402		4	0,205718	
5	0,174954		5	0,241353		5	0,220015		5	0,512344		5	0,212999	
6			6			6			6			6		
7	0,175824		7	0,2413028		7	0,220033		7	0,5074184		7	0,2107896	
EPSILON =0,000001										Iteraciones 100	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,165733		1	0,148124		1	0,332318		1	0,539165		1	0,204096	
2	0,170259		2	0,149206		2	0,32311		2	0,540576		2	0,195728	
3	0,167263		3	0,146454		3	0,322175		3	0,544457		3	0,204091	
4	0,168397		4	0,148341		4	0,322309		4	0,53954		4	0,203939	
5	0,168413		5	0,156166		5	0,330442		5	0,538123		5	0,196664	
6			6			6			6			6		
7	0,168013		7	0,1496582		7	0,3260708		7	0,5403722		7	0,2009036	
EPSILON =0,000001										Iteraciones 500	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,133625		1	0,117647		1	0,127067		1	0,123336		1	0,123393	
2	0,131203		2	0,120126		2	0,125801		2	0,122077		2	0,119859	
3	0,129567		3	0,118014		3	0,126338		3	0,127326		3	0,120301	
4	0,132079		4	0,121008		4	0,124033		4	0,121298		4	0,116832	
5	0,131926		5	0,117866		5	0,124446		5	0,124188		5	0,121597	
6			6			6			6			6		
7	0,13168		7	0,1189322		7	0,125537		7	0,123645		7	0,1203964	
EPSILON =0,01										Iteraciones 10	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,0144119		1	0,0137529		1	0,0120051		1	0,012742		1	0,00668001	
2	0,013978		2	0,013726		2	0,012697		2	0,0127161		2	0,00753808	
3	0,0140531		3	0,0135119		3	0,0124729		3	0,0125499		3	0,00721812	
4	0,0143471		4	0,0137591		4	0,01264		4	0,0128069		4	0,00731397	
5	0,014291		5	0,013377		5	0,012656		5	0,0132499		5	0,00734782	
6			6			6			6			6		
7	0,01421622		7	0,01362538		7	0,0124942		7	0,01281296		7	0,0072196	
EPSILON =0,01										Iteraciones 50	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,0123241		1	0,0123141		1	0,0110691		1	0,0104301		1	0,0108218	
2	0,0126121		2	0,0122271		2	0,011127		2	0,010602		2	0,0106149	
3	0,0123291		3	0,0126681		3	0,011317		3	0,010596		3	0,010381	
4	0,0124409		4	0,011982		4	0,0109799		4	0,0104659		4	0,010447	
5	0,0126781		5	0,0119948		5	0,010386		5	0,0102789		5	0,0103459	
6			6			6			6			6		
7	0,01247686		7	0,01223722		7	0,0109758		7	0,01047458		7	0,01052212	
EPSILON =0,01										Iteraciones 100	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,0107591		1	0,00965691		1	0,00870085		1	0,0100989		1	0,010139	
2	0,0102839		2	0,00963998		2	0,00968003		2	0,00883698		2	0,00963807	
3	0,0110059		3	0,00913501		3	0,00971198		3	0,00967407		3	0,00943804	
4	0,0105829		4	0,00885797		4	0,00945902		4	0,0103021		4	0,00946712	
5	0,0107801		5	0,00946903		5	0,00878		5	0,009938		5	0,00834703	
6			6			6			6			6		
7	0,01068238		7	0,00935178		7	0,00926638		7	0,00977001		7	0,00940585	
EPSILON =0,01										Iteraciones 500	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,0317941		1	0,031918		1	0,0314338		1	0,0325999		1	0,0320971	
2	0,0343161		2	0,0319819		2	0,0317569		2	0,0327389		2	0,033031	
3	0,032124		3	0,030947		3	0,0324152		3	0,0312481		3	0,032721	
4	0,0320129		4	0,031714		4	0,0325241		4	0,0324409		4	0,0325832	
5	0,032644		5	0,0326428		5	0,03107		5	0,0319679		5	0,033468	
6			6			6			6			6		
7	0,03257822		7	0,03184074		7	0,03184		7	0,03219914		7	0,03278006	

EPSILON =0,0001 — Iteraciones 10

FUNCION : BOOTH [-10,10][-10,10]

N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO
1	0,0220168	1	0,0213342	1	0,0208242	1	0,0209651	1	0,011152
2	0,0213189	2	0,0213139	2	0,019645	2	0,0195351	2	0,0108271
3	0,021939	3	0,0218201	3	0,0205569	3	0,0198021	3	0,0108531
4	0,022398	4	0,0209279	4	0,0204341	4	0,0210159	4	0,0114751
5	0,022444	5	0,0212119	5	0,0201299	5	0,020261	5	0,0111701
6		6		6		6		6	
7	0,02202334	7	0,0213216	7	0,02031802	7	0,02031584	7	0,01109548

EPSILON =0,0001 — Iteraciones 50

FUNCION : BOOTH [-10,10][-10,10]

N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO
1	0,0196078	1	0,0201292	1	0,0192521	1	0,0181029	1	0,018132
2	0,0202222	2	0,01999	2	0,0188711	2	0,01823	2	0,018733
3	0,019994	3	0,0209112	3	0,0186558	3	0,0180988	3	0,017673
4	0,02016	4	0,019717	4	0,0190759	4	0,0183139	4	0,01862
5	0,0198951	5	0,0200641	5	0,0192411	5	0,0180061	5	0,0186229
6		6		6		6		6	
7	0,01997582	7	0,0201623	7	0,0190192	7	0,01815034	7	0,01835618

EPSILON =0,0001 — Iteraciones 100

FUNCION : BOOTH [-10,10][-10,10]

N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO
1	0,0189412	1	0,017251	1	0,0172238	1	0,017107	1	0,012491
2	0,0185721	2	0,0168951	2	0,0161331	2	0,0167949	2	0,0122402
3	0,018172	3	0,0170321	3	0,0174501	3	0,0163581	3	0,0122652
4	0,018254	4	0,017036	4	0,0166459	4	0,017673	4	0,0123029
5	0,0177209	5	0,0169199	5	0,0169189	5	0,0169899	5	0,0123758
6		6		6		6		6	
7	0,01833204	7	0,01702682	7	0,01687436	7	0,01698458	7	0,01233502

EPSILON =0,0001 — Iteraciones 500

FUNCION : BOOTH [-10,10][-10,10]

N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO
1	0,0311909	1	0,0318201	1	0,0305779	1	0,032848	1	0,032968
2	0,031975	2	0,031235	2	0,0325658	2	0,0311711	2	0,0311189
3	0,0318339	3	0,0327201	3	0,0320241	3	0,0315099	3	0,0326388
4	0,0319099	4	0,0325	4	0,0339119	4	0,031229	4	0,0325229
5	0,0327311	5	0,03128	5	0,032299	5	0,0303969	5	0,032553
6		6		6		6		6	
7	0,03192816	7	0,03191104	7	0,03227574	7	0,03143098	7	0,03236032

EPSILON =0,000001 — Iteraciones 10

FUNCION : BOOTH [-10,10][-10,10]

N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO
1	0,0298159	1	0,0300641	1	0,027993	1	0,0281742	1	0,0145569
2	0,0296838	2	0,0293179	2	0,0286052	2	0,0292919	2	0,01456
3	0,030637	3	0,0300381	3	0,0283639	3	0,0281119	3	0,0142441
4	0,030128	4	0,029784	4	0,028616	4	0,0280511	4	0,0145209
5	0,029407	5	0,0294559	5	0,030396	5	0,028724	5	0,0145001
6		6		6		6		6	
7	0,02993434	7	0,029732	7	0,02879482	7	0,02847062	7	0,0144764

EPSILON =0,000001 — Iteraciones 50

FUNCION : BOOTH [-10,10][-10,10]

N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO
1	0,0282311	1	0,0283191	1	0,0274148	1	0,027421	1	0,026741
2	0,0289311	2	0,028264	2	0,0264549	2	0,0259838	2	0,026999
3	0,0284929	3	0,027931	3	0,026706	3	0,0263081	3	0,026428
4	0,0281019	4	0,0289519	4	0,026269	4	0,0277259	4	0,027034
5	0,0277469	5	0,02865	5	0,0265341	5	0,030045	5	0,0284379
6		6		6		6		6	
7	0,02830078	7	0,0284232	7	0,02667576	7	0,02749676	7	0,02712798

EPSILON =0,000001 — Iteraciones 100

FUNCION : BOOTH [-10,10][-10,10]

N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO
1	0,027097	1	0,0255198	1	0,0251141	1	0,025162	1	0,01616
2	0,0263159	2	0,0252519	2	0,0255179	2	0,024008	2	0,0164051
3	0,026659	3	0,0261781	3	0,026108	3	0,0248072	3	0,016263
4	0,0265312	4	0,0251789	4	0,024745	4	0,0248401	4	0,0162292
5	0,026479	5	0,024019	5	0,025214	5	0,024298	5	0,015353
6		6		6		6		6	
7	0,02661642	7	0,02522954	7	0,0253398	7	0,02462306	7	0,01608206

EPSILON =0,000001 — Iteraciones 500

FUNCION : BOOTH [-10,10][-10,10]

N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO
1	0,0326529	1	0,031095	1	0,031014	1	0,0348728	1	0,0308671
2	0,0322371	2	0,032584	2	0,0312171	2	0,0343251	2	0,0322678
3	0,0313091	3	0,032799	3	0,0317152	3	0,0319459	3	0,0314779
4	0,0320258	4	0,0326641	4	0,0310268	4	0,032172	4	0,0327442
5	0,0326798	5	0,032701	5	0,0316091	5	0,031357	5	0,031045
6		6		6		6		6	
7	0,03218094	7	0,03236862	7	0,03131644	7	0,03293456	7	0,0316804

EPSILON =0,01										Iteraciones 10	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0463109		1	0,0252032	1	0,0227611	1	0,023078	1	0,020993	
2	0,0461669		2	0,0261481	2	0,023001	2	0,022748	2	0,021234	
3	0,0452881		3	0,0247509	3	0,02279	3	0,0228128	3	0,0211141	
4	0,0465491		4	0,0254409	4	0,0232468	4	0,0224771	4	0,020505	
5	0,0472569		5	0,0254118	5	0,0233569	5	0,0231371	5	0,0204949	
6			6		6		6		6		
7	0,04631438		7	0,02539098	7	0,02303116	7	0,0228506	7	0,0208682	

EPSILON =0,01										Iteraciones 50	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,04566		1	0,0390019	1	0,037627	1	0,035728	1	0,0190609	
2	0,0474701		2	0,0397711	2	0,037035	2	0,035805	2	0,017771	
3	0,0462141		3	0,0390551	3	0,0371001	3	0,0366399	3	0,0195048	
4	0,047684		4	0,0387981	4	0,037137	4	0,036618	4	0,019155	
5	0,047729		5	0,0390179	5	0,037375	5	0,0355301	5	0,019244	
6			6		6		6		6		
7	0,04695144		7	0,03912882	7	0,03725482	7	0,0360642	7	0,01894714	

EPSILON =0,01										Iteraciones 100	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0460339		1	0,0386622	1	0,041285	1	0,0370829	1	0,027123	
2	0,0466719		2	0,0401752	2	0,0446181	2	0,037436	2	0,026983	
3	0,0467229		3	0,041189	3	0,0412309	3	0,038265	3	0,026855	
4	0,0460491		4	0,040411	4	0,0404251	4	0,0373299	4	0,0269279	
5	0,045944		5	0,0406981	5	0,039221	5	0,036128	5	0,0268888	
6			6		6		6		6		
7	0,04628436		7	0,0402271	7	0,04135602	7	0,03724836	7	0,02695554	

EPSILON =0,01										Iteraciones 500	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0336359		1	0,0342391	1	0,033957	1	0,035578	1	0,033937	
2	0,0346498		2	0,0340729	2	0,0338719	2	0,0332801	2	0,0358582	
3	0,034745		3	0,034132	3	0,03543	3	0,0339429	3	0,0332191	
4	0,034698		4	0,035074	4	0,0343261	4	0,0341561	4	0,0341899	
5	0,0333481		5	0,035665	5	0,03319	5	0,0340412	5	0,033977	
6			6		6		6		6		
7	0,03421536		7	0,0346366	7	0,034155	7	0,03419966	7	0,03423624	

EPSILON =0,0001										Iteraciones 10	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0838821		1	0,0449951	1	0,0419619	1	0,0437071	1	0,0381181	
2	0,0844381		2	0,0441999	2	0,043658	2	0,0426831	2	0,039124	
3	0,084059		3	0,046351	3	0,0418148	3	0,042614	3	0,0373809	
4	0,0878661		4	0,0441298	4	0,042753	4	0,042748	4	0,0381551	
5	0,082855		5	0,045131	5	0,0423	5	0,0432348	5	0,0382581	
6			6		6		6		6		
7	0,08462006		7	0,04496136	7	0,04249754	7	0,0429974	7	0,03820724	

EPSILON =0,0001										Iteraciones 50	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0855389		1	0,077111	1	0,0773132	1	0,0761149	1	0,0369601	
2	0,086798		2	0,0757082	2	0,0796771	2	0,076607	2	0,037374	
3	0,0831959		3	0,077666	3	0,0763249	3	0,0738711	3	0,03757	
4	0,0841639		4	0,0770762	4	0,0752521	4	0,075731	4	0,0371921	
5	0,08483		5	0,076658	5	0,075439	5	0,073704	5	0,03723	
6			6		6		6		6		
7	0,08490534		7	0,07684388	7	0,07680126	7	0,0752056	7	0,03726524	

EPSILON =0,0001										Iteraciones 100	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,084861		1	0,078177	1	0,079819	1	0,079313	1	0,0471981	
2	0,086483		2	0,0820029	2	0,0795829	2	0,0791399	2	0,047044	
3	0,08621		3	0,0789888	3	0,0806701	3	0,07568	3	0,0456669	
4	0,0849249		4	0,0803151	4	0,0804532	4	0,075567	4	0,047827	
5	0,0860162		5	0,080266	5	0,0769439	5	0,075608	5	0,0484221	
6			6		6		6		6		
7	0,08569902		7	0,07994996	7	0,07949382	7	0,07706158	7	0,04723162	

EPSILON =0,0001										Iteraciones 500	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,06903		1	0,060812	1	0,0625029	1	0,0453439	1	0,050216	
2	0,0673311		2	0,059279	2	0,061774	2	0,0491049	2	0,04845	
3	0,066741		3	0,061671	3	0,062752	3	0,0481079	3	0,0493572	
4	0,0690529		4	0,0605721	4	0,061713	4	0,0482271	4	0,0487139	
5	0,06704		5	0,060966	5	0,062248	5	0,049367	5	0,0486581	
6			6		6		6		6		
7	0,067839		7	0,06066002	7	0,06219798	7	0,04803016	7	0,04907904	

EPSILON =0,000001										Iteraciones 10	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,125699		1	0,0639241		1	0,0602622		1	0,0603511		1	0,0558419	
2	0,125501		2	0,064038		2	0,060272		2	0,0594661		2	0,056252	
3	0,130707		3	0,0630281		3	0,0607009		3	0,0602648		3	0,0557411	
4	0,12369		4	0,067157		4	0,0613079		4	0,063343		4	0,055876	
5	0,122785		5	0,062866		5	0,0613639		5	0,06055		5	0,057487	
6			6			6			6			6		
7	0,1256764		7	0,06420264		7	0,06078138		7	0,060795		7	0,0562396	

EPSILON =0,000001										Iteraciones 50	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,125206		1	0,120503		1	0,114665		1	0,112249		1	0,055171	
2	0,123254		2	0,114399		2	0,113984		2	0,114445		2	0,057338	
3	0,121878		3	0,11482		3	0,117987		3	0,111231		3	0,0560448	
4	0,123751		4	0,118123		4	0,113846		4	0,114849		4	0,053961	
5	0,124001		5	0,116064		5	0,111651		5	0,11266		5	0,0553012	
6			6			6			6			6		
7	0,123618		7	0,1167818		7	0,1144266		7	0,1130868		7	0,0555632	

EPSILON =0,000001										Iteraciones 100	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,125207		1	0,115718		1	0,117287		1	0,114065		1	0,0676	
2	0,121724		2	0,116093		2	0,120318		2	0,117716		2	0,065516	
3	0,123423		3	0,119187		3	0,117586		3	0,11575		3	0,0670588	
4	0,125078		4	0,115901		4	0,115907		4	0,113505		4	0,0681989	
5	0,121996		5	0,119104		5	0,117364		5	0,114253		5	0,067209	
6			6			6			6			6		
7	0,1234856		7	0,1172006		7	0,1176924		7	0,1150578		7	0,06711654	

EPSILON =0,000001										Iteraciones 500	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,105414		1	0,0995269		1	0,101859		1	0,069093		1	0,0697072	
2	0,107395		2	0,098485		2	0,100807		2	0,0690858		2	0,0679941	
3	0,11018		3	0,100404		3	0,102012		3	0,0679109		3	0,06951	
4	0,105589		4	0,0985289		4	0,0994501		4	0,0684021		4	0,068742	
5	0,10788		5	0,099144		5	0,100833		5	0,0684328		5	0,068917	
6			6			6			6			6		
7	0,1072916		7	0,09921776		7	0,10099222		7	0,06858492		7	0,06897406	

EPSILON =0,01										Iteraciones 10	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,0722249		1	0,0407438		1	0,0342629		1	0,0324252		1	0,03355	
2	0,069958		2	0,039392		2	0,0339658		2	0,0323832		2	0,032748	
3	0,070148		3	0,041784		3	0,0340791		3	0,0326779		3	0,0317161	
4	0,0688541		4	0,0390091		4	0,0350611		4	0,03371		4	0,032465	
5	0,071959		5	0,0390968		5	0,034152		5	0,032927		5	0,03198	
6			6			6			6			6		
7	0,0706288		7	0,04000514		7	0,03430418		7	0,03282466		7	0,03249182	

EPSILON =0,01										Iteraciones 50	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,0690939		1	0,039042		1	0,037926		1	0,0343859		1	0,0336311	
2	0,0688939		2	0,039722		2	0,037673		2	0,0373259		2	0,033916	
3	0,0688891		3	0,0386181		3	0,038583		3	0,0341661		3	0,032763	
4	0,0683959		4	0,039432		4	0,0394201		4	0,033685		4	0,0358448	
5	0,0711958		5	0,03878		5	0,038527		5	0,0335701		5	0,035336	
6			6			6			6			6		
7	0,06929372		7	0,03911882		7	0,03842582		7	0,0346266		7	0,03429818	

EPSILON =0,01										Iteraciones 100	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,0679581		1	0,0389631		1	0,0407012		1	0,0355599		1	0,033154	
2	0,0670838		2	0,0407691		2	0,040818		2	0,036963		2	0,0332031	
3	0,0700371		3	0,040386		3	0,040725		3	0,0377128		3	0,0335169	
4	0,0659139		4	0,0406771		4	0,040508		4	0,0353611		4	0,0343809	
5	0,0696609		5	0,0412362		5	0,041256		5	0,0373819		5	0,0317059	
6			6			6			6			6		
7	0,06813076		7	0,0404063		7	0,04080164		7	0,03659574		7	0,03319216	

EPSILON =0,01										Iteraciones 500	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,048315		1	0,0491221		1	0,0491691		1	0,0495131		1	0,0493429	
2	0,0485909		2	0,0508649		2	0,0510218		2	0,04862		2	0,0510101	
3	0,048538		3	0,0509191		3	0,0515451		3	0,0498619		3	0,049252	
4	0,0492458		4	0,0493419		4	0,047739		4	0,0486898		4	0,0516829	
5	0,049104		5	0,0510008		5	0,05147		5	0,0495629		5	0,0501921	
6			6			6			6			6		
7	0,04875874		7	0,05024976		7	0,050189		7	0,04924954		7	0,050296	

EPSILON =0,0001 — Iteraciones 10

FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]

N°	N PRO,	TIEMPO	N°	N PRO,	TIEMPO	N°	N PRO,	TIEMPO	N°	N PRO,	TIEMPO	N°	N PRO,	TIEMPO
1		0,081888	1		0,04671	1		0,0376301	1		0,0358529	1		0,035732
2		0,078691	2		0,0458009	2		0,0375099	2		0,0352001	2		0,0350161
3		0,07883	3		0,0479069	3		0,0374038	3		0,0350301	3		0,034735
4		0,078213	4		0,0463831	4		0,0365448	4		0,034657	4		0,0347469
5		0,0778639	5		0,045819	5		0,037513	5		0,0362132	5		0,034251
6			6			6			6			6		
7		0,07909718	7		0,04652398	7		0,03732032	7		0,03539066	7		0,0348962

EPSILON =0,0001 — Iteraciones 50

FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]

N°	N PRO,	TIEMPO	N°	N PRO,	TIEMPO	N°	N PRO,	TIEMPO	N°	N PRO,	TIEMPO	N°	N PRO,	TIEMPO
1		0,0788329	1		0,0448201	1		0,0694098	1		0,065449	1		0,0665212
2		0,077889	2		0,043035	2		0,0698941	2		0,0655131	2		0,065943
3		0,0781128	3		0,0446529	3		0,071497	3		0,0695319	3		0,066385
4		0,0772302	4		0,0435231	4		0,068547	4		0,0651779	4		0,064291
5		0,0790241	5		0,0452731	5		0,0673831	5		0,0660279	5		0,067426
6			6			6			6			6		
7		0,0782178	7		0,04426084	7		0,0693462	7		0,06633996	7		0,06611324

EPSILON =0,0001 — Iteraciones 100

FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]

N°	N PRO,	TIEMPO	N°	N PRO,	TIEMPO	N°	N PRO,	TIEMPO	N°	N PRO,	TIEMPO	N°	N PRO,	TIEMPO
1		0,0766821	1		0,043499	1		0,071481	1		0,0697219	1		0,064791
2		0,0752499	2		0,045403	2		0,0714788	2		0,066309	2		0,0626118
3		0,077683	3		0,0439589	3		0,071743	3		0,0693018	3		0,0644209
4		0,0766098	4		0,047091	4		0,072062	4		0,0699341	4		0,0648479
5		0,0789931	5		0,0456159	5		0,0712161	5		0,067477	5		0,0672882
6			6			6			6			6		
7		0,07704358	7		0,04511356	7		0,07159618	7		0,06854876	7		0,06479196

EPSILON =0,0001 — Iteraciones 500

FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]

N°	N PRO,	TIEMPO	N°	N PRO,	TIEMPO	N°	N PRO,	TIEMPO	N°	N PRO,	TIEMPO	N°	N PRO,	TIEMPO
1		0,0492051	1		0,049438	1		0,0494211	1		0,055501	1		0,053175
2		0,0494311	2		0,048851	2		0,0486181	2		0,0502682	2		0,049921
3		0,0510969	3		0,050755	3		0,049494	3		0,0496171	3		0,0558238
4		0,0485821	4		0,0491691	4		0,049361	4		0,04951	4		0,050977
5		0,0492949	5		0,0555661	5		0,048604	5		0,051321	5		0,0492671
6			6			6			6			6		
7		0,04952202	7		0,05075584	7		0,04909964	7		0,05124346	7		0,05183278

EPSILON =0,000001 — Iteraciones 10

FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]

N°	N PRO,	TIEMPO	N°	N PRO,	TIEMPO	N°	N PRO,	TIEMPO	N°	N PRO,	TIEMPO	N°	N PRO,	TIEMPO
1		0,0952308	1		0,053318	1		0,0386882	1		0,0373199	1		0,039057
2		0,086597	2		0,0529819	2		0,03795	2		0,040678	2		0,0395019
3		0,0866451	3		0,0556901	3		0,0389791	3		0,0386369	3		0,0359261
4		0,087724	4		0,052639	4		0,0387881	4		0,0372941	4		0,036483
5		0,0905759	5		0,0535271	5		0,0424879	5		0,038615	5		0,0372689
6			6			6			6			6		
7		0,08935456	7		0,05363122	7		0,03937866	7		0,03850878	7		0,03764738

EPSILON =0,000001 — Iteraciones 50

FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]

N°	N PRO,	TIEMPO	N°	N PRO,	TIEMPO	N°	N PRO,	TIEMPO	N°	N PRO,	TIEMPO	N°	N PRO,	TIEMPO
1		0,08548	1		0,0478401	1		0,109689	1		0,102453	1		0,0956819
2		0,0899782	2		0,0485921	2		0,0986869	2		0,0959129	2		0,0955262
3		0,0906019	3		0,04882	3		0,102954	3		0,09971	3		0,09881
4		0,0971861	4		0,0479629	4		0,100536	4		0,0976212	4		0,101626
5		0,0872788	5		0,0494881	5		0,0990269	5		0,0992892	5		0,0949931
6			6			6			6			6		
7		0,090105	7		0,04854064	7		0,10217856	7		0,09899726	7		0,09732744

EPSILON =0,000001 — Iteraciones 100

FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]

N°	N PRO,	TIEMPO	N°	N PRO,	TIEMPO	N°	N PRO,	TIEMPO	N°	N PRO,	TIEMPO	N°	N PRO,	TIEMPO
1		0,0881798	1		0,05161	1		0,101182	1		0,099333	1		0,0986619
2		0,0834088	2		0,0484099	2		0,104612	2		0,10127	2		0,0944309
3		0,0857852	3		0,0515819	3		0,103017	3		0,098599	3		0,0963719
4		0,0914011	4		0,0518489	4		0,11032	4		0,0987339	4		0,0948761
5		0,0851278	5		0,0517359	5		0,105682	5		0,0997241	5		0,0954411
6			6			6			6			6		
7		0,08678054	7		0,05103732	7		0,1049626	7		0,099532	7		0,09595638

EPSILON =0,000001 — Iteraciones 500

FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]

N°	N PRO,	TIEMPO	N°	N PRO,	TIEMPO	N°	N PRO,	TIEMPO	N°	N PRO,	TIEMPO	N°	N PRO,	TIEMPO
1		0,0522289	1		0,050504	1		0,0527642	1		0,0522311	1		0,0510459
2		0,0514331	2		0,049649	2		0,051398	2		0,052454	2		0,0532179
3		0,054832	3		0,052995	3		0,053048	3		0,0499601	3		0,054117
4		0,054868	4		0,0513239	4		0,05284	4		0,0496399	4		0,05055
5		0,0549471	5		0,051384	5		0,0517659	5		0,0522449	5		0,0498641
6			6			6			6			6		
7		0,05366182	7		0,05117118	7		0,05236322	7		0,051306	7		0,05175898

EPSILON =0,01										Iteraciones 10	
FUNCION : TRECCANI [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,0535641		1	0,0286851		1	0,0272281		1	0,0262311		1	0,0245609	
2	0,0600951		2	0,0296841		2	0,027122		2	0,0285459		2	0,0259712	
3	0,0535021		3	0,0295391		3	0,028929		3	0,026118		3	0,0256329	
4	0,0528529		4	0,0302839		4	0,027215		4	0,026803		4	0,0246029	
5	0,0552521		5	0,032527		5	0,02724		5	0,026607		5	0,0248759	
6			6			6			6			6		
7	0,05505326		7	0,03014384		7	0,02754682		7	0,026861		7	0,02512876	

EPSILON =0,01										Iteraciones 50	
FUNCION : TRECCANI [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,0519822		1	0,0303099		1	0,036845		1	0,0263181		1	0,024796	
2	0,0523379		2	0,031611		2	0,0364661		2	0,025775		2	0,0249538	
3	0,0561302		3	0,0303538		3	0,0362639		3	0,026217		3	0,023973	
4	0,0523379		4	0,0318611		4	0,036577		4	0,0261719		4	0,024781	
5	0,05338		5	0,0310872		5	0,0375891		5	0,0268409		5	0,02562	
6			6			6			6			6		
7	0,05323364		7	0,0310446		7	0,03674822		7	0,02626458		7	0,02482476	

EPSILON =0,01										Iteraciones 100	
FUNCION : TRECCANI [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,0519521		1	0,0310869		1	0,031302		1	0,0262229		1	0,025218	
2	0,0516648		2	0,034095		2	0,0296202		2	0,0269122		2	0,0254991	
3	0,0542481		3	0,0331969		3	0,0309839		3	0,026197		3	0,023684	
4	0,0517101		4	0,0334589		4	0,029974		4	0,027828		4	0,024785	
5	0,0530519		5	0,033848		5	0,0295811		5	0,0274448		5	0,025599	
6			6			6			6			6		
7	0,0525254		7	0,03313714		7	0,03029224		7	0,02692098		7	0,02495702	

EPSILON =0,01										Iteraciones 500	
FUNCION : TRECCANI [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,0394831		1	0,041106		1	0,042403		1	0,0391412		1	0,04144	
2	0,041748		2	0,0400369		2	0,0413289		2	0,0401762		2	0,0398459	
3	0,040895		3	0,0410919		3	0,0388501		3	0,0408421		3	0,0399179	
4	0,0419459		4	0,0423591		4	0,043047		4	0,040565		4	0,0415511	
5	0,0425091		5	0,0409951		5	0,040976		5	0,0415571		5	0,0417449	
6			6			6			6			6		
7	0,04131622		7	0,0411178		7	0,041321		7	0,04045632		7	0,04089996	

EPSILON =0,0001										Iteraciones 10	
FUNCION : TRECCANI [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,088177		1	0,060668		1	0,0572128		1	0,055284		1	0,053663	
2	0,0848041		2	0,0576119		2	0,057729		2	0,0548439		2	0,05423	
3	0,0834389		3	0,0604811		3	0,0560701		3	0,0538092		3	0,0538521	
4	0,0820901		4	0,0598011		4	0,0548739		4	0,0555201		4	0,0524328	
5	0,084698		5	0,058636		5	0,054307		5	0,0548542		5	0,0536351	
6			6			6			6			6		
7	0,08464162		7	0,05943962		7	0,05603856		7	0,05486228		7	0,0535626	

EPSILON =0,0001										Iteraciones 50	
FUNCION : TRECCANI [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,0814178		1	0,060303		1	0,0724909		1	0,053762		1	0,056133	
2	0,085907		2	0,0597181		2	0,0651841		2	0,0540259		2	0,0533812	
3	0,0809572		3	0,060632		3	0,0675819		3	0,0532289		3	0,0537949	
4	0,081846		4	0,059269		4	0,065742		4	0,0545499		4	0,053556	
5	0,0836551		5	0,059212		5	0,0649579		5	0,054235		5	0,0559871	
6			6			6			6			6		
7	0,08275662		7	0,05982682		7	0,06719136		7	0,05396034		7	0,05457044	

EPSILON =0,0001										Iteraciones 100	
FUNCION : TRECCANI [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,106825		1	0,0589142		1	0,0544741		1	0,0489829		1	0,0458121	
2	0,103455		2	0,0593441		2	0,0545599		2	0,0499079		2	0,046896	
3	0,101583		3	0,060075		3	0,056458		3	0,0497689		3	0,0459409	
4	0,101187		4	0,0617559		4	0,0565569		4	0,0487168		4	0,0461788	
5	0,106721		5	0,0597849		5	0,0545092		5	0,0502422		5	0,046061	
6			6			6			6			6		
7	0,1039542		7	0,05997482		7	0,05531162		7	0,04952374		7	0,04617776	

EPSILON =0,0001										Iteraciones 500	
FUNCION : TRECCANI [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,081033		1	0,06566		1	0,0574338		1	0,0555401		1	0,057215	
2	0,082262		2	0,0651021		2	0,0567961		2	0,056078		2	0,0556359	
3	0,083957		3	0,0650258		3	0,0561512		3	0,0549221		3	0,0550869	
4	0,086442		4	0,065094		4	0,056603		4	0,057724		4	0,055352	
5	0,0878501		5	0,0640092		5	0,056103		5	0,0563409		5	0,0564151	
6			6			6			6			6		
7	0,08430882		7	0,06497822		7	0,05661742		7	0,05612102		7	0,05594098	

EPSILON =0,000001										Iteraciones 10	
FUNCION : TRECCANI [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		
1	0,1599	1	0,0820739	1	0,0780029	1	0,076138	1	0,070266		
2	0,154962	2	0,0835161	2	0,0803881	2	0,0768342	2	0,072041		
3	0,164693	3	0,0859749	3	0,0798252	3	0,076668	3	0,0723591		
4	0,153371	4	0,0813551	4	0,0771439	4	0,0713582	4	0,0723438		
5	0,15531	5	0,0864	5	0,0767851	5	0,0766611	5	0,0721741		
6		6		6		6		6			
7	0,1576472	7	0,083864	7	0,07842904	7	0,0755319	7	0,0718368		
EPSILON =0,000001										Iteraciones 50	
FUNCION : TRECCANI [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		
1	0,151021	1	0,0815001	1	0,0860341	1	0,0725279	1	0,0712411		
2	0,162089	2	0,0828998	2	0,0833249	2	0,066972	2	0,0711861		
3	0,1544	3	0,0825851	3	0,0869169	3	0,0691831	3	0,06811		
4	0,152274	4	0,0886619	4	0,081774	4	0,0681169	4	0,0682938		
5	0,152158	5	0,083951	5	0,0921969	5	0,0688488	5	0,06914		
6		6		6		6		6			
7	0,1543884	7	0,08391958	7	0,08604936	7	0,06912974	7	0,0695942		
EPSILON =0,000001										Iteraciones 100	
FUNCION : TRECCANI [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		
1	0,160933	1	0,08605	1	0,0783458	1	0,0777521	1	0,0675809		
2	0,170442	2	0,0882249	2	0,0765688	2	0,0701449	2	0,0698478		
3	0,158313	3	0,0893459	3	0,085721	3	0,07112	3	0,0689991		
4	0,161041	4	0,0848122	4	0,0773809	4	0,073586	4	0,07339		
5	0,149847	5	0,0895798	5	0,0786519	5	0,0710192	5	0,0710809		
6		6		6		6		6			
7	0,1601152	7	0,08760256	7	0,07933368	7	0,07272444	7	0,07017974		
EPSILON =0,000001										Iteraciones 500	
FUNCION : TRECCANI [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		
1	0,133478	1	0,0922561	1	0,0820889	1	0,0826831	1	0,0800641		
2	0,136151	2	0,0900669	2	0,079627	2	0,082793	2	0,076175		
3	0,132858	3	0,092423	3	0,0816331	3	0,081393	3	0,078151		
4	0,133687	4	0,0918069	4	0,085392	4	0,0767291	4	0,07441		
5	0,130277	5	0,0889819	5	0,0821788	5	0,0806401	5	0,0764711		
6		6		6		6		6			
7	0,1332902	7	0,09110696	7	0,08218396	7	0,08084766	7	0,07705424		

EPSILON =0,01										Iteraciones 10	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]											
	N PRO.	2	N PRO.	3	N PRO.	4	N PRO.	5	N PRO.	6	
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		
1	5,92227	1	3,20283	1	2,54428	1	2,01052	1	1,94293		
2	5,92134	2	3,21772	2	2,54542	2	2,02668	2	1,97927		
3	6,07221	3	3,34115	3	2,47092	3	2,01502	3	1,98090		
4	6,11948	4	3,19090	4	2,49081	4	2,09658	4	1,95484		
5	5,87535	5	3,21465	5	2,55062	5	2,02469	5	1,93610		
6		6		6		6		6			
7	5,98213	7	3,23345	7	2,52041	7	2,03470	7	1,95881		
EPSILON =0,01										Iteraciones 50	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]											
	N PRO.	2	N PRO.	3	N PRO.	4	N PRO.	5	N PRO.	6	
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		
1	6,04778	1	3,95515	1	2,86721	1	3,09542	1	1,96147		
2	5,83420	2	3,95900	2	2,87399	2	3,09648	2	1,91348		
3	5,88932	3	4,51064	3	2,85408	3	3,09905	3	1,91507		
4	6,04623	4	4,43441	4	2,92697	4	3,20032	4	1,91449		
5	6,04103	5	3,95996	5	2,92098	5	3,08901	5	1,89139		
6		6		6		6		6			
7	5,97171	7	4,16383	7	2,88865	7	3,11606	7	1,91918		
EPSILON =0,01										Iteraciones 100	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]											
	N PRO,	2	N PRO,	3	N PRO.	4	N PRO.	5	N PRO.	6	
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		
1	5,92177	1	3,98834	1	2,85812	1	3,15706	1	1,89158		
2	5,86086	2	3,99712	2	2,92312	2	3,06398	2	2,06682		
3	6,03918	3	4,06039	3	2,86462	3	3,07001	3	1,90847		
4	5,87512	4	3,99256	4	2,84377	4	3,07801	4	1,90925		
5	6,03698	5	3,95778	5	2,87653	5	3,10547	5	1,95760		
6		6		6		6		6			
7	5,94678	7	3,99924	7	2,87323	7	3,09491	7	1,94674		
EPSILON =0,01										Iteraciones 500	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]											
	N PRO.	2	N PRO.	3	N PRO.	4	N PRO.	5	N PRO.	6	
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		
1	5,83076	1	3,63191	1	2,32223	1	1,90065	1	1,88813		
2	5,81619	2	3,62483	2	2,28922	2	2,01821	2	1,90339		
3	5,95015	3	3,64539	3	2,24345	3	1,95032	3	1,90771		
4	5,86230	4	3,61029	4	2,22655	4	1,91647	4	1,95812		
5	5,80934	5	3,61148	5	2,24173	5	1,89762	5	1,90533		
6		6		6		6		6			
7	5,85375	7	3,62478	7	2,26464	7	1,93665	7	1,91254		

EPSILON =0,0001									Iteraciones 10	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]										
	N PRO.	2	N PRO.	3	N PRO.	4	N PRO.	5	N PRO.	6
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	18,5549		1	11,4770	1	26,9167	1	21,4907	1	20,0346
2	18,6556		2	11,4876	2	27,1946	2	20,1833	2	20,1685
3	18,7995		3	11,4346	3	26,9002	3	20,7470	3	20,2831
4	18,7531		4	11,3837	4	28,0191	4	20,0856	4	21,7432
5	19,1576		5	11,4710	5	27,8879	5	20,1026	5	20,2679
6			6		6		6		6	
7	18,78414		7	11,45078	7	27,38370	7	20,52184	7	20,49946

EPSILON =0,0001									Iteraciones 50	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]										
	N PRO.	2	N PRO.	3	N PRO.	4	N PRO.	5	N PRO.	6
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	19,2133		1	20,1162	1	15,3244	1	14,8688	1	15,3005
2	19,2827		2	20,7533	2	15,6973	2	14,9399	2	15,4900
3	18,7189		3	20,2367	3	15,2194	3	15,1768	3	15,2654
4	19,0700		4	20,4487	4	15,1713	4	14,8756	4	14,9045
5	18,7880		5	20,2088	5	15,8467	5	14,9402	5	15,3004
6			6		6		6		6	
7	19,01458		7	20,35274	7	15,45182	7	14,96026	7	15,25216

EPSILON =0,0001									Iteraciones 100	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]										
	N PRO.	2	N PRO.	3	N PRO.	4	N PRO.	5	N PRO.	6
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	18,5710		1	14,45500	1	24,2969	1	24,3974	1	19,9450
2	18,5640		2	13,46690	2	24,3845	2	24,9617	2	20,0501
3	18,5376		3	13,73630	3	24,4953	3	24,4925	3	19,9707
4	19,2583		4	13,52250	4	24,4100	4	24,9820	4	20,0158
5	19,1557		5	13,49090	5	24,2664	5	25,3883	5	19,9602
6			6		6		6		6	
7	18,81732		7	13,73432	7	24,37062	7	24,84438	7	19,98836

EPSILON =0,0001									Iteraciones 500	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]										
	N PRO.	2	N PRO.	3	N PRO.	4	N PRO.	5	N PRO.	6
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	18,5038		1	21,8719	1	18,3548	1	15,9306	1	16,0853
2	18,5290		2	21,5860	2	18,5857	2	16,3944	2	16,3245
3	19,1244		3	20,7996	3	18,9403	3	16,0685	3	16,0296
4	18,6091		4	22,8594	4	18,4637	4	16,3796	4	15,9276
5	19,1008		5	20,9682	5	18,3599	5	16,0862	5	16,5908
6			6		6		6		6	
7	18,77342		7	21,61702	7	18,54088	7	16,17186	7	16,19156

EPSILON =0,000001									Iteraciones 10	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]										
	N PRO.	2	N PRO.	3	N PRO.	4	N PRO.	5	N PRO.	6
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	18,8813		1	11,4559	1	40,9306	1	34,0720	1	34,3572
2	18,9578		2	11,9708	2	40,7327	2	33,9688	2	36,2031
3	19,1160		3	12,6600	3	42,6051	3	34,8112	3	36,7042
4	18,7293		4	11,4723	4	41,1842	4	33,9406	4	35,1519
5	19,2985		5	11,5120	5	42,6642	5	34,7961	5	34,0559
6			6		6		6		6	
7	18,99658		7	11,81420	7	41,62336	7	34,31774	7	35,29446

EPSILON =0,000001									Iteraciones 50	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]										
	N PRO.	2	N PRO.	3	N PRO.	4	N PRO.	5	N PRO.	6
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	18,7349		1	33,9606	1	29,4048	1	29,0177	1	28,9973
2	18,8855		2	33,8134	2	28,8324	2	28,4341	2	29,1536
3	19,3567		3	36,0289	3	29,9912	3	28,4224	3	28,4302
4	18,8732		4	34,8601	4	28,7152	4	28,2858	4	28,1676
5	18,9696		5	34,2693	5	28,7776	5	29,0518	5	28,0879
6			6		6		6		6	
7	18,96398		7	34,58646	7	29,14424	7	28,64236	7	28,56732

EPSILON =0,000001									Iteraciones 100	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]										
	N PRO.	2	N PRO.	3	N PRO.	4	N PRO.	5	N PRO.	6
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	19,3979		1	13,6887	1	38,8367	1	38,2535	1	34,0832
2	18,7765		2	14,0517	2	39,6157	2	38,0820	2	35,9202
3	19,3796		3	14,2123	3	38,4794	3	40,7023	3	34,7743
4	18,7736		4	14,0181	4	37,9651	4	39,9823	4	34,1015
5	18,8661		5	14,0947	5	38,2199	5	38,4452	5	33,7776
6			6		6		6		6	
7	19,03874		7	14,01310	7	38,62336	7	39,09306	7	34,53136

EPSILON =0,000001									Iteraciones 500	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]										
	N PRO.	2	N PRO.	3	N PRO.	4	N PRO.	5	N PRO.	6
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	19,3187		1	34,8851	1	32,7021	1	29,3782	1	29,1432
2	19,2986		2	36,0686	2	32,6317	2	31,2409	2	30,1898
3	18,7286		3	34,4925	3	32,6454	3	29,3453	3	29,2930
4	18,7246		4	34,8234	4	32,0686	4	30,1513	4	30,1460
5	18,8739		5	35,8403	5	31,9716	5	29,5677	5	29,2349
6			6		6		6		6	
7	18,98888		7	35,22198	7	32,40388	7	29,93668	7	29,60138

DATOS DE ALGOTIMO CAJAS **RESULTADO CON RANKING UF FINAL**

EPSILON =0,01 — Iteraciones 10

FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]

N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO
1	0,159632	1	0,132062	1	0,09659	1	0,097945	1	0,051703
2	0,160566	2	0,12511	2	0,098665	2	0,0962369	2	0,0506241
3	0,158724	3	0,127462	3	0,0973001	3	0,0978079	3	0,0521619
4	0,158956	4	0,126985	4	0,0970039	4	0,101572	4	0,05283
5	0,159355	5	0,131982	5	0,098109	5	0,097683	5	0,0512621
6		6		6		6		6	
7	0,1594466	7	0,1287202	7	0,0975336	7	0,09824896	7	0,05171622

EPSILON =0,01 — Iteraciones 50

FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]

N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO
1	0,158367	1	0,10774	1	0,0740759	1	0,065269	1	0,063323
2	0,155926	2	0,108547	2	0,07793	2	0,0652611	2	0,0640631
3	0,155552	3	0,106178	3	0,075326	3	0,06637	3	0,0651569
4	0,163195	4	0,106445	4	0,0756781	4	0,064698	4	0,0648232
5	0,164381	5	0,106304	5	0,078238	5	0,0649402	5	0,0635219
6		6		6		6		6	
7	0,1594842	7	0,1070428	7	0,0762496	7	0,06530766	7	0,06417762

EPSILON =0,01 — Iteraciones 100

FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]

N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO
1	0,156829	1	0,110482	1	0,074368	1	0,0712352	1	0,061461
2	0,159366	2	0,112073	2	0,0737278	2	0,070636	2	0,0595069
3	0,165943	3	0,109549	3	0,076669	3	0,0727961	3	0,0617149
4	0,366983	4	0,109777	4	0,076679	4	0,070426	4	0,061475
5	0,157514	5	0,106996	5	0,0786469	5	0,069262	5	0,057435
6		6		6		6		6	
7	0,201327	7	0,1097754	7	0,07601814	7	0,07087106	7	0,06031856

EPSILON =0,01 — Iteraciones 500

FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]

N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO
1	0,13081	1	0,106132	1	0,0956531	1	0,08989	1	0,091202
2	0,127129	2	0,103002	2	0,0950139	2	0,0901589	2	0,0877891
3	0,131756	3	0,0984781	3	0,093817	3	0,0879509	3	0,0891578
4	0,130609	4	0,102	4	0,0941	4	0,09375	4	0,0943952
5	0,126248	5	0,104721	5	0,0953691	5	0,0900009	5	0,0915029
6		6		6		6		6	
7	0,1293104	7	0,10286662	7	0,09479062	7	0,09035014	7	0,0908094

EPSILON =0,0001 — Iteraciones 10

FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]

N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO
1	0,172582	1	0,141436	1	0,110801	1	0,108763	1	0,0597908
2	0,173242	2	0,140969	2	0,115478	2	0,111485	2	0,0599859
3	0,173788	3	0,145581	3	0,110439	3	0,111422	3	0,059829
4	0,172291	4	0,140374	4	0,109918	4	0,110714	4	0,059907
5	0,178345	5	0,138547	5	0,115099	5	0,111125	5	0,060143
6		6		6		6		6	
7	0,1740496	7	0,1413814	7	0,112347	7	0,1107018	7	0,05993114

EPSILON =0,0001 — Iteraciones 50

FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]

N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO
1	0,177882	1	0,14094	1	0,111061	1	0,111235	1	0,0601699
2	0,172239	2	0,141007	2	0,111703	2	0,111407	2	0,0602069
3	0,17601	3	0,140326	3	0,115195	3	0,112026	3	0,0579078
4	0,169632	4	0,142289	4	0,110422	4	0,110385	4	0,059839
5	0,172037	5	0,140749	5	0,113009	5	0,110894	5	0,059968
6		6		6		6		6	
7	0,17356	7	0,1410622	7	0,112278	7	0,1111894	7	0,05961832

EPSILON =0,0001 — Iteraciones 100

FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]

N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO
1	0,172669	1	0,126656	1	0,0835721	1	0,0832322	1	0,059752
2	0,171783	2	0,123223	2	0,082418	2	0,0855861	2	0,0600369
3	0,174101	3	0,122424	3	0,081897	3	0,0872531	3	0,0610669
4	0,170113	4	0,179557	4	0,08494	4	0,0833061	4	0,059983
5	0,37614	5	0,12159	5	0,0816591	5	0,0875881	5	0,059582
6		6		6		6		6	
7	0,2129612	7	0,13469	7	0,08289724	7	0,08539312	7	0,06008416

EPSILON =0,0001 — Iteraciones 500

FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]

N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO
1	0,149264	1	0,11784	1	0,095819	1	0,089926	1	0,0920601
2	0,138761	2	0,116864	2	0,092845	2	0,09549	2	0,0925119
3	0,350412	3	0,116194	3	0,0922451	3	0,0963378	3	0,0888309
4	0,371252	4	0,120371	4	0,095304	4	0,09444	4	0,089294
5	0,134331	5	0,1173	5	0,0965312	5	0,0941389	5	0,089184
6		6		6		6		6	
7	0,228804	7	0,1177138	7	0,09454886	7	0,09406654	7	0,09037618

EPSILON =0,000001										Iteraciones 10	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,18551		1	0,156564		1	0,131163		1	0,128408		1	0,069406	
2	0,194533		2	0,154285		2	0,127328		2	0,127285		2	0,0655942	
3	0,193518		3	0,156079		3	0,133289		3	0,126404		3	0,068229	
4	0,197077		4	0,1562		4	0,131558		4	0,13133		4	0,0651488	
5	0,194378		5	0,154231		5	0,1248		5	0,126095		5	0,0650141	
6			6			6			6			6		
7	0,1930032		7	0,1554718		7	0,1296276		7	0,1279044		7	0,06667842	
EPSILON =0,000001										Iteraciones 50	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,187541		1	0,123728		1	0,104228		1	0,0874782		1	0,092541	
2	0,18625		2	0,122556		2	0,103957		2	0,087678		2	0,0907261	
3	0,19246		3	0,127459		3	0,103964		3	0,0909519		3	0,0914001	
4	0,197277		4	0,126897		4	0,10274		4	0,092597		4	0,091135	
5	0,194317		5	0,127212		5	0,107915		5	0,0880201		5	0,0945301	
6			6			6			6			6		
7	0,191569		7	0,1255704		7	0,1045608		7	0,08934504		7	0,09206646	
EPSILON =0,000001										Iteraciones 100	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°			N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,399387		1	0,134503		1	0,09881		1	0,098366		1	0,068397	
2	0,408927		2	0,138072		2	0,101427		2	0,102488		2	0,0684681	
3	0,445163		3	0,136725		3	0,101057		3	0,0997		3	0,067836	
4	0,190419		4	0,138198		4	0,104293		4	0,103414		4	0,0676742	
5	0,184777		5	0,143008		5	0,101429		5	0,09934		5	0,0647938	
6			6			6			6			6		
7	0,3257346		7	0,1381012		7	0,1014032		7	0,1006616		7	0,06743382	
EPSILON =0,000001										Iteraciones 500	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,155971		1	0,156006		1	0,110949		1	0,112517		1	0,103181	
2	0,356767		2	0,158191		2	0,1148		2	0,111977		2	0,106866	
3	0,157142		3	0,163275		3	0,112982		3	0,108109		3	0,101851	
4	0,159494		4	0,129354		4	0,115448		4	0,108587		4	0,106773	
5	0,159591		5	0,135492		5	0,113879		5	0,109073		5	0,104833	
6			6			6			6			6		
7	0,197793		7	0,1484636		7	0,1136116		7	0,1100526		7	0,1047008	
EPSILON =0,01										Iteraciones 10	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,070081		1	0,0702491		1	0,242924		1	0,202357		1	0,198877	
2	0,0682991		2	0,064369		2	0,204998		2	0,196157		2	0,203628	
3	0,0690708		3	0,0663049		3	0,202801		3	0,202267		3	0,200221	
4	0,071095		4	0,0652709		4	0,201758		4	0,198712		4	0,19849	
5	0,0691409		5	0,064893		5	0,203713		5	0,196172		5	0,197958	
6			6			6			6			6		
7	0,06953736		7	0,06621738		7	0,2112388		7	0,199133		7	0,1998348	
EPSILON =0,01										Iteraciones 50	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,067888		1	0,0645468		1	0,058557		1	0,0518842		1	0,0550032	
2	0,067306		2	0,0641811		2	0,058738		2	0,0511858		2	0,0549679	
3	0,067076		3	0,0646191		3	0,0585101		3	0,0500159		3	0,0550301	
4	0,0694079		4	0,0636811		4	0,0585868		4	0,0505259		4	0,0563321	
5	0,06633		5	0,0650809		5	0,0587261		5	0,0501862		5	0,0549648	
6			6			6			6			6		
7	0,06760158		7	0,0644218		7	0,0586236		7	0,0507596		7	0,05525962	
EPSILON =0,01										Iteraciones 100	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,069222		1	0,0589499		1	0,053437		1	0,0565889		1	0,0460298	
2	0,0633502		2	0,056905		2	0,0528221		2	0,0566671		2	0,0440609	
3	0,0649669		3	0,0581481		3	0,051332		3	0,057225		3	0,0445859	
4	0,0623329		4	0,0582569		4	0,0526509		4	0,0565321		4	0,0444081	
5	0,0638039		5	0,0588019		5	0,0529881		5	0,0575531		5	0,0446348	
6			6			6			6			6		
7	0,06473518		7	0,05821236		7	0,05264602		7	0,05691324		7	0,0447439	
EPSILON =0,01										Iteraciones 500	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,067498		1	0,069351		1	0,0671551		1	0,0667341		1	0,06654	
2	0,0691991		2	0,0662041		2	0,069253		2	0,0663431		2	0,0667851	
3	0,0670681		3	0,066479		3	0,067889		3	0,06654		3	0,068305	
4	0,0692611		4	0,067297		4	0,066267		4	0,0653288		4	0,0664229	
5	0,073185		5	0,065444		5	0,06635		5	0,067003		5	0,068119	
6			6			6			6			6		
7	0,06924226		7	0,06695502		7	0,06738282		7	0,0663898		7	0,0672344	

EPSILON =0,0001									Iteraciones 10	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]										
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,12128		1	0,115127	1	0,286172	1	0,280474	1	0,283201
2	0,124342		2	0,115778	2	0,287923	2	0,289382	2	0,283566
3	0,12201		3	0,116201	3	0,286651	3	0,281387	3	0,279609
4	0,120543		4	0,115002	4	0,327637	4	0,283216	4	0,289547
5	0,121624		5	0,119408	5	0,294651	5	0,281345	5	0,280823
6			6		6		6		6	
7	0,1219598		7	0,1163032	7	0,2966068	7	0,2831608	7	0,2833492
EPSILON =0,0001									Iteraciones 50	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]										
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,118336		1	0,116327	1	0,104836	1	0,100771	1	0,108734
2	0,118151		2	0,118712	2	0,0997388	2	0,104368	2	0,105912
3	0,117987		3	0,115213	3	0,101197	3	0,101106	3	0,105766
4	0,121927		4	0,115259	4	0,102544	4	0,101305	4	0,106204
5	0,118048		5	0,119922	5	0,103975	5	0,100535	5	0,107835
6			6		6		6		6	
7	0,1188898		7	0,1170866	7	0,10245816	7	0,101617	7	0,1068902
EPSILON =0,0001									Iteraciones 100	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]										
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,117427		1	0,111174	1	0,103908	1	0,108096	1	0,094146
2	0,115295		2	0,109895	2	0,106564	2	0,111139	2	0,0940969
3	0,118494		3	0,111954	3	0,103253	3	0,108602	3	0,095818
4	0,114595		4	0,110139	4	0,105392	4	0,110555	4	0,0957069
5	0,114997		5	0,112724	5	0,106087	5	0,107652	5	0,0975089
6			6		6		6		6	
7	0,1161616		7	0,1111772	7	0,1050408	7	0,1092088	7	0,09545534
EPSILON =0,0001									Iteraciones 500	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]										
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,0847821		1	0,0772102	1	0,0729189	1	0,0711851	1	0,0695469
2	0,080225		2	0,0756381	2	0,076242	2	0,070926	2	0,0701008
3	0,078326		3	0,0747688	3	0,0731671	3	0,0708349	3	0,070127
4	0,07863		4	0,0749459	4	0,0731061	4	0,0710199	4	0,0696561
5	0,0779729		5	0,0769911	5	0,0757389	5	0,070801	5	0,0700531
6			6		6		6		6	
7	0,0799872		7	0,07591082	7	0,0742346	7	0,07095338	7	0,06989678
EPSILON =0,000001									Iteraciones 10	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]										
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,173369		1	0,172759	1	0,295283	1	0,280599	1	0,280165
2	0,172849		2	0,168042	2	0,286013	2	0,289546	2	0,281229
3	0,178725		3	0,168858	3	0,291767	3	0,283063	3	0,281253
4	0,171823		4	0,168153	4	0,285911	4	0,28894	4	0,284514
5	0,176494		5	0,167577	5	0,288384	5	0,28581	5	0,289706
6			6		6		6		6	
7	0,174652		7	0,1690778	7	0,2894716	7	0,2855916	7	0,2833734
EPSILON =0,000001									Iteraciones 50	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]										
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6
N°			N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,174646		1	0,167931	1	0,15329	1	0,157397	1	0,163338
2	0,174538		2	0,16848	2	0,153751	2	0,15379	2	0,158222
3	0,176228		3	0,167316	3	0,15234	3	0,156498	3	0,158497
4	0,175568		4	0,167964	4	0,153332	4	0,153918	4	0,157986
5	0,170078		5	0,172924	5	0,152635	5	0,153254	5	0,158087
6			6		6		6		6	
7	0,1742116		7	0,168923	7	0,1530696	7	0,1549714	7	0,159226
EPSILON =0,000001									Iteraciones 100	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]										
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,166531		1	0,161184	1	0,160083	1	0,159986	1	0,151203
2	0,165857		2	0,161276	2	0,155437	2	0,161582	2	0,146444
3	0,166805		3	0,161949	3	0,154542	3	0,159384	3	0,150714
4	0,165451		4	0,166447	4	0,155505	4	0,163511	4	0,151858
5	0,165335		5	0,161458	5	0,156591	5	0,163841	5	0,151565
6			6		6		6		6	
7	0,1659958		7	0,1624628	7	0,1564316	7	0,1616608	7	0,1503568
EPSILON =0,000001									Iteraciones 500	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]										
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,132357		1	0,126179	1	0,127019	1	0,121845	1	0,113916
2	0,130196		2	0,12716	2	0,125353	2	0,122888	2	0,113143
3	0,132628		3	0,129001	3	0,124768	3	0,123355	3	0,113685
4	0,130313		4	0,125575	4	0,125049	4	0,12516	4	0,114113
5	0,132011		5	0,127041	5	0,126897	5	0,121542	5	0,114416
6			6		6		6		6	
7	0,131501		7	0,1269912	7	0,1258172	7	0,122958	7	0,1138546

EPSILON =0,01										Iteraciones 10	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,014272		1	0,0128319	1	0,01284	1	0,0129809	1	0,00731397	
2	0,0140529		2	0,012712	2	0,0126181	2	0,0128069	2	0,00745606	
3	0,0136468		3	0,0137091	3	0,0126259	3	0,0126998	3	0,00739813	
4	0,0133958		4	0,0130649	4	0,012264	4	0,0127592	4	0,00735497	
5	0,0144279		5	0,013453	5	0,0127301	5	0,012455	5	0,00739789	
6			6		6		6		6		
7	0,01395908		7	0,01315418	7	0,01261562	7	0,01274036	7	0,0073842	
EPSILON =0,01										Iteraciones 50	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0124202		1	0,011853	1	0,0109739	1	0,0104599	1	0,0104611	
2	0,012563		2	0,0122168	2	0,0105519	2	0,0103719	2	0,010438	
3	0,012635		3	0,0121629	3	0,01088	3	0,010788	3	0,0100849	
4	0,0123711		4	0,012044	4	0,0109761	4	0,0105839	4	0,0104551	
5	0,0125461		5	0,012162	5	0,0112009	5	0,0102608	5	0,010287	
6			6		6		6		6		
7	0,01250708		7	0,01208774	7	0,01091656	7	0,0104929	7	0,01034522	
EPSILON =0,01										Iteraciones 100	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0105598		1	0,00956988	1	0,00982404	1	0,00951695	1	0,00930691	
2	0,010443		2	0,009552	2	0,009516	2	0,00956511	2	0,0106952	
3	0,00937295		3	0,00953507	3	0,00966501	3	0,00962496	3	0,00879002	
4	0,00981998		4	0,00975299	4	0,00961709	4	0,00938511	4	0,011127	
5	0,010427		5	0,00969481	5	0,00966191	5	0,00952196	5	0,00979304	
6			6		6		6		6		
7	0,010124546		7	0,00962095	7	0,00965681	7	0,00952282	7	0,00994243	
EPSILON =0,01										Iteraciones 500	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0312631		1	0,0311141	1	0,0327449	1	0,0332339	1	0,0331109	
2	0,0311949		2	0,0318551	2	0,0312061	2	0,031297	2	0,0327129	
3	0,032747		3	0,031867	3	0,03108	3	0,032939	3	0,0321929	
4	0,031847		4	0,033021	4	0,032052	4	0,033046	4	0,0339692	
5	0,0313339		5	0,031852	5	0,031791	5	0,0338809	5	0,0323958	
6			6		6		6		6		
7	0,03167718		7	0,03194184	7	0,0317748	7	0,03287936	7	0,03287634	
EPSILON =0,0001										Iteraciones 10	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0217638		1	0,0214992	1	0,020612	1	0,0204079	1	0,010963	
2	0,0226099		2	0,021471	2	0,020602	2	0,0206649	2	0,010551	
3	0,0218232		3	0,0217299	3	0,019964	3	0,01967	3	0,0109639	
4	0,0214679		4	0,0217462	4	0,020293	4	0,020335	4	0,010555	
5	0,0223958		5	0,0215161	5	0,0209651	5	0,0210621	5	0,0110719	
6			6		6		6		6		
7	0,02201212		7	0,02159248	7	0,02048722	7	0,02042798	7	0,01082096	
EPSILON =0,0001										Iteraciones 50	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0204608		1	0,0202901	1	0,0191309	1	0,018317	1	0,0183592	
2	0,019872		2	0,0198419	2	0,0192001	2	0,0182199	2	0,0181041	
3	0,0201991		3	0,0200861	3	0,018831	3	0,0191462	3	0,0191662	
4	0,0201001		4	0,0198679	4	0,0187972	4	0,018353	4	0,0183899	
5	0,020869		5	0,0199881	5	0,018703	5	0,0184081	5	0,018028	
6			6		6		6		6		
7	0,0203002		7	0,02001482	7	0,01893244	7	0,01848884	7	0,01840948	
EPSILON =0,0001										Iteraciones 100	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0184519		1	0,020468	1	0,0168531	1	0,016768	1	0,012692	
2	0,0183249		2	0,0171411	2	0,017102	2	0,0171971	2	0,012342	
3	0,0182939		3	0,0163031	3	0,016958	3	0,0171568	3	0,012753	
4	0,0183499		4	0,0172031	4	0,017148	4	0,0173202	4	0,0125771	
5	0,018451		5	0,0172639	5	0,0165691	5	0,0174022	5	0,0124581	
6			6		6		6		6		
7	0,01837432		7	0,01767584	7	0,01692604	7	0,01716886	7	0,01256444	
EPSILON =0,0001										Iteraciones 500	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,033031		1	0,0319681	1	0,031939	1	0,0312691	1	0,031703	
2	0,0322652		2	0,0317461	2	0,03107	2	0,0328829	2	0,0327909	
3	0,0317819		3	0,032408	3	0,033524	3	0,033308	3	0,0344081	
4	0,0327859		4	0,0320871	4	0,032032	4	0,0327289	4	0,032469	
5	0,0310471		5	0,031939	5	0,0313821	5	0,0323899	5	0,0327158	
6			6		6		6		6		
7	0,03218222		7	0,03202966	7	0,03198942	7	0,03251576	7	0,03281736	

EPSILON =0,000001										Iteraciones 10	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,0307019		1	0,02898		1	0,0288861		1	0,0287499		1	0,015486	
2	0,030551		2	0,0295758		2	0,029387		2	0,028908		2	0,014811	
3	0,0308251		3	0,0292511		3	0,0293639		3	0,029496		3	0,014349	
4	0,0301421		4	0,0288482		4	0,028152		4	0,0286419		4	0,0151129	
5	0,0294602		5	0,0294509		5	0,0281751		5	0,028616		5	0,0146439	
6			6			6			6			6		
7	0,03033606		7	0,0292212		7	0,02879282		7	0,02888236		7	0,01488056	
EPSILON =0,000001										Iteraciones 50	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,0274661		1	0,0287008		1	0,026716		1	0,026948		1	0,027391	
2	0,02753		2	0,0279291		2	0,027288		2	0,0265119		2	0,0268791	
3	0,028867		3	0,0297511		3	0,0261409		3	0,0261621		3	0,026572	
4	0,028029		4	0,0298309		4	0,027169		4	0,026715		4	0,0265019	
5	0,0283589		5	0,029264		5	0,0277889		5	0,0265131		5	0,0260251	
6			6			6			6			6		
7	0,0280502		7	0,02909518		7	0,02702056		7	0,02657002		7	0,02667382	
EPSILON =0,000001										Iteraciones 100	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,025991		1	0,0245779		1	0,0256121		1	0,0254228		1	0,0161319	
2	0,02578		2	0,0248199		2	0,0245931		2	0,025245		2	0,0165431	
3	0,026906		3	0,024967		3	0,0247989		3	0,0246651		3	0,0162489	
4	0,0264082		4	0,0250912		4	0,0248129		4	0,0248141		4	0,0161791	
5	0,026228		5	0,02544		5	0,0252051		5	0,0250521		5	0,0162649	
6			6			6			6			6		
7	0,02626264		7	0,0249792		7	0,02500442		7	0,02503982		7	0,01627358	
EPSILON =0,000001										Iteraciones 500	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,031239		1	0,0320401		1	0,0325952		1	0,0319619		1	0,031357	
2	0,0318391		2	0,0318339		2	0,0329781		2	0,032069		2	0,0336099	
3	0,0317559		3	0,0316958		3	0,031327		3	0,03315		3	0,0318301	
4	0,0312691		4	0,0321379		4	0,0321929		4	0,0329671		4	0,0328209	
5	0,0327539		5	0,0324981		5	0,0314989		5	0,0326581		5	0,0323679	
6			6			6			6			6		
7	0,0317714		7	0,03204116		7	0,03211842		7	0,03256122		7	0,03239716	
EPSILON =0,01										Iteraciones 10	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,0462379		1	0,0254359		1	0,0221529		1	0,0228732		1	0,0208519	
2	0,048501		2	0,0254371		2	0,022686		2	0,0227969		2	0,0203719	
3	0,046144		3	0,0251608		3	0,0230289		3	0,0229321		3	0,0204492	
4	0,0465689		4	0,0249231		4	0,022367		4	0,023087		4	0,0200109	
5	0,0461478		5	0,025259		5	0,0222549		5	0,0229371		5	0,0199442	
6			6			6			6			6		
7	0,04671992		7	0,02524318		7	0,02249794		7	0,02292526		7	0,02032562	
EPSILON =0,01										Iteraciones 50	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,0466919		1	0,0402758		1	0,0409489		1	0,03635		1	0,0192149	
2	0,046932		2	0,0398951		2	0,0373509		2	0,0361109		2	0,0192559	
3	0,047729		3	0,039851		3	0,037864		3	0,0361629		3	0,0197382	
4	0,0469668		4	0,038774		4	0,037513		4	0,0358632		4	0,0193441	
5	0,046885		5	0,039304		5	0,038542		5	0,0366442		5	0,0192931	
6			6			6			6			6		
7	0,04704094		7	0,03961998		7	0,03844376		7	0,03622624		7	0,01936924	
EPSILON =0,01										Iteraciones 100	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,0447762		1	0,041101		1	0,0412481		1	0,0374269		1	0,026581	
2	0,046113		2	0,0401452		2	0,039387		2	0,0378361		2	0,0253	
3	0,0459671		3	0,0411808		3	0,0406229		3	0,038743		3	0,025636	
4	0,0461161		4	0,0404029		4	0,040801		4	0,0383489		4	0,0268519	
5	0,046628		5	0,040354		5	0,040632		5	0,037071		5	0,0270691	
6			6			6			6			6		
7	0,04592008		7	0,04063678		7	0,0405382		7	0,03788518		7	0,0262876	
EPSILON =0,01										Iteraciones 500	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,035239		1	0,0352681		1	0,0332141		1	0,037524		1	0,0349422	
2	0,0341122		2	0,0339651		2	0,0348132		2	0,0351319		2	0,033391	
3	0,0359461		3	0,034044		3	0,035089		3	0,035502		3	0,0337949	
4	0,0351989		4	0,0347171		4	0,0333891		4	0,03459		4	0,0339599	
5	0,0359142		5	0,03405		5	0,034749		5	0,0349951		5	0,0325849	
6			6			6			6			6		
7	0,03528208		7	0,03440886		7	0,03425088		7	0,0355486		7	0,03373458	

EPSILON =0,0001										Iteraciones 10	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0867038		1	0,0454528	1	0,0444379	1	0,0419769	1	0,0382509	
2	0,0835462		2	0,044982	2	0,043705	2	0,0423229	2	0,03916	
3	0,0888791		3	0,044297	3	0,040905	3	0,0420611	3	0,0393791	
4	0,085181		4	0,044363	4	0,043093	4	0,0431812	4	0,0387571	
5	0,0848031		5	0,046073	5	0,041944	5	0,0411918	5	0,0388789	
6			6		6		6		6		
7	0,08582264		7	0,04503356	7	0,04281698	7	0,04214678	7	0,0388852	
EPSILON =0,0001										Iteraciones 50	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0865951		1	0,0766392	1	0,075947	1	0,0739241	1	0,03813	
2	0,085742		2	0,077379	2	0,075336	2	0,074245	2	0,0374382	
3	0,087007		3	0,0775461	3	0,075721	3	0,074147	3	0,0374169	
4	0,0866151		4	0,0780489	4	0,076113	4	0,0760839	4	0,0380619	
5	0,0859771		5	0,0792229	5	0,0758779	5	0,074039	5	0,037396	
6			6		6		6		6		
7	0,08638726		7	0,07776722	7	0,07579898	7	0,0744878	7	0,0376886	
EPSILON =0,0001										Iteraciones 100	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0854321		1	0,0773952	1	0,07832	1	0,0767438	1	0,0476139	
2	0,0854449		2	0,0810292	2	0,0770011	2	0,0785599	2	0,0465031	
3	0,0852869		3	0,079304	3	0,080651	3	0,0758111	3	0,0478868	
4	0,085377		4	0,0784519	4	0,0773561	4	0,0756831	4	0,0526881	
5	0,0843771		5	0,080555	5	0,085813	5	0,0758719	5	0,046982	
6			6		6		6		6		
7	0,0851836		7	0,07934706	7	0,07982824	7	0,07653396	7	0,04833478	
EPSILON =0,0001										Iteraciones 500	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,067785		1	0,0617411	1	0,0627661	1	0,0484838	1	0,048157	
2	0,0687571		2	0,061172	2	0,0633299	2	0,047945	2	0,0466621	
3	0,067513		3	0,0616429	3	0,0627398	3	0,04811	3	0,0479059	
4	0,068764		4	0,0609391	4	0,0644069	4	0,049469	4	0,048331	
5	0,0675461		5	0,0618019	5	0,061697	5	0,0484931	5	0,0479581	
6			6		6		6		6		
7	0,06807304		7	0,0614594	7	0,06298794	7	0,04850018	7	0,04780282	
EPSILON =0,000001										Iteraciones 10	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,122828		1	0,0647581	1	0,062161	1	0,0649679	1	0,056102	
2	0,125195		2	0,06515	2	0,0611329	2	0,0613291	2	0,0575511	
3	0,122491		3	0,0656519	3	0,0613809	3	0,0636759	3	0,056617	
4	0,121999		4	0,063499	4	0,0627131	4	0,0635049	4	0,055932	
5	0,124824		5	0,063916	5	0,062371	5	0,061413	5	0,055161	
6			6		6		6		6		
7	0,1234674		7	0,064595	7	0,06195178	7	0,06297816	7	0,05627262	
EPSILON =0,000001										Iteraciones 50	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,125477		1	0,116267	1	0,113463	1	0,115022	1	0,058007	
2	0,134691		2	0,114914	2	0,116266	2	0,115816	2	0,0557871	
3	0,122417		3	0,118726	3	0,112981	3	0,1123	3	0,0562222	
4	0,124312		4	0,11859	4	0,113284	4	0,111826	4	0,0580182	
5	0,123484		5	0,11471	5	0,113046	5	0,115236	5	0,0560479	
6			6		6		6		6		
7	0,1260762		7	0,1166414	7	0,113808	7	0,11404	7	0,05681648	
EPSILON =0,000001										Iteraciones 100	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,123091		1	0,116838	1	0,120062	1	0,115069	1	0,067394	
2	0,121167		2	0,120001	2	0,118165	2	0,117161	2	0,068141	
3	0,122689		3	0,119862	3	0,119712	3	0,11411	3	0,0694711	
4	0,122738		4	0,119282	4	0,11684	4	0,116519	4	0,0681181	
5	0,123547		5	0,117233	5	0,11657	5	0,113753	5	0,068413	
6			6		6		6		6		
7	0,1226464		7	0,1186432	7	0,1182698	7	0,1153224	7	0,06830744	
EPSILON =0,000001										Iteraciones 500	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,108034		1	0,100859	1	0,099637	1	0,069273	1	0,0680289	
2	0,107171		2	0,100953	2	0,100296	2	0,0695279	2	0,0698521	
3	0,106144		3	0,099786	3	0,100187	3	0,0691979	3	0,0692799	
4	0,104837		4	0,0996001	4	0,101859	4	0,0697901	4	0,069679	
5	0,107391		5	0,102379	5	0,100135	5	0,068738	5	0,0689762	
6			6		6		6		6		
7	0,1067154		7	0,10071542	7	0,1004228	7	0,06930538	7	0,06916322	

EPSILON =0,01										Iteraciones 10	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0721819		1	0,0394421	1	0,034236	1	0,0337131	1	0,032541	
2	0,072552		2	0,0397301	2	0,0341461	2	0,0341899	2	0,0312939	
3	0,0701871		3	0,0389071	3	0,0353479	3	0,0358331	3	0,033474	
4	0,070262		4	0,040375	4	0,0344	4	0,034086	4	0,0325458	
5	0,0689821		5	0,0398161	5	0,0333252	5	0,0341339	5	0,0325079	
6			6		6		6		6		
7	0,07083302		7	0,03965408	7	0,03429104	7	0,0343912	7	0,03247252	
EPSILON =0,01										Iteraciones 50	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0724521		1	0,0390601	1	0,037674	1	0,0335929	1	0,0326269	
2	0,0700991		2	0,038517	2	0,0371342	2	0,0322311	2	0,0333209	
3	0,0682759		3	0,0401449	3	0,0376191	3	0,03385	3	0,034519	
4	0,0681942		4	0,0393569	4	0,0387909	4	0,034513	4	0,0332351	
5	0,0705099		5	0,039808	5	0,038419	5	0,0344892	5	0,0333891	
6			6		6		6		6		
7	0,06990624		7	0,03937738	7	0,03792744	7	0,03373524	7	0,0334182	
EPSILON =0,01										Iteraciones 100	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,068871		1	0,0418432	1	0,0391481	1	0,0362952	1	0,03263	
2	0,068131		2	0,0409791	2	0,038034	2	0,036639	2	0,0324559	
3	0,0686522		3	0,0414739	3	0,0379591	3	0,035624	3	0,0331218	
4	0,0680501		4	0,040915	4	0,0370591	4	0,0358548	4	0,0334439	
5	0,0676901		5	0,0408831	5	0,0378971	5	0,035826	5	0,0307121	
6			6		6		6		6		
7	0,06827888		7	0,04121886	7	0,03801948	7	0,0360478	7	0,03247274	
EPSILON =0,01										Iteraciones 500	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,051512		1	0,0495801	1	0,0512452	1	0,0497329	1	cae	
2	0,049849		2	0,048485	2	0,0512452	2	0,0491328	2	cae	
3	0,0494399		3	0,0495889	3	0,0516169	3	0,052696	3	cae	
4	0,0486679		4	0,049547	4	0,0504	4	0,049521	4	cae	
5	0,048836		5	0,0491281	5	0,0496428	5	0,048907	5	cae	
6			6		6		6		6		
7	0,04966096		7	0,04926582	7	0,05083002	7	0,04999794	7	#DIV/0!	
EPSILON =0,0001										Iteraciones 10	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0787752		1	0,0463481	1	0,036696	1	0,0342751	1	0,0361929	
2	0,0792921		2	0,0459199	2	0,037827	2	0,0359499	2	0,034425	
3	0,080864		3	0,048156	3	0,037641	3	0,037009	3	0,0358031	
4	0,0787482		4	0,0469139	4	0,0351908	4	0,035897	4	0,0345929	
5	0,080308		5	0,0464072	5	0,0367291	5	0,0351579	5	0,0359509	
6			6		6		6		6		
7	0,0795975		7	0,04674902	7	0,03681678	7	0,03565778	7	0,03539296	
EPSILON =0,0001										Iteraciones 50	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0791039		1	0,043278	1	0,0447061	1	0,0412359	1	0,0404458	
2	0,078409		2	0,043422	2	0,045043	2	0,039701	2	0,0417361	
3	0,07827		3	0,046315	3	0,0444019	3	0,0413909	3	0,0402	
4	0,0772891		4	0,0432019	4	0,0445631	4	0,0404809	4	0,0405149	
5	0,0784459		5	0,043509	5	0,0430429	5	0,0404949	5	0,0405371	
6			6		6		6		6		
7	0,07830358		7	0,04394518	7	0,0443514	7	0,04066072	7	0,04068678	
EPSILON =0,0001										Iteraciones 100	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0770838		1	0,0453548	1	0,0398159	1	0,0388682	1	0,033232	
2	0,0745089		2	0,0456181	2	0,0399821	2	0,0387161	2	0,0330231	
3	0,0765591		3	0,0464959	3	0,0398149	3	0,0385852	3	0,0334859	
4	0,0777309		4	0,04442	4	0,037941	4	0,038666	4	0,0342731	
5	0,077944		5	0,043618	5	0,039705	5	0,0388651	5	0,0331991	
6			6		6		6		6		
7	0,07676534		7	0,04510136	7	0,03945178	7	0,03874012	7	0,03344264	
EPSILON =0,0001										Iteraciones 500	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0484891		1	0,0497589	1	0,0496171	1	0,0490181	1	cae	
2	0,0495031		2	0,051821	2	0,05003	2	0,0500882	2	cae	
3	0,0495369		3	0,0492711	3	0,0498431	3	0,0498261	3	cae	
4	0,0496621		4	0,0485821	4	0,0496671	4	0,050693	4	cae	
5	0,052912		5	0,0486541	5	0,0497248	5	0,049397	5	cae	
6			6		6		6		6		
7	0,05002064		7	0,04961744	7	0,04977642	7	0,04980448	7	#DIV/0!	

EPSILON =0,000001										Iteraciones 10	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,087501		1	0,052727	1	0,0388839	1	0,0371151	1	0,0372641	
2	0,0873921		2	0,0535669	2	0,0388222	2	0,036999	2	0,038398	
3	0,087996		3	0,0526049	3	0,0387011	3	0,0377331	3	0,0369852	
4	0,088017		4	0,053436	4	0,0381551	4	0,037534	4	0,0372319	
5	0,0870948		5	0,0578589	5	0,0399721	5	0,0377822	5	0,037045	
6			6		6		6		6		
7	0,08760018		7	0,05403874	7	0,03890688	7	0,03743268	7	0,03738484	
EPSILON =0,000001										Iteraciones 50	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,087389		1	0,048948	1	0,0457251	1	0,0391438	1	0,039907	
2	0,0873749		2	0,0479062	2	0,0446908	2	0,040576	2	0,041507	
3	0,087033		3	0,0480709	3	0,0446811	3	0,039366	3	0,0404601	
4	0,087728		4	0,047941	4	0,0460031	4	0,0400741	4	0,04176	
5	0,0876		5	0,0484869	5	0,0450342	5	0,040498	5	0,0405838	
6			6		6		6		6		
7	0,08742498		7	0,0482706	7	0,04522686	7	0,03993158	7	0,04084358	
EPSILON =0,000001										Iteraciones 100	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0871911		1	0,0507669	1	0,04354	1	0,0421031	1	0,0335128	
2	0,0855532		2	0,0511289	2	0,043431	2	0,0429158	2	0,0331039	
3	0,0886109		3	0,0500879	3	0,0426121	3	0,0434451	3	0,0343211	
4	0,0890179		4	0,0512681	4	0,0435078	4	0,0428011	4	0,033808	
5	0,088609		5	0,0512209	5	0,0413952	5	0,0430081	5	0,0338578	
6			6		6		6		6		
7	0,08779642		7	0,05089454	7	0,04289722	7	0,04285464	7	0,03372072	
EPSILON =0,000001										Iteraciones 500	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,052763		1	0,051111	1	0,050935	1	0,0496631	1	cae	
2	0,0532432		2	0,0512321	2	0,052428	2	0,0500159	2	cae	
3	0,053174		3	0,051621	3	0,0512941	3	0,0497971	3	cae	
4	0,0532351		4	0,0550778	4	0,053149	4	0,0509589	4	cae	
5	0,0518119		5	0,0515988	5	0,0500212	5	0,0499382	5	cae	
6			6		6		6		6		
7	0,05284544		7	0,05212814	7	0,05156546	7	0,05007464	7	#DIV/0!	
EPSILON =0,01										Iteraciones 10	
FUNCION : TRECCANI [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,056077		1	0,0308421	1	0,027	1	0,027128	1	0,024406	
2	0,056607		2	0,0294471	2	0,0272651	2	0,0268691	2	0,0249181	
3	0,05357		3	0,03001	3	0,0286291	3	0,0278029	3	0,024199	
4	0,0519981		4	0,0291719	4	0,0273211	4	0,0270231	4	0,0242059	
5	0,053797		5	0,0285249	5	0,026994	5	0,026943	5	0,025924	
6			6		6		6		6		
7	0,05440982		7	0,0295992	7	0,02744186	7	0,02715322	7	0,0247306	
EPSILON =0,01										Iteraciones 50	
FUNCION : TRECCANI [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,052875		1	0,0324042	1	0,0371709	1	0,0268421	1	0,0251629	
2	0,0521891		2	0,0305712	2	0,0365279	2	0,0262098	2	0,0252481	
3	0,052202		3	0,032079	3	0,0359211	3	0,025912	3	0,0250771	
4	0,0537381		4	0,030755	4	0,0356231	4	0,0256839	4	0,0248191	
5	0,0559199		5	0,0308042	5	0,0362759	5	0,0259418	5	0,0261829	
6			6		6		6		6		
7	0,05338482		7	0,03132272	7	0,03630378	7	0,02611792	7	0,02529802	
EPSILON =0,01										Iteraciones 100	
FUNCION : TRECCANI [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0532129		1	0,033987	1	0,0304692	1	0,0265129	1	0,0252512	
2	0,0521271		2	0,0333869	2	0,0308409	2	0,02683	2	0,023773	
3	0,054302		3	0,033742	3	0,0306289	3	0,0260179	3	0,0263779	
4	0,0521541		4	0,0318129	4	0,0302372	4	0,0258172	4	0,0236311	
5	0,053		5	0,0342629	5	0,0302219	5	0,0264709	5	0,0243649	
6			6		6		6		6		
7	0,05295922		7	0,03343834	7	0,03047962	7	0,02632978	7	0,02467962	
EPSILON =0,01										Iteraciones 500	
FUNCION : TRECCANI [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0422239		1	0,042418	1	0,040045	1	0,043601	1	0,0428929	
2	0,0433609		2	0,0408628	2	0,0400269	2	0,0407939	2	0,0418742	
3	0,043694		3	0,0412149	3	0,0411971	3	0,0403581	3	0,0425701	
4	0,043505		4	0,039927	4	0,040581	4	0,0409899	4	0,039784	
5	0,0432532		5	0,0393	5	0,0386341	5	0,043503	5	0,0382621	
6			6		6		6		6		
7	0,0432074		7	0,04074454	7	0,04009682	7	0,04184918	7	0,04107666	

EPSILON =0,0001 — Iteraciones 10

FUNCION : TRECCANI [-5,5][-5,5]

	N PRO, 2		N PRO, 3		N PRO, 4		N PRO, 5		N PRO, 6
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,105094	1	0,0583332	1	0,05318	1	0,050611	1	0,050066
2	0,106126	2	0,0586109	2	0,054306	2	0,0501118	2	0,0501099
3	0,108534	3	0,0563519	3	0,0525379	3	0,052283	3	0,048063
4	0,105278	4	0,0545099	4	0,0524962	4	0,050709	4	0,050163
5	0,105632	5	0,055598	5	0,053432	5	0,049051	5	0,0493908
6		6		6		6		6	
7	0,1061328	7	0,05668078	7	0,05319042	7	0,05055316	7	0,04955854

EPSILON =0,0001 — Iteraciones 50

FUNCION : TRECCANI [-5,5][-5,5]

	N PRO, 2		N PRO, 3		N PRO, 4		N PRO, 5		N PRO, 6
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,105596	1	0,0550449	1	0,059377	1	0,0469329	1	0,0484679
2	0,100517	2	0,0560911	2	0,0603318	2	0,0487621	2	0,046551
3	0,104159	3	0,0561049	3	0,060581	3	0,046324	3	0,0462079
4	0,101931	4	0,054997	4	0,0603189	4	0,0484619	4	0,0459499
5	0,10292	5	0,0587239	5	0,0582979	5	0,0483301	5	0,0470319
6		6		6		6		6	
7	0,1030246	7	0,05619236	7	0,05978132	7	0,0477622	7	0,04684172

EPSILON =0,0001 — Iteraciones 100

FUNCION : TRECCANI [-5,5][-5,5]

	N PRO, 2		N PRO, 3		N PRO, 4		N PRO, 5		N PRO, 6
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,104399	1	0,0618379	1	0,0560091	1	0,0486281	1	0,047034
2	0,106993	2	0,06217	2	0,0547602	2	0,050144	2	0,0472119
3	0,101488	3	0,0601559	3	0,0532999	3	0,047893	3	0,0473509
4	0,104869	4	0,0609879	4	0,053911	4	0,0500641	4	0,0461309
5	0,104324	5	0,0613739	5	0,054656	5	0,0493069	5	0,0480292
6		6		6		6		6	
7	0,1044146	7	0,06130512	7	0,05452724	7	0,04920722	7	0,04715138

EPSILON =0,0001 — Iteraciones 500

FUNCION : TRECCANI [-5,5][-5,5]

	N PRO, 2		N PRO, 3		N PRO, 4		N PRO, 5		N PRO, 6
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,085139	1	0,063525	1	0,0551898	1	0,0557849	1	0,0575662
2	0,0843959	2	0,0647111	2	0,0587559	2	0,0556841	2	0,0562661
3	0,0832641	3	0,0630999	3	0,057687	3	0,056726	3	0,0548489
4	0,0841339	4	0,064965	4	0,0566139	4	0,0543499	4	0,057323
5	0,0833061	5	0,066704	5	0,0580199	5	0,057327	5	0,0556381
6		6		6		6		6	
7	0,0840478	7	0,064601	7	0,0572533	7	0,05597438	7	0,05632846

EPSILON =0,000001 — Iteraciones 10

FUNCION : TRECCANI [-5,5][-5,5]

	N PRO, 2		N PRO, 3		N PRO, 4		N PRO, 5		N PRO, 6
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,163947	1	0,0823629	1	0,0809329	1	0,0745299	1	0,0744741
2	0,161742	2	0,0823889	2	0,0785048	2	0,0745339	2	0,073915
3	0,160502	3	0,0829542	3	0,0756481	3	0,0731599	3	0,0722649
4	0,153169	4	0,081043	4	0,079947	4	0,07215	4	0,0716789
5	0,155633	5	0,0809059	5	0,0779788	5	0,0723629	5	0,071496
6		6		6		6		6	
7	0,1589986	7	0,08193098	7	0,07860232	7	0,07334732	7	0,07276578

EPSILON =0,000001 — Iteraciones 50

FUNCION : TRECCANI [-5,5][-5,5]

	N PRO, 2		N PRO, 3		N PRO, 4		N PRO, 5		N PRO, 6
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,152139	1	0,0814219	1	0,081506	1	0,0682199	1	0,0689349
2	0,152913	2	0,0858562	2	0,0828528	2	0,0694549	2	0,067363
3	0,151283	3	0,0814021	3	0,084162	3	0,068383	3	0,0675271
4	0,156412	4	0,0814221	4	0,0877261	4	0,0720999	4	0,0684719
5	0,1602	5	0,0868149	5	0,0864131	5	0,0707321	5	0,0677922
6		6		6		6		6	
7	0,1545894	7	0,08338344	7	0,084532	7	0,06977796	7	0,06801782

EPSILON =0,000001 — Iteraciones 100

FUNCION : TRECCANI [-5,5][-5,5]

	N PRO, 2		N PRO, 3		N PRO, 4		N PRO, 5		N PRO, 6
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,150864	1	0,0874281	1	0,0791941	1	0,073081	1	0,0669918
2	0,155402	2	0,0851429	2	0,0807741	2	0,073472	2	0,068768
3	0,159512	3	0,088912	3	0,0835209	3	0,0726068	3	0,0676029
4	0,158602	4	0,086519	4	0,0801911	4	0,072989	4	0,0698409
5	0,15311	5	0,0864971	5	0,080097	5	0,0710611	5	0,0686731
6		6		6		6		6	
7	0,155498	7	0,08689982	7	0,08075544	7	0,07264198	7	0,06837534

EPSILON =0,000001 — Iteraciones 500

FUNCION : TRECCANI [-5,5][-5,5]

	N PRO, 2		N PRO, 3		N PRO, 4		N PRO, 5		N PRO, 6
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,132751	1	0,090482	1	0,081171	1	0,08005	1	0,077507
2	0,13825	2	0,088769	2	0,082381	2	0,0794902	2	0,0795619
3	0,13919	3	0,0872991	3	0,0828831	3	0,0766249	3	0,0822868
4	0,135954	4	0,088011	4	0,0854211	4	0,081044	4	0,0773258
5	0,137395	5	0,0910451	5	0,0829532	5	0,0769649	5	0,078794
6		6		6		6		6	
7	0,136708	7	0,08912124	7	0,08296188	7	0,0788348	7	0,0790951

EPSILON =0,01										Iteraciones 10	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]											
	N PRO.	2	N PRO.	3	N PRO.	4	N PRO.	5	N PRO.	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	5,89801		1	3,20400	1	2,45422	1	2,04303	1	1,92608	
2	5,89148		2	3,20744	2	2,44210	2	1,99846	2	1,92158	
3	5,84576		3	3,21320	3	2,43132	3	1,98902	3	1,89887	
4	5,83660		4	3,27512	4	2,50821	4	2,05685	4	1,89893	
5	5,85912		5	3,18108	5	2,51935	5	2,05415	5	1,88692	
6			6		6		6		6		
7	5,86619		7	3,21617	7	2,47104	7	2,02830	7	1,90648	

EPSILON =0,01										Iteraciones 50	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]											
	N PRO.	2	N PRO.	3	N PRO.	4	N PRO.	5	N PRO.	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	5,86541		1	3,93428	1	2,81346	1	3,06418	1	1,94106	
2	6,59642		2	3,96988	2	2,81279	2	3,08298	2	1,89858	
3	5,85001		3	4,15291	3	2,90607	3	3,07153	3	1,87686	
4	5,83276		4	3,95056	4	2,83178	4	3,14296	4	1,87873	
5	6,02191		5	3,93659	5	2,90064	5	3,06017	5	1,90206	
6			6		6		6		6		
7	6,03330		7	3,98884	7	2,85295	7	3,08436	7	1,89946	

EPSILON =0,01										Iteraciones 100	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]											
	N PRO,	2	N PRO,	3	N PRO.	4	N PRO.	5	N PRO.	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	5,87279		1	4,02391	1	2,82981	1	2,84939	1	2,01951	
2	6,00546		2	3,93049	2	2,89859	2	2,83354	2	2,07039	
3	6,22137		3	4,02641	3	2,81672	3	2,83301	3	2,03072	
4	6,08921		4	4,24372	4	2,81477	4	2,89583	4	2,03444	
5	5,84933		5	3,92819	5	2,89108	5	2,82725	5	2,07777	
6			6		6		6		6		
7	6,00763		7	4,03054	7	2,85019	7	2,84780	7	2,04657	

EPSILON =0,01										Iteraciones 500	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]											
	N PRO.	2	N PRO.	3	N PRO.	4	N PRO.	5	N PRO.	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	5,86230		1	3,69346	1	2,27423	1	1,89597	1	1,89191	
2	5,94490		2	3,61787	2	2,23151	2	1,87346	2	1,90946	
3	5,81627		3	3,59556	3	2,26827	3	1,87433	3	1,92948	
4	5,81938		4	3,59346	4	2,22114	4	1,93171	4	1,93914	
5	6,03196		5	3,59513	5	2,27072	5	1,88237	5	1,90084	
6			6		6		6		6		
7	5,89496		7	3,61910	7	2,25317	7	1,89157	7	1,91417	

EPSILON =0,0001										Iteraciones 10	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]											
	N PRO.	2	N PRO.	3	N PRO.	4	N PRO.	5	N PRO.	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	18,7026		1	11,7073	1	9,3599	1	7,5260	1	7,4912	
2	18,6422		2	11,4019	2	9,7052	2	7,4624	2	7,7371	
3	19,1047		3	11,4700	3	9,4235	3	7,4946	3	7,5220	
4	18,7436		4	11,7022	4	9,6749	4	7,4403	4	7,4684	
5	19,1322		5	11,3949	5	9,4610	5	7,5315	5	7,6747	
6			6		6		6		6		
7	18,86506		7	11,53526	7	9,52490	7	7,49096	7	7,57867	

EPSILON =0,0001										Iteraciones 50	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]											
	N PRO.	2	N PRO.	3	N PRO.	4	N PRO.	5	N PRO.	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	19,0938		1	11,9189	1	10,8062	1	11,1866	1	6,6128	
2	18,6068		2	11,9115	2	11,3867	2	11,5468	2	6,8115	
3	18,7058		3	11,9455	3	11,9256	3	11,2014	3	6,6299	
4	18,6990		4	11,8841	4	11,2745	4	11,2853	4	6,9545	
5	18,6498		5	11,8703	5	10,9763	5	11,3168	5	6,5784	
6			6		6		6		6		
7	18,75104		7	11,90606	7	11,27386	7	11,30738	7	6,71742	

EPSILON =0,0001										Iteraciones 100	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]											
	N PRO,	2	N PRO,	3	N PRO.	4	N PRO.	5	N PRO.	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	18,6798		1	13,4361	1	10,5728	1	10,6540	1	7,0556	
2	19,2915		2	13,6255	2	10,9304	2	10,8673	2	7,2598	
3	18,6537		3	13,5268	3	10,9119	3	10,6534	3	7,0287	
4	18,5670		4	13,4761	4	10,6187	4	10,9236	4	7,0801	
5	18,6502		5	13,5382	5	10,5708	5	10,7121	5	7,0840	
6			6		6		6		6		
7	18,76844		7	13,52054	7	10,72092	7	10,76208	7	7,10166	

EPSILON =0,0001										Iteraciones 500	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]											
	N PRO.	2	N PRO.	3	N PRO.	4	N PRO.	5	N PRO.	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	18,6855		1	10,6883	1	7,1082	1	5,7574	1	5,9516	
2	18,4428		2	10,3414	2	7,8077	2	5,9204	2	5,7747	
3	18,4544		3	10,6686	3	7,0284	3	5,7752	3	5,7665	
4	18,6782		4	12,0590	4	7,1016	4	5,7792	4	5,9322	
5	18,6534		5	10,6719	5	6,9580	5	5,8304	5	5,8277	
6			6		6		6		6		
7	18,58286		7	10,88584	7	7,20075	7	5,81252	7	5,85055	

EPSILON =0,000001									Iteraciones 10	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]										
	N PRO.	2		N PRO.	3		N PRO.	4	N PRO.	5		N PRO.	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO
1	19,3646		1	11,8210		1	9,6923		1	7,5184		1	7,6924
2	21,1355		2	11,4258		2	9,4300		2	7,4758		2	7,4486
3	19,3798		3	11,4636		3	9,4123		3	7,5338		3	7,4666
4	18,9602		4	11,5291		4	9,6566		4	7,5177		4	7,6972
5	18,8330		5	11,8210		5	9,3744		5	7,4946		5	7,7204
6			6			6			6			6	
7	19,53462		7	11,61210		7	9,51312		7	7,50804		7	7,60502
EPSILON =0,000001									Iteraciones 50	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]										
	N PRO.	2		N PRO.	3		N PRO.	4	N PRO.	5		N PRO.	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO
1	18,9197		1	12,2253		1	10,9589		1	11,6499		1	6,6988
2	18,7316		2	12,1507		2	10,9911		2	11,3700		2	6,8835
3	19,3491		3	12,4715		3	10,9016		3	11,2230		3	6,7702
4	18,8044		4	12,0790		4	10,9645		4	11,3819		4	6,8348
5	18,8183		5	12,3956		5	11,2894		5	11,2795		5	6,7484
6			6			6			6			6	
7	18,92462		7	12,26442		7	11,02110		7	11,38086		7	6,78714
EPSILON =0,000001									Iteraciones 100	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]										
	N PRO.	2		N PRO.	3		N PRO.	4	N PRO.	5		N PRO.	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO
1	18,9979		1	14,0857		1	10,7102		1	10,6495		1	7,0589
2	18,9320		2	14,0391		2	10,6584		2	10,7359		2	7,1697
3	18,9396		3	14,1517		3	10,5618		3	10,9121		3	7,2354
4	18,9783		4	13,6753		4	10,7308		4	10,5904		4	7,2442
5	19,3317		5	13,7470		5	10,9596		5	10,6287		5	7,1097
6			6			6			6			6	
7	19,03590		7	13,93976		7	10,72416		7	10,70332		7	7,16356
EPSILON =0,000001									Iteraciones 500	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]										
	N PRO.	2		N PRO.	3		N PRO.	4	N PRO.	5		N PRO.	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO
1	19,3442		1	10,5708		1	7,2248		1	5,8084		1	5,7482
2	18,7582		2	10,9150		2	6,9962		2	5,9430		2	5,9264
3	18,6700		3	10,6305		3	7,0732		3	5,7566		3	5,8086
4	18,7175		4	10,5316		4	7,0439		4	5,7940		4	5,8055
5	18,8145		5	10,5981		5	7,2210		5	5,9207		5	5,7757
6			6			6			6			6	
7	18,86088		7	10,64920		7	7,11181		7	5,84454		7	5,81287

DATOS ALGORTIMO CAJAS RESULTADO CON UF COMPARTIDO

EPSILON =0,01									Iteraciones 10	
ALGORTIMO CAJAS RESULTADO CON DISTRIBUCION EQULIBRADA										
	N PRO.	2		N PRO.	3		N PRO.	4	N PRO.	5		N PRO.	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO
1	0,166265		1	0,132112		1	0,100677		1	0,102698		1	0,0562119
2	0,165358		2	0,130507		2	0,102168		2	0,102691		2	0,0539591
3	0,165768		3	0,13323		3	0,103275		3	0,103423		3	0,0546961
4	0,171619		4	0,132588		4	0,101322		4	0,104949		4	0,05446
5	0,16577		5	0,13322		5	0,100985		5	0,100362		5	0,056881
6			6			6			6			6	
7	0,166956		7	0,1323314		7	0,1016854		7	0,1028246		7	0,05524162
EPSILON =0,01									Iteraciones 50	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]										
	N PRO.	2		N PRO.	3		N PRO.	4	N PRO.	5		N PRO.	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO
1	0,1623		1	0,111841		1	0,0785191		1	0,0662649		1	0,0681221
2	0,168275		2	0,119064		2	0,0784819		2	0,0691981		2	0,0681119
3	0,170553		3	0,113289		3	0,0782609		3	0,0663471		3	0,0659029
4	0,161356		4	0,112249		4	0,0785909		4	0,0658		4	0,067142
5	0,163449		5	0,113048		5	0,077143		5	0,0667088		5	0,0664661
6			6			6			6			6	
7	0,1651866		7	0,1138982		7	0,07819916		7	0,06686378		7	0,067149
EPSILON =0,01									Iteraciones 100	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]										
	N PRO.	2		N PRO.	3		N PRO.	4	N PRO.	5		N PRO.	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO
1	0,165458		1	0,116519		1	0,0750551		1	0,076426		1	0,0631919
2	0,363924		2	0,112407		2	0,07743		2	0,0734239		2	0,0618761
3	0,372896		3	0,112182		3	0,0790429		3	0,0731361		3	0,0620351
4	0,166888		4	0,113993		4	0,074672		4	0,0735199		4	0,063226
5	0,164156		5	0,117699		5	0,0796549		5	0,074415		5	0,061342
6			6			6			6			6	
7	0,2466644		7	0,11456		7	0,07717098		7	0,07418418		7	0,06233422
EPSILON =0,01									Iteraciones 500	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]										
	N PRO.	2		N PRO.	3		N PRO.	4	N PRO.	5		N PRO.	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO
1	0,3677		1	0,110653		1	0,0969632		1	0,0946059		1	0,0926878
2	0,325419		2	0,102804		2	0,097472		2	0,0933788		2	0,0948901
3	0,334768		3	0,111037		3	0,0990601		3	0,093107		3	0,095819
4	0,391357		4	0,107001		4	0,0932178		4	0,089417		4	0,088589
5	0,367886		5	0,110534		5	0,0972328		5	0,092901		5	0,0905001
6			6			6			6			6	
7	0,357426		7	0,1084058		7	0,09678918		7	0,09268194		7	0,0924972

EPSILON =0,0001 — Iteraciones 10

FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]

N PRO,		N PRO,		N PRO,		N PRO,		N PRO,	
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,179521	1	0,148434	1	0,1164	1	0,121891	1	0,064213
2	0,176778	2	0,144759	2	0,117976	2	0,114231	2	0,060771
3	0,187137	3	0,145861	3	0,115484	3	0,120306	3	0,0633562
4	0,17744	4	0,14464	4	0,119887	4	0,116682	4	0,0614059
5	0,177896	5	0,145008	5	0,119521	5	0,115522	5	0,0609469
6		6		6		6		6	
7	0,1797544	7	0,1457404	7	0,1178536	7	0,1177264	7	0,0621386

EPSILON =0,0001 — Iteraciones 50

FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]

N PRO,		N PRO,		N PRO,		N PRO,		N PRO,	
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,178829	1	0,120616	1	0,09234	1	0,076309	1	0,079236
2	0,174766	2	0,118325	2	0,093116	2	0,076236	2	0,080734
3	0,183551	3	0,118591	3	0,093348	3	0,0757959	3	0,083364
4	0,177356	4	0,121098	4	0,0960832	4	0,0761912	4	0,0818839
5	0,180282	5	0,119784	5	0,0952818	5	0,0761769	5	0,0832961
6		6		6		6		6	
7	0,1789568	7	0,1196828	7	0,0940338	7	0,0761418	7	0,0817028

EPSILON =0,0001 — Iteraciones 100

FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]

N PRO,		N PRO,		N PRO,		N PRO,		N PRO,	
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,17903	1	0,126959	1	0,0861201	1	0,09061	1	0,0621588
2	0,181032	2	0,129533	2	0,085747	2	0,088131	2	0,060051
3	0,184513	3	0,128501	3	0,0881698	3	0,087532	3	0,061667
4	0,184883	4	0,128141	4	0,0862739	4	0,0887408	4	0,0613241
5	0,177541	5	0,125942	5	0,0895269	5	0,090395	5	0,061583
6		6		6		6		6	
7	0,1813998	7	0,1278152	7	0,08716754	7	0,08908176	7	0,06135678

EPSILON =0,0001 — Iteraciones 500

FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]

N PRO,		N PRO,		N PRO,		N PRO,		N PRO,	
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,145888	1	0,119937	1	0,101625	1	0,0978861	1	0,0922549
2	0,144919	2	0,115849	2	0,105042	2	0,0952561	2	0,0927262
3	0,146539	3	0,119791	3	0,101293	3	0,099041	3	0,0938611
4	0,151217	4	0,143625	4	0,102166	4	0,095866	4	0,0940881
5	0,146779	5	0,122933	5	0,0981262	5	0,0993631	5	0,095238
6		6		6		6		6	
7	0,1470684	7	0,124427	7	0,10165044	7	0,09748246	7	0,09363366

EPSILON =0,000001 — Iteraciones 10

FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]

N PRO,		N PRO,		N PRO,		N PRO,		N PRO,	
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	
1	0,199801	1	0,161767	1	0,137469	1	0,137472	1	0,0711689
2	0,195538	2	0,16355	2	0,131527	2	0,134788	2	0,0726011
3	0,195738	3	[-0,0898421,-0,0	3	0,133547	3	0,132812	3	0,0735321
4	0,193861	4	0,161453	4	0,131884	4	0,132183	4	0,070715
5	0,193227	5	0,168393	5	0,131825	5	0,137392	5	0,0707052
6		6		6		6		6	
7	0,195633	7	0,16379075	7	0,1332504	7	0,1349294	7	0,07174446

EPSILON =0,000001 — Iteraciones 50

FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]

N PRO,		N PRO,		N PRO,		N PRO,		N PRO,	
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,192727	1	0,128712	1	0,109944	1	0,096194	1	0,09798
2	0,194147	2	0,127777	2	0,109293	2	0,0960591	2	0,0966232
3	0,192538	3	0,12806	3	0,114639	3	0,0935941	3	0,096524
4	0,201502	4	0,132232	4	0,109972	4	0,095062	4	0,0962319
5	0,192175	5	0,127499	5	0,109641	5	0,0932448	5	0,0977681
6		6		6		6		6	
7	0,1946178	7	0,128856	7	0,1106978	7	0,0948308	7	0,09702544

EPSILON =0,000001 — Iteraciones 100

FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]

N PRO,		N PRO,		N PRO,		N PRO,		N PRO,	
N°		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,193715	1	0,145527	1	0,103907	1	0,10251	1	0,071429
2	0,192549	2	0,1421	2	0,107987	2	0,112381	2	0,0719199
3	0,194489	3	0,144968	3	0,102889	3	0,109558	3	0,0719419
4	0,192861	4	0,144979	4	0,10231	4	0,107329	4	0,071882
5	0,195873	5	0,148842	5	0,101691	5	0,106357	5	0,0710678
6		6		6		6		6	
7	0,1938974	7	0,1452832	7	0,1037568	7	0,107627	7	0,07164812

EPSILON =0,000001 — Iteraciones 500

FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]

N PRO,		N PRO,		N PRO,		N PRO,		N PRO,	
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,165157	1	0,167083	1	0,116846	1	0,115934	1	0,107894
2	0,371074	2	0,139823	2	0,116401	2	0,112675	2	0,115593
3	0,363015	3	0,138238	3	0,116089	3	0,117886	3	0,108526
4	0,37053	4	0,138855	4	0,125454	4	0,114954	4	0,108965
5	0,160963	5	0,140129	5	0,118385	5	0,117663	5	0,108614
6		6		6		6		6	
7	0,2861478	7	0,1448256	7	0,118635	7	0,1158224	7	0,1099184

EPSILON =0,01										Iteraciones 10	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0798092		1	0,0741689	1	0,265558	1	0,249294	1	0,249967	
2	0,0796208		2	0,0724671	2	0,261612	2	0,258072	2	0,250254	
3	0,07723		3	0,0720491	3	0,261105	3	0,250161	3	0,249605	
4	0,090266		4	0,0724931	4	0,261685	4	0,249626	4	0,253552	
5	0,0773771		5	0,0714159	5	0,260409	5	0,250918	5	0,2498	
6			6		6		6		6		
7	0,08086062		7	0,07251882	7	0,2620738	7	0,2516142	7	0,2506356	
EPSILON =0,01										Iteraciones 50	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0771861		1	0,0652461	1	0,0644879	1	0,061492	1	0,0583491	
2	0,0772359		2	0,0646899	2	0,065726	2	0,0604832	2	0,0568352	
3	0,0775619		3	0,0642231	3	0,0643461	3	0,061698	3	0,0571809	
4	0,0754449		4	0,0662239	4	0,0662949	4	0,059911	4	0,0627789	
5	0,0782149		5	0,065779	5	0,0645959	5	0,059886	5	0,0593228	
6			6		6		6		6		
7	0,07712874		7	0,0652324	7	0,06509016	7	0,06069404	7	0,05889338	
EPSILON =0,01										Iteraciones 100	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,073549		1	0,0991559	1	0,0929811	1	0,0810978	1	0,087924	
2	0,0722342		2	0,097841	2	0,09321	2	0,0808549	2	0,0881791	
3	0,073065		3	0,096494	3	0,0932801	3	0,0832288	3	0,0884209	
4	0,0706308		4	0,0980811	4	0,093724	4	0,0797021	4	0,087225	
5	0,0732679		5	0,0982699	5	0,0912859	5	0,0812781	5	0,087754	
6			6		6		6		6		
7	0,07254938		7	0,09796838	7	0,09289622	7	0,08123234	7	0,0879006	
EPSILON =0,01										Iteraciones 500	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0659649		1	0,0673671	1	0,0656891	1	0,06657	1	0,0675371	
2	0,0659692		2	0,069272	2	0,0655339	2	0,0663559	2	0,0672269	
3	0,066483		3	0,0676231	3	0,0672081	3	0,068614	3	0,0668001	
4	0,0669761		4	0,065506	4	0,0665441	4	0,0662181	4	0,0670612	
5	0,0665529		5	0,0690401	5	0,0661049	5	0,0660751	5	0,065356	
6			6		6		6		6		
7	0,06638922		7	0,06776166	7	0,06621602	7	0,06676662	7	0,06679626	
EPSILON =0,0001										Iteraciones 10	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,135692		1	0,126438	1	0,609648	1	0,598753	1	0,601694	
2	0,128943		2	0,12646	2	0,626903	2	0,599094	2	0,599597	
3	0,128057		3	0,126496	3	0,608366	3	0,617746	3	0,602657	
4	0,127886		4	0,123035	4	0,608319	4	0,617584	4	0,599113	
5	0,133325		5	0,122717	5	0,62413	5	0,602762	5	0,614584	
6			6		6		6		6		
7	0,1307806		7	0,1250292	7	0,6154732	7	0,6071878	7	0,603529	
EPSILON =0,0001										Iteraciones 50	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,126725		1	0,118789	1	0,114573	1	0,11211	1	0,0864179	
2	0,126901		2	0,114613	2	0,117653	2	0,11113	2	0,0862749	
3	0,130782		3	0,114617	3	0,116835	3	0,111324	3	0,0863779	
4	0,125623		4	0,119377	4	0,116159	4	0,111083	4	0,0890572	
5	0,131173		5	0,115606	5	0,131794	5	0,111909	5	0,0851729	
6			6		6		6		6		
7	0,1282408		7	0,1166004	7	0,1194028	7	0,1115112	7	0,08666016	
EPSILON =0,0001										Iteraciones 100	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,12141		1	0,112077	1	0,105551	1	0,101993	1	0,111526	
2	0,126745		2	0,111771	2	0,105498	2	0,098429	2	0,117621	
3	0,122537		3	0,112537	3	0,108647	3	0,0992081	3	0,115594	
4	0,127316		4	0,112059	4	0,104846	4	0,100796	4	0,114489	
5	0,12333		5	0,11212	5	0,10468	5	0,100102	5	0,111611	
6			6		6		6		6		
7	0,1242676		7	0,1121128	7	0,1058444	7	0,10010562	7	0,1141682	
EPSILON =0,0001										Iteraciones 500	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,080935		1	0,0757792	1	0,0762279	1	0,0745661	1	0,0738611	
2	0,083905		2	0,0739169	2	0,0762391	2	0,0742719	2	0,0721281	
3	0,0812449		3	0,077687	3	0,076252	3	0,0735002	3	0,0758691	
4	0,081218		4	0,073581	4	0,0781901	4	0,0737438	4	0,072258	
5	0,0811589		5	0,072834	5	0,076009	5	0,072598	5	0,0718889	
6			6		6		6		6		
7	0,08169236		7	0,07475962	7	0,07658362	7	0,073736	7	0,07320104	

EPSILON =0,000001										Iteraciones 10	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,183933		1	0,174671		1	0,609875		1	0,602924		1	0,601397	
2	0,179636		2	0,174581		2	0,608345		2	0,61008		2	0,598652	
3	0,183595		3	0,177602		3	0,605817		3	0,604368		3	0,614193	
4	0,18102		4	0,178722		4	0,634729		4	0,61662		4	0,598418	
5	0,17938		5	0,174526		5	0,611365		5	0,600282		5	0,616708	
6			6			6			6			6		
7	0,1815128		7	0,1760204		7	0,6140262		7	0,6068548		7	0,6058736	
EPSILON =0,000001										Iteraciones 50	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,177627		1	0,167388		1	0,166829		1	0,167763		1	0,140154	
2	0,177785		2	0,170503		2	0,17136		2	0,16298		2	0,137094	
3	0,177539		3	0,168894		3	0,172768		3	0,163539		3	0,140972	
4	0,177941		4	0,170973		4	0,167381		4	0,161811		4	0,137224	
5	0,177371		5	0,166693		5	0,167418		5	0,162787		5	0,140619	
6			6			6			6			6		
7	0,1776526		7	0,1688902		7	0,1691512		7	0,163776		7	0,1392126	
EPSILON =0,000001										Iteraciones 100	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,17831		1	0,155941		1	0,157814		1	0,153132		1	0,152388	
2	0,179017		2	0,157963		2	0,155848		2	0,153191		2	0,151247	
3	0,177388		3	0,154339		3	0,155098		3	0,153443		3	0,152555	
4	0,173136		4	0,154376		4	0,154607		4	0,150252		4	0,155749	
5	0,178002		5	0,156343		5	0,154658		5	0,153864		5	0,152708	
6			6			6			6			6		
7	0,1771706		7	0,1557924		7	0,155605		7	0,1527764		7	0,1529294	
EPSILON =0,000001										Iteraciones 500	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,134736		1	0,121958		1	0,127653		1	0,124604		1	0,120681	
2	0,132275		2	0,124128		2	0,127382		2	0,126985		2	0,120345	
3	0,133623		3	0,123391		3	0,131333		3	0,125101		3	0,120232	
4	0,133639		4	0,121831		4	0,128947		4	0,125522		4	0,121981	
5	0,131742		5	0,121565		5	0,129661		5	0,126323		5	0,122009	
6			6			6			6			6		
7	0,133203		7	0,1225746		7	0,1289952		7	0,125707		7	0,1210496	
EPSILON =0,01										Iteraciones 10	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,016335		1	0,0155408		1	0,0149362		1	0,0153091		1	0,0107942	
2	0,016222		2	0,0156629		2	0,0149231		2	0,0153692		2	0,011924	
3	0,0161991		3	0,0155048		3	0,014874		3	0,0153868		3	0,010956	
4	0,0167868		4	0,0159998		4	0,014807		4	0,01509		4	0,0102661	
5	0,016607		5	0,015713		5	0,0148029		5	0,0152431		5	0,010067	
6			6			6			6			6		
7	0,01642998		7	0,01568426		7	0,01486864		7	0,01527964		7	0,01080146	
EPSILON =0,01										Iteraciones 50	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,0140071		1	0,013911		1	0,0130389		1	0,012183		1	0,01213	
2	0,012702		2	0,013941		2	0,0126431		2	0,0118082		2	0,0121379	
3	0,013638		3	0,013967		3	0,0127518		3	0,0121858		3	0,0124111	
4	0,0138218		4	0,0139878		4	0,0127461		4	0,012603		4	0,0125029	
5	0,0134621		5	0,013701		5	0,0126901		5	0,012393		5	0,013041	
6			6			6			6			6		
7	0,0135262		7	0,01390156		7	0,012774		7	0,0122346		7	0,01244458	
EPSILON =0,01										Iteraciones 100	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,0113258		1	0,0101979		1	0,010046		1	0,0100541		1	0,00974703	
2	0,0110521		2	0,0102141		2	0,0100641		2	0,0101268		2	0,00995708	
3	0,010973		3	0,009583		3	0,010155		3	0,010339		3	0,0098021	
4	0,0109441		4	0,0101662		4	0,0100939		4	0,0101929		4	0,0100191	
5	0,011014		5	0,010227		5	0,0101998		5	0,0101039		5	0,0121901	
6			6			6			6			6		
7	0,0110618		7	0,01007764		7	0,01011176		7	0,01016334		7	0,01034308	
EPSILON =0,01										Iteraciones 500	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,0318539		1	0,0332422		1	0,0320549		1	0,0346551		1	0,0334229	
2	0,032531		2	0,0318611		2	0,0320351		2	0,0318539		2	0,0346439	
3	0,0317421		3	0,0330958		3	0,032145		3	0,0320861		3	0,0340631	
4	0,032624		4	0,0321472		4	0,0321648		4	0,0331562		4	0,032527	
5	0,03336		5	0,0322511		5	0,0325689		5	0,0323119		5	0,033447	
6			6			6			6			6		
7	0,0324222		7	0,03251948		7	0,03219374		7	0,03281264		7	0,03362078	

EPSILON =0,0001										Iteraciones 10	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,0258911		1	0,024735		1	0,0238988		1	0,02578		1	0,015039	
2	0,02547		2	0,024395		2	0,024277		2	0,0245981		2	0,016247	
3	0,026449		3	0,0254719		3	0,0242541		3	0,0245709		3	0,015255	
4	0,0255189		4	0,0247121		4	0,0244		4	0,025044		4	0,0148358	
5	0,0251279		5	0,0246689		5	0,024184		5	0,024863		5	0,015861	
6			6			6			6			6		
7	0,02569138		7	0,02479658		7	0,02420278		7	0,0249712		7	0,01544756	

EPSILON =0,0001										Iteraciones 50	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,023134		1	0,0232198		1	0,022481		1	0,021323		1	0,0221379	
2	0,0233579		2	0,0229909		2	0,022073		2	0,0219829		2	0,0218029	
3	0,0229001		3	0,023515		3	0,0224502		3	0,0212331		3	0,02123	
4	0,023263		4	0,0234139		4	0,0223739		4	0,021631		4	0,0228932	
5	0,023973		5	0,023416		5	0,021358		5	0,0213082		5	0,0218699	
6			6			6			6			6		
7	0,0233256		7	0,02331112		7	0,02214722		7	0,02149564		7	0,02198678	

EPSILON =0,0001										Iteraciones 100	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,020293		1	0,0192151		1	0,019361		1	0,019316		1	0,0150921	
2	0,0203421		2	0,0192249		2	0,019279		2	0,0212901		2	0,014945	
3	0,0196109		3	0,019228		3	0,019444		3	0,019531		3	0,0156329	
4	0,0202811		4	0,019285		4	0,0194669		4	0,0210681		4	0,015944	
5	0,0204971		5	0,019155		5	0,01986		5	0,0186708		5	0,0154049	
6			6			6			6			6		
7	0,02020484		7	0,0192216		7	0,01948218		7	0,0199752		7	0,01540378	

EPSILON =0,0001										Iteraciones 500	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,033457		1	0,0327082		1	0,0325189		1	0,0330341		1	0,0340331	
2	0,032418		2	0,033422		2	0,032522		2	0,0335422		2	0,0340111	
3	0,032901		3	0,032306		3	0,033191		3	0,0350928		3	0,0336571	
4	0,031846		4	0,0334198		4	0,032253		4	0,0334349		4	0,032568	
5	0,032747		5	0,031662		5	0,0317349		5	0,0335131		5	0,0336049	
6			6			6			6			6		
7	0,0326738		7	0,0327036		7	0,03244396		7	0,03372342		7	0,03357484	

EPSILON =0,000001										Iteraciones 10	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,0342991		1	0,0344329		1	0,033097		1	0,0338049		1	0,0200229	
2	0,0351162		2	0,035331		2	0,033798		2	0,0356789		2	0,0209711	
3	0,0332658		3	0,034457		3	0,0342069		3	0,0340919		3	0,021857	
4	0,034965		4	0,0345941		4	0,03438		4	0,0340159		4	0,0205059	
5	0,03457		5	0,033977		5	0,033067		5	0,0332661		5	0,0224068	
6			6			6			6			6		
7	0,03444322		7	0,0345584		7	0,03370978		7	0,03417154		7	0,02115274	

EPSILON =0,000001										Iteraciones 50	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,0326309		1	0,0331521		1	0,031708		1	0,0312879		1	0,030817	
2	0,0319729		2	0,032829		2	0,031249		2	0,030581		2	0,0313799	
3	0,0315032		3	0,033016		3	0,031446		3	0,031069		3	0,0313501	
4	0,0322189		4	0,0332348		4	0,0313258		4	0,030911		4	0,033237	
5	0,0325131		5	0,0328381		5	0,031822		5	0,031626		5	0,032011	
6			6			6			6			6		
7	0,0321678		7	0,033014		7	0,03151016		7	0,03109498		7	0,031759	

EPSILON =0,000001										Iteraciones 100	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,0285079		1	0,0288529		1	0,028903		1	0,0287721		1	0,0199108	
2	0,0296471		2	0,0285432		2	0,0296781		2	0,0286789		2	0,020304	
3	0,0303471		3	0,028991		3	0,028204		3	0,0290208		3	0,020139	
4	0,0302541		4	0,0296102		4	0,0285759		4	0,0289121		4	0,0207729	
5	0,0300651		5	0,0296261		5	0,028064		5	0,028688		5	0,0203969	
6			6			6			6			6		
7	0,02976426		7	0,02912468		7	0,028685		7	0,02881438		7	0,02030472	

EPSILON =0,000001										Iteraciones 500	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,0331509		1	0,032196		1	0,0333419		1	0,0324769		1	0,0327981	
2	0,0325179		2	0,032748		2	0,032958		2	0,0328441		2	0,0326529	
3	0,0324109		3	0,0325701		3	0,0324171		3	0,0320671		3	0,032598	
4	0,0324218		4	0,0330889		4	0,0334361		4	0,032686		4	0,032517	
5	0,0318069		5	0,033469		5	0,0335472		5	0,0333591		5	0,0338202	
6			6			6			6			6		
7	0,03246168		7	0,0328144		7	0,03314006		7	0,03268664		7	0,03287724	

EPSILON =0,01										Iteraciones 10	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0503669		1	0,028199	1	0,0260789	1	0,02527	1	0,021884	
2	0,05021		2	0,027823	2	0,0255911	2	0,025393	2	0,0222492	
3	0,0504661		3	0,0282359	3	0,0261121	3	0,0253398	3	0,0217161	
4	0,0507958		4	0,028635	4	0,0251071	4	0,025707	4	0,0218091	
5	0,0508549		5	0,0279551	5	0,026015	5	0,025821	5	0,0218949	
6			6		6		6		6		
7	0,05053874		7	0,0281696	7	0,02578084	7	0,02550616	7	0,02191066	

EPSILON =0,01										Iteraciones 50	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0495169		1	0,0422251	1	0,039906	1	0,039201	1	0,02388	
2	0,0512609		2	0,041677	2	0,0424221	2	0,039983	2	0,021219	
3	0,0500131		3	0,0421641	3	0,040571	3	0,039799	3	0,0227349	
4	0,050622		4	0,0412731	4	0,0412319	4	0,0401189	4	0,0239911	
5	0,0499721		5	0,0421429	5	0,04038	5	0,038944	5	0,023279	
6			6		6		6		6		
7	0,050277		7	0,04189644	7	0,0409022	7	0,03960918	7	0,0230208	

EPSILON =0,01										Iteraciones 100	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0493011		1	0,045841	1	0,0438972	1	0,0395529	1	0,0280261	
2	0,0497999		2	0,0431862	2	0,0444	2	0,0405359	2	0,0277879	
3	0,0503459		3	0,0423	3	0,043251	3	0,0404871	3	0,0277541	
4	0,04965		4	0,0429528	4	0,044383	4	0,04159	4	0,027936	
5	0,049783		5	0,043283	5	0,043407	5	0,040324	5	0,028542	
6			6		6		6		6		
7	0,04977598		7	0,0435126	7	0,04386764	7	0,04049798	7	0,02800922	

EPSILON =0,01										Iteraciones 500	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,035414		1	0,034369	1	0,033895	1	0,0348659	1	0,035675	
2	0,0354009		2	0,035212	2	0,0333731	2	0,0349009	2	0,033514	
3	0,0358591		3	0,0344059	3	0,0345478	3	0,0341492	3	0,0364201	
4	0,0356991		4	0,034694	4	0,033978	4	0,033633	4	0,0334761	
5	0,0351732		5	0,0343931	5	0,034281	5	0,0330911	5	0,0339639	
6			6		6		6		6		
7	0,03550926		7	0,0346148	7	0,03401498	7	0,03412802	7	0,03460982	

EPSILON =0,0001										Iteraciones 10	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0923288		1	0,0493529	1	0,046828	1	0,0460689	1	0,0407891	
2	0,0916901		2	0,049067	2	0,0468831	2	0,0466828	2	0,0409219	
3	0,0947478		3	0,0494049	3	0,0474479	3	0,046325	3	0,040683	
4	0,091136		4	0,0489111	4	0,047153	4	0,0473461	4	0,0403121	
5	0,0915611		5	0,049015	5	0,046597	5	0,0475819	5	0,04141	
6			6		6		6		6		
7	0,09229276		7	0,04915018	7	0,0469818	7	0,04680094	7	0,04082322	

EPSILON =0,0001										Iteraciones 50	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,090173		1	0,084564	1	0,081198	1	0,082443	1	0,0460989	
2	0,093297		2	0,086664	2	0,0816779	2	0,0807781	2	0,045001	
3	0,0902281		3	0,084244	3	0,0822959	3	0,08026	3	0,043051	
4	0,0920432		4	0,085047	4	0,0812478	4	0,0820041	4	0,0425558	
5	0,0894251		5	0,0860031	5	0,0816879	5	0,0817571	5	0,0438058	
6			6		6		6		6		
7	0,09103328		7	0,08530442	7	0,0816215	7	0,08144846	7	0,0441025	

EPSILON =0,0001										Iteraciones 100	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0905149		1	0,0855711	1	0,0845771	1	0,0837021	1	0,0504351	
2	0,0931849		2	0,0853021	2	0,0849321	2	0,0819111	2	0,050843	
3	0,091253		3	0,085722	3	0,0853801	3	0,090256	3	0,0483911	
4	0,0923738		4	0,086803	4	0,084717	4	0,0815849	4	0,0501971	
5	0,092205		5	0,0861671	5	0,0843801	5	0,083483	5	0,0494418	
6			6		6		6		6		
7	0,09190632		7	0,08591306	7	0,08479728	7	0,08418742	7	0,04986162	

EPSILON =0,0001										Iteraciones 500	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0707459		1	0,06373	1	0,06604	1	0,049078	1	0,0506489	
2	0,0724959		2	0,0652342	2	0,064419	2	0,049372	2	0,0510252	
3	0,070226		3	0,0636461	3	0,0650759	3	0,0502949	3	0,049608	
4	0,069792		4	0,065146	4	0,0647631	4	0,04936	4	0,054827	
5	0,0701771		5	0,0628638	5	0,0648651	5	0,050262	5	0,050483	
6			6		6		6		6		
7	0,07068738		7	0,06412402	7	0,06503262	7	0,04967338	7	0,05131842	

EPSILON =0,000001									Iteraciones 10	
FUNCION : MATYAS [-10,10][-10,10]										
	N PRO,	2		N PRO,	3		N PRO,	4	N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO
1	0,138106		1	0,0707049		1	0,0687971		1	0,0699191		1	0,059124
2	0,135368		2	0,0694649		2	0,0689371		2	0,068536		2	0,060951
3	0,13273		3	0,0697		3	0,0679178		3	0,0675821		3	0,0590701
4	0,136464		4	0,071553		4	0,0696568		4	0,0683589		4	0,0589728
5	0,136136		5	0,0715718		5	0,0686071		5	0,068471		5	0,0612018
6			6			6			6			6	
7	0,1357608		7	0,07059892		7	0,06878318		7	0,06857342		7	0,05986394

EPSILON =0,000001									Iteraciones 50	
FUNCION : MATYAS [-10,10][-10,10]										
	N PRO,	2		N PRO,	3		N PRO,	4	N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO
1	0,132577		1	0,126952		1	0,123358		1	0,122967		1	0,0668941
2	0,135773		2	0,125254		2	0,125119		2	0,122828		2	0,0670011
3	0,131778		3	0,127982		3	0,125535		3	0,120982		3	0,0662351
4	0,132286		4	0,126205		4	0,128367		4	0,124393		4	0,067415
5	0,136029		5	0,124847		5	0,122693		5	0,12242		5	0,065887
6			6			6			6			6	
7	0,1336886		7	0,126248		7	0,1250144		7	0,122718		7	0,06668646

EPSILON =0,000001									Iteraciones 100	
FUNCION : MATYAS [-10,10][-10,10]										
	N PRO,	2		N PRO,	3		N PRO,	4	N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO
1	0,133246		1	0,127596		1	0,129276		1	0,123273		1	0,0726619
2	0,131391		2	0,126474		2	0,128269		2	0,126079		2	0,072787
3	0,132007		3	0,129688		3	0,126362		3	0,124735		3	0,0729909
4	0,13407		4	0,129186		4	0,127719		4	0,121905		4	0,069895
5	0,133464		5	0,126235		5	0,125102		5	0,12289		5	0,0706091
6			6			6			6			6	
7	0,1328356		7	0,1278358		7	0,1273456		7	0,1237764		7	0,07178878

EPSILON =0,000001									Iteraciones 500	
FUNCION : MATYAS [-10,10][-10,10]										
	N PRO,	2		N PRO,	3		N PRO,	4	N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO
1	0,112051		1	0,105704		1	0,106073		1	0,0701981		1	0,0709729
2	0,119661		2	0,108789		2	0,106891		2	0,071018		2	0,072176
3	0,112704		3	0,106531		3	0,10732		3	0,0706029		3	0,0709529
4	0,111678		4	0,106063		4	0,106746		4	0,0712729		4	0,0709369
5	0,110912		5	0,105511		5	0,106362		5	0,070986		5	0,070966
6			6			6			6			6	
7	0,1134012		7	0,1065196		7	0,1066784		7	0,07081558		7	0,07120094

EPSILON =0,01									Iteraciones 10	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]										
	N PRO,	2		N PRO,	3		N PRO,	4	N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO
1	0,11348		1	0,109679		1	0,108386		1	0,0708008		1	0,0711591
2	0,111102		2	0,107705		2	0,110046		2	0,0703759		2	0,07131
3	0,108797		3	0,105968		3	0,106316		3	0,072583		3	0,0716679
4	0,112476		4	0,106935		4	0,106172		4	0,070529		4	0,0721481
5	0,117487		5	0,107208		5	0,107856		5	0,0713491		5	0,0704131
6			6			6			6			6	
7	0,1126684		7	0,107499		7	0,1077552		7	0,07112756		7	0,07133964

EPSILON =0,01									Iteraciones 50	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]										
	N PRO,	2		N PRO,	3		N PRO,	4	N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO
1	0,0726261		1	0,0429201		1	0,0389421		1	0,035089		1	0,0355301
2	0,0740449		2	0,0420611		2	0,0390842		2	0,0348389		2	0,0342431
3	0,073436		3	0,0428832		3	0,0388401		3	0,0349441		3	0,034539
4	0,0713241		4	0,0415621		4	0,0400209		4	0,0361209		4	0,0347791
5	0,0738189		5	0,0420759		5	0,0385108		5	0,034533		5	0,034492
6			6			6			6			6	
7	0,07305		7	0,04230048		7	0,03907962		7	0,03510518		7	0,03471666

EPSILON =0,01									Iteraciones 100	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]										
	N PRO,	2		N PRO,	3		N PRO,	4	N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO
1	0,070421		1	0,043731		1	0,0403738		1	0,036761		1	0,0382309
2	0,0711801		2	0,044863		2	0,040066		2	0,0381811		2	0,033751
3	0,069078		3	0,043525		3	0,0403972		3	0,0371711		3	0,0378242
4	0,0727251		4	0,044667		4	0,0392191		4	0,0370851		4	0,0338299
5	0,0724101		5	0,043014		5	0,040669		5	0,037159		5	0,0344989
6			6			6			6			6	
7	0,07116286		7	0,04396		7	0,04014502		7	0,03727146		7	0,03562698

EPSILON =0,01									Iteraciones 500	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]										
	N PRO,	2		N PRO,	3		N PRO,	4	N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO
1	0,0492859		1	0,0518429		1	0,049264		1	0,0491619		1	0,052346
2	0,0508089		2	0,049804		2	0,0490649		2	0,05023		2	0,0506999
3	0,049727		3	0,0499871		3	0,0494349		3	0,0484521		3	0,0501058
4	0,0492611		4	0,0502		4	0,052186		4	0,0499482		4	0,049829
5	0,051919		5	0,0496199		5	0,0489061		5	0,0513599		5	0,053776
6			6			6			6			6	
7	0,05020038		7	0,05029078		7	0,04977118		7	0,04983042		7	0,05135134

EPSILON =0,0001										Iteraciones 10	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0874732		1	0,05128	1	0,040695	1	0,0403519	1	0,0386999	
2	0,0837569		2	0,050652	2	0,040679	2	0,0389509	2	0,0405228	
3	0,087153		3	0,0529392	3	0,040328	3	0,0389102	3	0,0392051	
4	0,0868189		4	0,049509	4	0,0398529	4	0,0395498	4	0,0394628	
5	0,086781		5	0,0504348	5	0,040087	5	0,039068	5	0,038681	
6			6		6		6		6		
7	0,0863966		7	0,050963	7	0,04032838	7	0,03936616	7	0,03931432	
EPSILON =0,0001										Iteraciones 50	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0816309		1	0,049721	1	0,045696	1	0,042572	1	0,042412	
2	0,0825911		2	0,0496371	2	0,045531	2	0,0425811	2	0,0414631	
3	0,081789		3	0,0483668	3	0,04653	3	0,04267	3	0,0413702	
4	0,082736		4	0,0494399	4	0,046541	4	0,0415361	4	0,0420918	
5	0,085196		5	0,0499651	5	0,04618	5	0,0420289	5	0,0413251	
6			6		6		6		6		
7	0,0827886		7	0,04942598	7	0,0460956	7	0,04227762	7	0,04173244	
EPSILON =0,0001										Iteraciones 100	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0821171		1	0,0498772	1	0,0430889	1	0,0434861	1	0,034832	
2	0,083847		2	0,0508249	2	0,044347	2	0,0432999	2	0,0339191	
3	0,0803599		3	0,0500641	3	0,042824	3	0,0435719	3	0,0344889	
4	0,083153		4	0,050046	4	0,0433779	4	0,0418999	4	0,0340378	
5	0,080303		5	0,0495119	5	0,0436239	5	0,042279	5	0,0354149	
6			6		6		6		6		
7	0,081956		7	0,05006482	7	0,04345234	7	0,04290736	7	0,03453854	
EPSILON =0,0001										Iteraciones 500	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0503349		1	0,049866	1	0,0491672	1	0,0501699	1	0,050252	
2	0,0502651		2	0,0495741	2	0,05001	2	0,0491161	2	0,0520539	
3	0,0498569		3	0,0490332	3	0,0502679	3	0,052048	3	0,049427	
4	0,0492971		4	0,0496371	4	0,0516648	4	0,050082	4	0,0543079	
5	0,0502481		5	0,0495889	5	0,049334	5	0,04931	5	0,0507128	
6			6		6		6		6		
7	0,05000042		7	0,04953986	7	0,05008878	7	0,0501452	7	0,05135072	
EPSILON =0,000001										Iteraciones 10	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0931568		1	0,057852	1	0,0450189	1	0,0423548	1	0,0420952	
2	0,093646		2	0,0593839	2	0,04527	2	0,0427639	2	0,041934	
3	0,0978639		3	0,0579989	3	0,0433829	3	0,0423801	3	0,0425541	
4	0,0934072		4	0,0586588	4	0,0437891	4	0,042443	4	0,0440869	
5	0,0943692		5	0,0589662	5	0,0432189	5	0,0424139	5	0,0424001	
6			6		6		6		6		
7	0,09448862		7	0,05857196	7	0,04413596	7	0,04247114	7	0,04261406	
EPSILON =0,000001										Iteraciones 50	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,096503		1	0,054812	1	0,0476551	1	0,0416682	1	0,0419831	
2	0,090349		2	0,0563819	2	0,047112	2	0,0432549	2	0,0417869	
3	0,096086		3	0,0542419	3	0,047338	3	0,0418241	3	0,0410199	
4	0,0934832		4	0,0548151	4	0,0473239	4	0,0418911	4	0,042958	
5	0,094007		5	0,0541911	5	0,046438	5	0,0425329	5	0,0415971	
6			6		6		6		6		
7	0,09408564		7	0,0548884	7	0,0471734	7	0,04223424	7	0,041869	
EPSILON =0,000001										Iteraciones 100	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0919561		1	0,058363	1	0,050288	1	0,0509951	1	0,0363221	
2	0,0905938		2	0,057116	2	0,0497999	2	0,050704	2	0,0400839	
3	0,089349		3	0,0574291	3	0,0498118	3	0,0499151	3	0,0345402	
4	0,0927949		4	0,056627	4	0,0499802	4	0,0508959	4	0,03599	
5	0,0919211		5	0,056082	5	0,0498052	5	0,0503221	5	0,035388	
6			6		6		6		6		
7	0,09132298		7	0,05712342	7	0,04993702	7	0,05056644	7	0,03646484	
EPSILON =0,000001										Iteraciones 500	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0559931		1	0,052705	1	0,0527842	1	0,053849	1	0,0520921	
2	0,0565639		2	0,054733	2	0,0556941	2	0,051851	2	0,0520282	
3	0,056251		3	0,054451	3	0,053226	3	0,0509801	3	0,0523829	
4	0,0544438		4	0,0543799	4	0,0529969	4	0,0534081	4	0,0547049	
5	0,0542672		5	0,0516131	5	0,0526462	5	0,0518389	5	0,0532999	
6			6		6		6		6		
7	0,0555038		7	0,0535764	7	0,05346948	7	0,05238542	7	0,0529016	

EPSILON =0,01										Iteraciones 10		
FUNCION : TRECCANI [-5,5][-5,5]												
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		N°	TIEMPO	
1	0,057703		1	0,0326109	1	0,0297089	1	0,0295119		1	0,0258532	
2	0,059454		2	0,0319211	2	0,029901	2	0,029953		2	0,0271201	
3	0,0572331		3	0,0327599	3	0,029665	3	0,0302491		3	0,026016	
4	0,0584531		4	0,0323169	4	0,0301969	4	0,0331409		4	0,027343	
5	0,0574429		5	0,0312941	5	0,0294871	5	0,0307429		5	0,0260699	
6			6		6		6			6		
7	0,05805722		7	0,03218058	7	0,02979178	7	0,03071956		7	0,02648044	
EPSILON =0,01										Iteraciones 50		
FUNCION : TRECCANI [-5,5][-5,5]												
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		N°	TIEMPO	
1	0,0565801		1	0,0335319	1	0,0381379	1	0,027992		1	0,0289102	
2	0,0569191		2	0,0330651	2	0,0403152	2	0,0281811		2	0,0277321	
3	0,0556009		3	0,0336308	3	0,0387609	3	0,0278161		3	0,0296409	
4	0,0564458		4	0,032824	4	0,03881	4	0,028641		4	0,0272079	
5	0,0591161		5	0,0333111	5	0,0385621	5	0,027195		5	0,0281682	
6			6		6		6			6		
7	0,0569324		7	0,03327258	7	0,03891722	7	0,02796504		7	0,02833186	
EPSILON =0,01										Iteraciones 100		
FUNCION : TRECCANI [-5,5][-5,5]												
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		N°	TIEMPO	
1	0,0562241		1	0,0346172	1	0,0318069	1	0,0282679		1	0,0268328	
2	0,0581141		2	0,036119	2	0,0329151	2	0,0283222		2	0,0264339	
3	0,055325		3	0,036422	3	0,032716	3	0,0278299		3	0,0258758	
4	0,262801		4	0,037415	4	0,032145	4	0,028625		4	0,0272789	
5	0,055393		5	0,0356588	5	0,0311489	5	0,0319891		5	0,0268419	
6			6		6		6			6		
7	0,09757144		7	0,0360464	7	0,03214638	7	0,02900682		7	0,02665266	
EPSILON =0,01										Iteraciones 500		
FUNCION : TRECCANI [-5,5][-5,5]												
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		N°	TIEMPO	
1	0,0415931		1	0,042866	1	0,039993	1	0,0406849		1	0,044863	
2	0,041317		2	0,0432131	2	0,04059	2	0,0413811		2	0,040123	
3	0,0422671		3	0,0390699	3	0,0394199	3	0,0418451		3	0,04335	
4	0,0420971		4	0,0410719	4	0,041749	4	0,043026		4	0,043756	
5	0,0423729		5	0,041223	5	0,0419431	5	0,0419269		5	0,044682	
6			6		6		6			6		
7	0,04192944		7	0,04148878	7	0,040739	7	0,0417728		7	0,0433548	
EPSILON =0,0001										Iteraciones 10		
FUNCION : TRECCANI [-5,5][-5,5]												
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		N°	TIEMPO	
1	0,112945		1	0,0601239	1	0,057199	1	0,05632		1	0,0517581	
2	0,113446		2	0,0600441	2	0,0582261	2	0,0567861		2	0,0519328	
3	0,117533		3	0,0625651	3	0,0579629	3	0,0575771		3	0,053947	
4	0,110852		4	0,0592649	4	0,0575101	4	0,055259		4	0,0514531	
5	0,114221		5	0,062413	5	0,056473	5	0,0576952		5	0,0515099	
6			6		6		6			6		
7	0,1137994		7	0,0608822	7	0,05747422	7	0,05672748		7	0,05212018	
EPSILON =0,0001										Iteraciones 50		
FUNCION : TRECCANI [-5,5][-5,5]												
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		N°	TIEMPO	
1	0,113005		1	0,061523	1	0,064337	1	0,0523241		1	0,0549822	
2	0,112998		2	0,0613661	2	0,065274	2	0,0542049		2	0,0512941	
3	0,111016		3	0,061219	3	0,0663409	3	0,050581		3	0,053334	
4	0,111296		4	0,061991	4	0,0630519	4	0,051656		4	0,0543928	
5	0,110974		5	0,0635679	5	0,0670488	5	0,0507131		5	0,053277	
6			6		6		6			6		
7	0,1118578		7	0,0619334	7	0,06521052	7	0,05189582		7	0,05345602	
EPSILON =0,0001										Iteraciones 100		
FUNCION : TRECCANI [-5,5][-5,5]												
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		N°	TIEMPO	
1	0,111449		1	0,0675871	1	0,059212	1	0,0550878		1	0,0523291	
2	0,110534		2	0,065043	2	0,0579541	2	0,0540431		2	0,051156	
3	0,114651		3	0,0655692	3	0,0597951	3	0,056093		3	0,051718	
4	0,10996		4	0,0645862	4	0,059386	4	0,0569491		4	0,052767	
5	0,105024		5	0,0680449	5	0,057502	5	0,054491		5	0,05142	
6			6		6		6			6		
7	0,1103236		7	0,06616608	7	0,05876984	7	0,0553328		7	0,05187802	
EPSILON =0,0001										Iteraciones 500		
FUNCION : TRECCANI [-5,5][-5,5]												
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		N°	TIEMPO	
1	0,0885601		1	0,0663829	1	0,0578661	1	0,058882		1	0,0571752	
2	0,087338		2	0,0663269	2	0,0580711	2	0,059052		2	0,0570271	
3	0,0857041		3	0,0662839	3	0,0584688	3	0,0571911		3	0,057189	
4	0,0875361		4	0,0658901	4	0,0585001	4	0,0595062		4	0,0572629	
5	0,088696		5	0,0666091	5	0,0580549	5	0,0608139		5	0,0572529	
6			6		6		6			6		
7	0,08756686		7	0,06629858	7	0,0581922	7	0,05908904		7	0,05718142	

EPSILON =0,000001 — **Iteraciones 10**

FUNCION : TRECCANI [-5,5][-5,5]

	N PRO, 2		N PRO, 3		N PRO, 4		N PRO, 5		N PRO, 6	
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,164129	1	0,0865998	1	0,0850081	1	0,086369	1	0,0752759
2	0,163742	2	0,0885551	2	0,082083	2	0,0833769	2	0,0781341
3	0,164755	3	0,0867929	3	0,0869339	3	0,081892	3	0,0797
4	0,17232	4	0,0879629	4	0,0823729	4	0,085279	4	0,0753632
5	0,167108	5	0,0913429	5	0,084744	5	0,081171	5	0,0771871
6		6		6		6		6	
7	0,1664108	7	0,08825072	7	0,08422838	7	0,08361758	7	0,07713206

EPSILON =0,000001 — **Iteraciones 50**

FUNCION : TRECCANI [-5,5][-5,5]

	N PRO, 2		N PRO, 3		N PRO, 4		N PRO, 5		N PRO, 6	
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,165986	1	0,0937879	1	0,088331	1	0,07529	1	0,078584
2	0,170785	2	0,0908821	2	0,0934401	2	0,0759282	2	0,0814559
3	0,172216	3	0,0919969	3	0,088424	3	0,0752962	3	0,0757451
4	0,167265	4	0,0908041	4	0,088578	4	0,0763459	4	0,0779312
5	0,164314	5	0,090724	5	0,0908539	5	0,0752831	5	0,075048
6		6		6		6		6	
7	0,1681132	7	0,091639	7	0,0899254	7	0,07562868	7	0,07775284

EPSILON =0,000001 — **Iteraciones 100**

FUNCION : TRECCANI [-5,5][-5,5]

	N PRO, 2		N PRO, 3		N PRO, 4		N PRO, 5		N PRO, 6	
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,163669	1	0,098326	1	0,084358	1	0,0789988	1	0,077703
2	0,166164	2	0,094445	2	0,088851	2	0,080704	2	0,0757382
3	0,161115	3	0,0979679	3	0,0844541	3	0,0792761	3	0,0770299
4	0,163653	4	0,093435	4	0,0856011	4	0,079339	4	0,07581
5	0,17202	5	0,0963249	5	0,0844629	5	0,0789571	5	0,0738511
6		6		6		6		6	
7	0,1653242	7	0,09609976	7	0,08554542	7	0,079455	7	0,07602644

EPSILON =0,000001 — **Iteraciones 500**

FUNCION : TRECCANI [-5,5][-5,5]

	N PRO, 2		N PRO, 3		N PRO, 4		N PRO, 5		N PRO, 6	
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,138948	1	0,096468	1	0,087604	1	0,0832241	1	0,0798609
2	0,139491	2	0,0974391	2	0,090065	2	0,0818119	2	0,082227
3	0,140605	3	0,0990329	3	0,095356	3	0,084796	3	0,0810151
4	0,14181	4	0,0957739	4	0,085959	4	0,082828	4	0,0822589
5	0,143207	5	0,095856	5	0,0878291	5	0,08375	5	0,0791769
6		6		6		6		6	
7	0,1408122	7	0,09691398	7	0,08936262	7	0,083282	7	0,08090776

EPSILON =0,01 — **Iteraciones 10**

FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]

	N PRO. 2		N PRO. 3		N PRO. 4		N PRO. 5		N PRO. 6	
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	5,93309	1	3,24707	1	2,55266	1	2,03394	1	1,96997
2	5,97023	2	3,36600	2	2,46885	2	2,14488	2	1,97807
3	5,98906	3	3,32586	3	2,58146	3	2,06614	3	1,97975
4	6,03802	4	3,22845	4	2,55882	4	2,02392	4	1,98195
5	5,96958	5	3,32799	5	2,50426	5	2,03519	5	1,93207
6		6		6		6		6	
7	5,9800	7	3,2991	7	2,5332	7	2,0608	7	1,9684

EPSILON =0,01 — **Iteraciones 50**

FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]

	N PRO. 2		N PRO. 3		N PRO. 4		N PRO. 5		N PRO. 6	
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	6,16942	1	4,02025	1	2,95244	1	3,11452	1	1,91186
2	6,05363	2	4,04273	2	3,14719	2	3,19118	2	1,97831
3	5,94940	3	4,06383	3	2,87260	3	3,15992	3	1,92760
4	5,97052	4	4,12563	4	2,99280	4	3,13725	4	1,91616
5	6,10959	5	4,04287	5	2,89713	5	3,12194	5	1,92733
6		6		6		6		6	
7	6,0505	7	4,0591	7	2,9724	7	3,1450	7	1,9323

EPSILON =0,01 — **Iteraciones 100**

FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]

	N PRO, 2		N PRO, 3		N PRO. 4		N PRO. 5		N PRO. 6	
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	6,17339	1	3,96922	1	2,92991	1	2,87507	1	2,06196
2	6,04414	2	4,01132	2	2,91044	2	2,85775	2	2,25656
3	6,17801	3	4,30543	3	2,85578	3	2,84698	3	2,05765
4	6,16857	4	3,99561	4	2,87411	4	2,86990	4	2,08161
5	6,03531	5	3,98320	5	2,93951	5	2,86112	5	2,10799
6		6		6		6		6	
7	6,1199	7	4,0530	7	2,9020	7	2,8622	7	2,1132

EPSILON =0,01 — **Iteraciones 500**

FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]

	N PRO. 2		N PRO. 3		N PRO. 4		N PRO. 5		N PRO. 6	
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	6,06163	1	3,75629	1	2,32020	1	1,91173	1	1,93331
2	5,93295	2	3,77575	2	2,29580	2	1,93601	2	1,96485
3	5,91686	3	3,68016	3	2,25905	3	1,90529	3	1,92023
4	5,84617	4	3,74108	4	2,39213	4	1,94583	4	1,96442
5	6,02982	5	3,77014	5	2,32340	5	1,92452	5	1,93082
6		6		6		6		6	
7	5,9575	7	3,7447	7	2,3181	7	1,9247	7	1,9427

EPSILON =0,0001										Iteraciones 10	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]											
	N PRO.	2	N PRO.	3	N PRO.	4	N PRO.	5	N PRO.	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	19,0562		1	11,5908	1	9,8364	1	7,8175	1	7,8214	
2	19,4870		2	11,5716	2	9,8727	2	7,8337	2	7,8084	
3	19,1318		3	11,5737	3	9,5823	3	7,8003	3	7,8179	
4	18,8879		4	11,7282	4	9,6400	4	7,6233	4	7,9088	
5	18,8270		5	11,7140	5	9,6612	5	7,6124	5	7,5686	
6			6		6		6		6		
7	19,0780		7	11,6357	7	9,7185	7	7,7374	7	7,7850	

EPSILON =0,0001										Iteraciones 50	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]											
	N PRO.	2	N PRO.	3	N PRO.	4	N PRO.	5	N PRO.	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	18,9102		1	12,2538	1	11,3437	1	11,4360	1	6,7861	
2	18,9774		2	12,2006	2	11,1642	2	11,4210	2	6,7287	
3	18,9898		3	12,0838	3	12,7840	3	11,4634	3	6,7907	
4	18,9902		4	12,1351	4	11,4058	4	11,4960	4	6,7004	
5	19,4157		5	12,3203	5	11,1270	5	11,4145	5	6,7566	
6			6		6		6		6		
7	19,0567		7	12,1987	7	11,5649	7	11,4462	7	6,7525	

EPSILON =0,0001										Iteraciones 100	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]											
	N PRO.	2	N PRO.	3	N PRO.	4	N PRO.	5	N PRO.	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	19,5039		1	13,8067	1	10,9260	1	10,7982	1	7,3634	
2	19,1085		2	13,8492	2	11,0937	2	10,8540	2	7,1638	
3	18,9072		3	13,7090	3	11,0034	3	10,9174	3	7,3878	
4	18,8438		4	13,9475	4	10,8344	4	10,8782	4	7,1633	
5	18,8540		5	13,8466	5	11,0992	5	11,0905	5	7,2213	
6			6		6		6		6		
7	19,0435		7	13,8318	7	10,9913	7	10,9077	7	7,2599	

EPSILON =0,0001										Iteraciones 500	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]											
	N PRO.	2	N PRO.	3	N PRO.	4	N PRO.	5	N PRO.	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	18,8394		1	10,8580	1	7,3574	1	5,8904	1	5,8517	
2	18,8220		2	10,5331	2	7,2290	2	6,0771	2	5,8872	
3	19,3759		3	10,6421	3	7,0959	3	5,9831	3	5,9447	
4	18,8449		4	10,6078	4	7,1028	4	5,8617	4	5,9018	
5	18,9583		5	10,5257	5	7,0156	5	5,8475	5	5,9132	
6			6		6		6		6		
7	18,9681		7	10,6333	7	7,1602	7	5,9319	7	5,8997	

EPSILON =0,000001										Iteraciones 10	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]											
	N PRO.	2	N PRO.	3	N PRO.	4	N PRO.	5	N PRO.	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	19,7148		1	11,6663	1	9,6701	1	7,8006	1	7,6563	
2	19,2347		2	11,8981	2	9,5933	2	7,6641	2	7,6339	
3	19,3261		3	12,1003	3	9,8561	3	7,6335	3	7,6714	
4	19,3035		4	11,7811	4	9,6605	4	7,8373	4	7,6527	
5	19,2498		5	11,9077	5	9,8825	5	7,6503	5	7,7754	
6			6		6		6		6		
7	19,3658		7	11,8707	7	9,7325	7	7,7171	7	7,6779	

EPSILON =0,000001										Iteraciones 50	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]											
	N PRO.	2	N PRO.	3	N PRO.	4	N PRO.	5	N PRO.	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	19,6885		1	12,7466	1	11,1606	1	11,4290	1	6,8558	
2	19,0540		2	12,5389	2	11,3185	2	11,4947	2	6,8301	
3	19,0492		3	12,3677	3	11,2447	3	11,8129	3	6,8343	
4	19,6895		4	12,3421	4	11,2214	4	11,4310	4	6,9829	
5	19,7790		5	12,9366	5	11,5024	5	11,8965	5	6,8494	
6			6		6		6		6		
7	19,4520		7	12,5864	7	11,2895	7	11,6128	7	6,8705	

EPSILON =0,000001										Iteraciones 100	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]											
	N PRO.	2	N PRO.	3	N PRO.	4	N PRO.	5	N PRO.	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	19,4576		1	14,1516	1	10,8834	1	10,8619	1	7,3673	
2	19,1957		2	13,8899	2	10,8119	2	11,1317	2	7,1516	
3	19,3355		3	13,8858	3	10,8383	3	10,9354	3	7,2173	
4	19,2700		4	13,8762	4	11,0849	4	11,0894	4	7,1976	
5	19,0833		5	13,9185	5	10,8965	5	10,8024	5	7,1665	
6			6		6		6		6		
7	19,2684		7	13,9444	7	10,9030	7	10,9642	7	7,2200	

EPSILON =0,000001										Iteraciones 500	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]											
	N PRO.	2	N PRO.	3	N PRO.	4	N PRO.	5	N PRO.	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	19,1932		1	10,7785	1	7,1526	1	5,8856	1	6,0379	
2	19,2035		2	10,7765	2	7,1164	2	6,0268	2	5,8784	
3	19,1707		3	10,7880	3	7,2027	3	5,9027	3	5,9270	
4	19,1821		4	11,0940	4	7,1596	4	5,9148	4	6,1915	
5	19,4220		5	10,7944	5	7,1950	5	6,0882	5	6,1915	
6			6		6		6		6		
7	19,2343		7	10,8463	7	7,1652	7	5,9636	7	6,0453	

DATOS ALGORITMO CAJAS RESULTADO CON UF FINAL Y BALANCEO DE CARGA DINAMICO ENFOCADO A LA DERECHA

EPSILON =0,01											Iteraciones 10	
ALGORTIMO CAJAS RESULTADO CON DISTRIBUCION EQULIBRADA												
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		
1	0,191666		1	0,154537	1	0,124859	1	0,125977	1	0,066674		
2	0,197871		2	0,156331	2	0,120082	2	0,126337	2	0,0659359		
3	0,198267		3	0,155283	3	0,12079	3	0,122063	3	0,066072		
4	0,191064		4	0,157147	4	0,120274	4	0,122106	4	0,0682352		
5	0,192933		5	0,154877	5	0,120173	5	0,122673	5	0,0655298		
6			6		6		6		6			
7	0,1943602		7	0,155635	7	0,1212356	7	0,1238312	7	0,06648938		
EPSILON =0,01											Iteraciones 50	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]												
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		
1	0,190452		1	0,126184	1	0,0872109	1	0,077781	1	0,0785739		
2	0,188485		2	0,125786	2	0,091677	2	0,0771241	2	0,0755899		
3	0,187705		3	0,125009	3	0,0907629	3	0,0767961	3	0,0786641		
4	0,187749		4	0,127689	4	0,0921321	4	0,0777831	4	0,0759568		
5	0,19522		5	0,123774	5	0,0876789	5	0,0774291	5	0,0765789		
6			6		6		6		6			
7	0,1899222		7	0,1256884	7	0,08989236	7	0,07738268	7	0,07707272		
EPSILON =0,01											Iteraciones 100	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]												
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		
1	0,191099		1	0,125995	1	0,0851049	1	0,086189	1	0,0709639		
2	0,186403		2	0,125642	2	0,086494	2	0,0813949	2	0,0712891		
3	0,18696		3	0,125134	3	0,087666	3	0,084408	3	0,0707591		
4	0,194046		4	0,125368	4	0,084913	4	0,0840931	4	0,0705149		
5	0,187408		5	0,124432	5	0,0854349	5	0,082808	5	0,0708091		
6			6		6		6		6			
7	0,1891832		7	0,1253142	7	0,08592256	7	0,0837786	7	0,07086722		
EPSILON =0,01											Iteraciones 500	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]												
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		
1	0,140764		1	0,139384	1	0,103868	1	0,096535	1	0,09799		
2	0,140179		2	0,114828	2	0,103096	2	0,096746	2	0,103093		
3	0,139657		3	0,111185	3	0,103322	3	0,104391	3	0,0966592		
4	0,139709		4	0,109619	4	0,102462	4	0,0989509	4	0,101344		
5	0,136225		5	0,11153	5	0,105588	5	0,0992019	5	0,10192		
6			6		6		6		6			
7	0,1393068		7	0,1173092	7	0,1036672	7	0,09916496	7	0,10020124		
EPSILON =0,0001											Iteraciones 10	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]												
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		
1	0,214573		1	0,17133	1	0,138052	1	0,140169	1	0,077013		
2	0,209672		2	0,177897	2	0,142556	2	0,146031	2	0,0752928		
3	0,208244		3	0,17151	3	0,138778	3	0,13884	3	0,0751829		
4	0,216602		4	0,177103	4	0,138341	4	0,142111	4	0,074342		
5	0,210057		5	0,172748	5	0,138911	5	0,143903	5	0,0745029		
6			6		6		6		6			
7	0,2118296		7	0,1741176	7	0,1393276	7	0,1422108	7	0,07526672		
EPSILON =0,0001											Iteraciones 50	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]												
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		
1	0,204553		1	0,137419	1	0,104382	1	0,086185	1	0,0936148		
2	0,205617		2	0,139648	2	0,108221	2	0,0869191	2	0,0941231		
3	0,206634		3	0,137309	3	0,104313	3	0,0882761	3	0,0954552		
4	0,212044		4	0,13395	4	0,1087	4	0,0859728	4	0,0927701		
5	0,212195		5	0,13362	5	0,104604	5	0,086288	5	0,0935829		
6			6		6		6		6			
7	0,2082086		7	0,1363892	7	0,106044	7	0,0867282	7	0,09390922		
EPSILON =0,0001											Iteraciones 100	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]												
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		
1	0,205226		1	0,141991	1	0,100587	1	0,100274	1	0,0700359		
2	0,20349		2	0,140982	2	0,0970929	2	0,103129	2	0,0704882		
3	0,205486		3	0,142409	3	0,0991101	3	0,0974939	3	0,072057		
4	0,205053		4	0,142267	4	0,0942328	4	0,103016	4	0,0707879		
5	0,204385		5	0,141841	5	0,0947192	5	0,100758	5	0,0712771		
6			6		6		6		6			
7	0,204728		7	0,141898	7	0,0971484	7	0,10093418	7	0,07092922		
EPSILON =0,0001											Iteraciones 500	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]												
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		
1	0,15733		1	0,127911	1	0,105669	1	0,102161	1	0,0997999		
2	0,156133		2	0,130312	2	0,10605	2	0,106058	2	0,0996199		
3	0,163019		3	0,131426	3	0,109221	3	0,104272	3	0,103077		
4	0,156622		4	0,129034	4	0,105391	4	0,103934	4	0,101949		
5	0,155898		5	0,130461	5	0,10563	5	0,103641	5	0,097713		
6			6		6		6		6			
7	0,1578004		7	0,1298288	7	0,1063922	7	0,1040132	7	0,10043176		

EPSILON =0,000001										Iteraciones 10	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°		
1	0,238084		1	0,190344		1	0,158834		1	0,159283		1	0,0851238	
2	0,227542		2	0,192365		2	0,156541		2	0,158102		2	0,0885341	
3	0,228678		3	0,190794		3	0,157515		3	0,160755		3	0,0839791	
4	0,237385		4	0,1918		4	0,157728		4	0,159697		4	0,0850511	
5	0,229419		5	0,193335		5	0,162651		5	0,159713		5	0,084801	
6			6			6			6			6		
7	0,2322216		7	0,1917276		7	0,1586538		7	0,15951		7	0,08549782	

EPSILON =0,000001										Iteraciones 50	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,225981		1	0,143251		1	0,123314		1	0,107417		1	0,112498	
2	0,223794		2	0,143589		2	0,122645		2	0,105709		2	0,11271	
3	0,233297		3	0,143546		3	0,122607		3	0,109152		3	0,115962	
4	0,223652		4	0,143021		4	0,121201		4	0,109577		4	0,111287	
5	0,224912		5	0,142853		5	0,125062		5	0,107344		5	0,112065	
6			6			6			6			6		
7	0,2263272		7	0,143252		7	0,1229658		7	0,1078398		7	0,1129044	

EPSILON =0,000001										Iteraciones 100	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°			N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,226666		1	0,160915		1	0,114415		1	0,119575		1	0,0799799	
2	0,222395		2	0,159645		2	0,122244		2	0,122908		2	0,082222	
3	0,219266		3	0,165444		3	0,115006		3	0,120118		3	0,0819449	
4	0,22341		4	0,164949		4	0,116023		4	0,118314		4	0,0851588	
5	0,225103		5	0,165874		5	0,115465		5	0,119857		5	0,0826778	
6			6			6			6			6		
7	0,223368		7	0,1633654		7	0,1166306		7	0,1201544		7	0,08239668	

EPSILON =0,000001										Iteraciones 500	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,175438		1	0,172457		1	0,124805		1	0,127566		1	0,120911	
2	0,181041		2	0,147562		2	0,127241		2	0,121631		2	0,11835	
3	0,176978		3	0,143412		3	0,122235		3	0,122364		3	0,122343	
4	0,168505		4	0,182938		4	0,125295		4	0,125797		4	0,118908	
5	0,179661		5	0,150404		5	0,123119		5	0,120926		5	0,117546	
6			6			6			6			6		
7	0,1763246		7	0,1593546		7	0,124539		7	0,1236568		7	0,1196116	

EPSILON =0,01										Iteraciones 10	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,080215		1	0,0767131		1	0,227767		1	0,22898		1	0,225394	
2	0,080894		2	0,0780909		2	0,233764		2	0,22412		2	0,226023	
3	0,0818501		3	0,0779181		3	0,234963		3	0,222941		3	0,226303	
4	0,082022		4	0,0770121		4	0,230077		4	0,226022		4	0,227459	
5	0,081928		5	0,076762		5	0,234186		5	0,222548		5	0,225229	
6			6			6			6			6		
7	0,08138182		7	0,07729924		7	0,2321514		7	0,2249222		7	0,2260816	

EPSILON =0,01										Iteraciones 50	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,0767729		1	0,0736248		1	0,06669		1	0,059442		1	0,065387	
2	0,078263		2	0,0732942		2	0,065737		2	0,060405		2	0,064544	
3	0,0761461		3	0,0736079		3	0,066205		3	0,058238		3	0,066046	
4	0,0785072		4	0,07212		4	0,0668371		4	0,058794		4	0,065444	
5	0,077795		5	0,074234		5	0,0728621		5	0,05918		5	0,065325	
6			6			6			6			6		
7	0,07749684		7	0,07337618		7	0,06766624		7	0,0592118		7	0,0653492	

EPSILON =0,01										Iteraciones 100	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,0726631		1	0,0661571		1	0,0606508		1	0,064883		1	0,05181	
2	0,0717089		2	0,065464		2	0,0617492		2	0,0657172		2	0,052866	
3	0,070981		3	0,0648131		3	0,059037		3	0,0661399		3	0,053674	
4	0,0726068		4	0,0658691		4	0,0613461		4	0,0641859		4	0,052197	
5	EL TIEMPO DE E		5	0,06552		5	0,060735		5	0,0668249		5	0,0534909	
6			6			6			6			6		
7	0,07198995		7	0,06556466		7	0,06070362		7	0,06555018		7	0,05280758	

EPSILON =0,01										Iteraciones 500	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,0684171		1	0,0696571		1	0,070745		1	0,070483		1	0,0742209	
2	0,0686121		2	0,0710602		2	0,0694339		2	0,072598		2	0,07197	
3	0,068361		3	0,0700259		3	0,0718269		3	0,071898		3	0,0710709	
4	0,0692251		4	0,0702369		4	0,068295		4	0,071146		4	0,072351	
5	0,0684171		5	0,069165		5	0,072017		5	0,072583		5	0,072325	
6			6			6			6			6		
7	0,06860648		7	0,07002902		7	0,07046356		7	0,0717416		7	0,07238756	

EPSILON =0,0001										Iteraciones 10	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,1405		1	0,137987	1	0,324696	1	0,317535	1	0,323354	
2	0,139596		2	0,137167	2	0,331438	2	0,318845	2	0,333222	
3	0,140956		3	0,141562	3	0,322179	3	0,328127	3	0,322436	
4	0,144038		4	0,138147	4	0,324843	4	0,318848	4	0,322982	
5	0,14036		5	0,141407	5	0,323267	5	0,319559	5	0,330362	
6			6		6		6		6		
7	0,14109		7	0,139254	7	0,3252846	7	0,3205828	7	0,3264712	

EPSILON =0,0001										Iteraciones 50	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,139966		1	0,131327	1	0,116387	1	0,118529	1	0,125581	
2	0,136513		2	0,132664	2	0,116464	2	0,119145	2	0,124996	
3	0,138105		3	0,134443	3	0,117505	3	0,117561	3	0,124706	
4	0,136506		4	0,135084	4	0,116796	4	0,118644	4	0,128644	
5	0,136396		5	0,131677	5	0,119387	5	0,119429	5	0,124783	
6			6		6		6		6		
7	0,1374972		7	0,133039	7	0,1173078	7	0,1186616	7	0,125742	

EPSILON =0,0001										Iteraciones 100	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,131358		1	0,124082	1	0,121835	1	0,125423	1	0,11369	
2	0,131757		2	0,126068	2	0,118907	2	0,125825	2	0,112678	
3	0,131522		3	0,126657	3	0,121073	3	0,128726	3	0,113036	
4	0,134416		4	0,124311	4	0,118899	4	0,126955	4	0,114394	
5	0,132215		5	0,124178	5	0,118124	5	0,125957	5	0,112532	
6			6		6		6		6		
7	0,1322536		7	0,1250592	7	0,1197676	7	0,1265772	7	0,113266	

EPSILON =0,0001										Iteraciones 500	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0802629		1	0,078593	1	0,0799999	1	0,0754139	1	0,076117	
2	0,081182		2	0,0808961	2	0,0797579	2	0,0754669	2	0,0744741	
3	0,0821609		3	0,077925	3	0,07862	3	0,0756621	3	0,0759408	
4	0,079504		4	0,078624	4	0,078289	4	0,0759568	4	0,0745611	
5	0,0822818		5	0,0805402	5	0,08003	5	0,0747879	5	0,0765851	
6			6		6		6		6		
7	0,08107832		7	0,07931566	7	0,07933936	7	0,07545752	7	0,07553562	

EPSILON =0,000001										Iteraciones 10	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,203695		1	0,200358	1	0,33157	1	0,319316	1	0,321285	
2	0,206395		2	0,205159	2	0,331257	2	0,316817	2	0,320988	
3	0,201		3	0,198577	3	0,33374	3	0,320438	3	0,330797	
4	0,200152		4	0,200339	4	0,331522	4	0,319294	4	0,320328	
5	0,20581		5	0,204801	5	0,326075	5	0,327682	5	0,320746	
6			6		6		6		6		
7	0,2034104		7	0,2018468	7	0,3308328	7	0,3207094	7	0,3228288	

EPSILON =0,000001										Iteraciones 50	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°			N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,198322		1	0,191699	1	0,182879	1	0,178041	1	0,189364	
2	0,204462		2	0,192554	2	0,176083	2	0,177581	2	0,185708	
3	0,198782		3	0,190508	3	0,176952	3	0,178093	3	0,185836	
4	0,203031		4	0,190037	4	0,177251	4	0,177868	4	0,189359	
5	0,202254		5	0,19545	5	0,176187	5	0,179131	5	0,186677	
6			6		6		6		6		
7	0,2013702		7	0,1920496	7	0,1778704	7	0,1781428	7	0,1873888	

EPSILON =0,000001										Iteraciones 100	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,192038		1	0,183009	1	0,182943	1	0,186343	1	0,174267	
2	0,197613		2	0,182915	2	0,17714	2	0,19002	2	0,178862	
3	0,192863		3	0,18433	3	0,17842	3	0,187079	3	0,173507	
4	0,193903		4	0,184637	4	0,17776	4	0,191213	4	0,175332	
5	0,19733		5	0,191042	5	0,179594	5	0,186215	5	0,178298	
6			6		6		6		6		
7	0,1947494		7	0,1851866	7	0,1791714	7	0,188174	7	0,1760532	

EPSILON =0,000001										Iteraciones 500	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,144153		1	0,14065	1	0,137684	1	0,137956	1	0,129188	
2	0,142317		2	0,141407	2	0,13851	2	0,134597	2	0,128808	
3	0,143842		3	0,143004	3	0,138546	3	0,132841	3	0,126939	
4	0,146649		4	0,141044	4	0,144679	4	0,133173	4	0,128545	
5	0,145228		5	0,137759	5	0,14139	5	0,133419	5	0,126465	
6			6		6		6		6		
7	0,1444378		7	0,1407728	7	0,1401618	7	0,1343972	7	0,127989	

EPSILON =0,01										Iteraciones 10		
FUNCION : BOOTH [-10,10][-10,10]												
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		N°	TIEMPO	
1	0,018533		1	0,0183589	1	0,0184541	1	0,019742		1	0,0142162	
2	0,018187		2	0,018374	2	0,0181451	2	0,0190251		2	0,0146861	
3	0,0182779		3	0,018225	3	0,017988	3	0,0196371		3	0,0141721	
4	0,0179212		4	0,0176542	4	0,0181	4	0,019665		4	0,0140328	
5	0,018435		5	0,01824	5	0,0177791	5	0,0194421		5	0,014374	
6			6		6		6			6		
7	0,01827082		7	0,01817042	7	0,01809326	7	0,01950226		7	0,01429624	

EPSILON =0,01										Iteraciones 50		
FUNCION : BOOTH [-10,10][-10,10]												
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		N°	TIEMPO	
1	0,015234		1	0,01612	1	0,01508	1	0,015475		1	0,015939	
2	0,0151742		2	0,0155718	2	0,0156901	2	0,015379		2	0,0167332	
3	0,014466		3	0,0157599	3	0,015599	3	0,015249		3	0,0164959	
4	0,015507		4	0,016078	4	0,015105	4	0,014854		4	0,0160701	
5	0,014348		5	0,0162091	5	0,0150108	5	0,014981		5	0,0172181	
6			6		6		6			6		
7	0,01494584		7	0,01594776	7	0,01529698	7	0,0151876		7	0,01649126	

EPSILON =0,01										Iteraciones 100		
FUNCION : BOOTH [-10,10][-10,10]												
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		N°	TIEMPO	
1	0,0108509		1	0,0121212	1	0,0127692	1	0,0135229		1	0,014487	
2	0,0114501		2	0,01193	2	0,012661	2	0,0140259		2	0,011549	
3	0,0114181		3	0,0123591	3	0,0117831	3	0,0135889		3	0,0147321	
4	0,0108991		4	0,0122149	4	0,0120928	4	0,0133569		4	0,014184	
5	0,0110128		5	0,011997	5	0,0126679	5	0,013994		5	0,014226	
6			6		6		6			6		
7	0,0111262		7	0,01212444	7	0,0123948	7	0,01369772		7	0,01383562	

EPSILON =0,01										Iteraciones 500		
FUNCION : BOOTH [-10,10][-10,10]												
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		N°	TIEMPO	
1	0,0340822		1	0,034719	1	0,0362248	1	0,037276		1	0,0354478	
2	0,0322409		2	0,0338058	2	0,0362189	2	0,03392		2	0,0381799	
3	0,034302		3	0,0342391	3	0,035598	3	0,0360901		3	0,034857	
4	0,0340569		4	0,034194	4	0,0348461	4	0,0360749		4	0,0370841	
5	0,033782		5	0,034106	5	0,0353549	5	0,0363581		5	0,037375	
6			6		6		6			6		
7	0,0336928		7	0,03421278	7	0,03564854	7	0,03594382		7	0,03658876	

EPSILON =0,0001										Iteraciones 10		
FUNCION : BOOTH [-10,10][-10,10]												
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		N°	TIEMPO	
1	0,029104		1	0,0290191	1	0,0291939	1	0,0304132		1	0,0206268	
2	0,0289562		2	0,0299518	2	0,0284338	2	0,0297658		2	0,0202839	
3	0,029381		3	0,029489	3	0,0288012	3	0,0298359		3	0,019918	
4	0,028698		4	0,029603	4	0,0289629	4	0,030252		4	0,019866	
5	0,0288651		5	0,02916	5	0,029134	5	0,030755		5	0,0200131	
6			6		6		6			6		
7	0,02900086		7	0,02944458	7	0,02890516	7	0,03020438		7	0,02014156	

EPSILON =0,0001										Iteraciones 50		
FUNCION : BOOTH [-10,10][-10,10]												
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		N°	TIEMPO	
1	0,0269139		1	0,0266562	1	0,0262341	1	0,026324		1	0,0270562	
2	0,0260298		2	0,0263431	2	0,0257089	2	0,026814		2	0,0271571	
3	0,0264091		3	0,0273161	3	0,0266101	3	0,0260491		3	0,028115	
4	0,025948		4	0,0263941	4	0,0268588	4	0,026484		4	0,0263121	
5	0,0262022		5	0,026185	5	0,026499	5	0,0267642		5	0,0272038	
6			6		6		6			6		
7	0,0263006		7	0,0265789	7	0,02638218	7	0,02648706		7	0,02716884	

EPSILON =0,0001										Iteraciones 100		
FUNCION : BOOTH [-10,10][-10,10]												
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		N°	TIEMPO	
1	0,0222549		1	0,0215569	1	0,02229	1	0,0228479		1	0,019428	
2	0,0224118		2	0,0215809	2	0,0229361	2	0,0230861		2	0,0188949	
3	0,021574		3	0,0219791	3	0,0220101	3	0,0232201		3	0,0188189	
4	0,0226841		4	0,021517	4	0,0226221	4	0,024472		4	0,0196521	
5	0,0217988		5	0,0220449	5	0,022717	5	0,023267		5	0,018811	
6			6		6		6			6		
7	0,02214472		7	0,02173576	7	0,02251506	7	0,02337862		7	0,01912098	

EPSILON =0,0001										Iteraciones 500		
FUNCION : BOOTH [-10,10][-10,10]												
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		N°	TIEMPO	
1	0,0341139		1	0,0353949	1	0,037899	1	0,0352731		1	0,0383129	
2	0,0333459		2	0,034533	2	0,0355141	2	0,0362899		2	0,0387061	
3	0,0330899		3	0,034936	3	0,0354869	3	0,036056		3	0,0380249	
4	0,0333419		4	0,035351	4	0,035625	4	0,0367892		4	0,037075	
5	0,0332921		5	0,034059	5	0,0375099	5	0,0370901		5	0,0377791	
6			6		6		6			6		
7	0,03343674		7	0,03485478	7	0,03640698	7	0,03629966		7	0,0379796	

EPSILON =0,000001									Iteraciones 10	
FUNCION : BOOTH [-10,10][-10,10]										
	N PRO,	2		N PRO,	3		N PRO,	4	N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO
1	0,040653		1	0,0407839		1	0,040122		1	0,0434299		1	0,0267169
2	0,0403659		2	0,0407159		2	0,0416949		2	0,0422189		2	0,02672
3	0,0400929		3	0,040113		3	0,0408971		3	0,042079		3	0,0260088
4	0,0398829		4	0,039993		4	0,0409062		4	0,0417709		4	0,0259089
5	0,040648		5	0,0395482		5	0,0403831		5	0,0425599		5	0,0261021
6			6			6			6			6	
7	0,04032854		7	0,0402308		7	0,04080066		7	0,04241172		7	0,02629134

EPSILON =0,000001									Iteraciones 50	
FUNCION : BOOTH [-10,10][-10,10]										
	N PRO,	2		N PRO,	3		N PRO,	4	N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO
1	0,0370431		1	0,0388999		1	0,0376718		1	0,0374339		1	0,038846
2	0,0372982		2	0,037677		2	0,038197		2	0,0378239		2	0,0388691
3	0,0364799		3	0,0372279		3	0,038214		3	0,0381429		3	0,037514
4	0,0369542		4	0,0372479		4	0,037349		4	0,037776		4	0,0395269
5	0,0376821		5	0,036289		5	0,0374551		5	0,037308		5	0,0385859
6			6			6			6			6	
7	0,0370915		7	0,03746834		7	0,03777738		7	0,03769694		7	0,03866838

EPSILON =0,000001									Iteraciones 100	
FUNCION : BOOTH [-10,10][-10,10]										
	N PRO,	2		N PRO,	3		N PRO,	4	N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO
1	0,032814		1	0,0329509		1	0,0333059		1	0,0347581		1	0,0247228
2	0,0336161		2	0,03317		2	0,0343671		2	0,0350668		2	0,0246069
3	0,0326939		3	0,0321441		3	0,0336649		3	0,0354779		3	0,0252092
4	0,033093		4	0,0328219		4	0,0337341		4	0,0347011		4	0,0251229
5	0,033112		5	0,032685		5	0,0338302		5	0,0345912		5	0,0268631
6			6			6			6			6	
7	0,0330658		7	0,03275438		7	0,03378044		7	0,03491902		7	0,02530498

EPSILON =0,000001									Iteraciones 500	
FUNCION : BOOTH [-10,10][-10,10]										
	N PRO,	2		N PRO,	3		N PRO,	4	N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO
1	0,032707		1	0,036171		1	0,0345421		1	0,0360911		1	0,0365598
2	0,033987		2	0,0355461		2	0,0350959		2	0,0378542		2	0,0370231
3	0,0341539		3	0,033906		3	0,0364642		3	0,035881		3	0,0368741
4	0,034246		4	0,0325131		4	0,0353441		4	0,035363		4	0,0375922
5	0,0333002		5	0,0356698		5	0,0339661		5	0,0362971		5	0,0361309
6			6			6			6			6	
7	0,03367882		7	0,0347612		7	0,03508248		7	0,03629728		7	0,03683602

EPSILON =0,01									Iteraciones 10	
FUNCION : MATYAS [-10,10][-10,10]										
	N PRO,	2		N PRO,	3		N PRO,	4	N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO
1	0,0592008		1	0,0315869		1	0,029875		1	0,031487		1	0,0284441
2	0,05582		2	0,031683		2	0,0301628		2	0,0313001		2	0,0275302
3	0,056031		3	0,031791		3	0,0298719		3	0,0311		3	0,02775
4	0,0558901		4	0,0318019		4	0,0303309		4	0,0313671		4	0,0281711
5	0,0552511		5	0,0321651		5	0,0297408		5	0,0317101		5	0,028115
6			6			6			6			6	
7	0,0564386		7	0,03180558		7	0,02999628		7	0,03139286		7	0,02800208

EPSILON =0,01									Iteraciones 50	
FUNCION : MATYAS [-10,10][-10,10]										
	N PRO,	2		N PRO,	3		N PRO,	4	N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO
1	0,053241		1	0,047766		1	0,046593		1	0,0476758		1	0,028291
2	0,0552762		2	0,0480528		2	0,047729		2	0,0458729		2	0,0279539
3	0,0554819		3	0,047904		3	0,0469582		3	0,046592		3	0,0290849
4	0,0568559		4	0,0493281		4	0,04603		4	0,04652		4	0,0276241
5	0,056365		5	0,048475		5	0,0467		5	0,0462611		5	0,0280001
6			6			6			6			6	
7	0,055444		7	0,04830518		7	0,04680204		7	0,04658436		7	0,0281908

EPSILON =0,01									Iteraciones 100	
FUNCION : MATYAS [-10,10][-10,10]										
	N PRO,	2		N PRO,	3		N PRO,	4	N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO
1	0,0527258		1	0,0473931		1	0,0493588		1	0,0484581		1	0,034514
2	0,053889		2	0,047241		2	0,049865		2	0,0479419		2	0,0346808
3	0,054249		3	0,049705		3	0,0507209		3	0,0478621		3	0,0341241
4	0,0542569		4	0,0499778		4	0,051791		4	0,0482171		4	0,034621
5	0,0529518		5	0,048481		5	0,0504291		5	0,0492949		5	0,0344341
6			6			6			6			6	
7	0,0536145		7	0,04855958		7	0,05043296		7	0,04835482		7	0,0344748

EPSILON =0,01									Iteraciones 500	
FUNCION : MATYAS [-10,10][-10,10]										
	N PRO,	2		N PRO,	3		N PRO,	4	N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO
1	0,0360529		1	0,037225		1	0,0373211		1	0,039696		1	0,0384231
2	0,0359511		2	0,0377531		2	0,035959		2	0,0367939		2	0,0409009
3	0,0359631		3	0,0371399		3	0,0386579		3	0,0388958		3	0,040221
4	0,0362089		4	0,0365839		4	0,0361431		4	0,039315		4	0,038316
5	0,0377212		5	0,03775		5	0,0366831		5	0,038137		5	0,038753
6			6			6			6			6	
7	0,03637944		7	0,03729038		7	0,03695284		7	0,03856754		7	0,0393228

EPSILON =0,0001 — Iteraciones 10

FUNCION : MATYAS [-10,10][-10,10]

N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO
1	0,104212	1	0,0566189	1	0,053436	1	0,0557451	1	0,0500629
2	0,102888	2	0,0560601	2	0,0547948	2	0,054112	2	0,0503809
3	0,106883	3	0,0554681	3	0,053993	3	0,0568409	3	0,0505049
4	0,105879	4	0,056437	4	0,0530519	4	0,0555871	4	0,049377
5	0,104044	5	0,056	5	0,054378	5	0,056103	5	0,049758
6		6		6		6		6	
7	0,1047812	7	0,05611682	7	0,05393074	7	0,05567762	7	0,05001674

EPSILON =0,0001 — Iteraciones 50

FUNCION : MATYAS [-10,10][-10,10]

N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO
1	0,105271	1	0,0942628	1	0,0936639	1	0,0960882	1	0,0519412
2	0,105759	2	0,0965781	2	0,095782	2	0,095165	2	0,055464
3	0,104906	3	0,0945461	3	0,095124	3	0,0947831	3	0,052135
4	0,102268	4	0,095211	4	0,0944209	4	0,0935221	4	0,054589
5	0,106071	5	0,094898	5	0,0936301	5	0,097054	5	0,055403
6		6		6		6		6	
7	0,104855	7	0,0950992	7	0,09452418	7	0,09532248	7	0,05390644

EPSILON =0,0001 — Iteraciones 100

FUNCION : MATYAS [-10,10][-10,10]

N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO
1	0,101847	1	0,098676	1	0,0957789	1	0,0996768	1	0,0587411
2	0,101677	2	0,0953009	2	0,0978758	2	0,0946121	2	0,0604091
3	0,101962	3	0,0951741	3	0,0965841	3	0,098489	3	0,0602131
4	0,104855	4	0,098053	4	0,0981491	4	0,103484	4	0,0602262
5	0,103778	5	0,0959389	5	0,097163	5	0,094681	5	0,0594392
6		6		6		6		6	
7	0,1028238	7	0,09662858	7	0,09711018	7	0,09818858	7	0,05980574

EPSILON =0,0001 — Iteraciones 500

FUNCION : MATYAS [-10,10][-10,10]

N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO
1	0,0769489	1	0,070962	1	0,073195	1	0,055774	1	0,056818
2	0,0764859	2	0,0704801	2	0,0731959	2	0,0553999	2	0,0579839
3	0,0792401	3	0,0706639	3	0,0730488	3	0,0567861	3	0,0567291
4	0,075789	4	0,0695951	4	0,0731368	4	0,0557768	4	0,0571051
5	0,075386	5	0,0705051	5	0,0729501	5	0,0563812	5	0,057313
6		6		6		6		6	
7	0,07676998	7	0,07044124	7	0,07310532	7	0,0560236	7	0,05718982

EPSILON =0,000001 — Iteraciones 10

FUNCION : MATYAS [-10,10][-10,10]

N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO
1	0,154948	1	0,087769	1	0,078826	1	0,0812149	1	0,0717719
2	0,15222	2	0,079736	2	0,0798619	2	0,0824878	2	0,072603
3	0,154034	3	0,0793099	3	0,0782819	3	0,080235	3	0,0728481
4	0,155253	4	0,0807209	4	0,0788341	4	0,080034	4	0,0728772
5	0,152599	5	0,079231	5	0,0794649	5	0,0823441	5	0,0730751
6		6		6		6		6	
7	0,1538108	7	0,08135336	7	0,07905376	7	0,08126316	7	0,07263506

EPSILON =0,000001 — Iteraciones 50

FUNCION : MATYAS [-10,10][-10,10]

N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO
1	0,154101	1	0,142191	1	0,181481	1	0,143571	1	0,078805
2	0,154524	2	0,142457	2	0,142088	2	0,1436	2	0,0789609
3	0,151359	3	0,141237	3	0,143685	3	0,143861	3	0,0791938
4	0,149495	4	0,143216	4	0,141782	4	0,143027	4	0,0790842
5	0,15035	5	0,141232	5	0,141123	5	0,143133	5	0,079881
6		6		6		6		6	
7	0,1519658	7	0,1420666	7	0,1500318	7	0,1434384	7	0,07918498

EPSILON =0,000001 — Iteraciones 100

FUNCION : MATYAS [-10,10][-10,10]

N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO
1	0,150349	1	0,144222	1	0,147384	1	0,145158	1	0,086832
2	0,148873	2	0,140677	2	0,149044	2	0,14266	2	0,0865939
3	0,148546	3	0,144279	3	0,146508	3	0,14456	3	0,0851421
4	0,148666	4	0,148632	4	0,147755	4	0,142505	4	0,084523
5	0,150023	5	0,143607	5	0,144941	5	0,146234	5	0,0836489
6		6		6		6		6	
7	0,1492914	7	0,1442834	7	0,1471264	7	0,1442234	7	0,08534798

EPSILON =0,000001 — Iteraciones 500

FUNCION : MATYAS [-10,10][-10,10]

N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO	N°	N PRO, TIEMPO
1	0,125022	1	0,119147	1	0,119523	1	0,079926	1	0,0827229
2	0,124447	2	0,12445	2	0,124916	2	0,0803301	2	0,081624
3	0,124176	3	0,116725	3	0,120201	3	0,080699	3	0,0811219
4	0,124221	4	0,117406	4	0,118469	4	0,0818038	4	0,081188
5	0,123165	5	0,118301	5	0,121624	5	0,0797162	5	0,0817931
6		6		6		6		6	
7	0,1242062	7	0,1192058	7	0,1209466	7	0,08049502	7	0,08168998

EPSILON =0,01										Iteraciones 10	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0851331		1	0,047859	1	0,0433371	1	0,044348	1	0,0440829	
2	0,0843341		2	0,049022	2	0,0430729	2	0,043468	2	0,0443671	
3	0,0869269		3	0,0481761	3	0,043247	3	0,043263	3	0,044184	
4	0,0843749		4	0,0482311	4	0,04421	4	0,0442359	4	0,0442629	
5	0,0862381		5	0,0483172	5	0,0429399	5	0,0437829	5	0,043838	
6			6		6		6		6		
7	0,08540142		7	0,04832108	7	0,04336138	7	0,04381956	7	0,04414698	

EPSILON =0,01										Iteraciones 50	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,081269		1	0,047657	1	0,0449331	1	0,0418851	1	0,0428121	
2	0,084342		2	0,047343	2	0,045856	2	0,0410311	2	0,042625	
3	0,0822549		3	0,0483079	3	0,047076	3	0,0407791	3	0,0434148	
4	0,08181		4	0,050101	4	0,0457771	4	0,04248	4	0,0425081	
5	0,079766		5	0,0491478	5	0,0452859	5	0,045419	5	0,0423331	
6			6		6		6		6		
7	0,08188838		7	0,04851134	7	0,04578562	7	0,04231886	7	0,04273862	

EPSILON =0,01										Iteraciones 100	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0813251		1	0,050416	1	0,0446661	1	0,044595	1	0,0396311	
2	0,0800679		2	0,0501029	2	0,0457499	2	0,0432479	2	0,039655	
3	0,076113		3	0,0496018	3	0,0449998	3	0,0444829	3	0,039778	
4	0,0794339		4	0,04809	4	0,0437739	4	0,043684	4	0,040504	
5	0,0813088		5	0,049237	5	0,0451391	5	0,0433378	5	0,0388882	
6			6		6		6		6		
7	0,07964974		7	0,04948954	7	0,04486576	7	0,04386952	7	0,03969126	

EPSILON =0,01										Iteraciones 500	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0508969		1	0,054467	1	0,052814	1	0,0546031	1	0,0547421	
2	0,0515649		2	0,0530221	2	0,0535741	2	0,0545831	2	0,055527	
3	0,0509601		3	0,0518148	3	0,0533559	3	0,0536609	3	0,054575	
4	0,0504651		4	0,0547121	4	0,0533948	4	0,054122	4	0,0548639	
5	0,0510991		5	0,0516059	5	0,0516748	5	0,0544989	5	0,0570831	
6			6		6		6		6		
7	0,05099722		7	0,05312438	7	0,05296272	7	0,0542936	7	0,05535822	

EPSILON =0,0001										Iteraciones 10	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0987329		1	0,0597351	1	0,0485818	1	0,04653	1	0,049201	
2	0,0951629		2	0,056144	2	0,0477939	2	0,0474691	2	0,048012	
3	0,098428		3	0,0589252	3	0,0465529	3	0,04724	3	0,0483088	
4	0,0968161		4	0,0562539	4	0,0475249	4	0,0473449	4	0,0490911	
5	0,0973761		5	0,0568321	5	0,0468681	5	0,046638	5	0,048934	
6			6		6		6		6		
7	0,0973032		7	0,05757806	7	0,04746432	7	0,0470444	7	0,04870938	

EPSILON =0,0001										Iteraciones 50	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0964818		1	0,0548549	1	0,053515	1	0,0502331	1	0,0525601	
2	0,0941801		2	0,055593	2	0,0532148	2	0,0502322	2	0,053252	
3	0,0959201		3	0,056031	3	0,0560751	3	0,053453	3	0,0516272	
4	0,0937228		4	0,05404	4	0,0547118	4	0,050225	4	0,0516448	
5	0,0942221		5	0,0556619	5	0,054085	5	0,0504708	5	0,0512609	
6			6		6		6		6		
7	0,09490538		7	0,05523616	7	0,05432034	7	0,05092282	7	0,052069	

EPSILON =0,0001										Iteraciones 100	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,091553		1	0,0570388	1	0,0495992	1	0,051661	1	0,040365	
2	0,091192		2	0,0568831	2	0,0491202	2	0,050611	2	0,0407588	
3	0,0932		3	0,056637	3	0,05159	3	0,0545771	3	0,041373	
4	0,0930748		4	0,0576129	4	0,049618	4	0,0511069	4	0,0416589	
5	0,0910289		5	0,0567942	5	0,049042	5	0,050971	5	0,0407059	
6			6		6		6		6		
7	0,09200974		7	0,0569932	7	0,04979388	7	0,0517854	7	0,04097232	

EPSILON =0,0001										Iteraciones 500	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	0,0524039		1	0,0524309	1	0,0529859	1	0,053885	1	0,0537691	
2	0,0512941		2	0,0516629	2	0,0526779	2	0,054379	2	0,055598	
3	0,0531371		3	0,052217	3	0,054199	3	0,053525	3	0,0538092	
4	0,0529771		4	0,052866	4	0,0531299	4	0,056762	4	0,0551448	
5	0,0509231		5	0,0524421	5	0,0543358	5	0,0528281	5	0,05514	
6			6		6		6		6		
7	0,05214706		7	0,05232378	7	0,0534657	7	0,05427582	7	0,05469222	

EPSILON =0,000001									Iteraciones 10	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]										
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,111778		1	0,068146	1	0,0517361	1	0,0508931	1	0,052767
2	0,10813		2	0,0678689	2	0,0503399	2	0,051662	2	0,0514438
3	0,109573		3	0,066566	3	0,0504131	3	0,050947	3	0,0517459
4	0,11288		4	0,0670629	4	0,0502281	4	0,050724	4	0,052444
5	0,107815		5	0,0682909	5	0,05075	5	0,05092	5	0,051707
6			6		6		6		6	
7	0,1100352		7	0,06758694	7	0,05069344	7	0,05102922	7	0,05202154
EPSILON =0,000001									Iteraciones 50	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]										
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,104581		1	0,0621939	1	0,0562689	1	0,0503151	1	0,0509231
2	0,106278		2	0,061626	2	0,054599	2	0,050889	2	0,0528381
3	0,10483		3	0,0622811	3	0,0540509	3	0,051312	3	0,0525
4	0,105003		4	0,0640481	4	0,0552051	4	0,049773	4	0,0512271
5	0,110041		5	0,0631461	5	0,0556841	5	0,051548	5	0,0517812
6			6		6		6		6	
7	0,1061466		7	0,06265904	7	0,0551616	7	0,05076742	7	0,0518539
EPSILON =0,000001									Iteraciones 100	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]										
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,106029		1	0,0653791	1	0,057379	1	0,0636029	1	0,0434749
2	0,103131		2	0,0651062	2	0,058352	2	0,0583889	2	0,0426559
3	0,105413		3	0,0640211	3	0,0573649	3	0,058223	3	0,0421519
4	0,103219		4	0,0641782	4	0,057198	4	0,0584021	4	0,0427351
5	0,103021		5	0,0642478	5	0,057235	5	0,0589068	5	0,0426669
6			6		6		6		6	
7	0,1041626		7	0,06458648	7	0,05750578	7	0,05950474	7	0,04273694
EPSILON =0,000001									Iteraciones 500	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]										
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,0565221		1	0,055603	1	0,0556421	1	0,0579782	1	0,0564742
2	0,0560989		2	0,0550549	2	0,0579941	2	0,056066	2	0,056807
3	0,0579541		3	0,0556312	3	0,0573611	3	0,054848	3	0,0568402
4	0,0581789		4	0,0557358	4	0,0563369	4	0,0558851	4	0,058949
5	0,0563231		5	0,0555611	5	0,056226	5	0,055388	5	0,058713
6			6		6		6		6	
7	0,05701542		7	0,0555172	7	0,05671204	7	0,05603306	7	0,05755668
EPSILON =0,01									Iteraciones 10	
FUNCION : TRECCANI [-5,5][-5,5]										
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,066509		1	0,0371311	1	0,0355809	1	0,0369391	1	0,035723
2	0,0684741		2	0,0383551	2	0,035995	2	0,0377691	2	0,03549
3	0,0656118		3	0,0377769	3	0,0362151	3	0,0368311	3	0,034816
4	0,067266		4	0,0730081	4	0,0360012	4	0,03703	4	0,0360968
5	0,0677662		5	0,037389	5	0,0352211	5	0,037329	5	0,0343449
6			6		6		6		6	
7	0,06712542		7	0,04473204	7	0,03580266	7	0,03717966	7	0,03529414
EPSILON =0,01									Iteraciones 50	
FUNCION : TRECCANI [-5,5][-5,5]										
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,066752		1	0,039458	1	0,046874	1	0,032989	1	0,0341389
2	0,0655		2	0,0390382	2	0,0466909	2	0,0333941	2	0,033813
3	0,0660501		3	0,0393038	3	0,0460441	3	0,034266	3	0,0350201
4	0,0650239		4	0,0391922	4	0,0461419	4	0,032697	4	0,03368
5	0,0654321		5	0,038362	5	0,0467539	5	0,034301	5	0,0339389
6			6		6		6		6	
7	0,06575162		7	0,03907084	7	0,04650096	7	0,03352942	7	0,03411818
EPSILON =0,01									Iteraciones 100	
FUNCION : TRECCANI [-5,5][-5,5]										
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,0637491		1	0,039758	1	0,0390141	1	0,0342839	1	0,0325589
2	0,0656199		2	0,0403011	2	0,038712	2	0,0339849	2	0,0323291
3	0,060895		3	0,0402319	3	0,038764	3	0,034838	3	0,0318201
4	0,0637279		4	0,041728	4	0,0385051	4	0,033942	4	0,0331991
5	0,0652139		5	0,040643	5	0,038554	5	0,0328751	5	0,0329881
6			6		6		6		6	
7	0,06384116		7	0,0405324	7	0,03870984	7	0,03398478	7	0,03257906
EPSILON =0,01									Iteraciones 500	
FUNCION : TRECCANI [-5,5][-5,5]										
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO
1	0,042819		1	0,0427861	1	0,0445631	1	0,0462608	1	0,0449269
2	0,044431		2	0,042984	2	0,0435731	2	0,0443518	2	0,0457599
3	0,0450609		3	0,0449769	3	0,0427279	3	0,0484049	3	0,0447381
4	0,0444169		4	0,0456281	4	0,042892	4	0,045599	4	0,0461621
5	0,0426581		5	0,043499	5	0,0454161	5	0,047797	5	0,0460091
6			6		6		6		6	
7	0,04387718		7	0,04397482	7	0,04383444	7	0,0464827	7	0,04551922

EPSILON =0,0001 Iteraciones 10

FUNCION : TRECCANI [-5,5][-5,5]

N PRO,		2	N PRO,		3	N PRO,		4	N PRO,		5	N PRO,		6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,135594		1	0,071619		1	0,0677409		1	0,06811		1	0,0680509	
2	0,132639		2	0,0734651		2	0,067986		2	0,0683839		2	0,0653911	
3	0,139978		3	0,0741479		3	0,0736041		3	0,0704551		3	0,0665331	
4	0,131668		4	0,0705011		4	0,068104		4	0,066572		4	0,066062	
5	0,135204		5	0,0716681		5	0,0709929		5	0,06778		5	0,0678182	
6			6			6			6			6		
7	0,1350166		7	0,07228024		7	0,06968558		7	0,0682602		7	0,06677106	

EPSILON =0,0001 Iteraciones 50

FUNCION : TRECCANI [-5,5][-5,5]

N PRO,		2	N PRO,		3	N PRO,		4	N PRO,		5	N PRO,		6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,133097		1	0,0739691		1	0,0760651		1	0,0613861		1	0,0630941	
2	0,12871		2	0,0738339		2	0,0774388		2	0,0615921		2	0,0633168	
3	0,12779		3	0,0733531		3	0,0743768		3	0,0611031		3	0,0622509	
4	0,128425		4	0,07569		4	0,076786		4	0,0618951		4	0,0656419	
5	0,130081		5	0,0722709		5	0,0756299		5	0,061542		5	0,0639079	
6			6			6			6			6		
7	0,1296206		7	0,0738234		7	0,07605932		7	0,06150368		7	0,06364232	

EPSILON =0,0001 Iteraciones 100

FUNCION : TRECCANI [-5,5][-5,5]

N PRO,		2	N PRO,		3	N PRO,		4	N PRO,		5	N PRO,		6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,131323		1	0,07111		1	0,071547		1	0,0638292		1	0,0628459	
2	0,128404		2	0,0724208		2	0,0709388		2	0,0650489		2	0,0628221	
3	0,127957		3	0,073828		3	0,068151		3	0,0646329		3	0,0607371	
4	0,125964		4	0,074944		4	0,070931		4	0,0641861		4	0,0631981	
5	0,127954		5	0,0764949		5	0,0692229		5	0,0612419		5	0,0628541	
6			6			6			6			6		
7	0,1283204		7	0,07375954		7	0,07015814		7	0,0637878		7	0,06249146	

EPSILON =0,0001 Iteraciones 500

FUNCION : TRECCANI [-5,5][-5,5]

N PRO,		2	N PRO,		3	N PRO,		4	N PRO,		5	N PRO,		6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,0946281		1	0,071439		1	0,065727		1	0,065546		1	0,0649381	
2	0,096741		2	0,0722148		2	0,0661581		2	0,065119		2	0,0667329	
3	0,0968049		3	0,072557		3	0,0639629		3	0,0652759		3	0,0667272	
4	0,0986979		4	0,0715919		4	0,0641761		4	0,067075		4	0,0647709	
5	0,095695		5	0,073046		5	0,0638978		5	0,0665619		5	0,065882	
6			6			6			6			6		
7	0,09651338		7	0,07216974		7	0,06478438		7	0,06591556		7	0,06581022	

EPSILON =0,000001 Iteraciones 10

FUNCION : TRECCANI [-5,5][-5,5]

N PRO,		2	N PRO,		3	N PRO,		4	N PRO,		5	N PRO,		6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,192542		1	0,106769		1	0,101102		1	0,0991399		1	0,097616	
2	0,201616		2	0,106939		2	0,100015		2	0,099339		2	0,102876	
3	0,198368		3	0,103947		3	0,104474		3	0,102036		3	0,0958109	
4	0,196247		4	0,107133		4	0,101014		4	0,0989132		4	0,102857	
5	0,191217		5	0,10922		5	0,104238		5	0,098341		5	0,096559	
6			6			6			6			6		
7	0,195998		7	0,1068016		7	0,1021686		7	0,09955382		7	0,09914378	

EPSILON =0,000001 Iteraciones 50

FUNCION : TRECCANI [-5,5][-5,5]

N PRO,		2	N PRO,		3	N PRO,		4	N PRO,		5	N PRO,		6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,202176		1	0,112489		1	0,108579		1	0,0888591		1	0,091224	
2	0,199151		2	0,108665		2	0,104179		2	0,0897849		2	0,091109	
3	0,192957		3	0,111999		3	0,103204		3	0,090549		3	0,0925169	
4	0,188866		4	0,107208		4	0,104177		4	0,0899711		4	0,0902259	
5	0,194476		5	0,112326		5	0,104433		5	0,0933518		5	0,0914581	
6			6			6			6			6		
7	0,1955252		7	0,1105374		7	0,1049144		7	0,09050318		7	0,09130678	

EPSILON =0,000001 Iteraciones 100

FUNCION : TRECCANI [-5,5][-5,5]

N PRO,		2	N PRO,		3	N PRO,		4	N PRO,		5	N PRO,		6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,196423		1	0,10803		1	0,0992548		1	0,0950658		1	0,0900791	
2	0,198623		2	0,106566		2	0,101114		2	0,0959458		2	0,0888221	
3	0,198491		3	0,108232		3	0,100745		3	0,0933781		3	0,0914941	
4	0,188484		4	0,10851		4	0,1007		4	0,0972691		4	0,092376	
5	0,19247		5	0,109423		5	0,102826		5	0,0920322		5	0,0901558	
6			6			6			6			6		
7	0,1948982		7	0,1081522		7	0,10092796		7	0,0947382		7	0,09058542	

EPSILON =0,000001 Iteraciones 500

FUNCION : TRECCANI [-5,5][-5,5]

N PRO,		2	N PRO,		3	N PRO,		4	N PRO,		5	N PRO,		6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,157527		1	0,106638		1	0,099581		1	0,09409		1	0,095319	
2	0,161305		2	0,104863		2	0,0981519		2	0,095819		2	0,0965819	
3	0,157365		3	0,108099		3	0,0981369		3	0,095495		3	0,0930908	
4	0,160417		4	0,105041		4	0,0987649		4	0,0927892		4	0,0941439	
5	0,163357		5	0,105349		5	0,0967491		5	0,0930488		5	0,099	
6			6			6			6			6		
7	0,1599942		7	0,105998		7	0,09827676		7	0,0942484		7	0,09562712	

EPSILON =0,01									Iteraciones 10	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]										
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	6,65717	1	3,49984	1	2,68581	1	2,23559	1	2,15925	
2	6,86954	2	3,47342	2	2,70716	2	2,24184	2	2,10137	
3	6,66621	3	3,47456	3	2,68085	3	2,22133	3	2,10681	
4	6,67356	4	3,47213	4	2,66664	4	2,29212	4	2,10497	
5	6,74060	5	3,57841	5	2,70834	5	2,23205	5	2,15058	
6		6		6		6		6		
7	6,7214	7	3,4997	7	2,6898	7	2,2446	7	2,1246	

EPSILON =0,01									Iteraciones 50	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]										
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	6,66368	1	4,33787	1	3,14432	1	3,37612	1	2,10142	
2	6,74233	2	4,31123	2	3,23679	2	3,50156	2	2,08010	
3	6,73053	3	4,46500	3	3,12453	3	3,36686	3	2,13914	
4	6,78329	4	4,35660	4	3,13176	4	3,36807	4	2,09210	
5	6,86456	5	4,34171	5	3,19977	5	3,46816	5	2,10595	
6		6		6		6		6		
7	6,7569	7	4,3625	7	3,1674	7	3,4162	7	2,1037	

EPSILON =0,01									Iteraciones 100	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]										
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	6,68480	1	4,51549	1	3,22323	1	3,33044	1	2,23312	
2	6,67095	2	4,63055	2	3,31844	2	3,32301	2	2,23131	
3	6,91013	3	4,49442	3	3,23861	3	3,26529	3	2,24108	
4	6,87806	4	4,63371	4	3,27094	4	3,24571	4	2,26082	
5	6,88325	5	4,52908	5	4,62396	5	3,24732	5	2,24016	
6		6		6		6		6		
7	6,8054	7	4,5607	7	3,5350	7	3,2824	7	2,2413	

EPSILON =0,01									Iteraciones 500	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]										
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	6,82552	1	3,88809	1	2,44447	1	2,08387	1	2,07046	
2	6,79583	2	3,91804	2	2,48512	2	2,11711	2	2,26721	
3	6,61243	3	3,89792	3	2,41997	3	2,07990	3	2,10928	
4	6,58045	4	3,94201	4	2,48357	4	2,07933	4	2,06607	
5	6,62951	5	3,94493	5	2,40139	5	2,06505	5	2,06428	
6		6		6		6		6		
7	6,6887	7	3,9182	7	2,4469	7	2,0851	7	2,1155	

EPSILON =0,0001									Iteraciones 10	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]										
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	21,6345	1	13,3189	1	10,4093	1	7,1062	1	6,6477	
2	21,5749	2	12,6463	2	10,5427	2	6,8507	2	6,7904	
3	21,5195	3	12,6762	3	10,4757	3	6,9294	3	7,0382	
4	21,5047	4	12,7162	4	10,5060	4	6,8725	4	6,8820	
5	21,8000	5	12,5982	5	10,4721	5	6,8297	5	6,6650	
6		6		6		6		6		
7	21,6067	7	12,7912	7	10,4812	7	6,9177	7	6,8046	

EPSILON =0,0001									Iteraciones 50	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]										
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	22,0798	1	12,1051	1	8,3381	1	8,0601	1	5,8460	
2	22,6435	2	11,8555	2	8,6191	2	8,5932	2	5,8068	
3	22,1073	3	12,0958	3	8,5965	3	8,3521	3	5,9338	
4	21,5269	4	11,7980	4	8,4397	4	8,3149	4	5,9042	
5	21,8446	5	12,1761	5	8,7559	5	8,4077	5	5,8301	
6		6		6		6		6		
7	22,0404	7	12,0061	7	8,5499	7	8,3456	7	5,8642	

EPSILON =0,0001									Iteraciones 100	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]										
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	21,7548	1	15,8702	1	8,8106	1	6,8909	1	5,1603	
2	21,4070	2	15,6814	2	7,8503	2	6,3402	2	5,1924	
3	21,5327	3	15,7392	3	7,9957	3	6,3219	3	5,7635	
4	21,4424	4	15,7867	4	7,9496	4	6,8330	4	5,7090	
5	22,2128	5	15,9651	5	7,8846	5	6,8834	5	5,8022	
6		6		6		6		6		
7	21,6699	7	15,8085	7	8,0982	7	6,6539	7	5,5255	

EPSILON =0,0001									Iteraciones 500	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]										
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	21,4077	1	11,0749	1	7,6517	1	5,7898	1	5,7383	
2	21,7633	2	11,1394	2	7,6128	2	5,9605	2	5,7050	
3	21,4539	3	10,9705	3	7,7434	3	5,7983	3	5,6876	
4	21,3515	4	10,8107	4	7,5612	4	5,7808	4	5,8800	
5	21,4609	5	10,7956	5	7,6660	5	6,0077	5	5,7349	
6		6		6		6		6		
7	21,4875	7	10,9582	7	7,6470	7	5,8674	7	5,7492	

EPSILON =0,000001										Iteraciones 10	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	21,7456		1	12,8454		1	10,6596		1	7,1544		1	6,8972	
2	21,7666		2	12,8706		2	10,7540		2	7,0408		2	6,8588	
3	21,6843		3	12,8830		3	10,4617		3	6,9608		3	6,9219	
4	21,8468		4	13,0418		4	10,4617		4	7,1205		4	7,2911	
5	22,6933		5	13,2115		5	10,7804		5	7,1363		5	7,2468	
6			6			6			6			6		
7	21,9473		7	12,9705		7	10,6235		7	7,0826		7	7,0432	

EPSILON =0,000001										Iteraciones 50	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	20,9250		1	11,8213		1	8,7203		1	7,6831		1	5,8754	
2	21,0279		2	11,9107		2	8,4686		2	8,1525		2	5,9921	
3	20,7440		3	11,8399		3	8,9672		3	7,7839		3	5,9051	
4	21,3481		4	12,2111		4	8,3550		4	7,8222		4	5,9690	
5	20,8517		5	11,9708		5	8,2852		5	7,8296		5	5,9895	
6			6			6			6			6		
7	20,9793		7	11,9508		7	8,5593		7	7,8542		7	5,9462	

EPSILON =0,000001										Iteraciones 100	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	21,1801		1	14,7818		1	7,6317		1	6,0483		1	5,2413	
2	19,9781		2	14,3520		2	7,8221		2	5,8238		2	5,2680	
3	19,8209		3	14,1008		3	7,5671		3	6,3194		3	5,2899	
4	19,5779		4	14,0793		4	7,6828		4	6,8463		4	5,3400	
5	22,8076		5	14,6001		5	7,2716		5	6,0099		5	5,3518	
6			6			6			6			6		
7	20,6729		7	14,3828		7	7,5951		7	6,2095		7	5,2982	

EPSILON =0,000001										Iteraciones 500	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	19,7550		1	10,2167		1	7,1271		1	5,9632		1	5,5343	
2	19,1382		2	10,0305		2	7,2380		2	5,5550		2	5,5688	
3	19,3287		3	10,0295		3	7,1216		3	5,5819		3	5,5782	
4	19,2753		4	10,0256		4	7,1599		4	5,5494		4	5,6625	
5	19,7926		5	10,0340		5	7,1626		5	5,6115		5	5,6423	
6			6			6			6			6		
7	19,4580		7	10,0673		7	7,1619		7	5,6522		7	5,5972	

DATOS ALGORTIMO CAJA RESULTADO (2007, Muñoz-Peña)

EPSILON =0,01											Iteraciones 10	
ALGORTIMO CAJAS RESULTADO CON DISTRIBUCION EQULIBRADA												
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		
1	0,16043		1	0,133019	1	0,0973411	1	0,0986631	1	0,052978		
2	0,165217		2	0,132185	2	0,0980239	2	0,0984471	2	0,0521009		
3	0,157236		3	0,133116	3	0,100235	3	0,103657	3	0,0523119		
4	0,165559		4	0,134207	4	0,103194	4	0,099411	4	0,0510221		
5	0,158544		5	0,129303	5	0,10012	5	0,100124	5	0,050523		
6			6		6		6		6			
7	0,1613972		7	0,132366	7	0,0997828	7	0,10006044	7	0,05178718		
EPSILON =0,01											Iteraciones 50	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]												
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		
1	0,157383		1	0,110721	1	0,291265	1	cae	1	cae		
2	0,156889		2	0,107632	2	0,295945	2		2			
3	0,160226		3	0,108234	3	0,290649	3		3			
4	0,158561		4	0,105091	4	0,28609	4		4			
5	0,156321		5	0,11058	5	0,284689	5		5			
6			6		6		6		6			
7	0,157876		7	0,1084516	7	0,2897276	7	#DIV/0!	7	#DIV/0!		
EPSILON =0,01											Iteraciones 100	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]												
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		
1	0,153992		1		1	0,194208	1		1			
2	0,153359		2		2	0,187237	2		2			
3	0,159321		3		3	0,184505	3		3			
4	0,153093		4		4	0,188519	4		4			
5	0,158231		5		5	0,187594	5		5			
6			6		6		6		6			
7	0,1555992		7	#DIV/0!	7	0,1884126	7	#DIV/0!	7	#DIV/0!		
EPSILON =0,01											Iteraciones 500	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]												
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		
1	0,122261		1		1		1		1			
2	0,125795		2		2		2		2			
3	0,12739		3		3		3		3			
4	0,125372		4		4		4		4			
5	0,122596		5		5		5		5			
6			6		6		6		6			
7	0,1246828		7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	7	#DIV/0!		
EPSILON =0,0001											Iteraciones 10	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]												
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		
1	0,173207		1	0,142496	1	0,116166	1	0,113878	1	0,056201		
2	0,182103		2	0,142889	2	0,120243	2	0,116247	2	0,0600219		
3	0,183437		3	0,18044	3	0,117599	3	0,116079	3	0,0588281		
4	0,17729		4	0,1471	4	0,118311	4	0,118949	4	0,0592492		
5	0,176439		5	0,150502	5	0,11577	5	0,119901	5	0,0589862		
6			6		6		6		6			
7	0,1784952		7	0,1526854	7	0,1176178	7	0,1170108	7	0,05865728		
EPSILON =0,0001											Iteraciones 50	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]												
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		
1	0,17378		1	0,123578	1	0,639314	1	0,369682	1	cae		
2	0,171296		2	0,122716	2	0,654546	2	0,371178	2			
3	0,178166		3	0,123235	3	0,630691	3	0,354514	3			
4	0,179216		4	0,123094	4	0,654472	4	0,36047	4			
5	0,174659		5	0,121826	5	0,631913	5	0,364763	5			
6			6		6		6		6			
7	0,1754234		7	0,1228898	7	0,6421872	7	0,3641214	7	#DIV/0!		
EPSILON =0,0001											Iteraciones 100	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]												
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		
1	0,16927		1	cae	1	0,186856	1	cae	1	cae		
2	0,166045		2		2	0,18714	2		2			
3	0,177761		3		3	0,187687	3		3			
4	0,179839		4		4	0,183852	4		4			
5	0,175334		5		5	0,194052	5		5			
6			6		6		6		6			
7	0,1736498		7	#DIV/0!	7	0,1879174	7	#DIV/0!	7	#DIV/0!		
EPSILON =0,0001											Iteraciones 500	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]												
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6		
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO		
1	0,139666		1	cae	1	cae	1	cae	1	cae		
2	0,139356		2		2		2		2			
3	0,140385		3		3		3		3			
4	0,141661		4		4		4		4			
5	0,142188		5		5		5		5			
6			6		6		6		6			
7	0,1406512		7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	7	#DIV/0!		

EPSILON =0,000001										Iteraciones 10	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°		
1	0,212879		1	0,165587		1	0,13915		1	0,133489		1	cae	
2	0,196942		2	0,164595		2	0,135531		2	0,13562		2		
3	0,195795		3	0,169815		3	0,135144		3	0,134472		3		
4	0,197192		4	0,163986		4	0,136019		4	0,143474		4		
5	0,193076		5	0,164645		5	0,138824		5	0,134365		5		
6			6			6			6			6		
7	0,1991768		7	0,1657256		7	0,1369336		7	0,136284		7	#DIV/0!	
EPSILON =0,000001										Iteraciones 50	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,189345		1	0,139654		1	cae		1	0,49708		1	cae	
2	0,194083		2	0,13992		2			2	0,490103		2		
3	0,192125		3	0,142259		3			3	0,49968		3		
4	0,198832		4	0,151142		4			4	0,515515		4		
5	0,191631		5	0,141177		5			5	0,498811		5		
6			6			6			6			6		
7	0,1932032		7	0,1428304		7	#DIV/0!		7	0,5002378		7	#DIV/0!	
EPSILON =0,000001										Iteraciones 100	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°			N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,188872		1	cae		1	cae		1	cae		1	cae	
2	0,188539		2			2			2			2		
3	0,185497		3			3			3			3		
4	0,196405		4			4			4			4		
5	0,194903		5			5			5			5		
6			6			6			6			6		
7	0,1908432		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!	
EPSILON =0,000001										Iteraciones 500	
FUNCION :SIX_HUM_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	0,160397		1	cae		1	cae		1	cae		1	cae	
2	0,161193		2			2			2			2		
3	0,160524		3			3			3			3		
4	0,15938		4			4			4			4		
5	0,158992		5			5			5			5		
6			6			6			6			6		
7	0,1600972			#DIV/0!			#DIV/0!			#DIV/0!			#DIV/0!	
EPSILON =0,01										Iteraciones 10	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	cae		1	0,409911		1	0,281961		1	0,268	
2			2			2	0,413207		2	0,284849		2	0,268724	
3			3			3	0,410357		3	0,288107		3	0,275479	
4			4			4	0,413124		4	0,280718		4	0,259445	
5			5			5	0,413559		5	0,288459		5	0,263561	
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	0,4120316		7	0,2848188		7	0,2670418	
EPSILON =0,01										Iteraciones 50	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	0,10238		1	0,291474		1	0,16507		1	0,178272	
2			2	0,105052		2	0,300264		2	0,166245		2	0,183785	
3			3	0,102724		3	0,296472		3	0,165424		3	0,185059	
4			4	0,105046		4	0,292305		4	0,169271		4	0,178545	
5			5	0,100595		5	0,293443		5	0,16982		5	0,175927	
6			6			6			6			6		
7	#DIV/0!		7	0,1031594		7	0,2947916		7	0,167166		7	0,1803176	
EPSILON =0,01										Iteraciones 100	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	cae		1	cae		1	0,372176		1	0,350429	
2			2			2			2	0,356103		2	0,360646	
3			3			3			3	0,348775		3	0,349431	
4			4			4			4	0,358113		4	0,35532	
5			5			5			5	0,38034		5	0,352339	
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	0,3631014		7	0,353633	
EPSILON =0,01										Iteraciones 500	
FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	0,063412		1	0,0698922		1	0,0656891		1	0,064332	
2			2	0,062932		2	0,0640669		2	0,070889		2	0,066438	
3			3	0,0629878		3	0,065052		3	0,063344		3	0,0704241	
4			4	0,0717919		4	0,0644629		4	0,0644622		4	0,063349	
5			5	0,0629249		5	0,0641968		5	0,065182		5	0,0633671	
6			6			6			6			6		
7	#DIV/0!		7	0,06480972		7	0,06553416		7	0,06591326		7	0,06558204	

EPSILON =0,0001 — Iteraciones 10

FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]

	N PRO, 2		N PRO, 3		N PRO, 4		N PRO, 5		N PRO, 6	
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	cae	1	cae	1	cae	1	cae	1	cae	
2		2		2		2		2		
3		3		3		3		3		
4		4		4		4		4		
5		5		5		5		5		
6		6		6		6		6		
7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	

EPSILON =0,0001 — Iteraciones 50

FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]

	N PRO, 2		N PRO, 3		N PRO, 4		N PRO, 5		N PRO, 6	
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	cae	1	cae	1	cae	1	cae	1	cae	
2		2		2		2		2		
3		3		3		3		3		
4		4		4		4		4		
5		5		5		5		5		
6		6		6		6		6		
7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	

EPSILON =0,0001 — Iteraciones 100

FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]

	N PRO, 2		N PRO, 3		N PRO, 4		N PRO, 5		N PRO, 6	
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	cae	1	cae	1	cae	1	cae	1	cae	
2		2		2		2		2		
3		3		3		3		3		
4		4		4		4		4		
5		5		5		5		5		
6		6		6		6		6		
7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	

EPSILON =0,0001 — Iteraciones 500

FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]

	N PRO, 2		N PRO, 3		N PRO, 4		N PRO, 5		N PRO, 6	
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	cae	1	0,0837379	1	cae	1	0,086551	1	0,0739	
2		2	0,0827389	2		2	0,0864499	2	0,0743301	
3		3	0,0920761	3		3	0,094142	3	0,080436	
4		4	0,0900321	4		4	0,0863779	4	0,0758669	
5		5	0,091471	5		5	0,0870731	5	0,074326	
6		6		6		6		6		
7	#DIV/0!	7	0,0880112	7	#DIV/0!	7	0,08811878	7	0,0757718	

EPSILON =0,000001 — Iteraciones 10

FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]

	N PRO, 2		N PRO, 3		N PRO, 4		N PRO, 5		N PRO, 6	
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	cae	1	cae	1	cae	1	cae	1	cae	
2		2		2		2		2		
3		3		3		3		3		
4		4		4		4		4		
5		5		5		5		5		
6		6		6		6		6		
7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	

EPSILON =0,000001 — Iteraciones 50

FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]

	N PRO, 2		N PRO, 3		N PRO, 4		N PRO, 5		N PRO, 6	
N°		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	cae	1	cae	1	cae	1	cae	1	cae	
2		2		2		2		2		
3		3		3		3		3		
4		4		4		4		4		
5		5		5		5		5		
6		6		6		6		6		
7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	

EPSILON =0,000001 — Iteraciones 100

FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]

	N PRO, 2		N PRO, 3		N PRO, 4		N PRO, 5		N PRO, 6	
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	cae	1	cae	1	cae	1	cae	1	cae	
2		2		2		2		2		
3		3		3		3		3		
4		4		4		4		4		
5		5		5		5		5		
6		6		6		6		6		
7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	

EPSILON =0,000001 — Iteraciones 500

FUNCION : BEALE_1 [-4,5,4,5][-4,5,4,5]

	N PRO, 2		N PRO, 3		N PRO, 4		N PRO, 5		N PRO, 6	
N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1		1	0,160054	1		1	0,232582	1	0,117753	
2		2	0,164228	2		2	0,231891	2	0,118876	
3		3	0,160332	3		3	0,256937	3	0,117387	
4		4	0,162271	4		4	0,237016	4	0,11789	
5		5	0,162817	5		5	0,230897	5	0,118941	
6		6		6		6		6		
7	#DIV/0!	7	0,1619404	7	#DIV/0!	7	0,2378646	7	0,1181694	

EPSILON =0,01										Iteraciones 10	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	cae		1	cae		1	cae		1	cae	
2			2			2			2			2		
3			3			3			3			3		
4			4			4			4			4		
5			5			5			5			5		
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!	

EPSILON =0,01										Iteraciones 50	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	cae		1	cae		1	cae		1	cae	
2			2			2			2			2		
3			3			3			3			3		
4			4			4			4			4		
5			5			5			5			5		
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!	

EPSILON =0,01										Iteraciones 100	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	cae		1	cae		1	cae		1	cae	
2			2			2			2			2		
3			3			3			3			3		
4			4			4			4			4		
5			5			5			5			5		
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!	

EPSILON =0,01										Iteraciones 500	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	cae		1	0,0310001		1	0,03161		1	0,0325429	
2			2			2	0,0308321		2	0,032125		2	0,031991	
3			3			3	0,031049		3	0,0305028		3	0,031842	
4			4			4	0,0317969		4	0,0305231		4	0,0314989	
5			5			5	0,030849		5	0,0322659		5	0,0336189	
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	0,03110542		7	0,03140536		7	0,03229874	

EPSILON =0,0001										Iteraciones 10	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	cae		1	cae		1	cae		1	cae	
2			2			2			2			2		
3			3			3			3			3		
4			4			4			4			4		
5			5			5			5			5		
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!	

EPSILON =0,0001										Iteraciones 50	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	cae		1	cae		1	cae		1	cae	
2			2			2			2			2		
3			3			3			3			3		
4			4			4			4			4		
5			5			5			5			5		
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!	

EPSILON =0,0001										Iteraciones 100	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	cae		1	cae		1	cae		1	cae	
2			2			2			2			2		
3			3			3			3			3		
4			4			4			4			4		
5			5			5			5			5		
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!	

EPSILON =0,0001										Iteraciones 500	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	cae		1	0,0322099		1	0,031296		1	0,032335	
2			2			2	0,0311811		2	0,031322		2	0,035537	
3			3			3	0,0304432		3	0,031184		3	0,0314908	
4			4			4	0,0322599		4	0,0312829		4	0,031512	
5			5			5	0,0348721		5	0,0314641		5	0,0317349	
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	0,03219324		7	0,0313098		7	0,03252194	

EPSILON =0,000001										Iteraciones 10	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	cae		1	cae	1	cae	1	cae	1	cae	
2			2		2		2		2		
3			3		3		3		3		
4			4		4		4		4		
5			5		5		5		5		
6			6		6		6		6		
7	#DIV/0!		7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	

EPSILON =0,000001										Iteraciones 50	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	cae		1	cae	1	cae	1	cae	1	cae	
2			2		2		2		2		
3			3		3		3		3		
4			4		4		4		4		
5			5		5		5		5		
6			6		6		6		6		
7	#DIV/0!		7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	

EPSILON =0,000001										Iteraciones 100	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	cae		1	cae	1	cae	1	cae	1	cae	
2			2		2		2		2		
3			3		3		3		3		
4			4		4		4		4		
5			5		5		5		5		
6			6		6		6		6		
7	#DIV/0!		7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	

EPSILON =0,000001										Iteraciones 500	
FUNCION : BOOTH [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	cae		1	cae	1	0,030509	1	0,0313199	1	0,0315361	
2			2		2	0,0310552	2	0,0306852	2	0,030935	
3			3		3	0,0310361	3	0,0314131	3	0,0303211	
4			4		4	0,0316129	4	0,0306342	4	0,0329161	
5			5		5	0,030252	5	0,0320039	5	0,0308628	
6			6		6		6		6		
7	#DIV/0!		7	#DIV/0!	7	0,03089304	7	0,03121126	7	0,03131422	

EPSILON =0,01										Iteraciones 10	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	cae		1	cae	1	cae	1	cae	1	cae	
2			2		2		2		2		
3			3		3		3		3		
4			4		4		4		4		
5			5		5		5		5		
6			6		6		6		6		
7	#DIV/0!		7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	

EPSILON =0,01										Iteraciones 50	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	cae		1	cae	1	cae	1	cae	1	cae	
2			2		2		2		2		
3			3		3		3		3		
4			4		4		4		4		
5			5		5		5		5		
6			6		6		6		6		
7	#DIV/0!		7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	

EPSILON =0,01										Iteraciones 100	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	cae		1	cae	1	cae	1	cae	1	cae	
2			2		2		2	cae	2		
3			3		3		3		3		
4			4		4		4		4		
5			5		5		5		5		
6			6		6		6		6		
7	#DIV/0!		7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	

EPSILON =0,01										Iteraciones 500	
FUNCION : MATYAS [-10,10][-10,10]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	cae		1	cae	1	cae	1	cae	1	cae	
2			2		2		2		2		
3			3		3		3		3		
4			4		4		4		4		
5			5		5		5		5		
6			6		6		6		6		
7	#DIV/0!		7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	

EPSILON =0,0001										Iteraciones 10		
FUNCION : MATYAS [-10,10][-10,10]												
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	cae		1	cae		1	cae		1	cae	
2			2			2			2			2		
3			3			3			3			3		
4			4			4			4			4		
5			5			5			5			5		
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!	

EPSILON =0,0001										Iteraciones 50		
FUNCION : MATYAS [-10,10][-10,10]												
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	cae		1	cae		1	cae		1	cae	
2			2			2			2			2		
3			3			3			3			3		
4			4			4			4			4		
5			5			5			5			5		
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!	

EPSILON =0,0001										Iteraciones 100		
FUNCION : MATYAS [-10,10][-10,10]												
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	cae		1	cae		1	cae		1	cae	
2			2			2			2			2		
3			3			3			3			3		
4			4			4			4			4		
5			5			5			5			5		
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!	

EPSILON =0,0001										Iteraciones 500		
FUNCION : MATYAS [-10,10][-10,10]												
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	cae		1	cae		1	cae		1	cae	
2			2			2			2			2		
3			3			3			3			3		
4			4			4			4			4		
5			5			5			5			5		
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!	

EPSILON =0,000001										Iteraciones 10		
FUNCION : MATYAS [-10,10][-10,10]												
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	cae		1	cae		1	cae		1	cae	
2			2			2			2			2		
3			3			3			3			3		
4			4			4			4			4		
5			5			5			5			5		
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!	

EPSILON =0,000001										Iteraciones 50		
FUNCION : MATYAS [-10,10][-10,10]												
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	cae		1	cae		1	cae		1		
2			2			2			2			2		
3			3			3			3			3		
4			4			4			4			4		
5			5			5			5			5		
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!	

EPSILON =0,000001										Iteraciones 100		
FUNCION : MATYAS [-10,10][-10,10]												
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	cae		1	cae		1	cae		1	cae	
2			2			2			2			2		
3			3			3			3			3		
4			4			4			4			4		
5			5			5			5			5		
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!	

EPSILON =0,000001										Iteraciones 500		
FUNCION : MATYAS [-10,10][-10,10]												
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	cae		1	cae		1	cae		1	cae	
2			2			2			2			2		
3			3			3			3			3		
4			4			4			4			4		
5			5			5			5			5		
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!	

EPSILON =0,01										Iteraciones 10	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	cae		1	0,036134	1	0,0342751	1	0,03756	1	cae	
2			2	0,036659	2	0,0348039	2	0,0353251	2		
3			3	0,0362639	3	0,0351901	3	0,035466	3		
4			4	0,0358162	4	0,0347841	4	0,0355191	4		
5			5	0,0360839	5	0,033885	5	0,0352859	5		
6			6		6		6		6		
7	#DIV/0!		7	0,0361914	7	0,03458764	7	0,03583122	7	#DIV/0!	

EPSILON =0,01										Iteraciones 50	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	cae		1	cae	1	cae	1	cae	1	cae	
2			2		2		2		2		
3			3		3		3		3		
4			4		4		4		4		
5			5		5		5		5		
6			6		6		6		6		
7	#DIV/0!		7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	

EPSILON =0,01										Iteraciones 100	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	cae		1	cae	1	0,0525591	1	0,0488398	1	0,051548	
2			2		2	0,0527868	2	0,049098	2	0,0515239	
3			3		3	0,0547619	3	0,0495651	3	0,050416	
4			4		4	0,0526102	4	0,049078	4	0,0514939	
5			5		5	0,053477	5	0,0486598	5	0,051471	
6			6		6		6		6		
7	#DIV/0!		7	#DIV/0!	7	0,053239	7	0,04904814	7	0,05129056	

EPSILON =0,01										Iteraciones 500	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	cae		1	cae	1	0,0502069	1	cae	1	cae	
2			2		2	0,0504358	2		2		
3			3		3	0,0482922	3		3		
4			4		4	0,0483952	4		4		
5			5		5	0,0485971	5		5		
6			6		6		6		6		
7	#DIV/0!		7	#DIV/0!	7	0,04918544	7	#DIV/0!	7	#DIV/0!	

EPSILON =0,0001										Iteraciones 10	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	cae		1	0,0404019	1	0,0365601	1	0,041482	1	cae	
2			2	0,0439091	2	0,036695	2	0,0398669	2		
3			3	0,0397649	3	0,0380161	3	0,041441	3		
4			4	0,0416191	4	0,037034	4	0,0419421	4		
5			5	0,0407612	5	0,0362861	5	0,0410678	5		
6			6		6		6		6		
7	#DIV/0!		7	0,04129124	7	0,03691826	7	0,04115996	7	#DIV/0!	

EPSILON =0,0001										Iteraciones 50	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	cae		1	cae	1	cae	1	cae	1	cae	
2			2		2		2		2		
3			3		3		3		3		
4			4		4		4		4		
5			5		5		5		5		
6			6		6		6		6		
7	#DIV/0!		7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	

EPSILON =0,0001										Iteraciones 100	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	cae		1	cae	1	0,0771031	1	0,073911	1	0,0741849	
2			2		2	0,0764048	2	0,0729759	2	0,075181	
3			3		3	0,076165	3	0,07481	3	0,0735769	
4			4		4	0,0789671	4	0,0735788	4	0,0735569	
5			5		5	0,0779989	5	0,073132	5	0,0752079	
6			6		6		6		6		
7	#DIV/0!		7	#DIV/0!	7	0,07732778	7	0,07368154	7	0,07434152	

EPSILON =0,0001										Iteraciones 500	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	cae		1	cae	1	0,0504649	1	cae	1	cae	
2			2		2	0,049484	2		2		
3			3		3	0,051029	3		3		
4			4		4	0,0474851	4		4		
5			5		5	0,0478499	5		5		
6			6		6		6		6		
7	#DIV/0!		7	#DIV/0!	7	0,04926258	7	#DIV/0!	7	#DIV/0!	

EPSILON =0,000001										Iteraciones 10	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	0,044697		1	0,0405281		1	0,044452		1	cae	
2			2	0,0462041		2	0,0393021		2	0,0464852		2		
3			3	0,0454271		3	0,03847		3	0,045218		3		
4			4	0,0456579		4	0,0393898		4	0,0445352		4		
5			5	0,0450091		5	0,0381911		5	0,04532		5		
6			6			6			6			6		
7	#DIV/0!		7	0,04539904		7	0,03917622		7	0,04520208		7	#DIV/0!	

EPSILON =0,000001										Iteraciones 50	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	cae		1	cae		1	cae		1	cae	
2			2			2			2			2		
3			3			3			3			3		
4			4			4			4			4		
5			5			5			5			5		
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!	

EPSILON =0,000001										Iteraciones 100	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	cae		1	0,111268		1	0,105419		1	0,108891	
2			2			2	0,109674		2	0,104738		2	0,105725	
3			3			3	0,115595		3	0,105248		3	0,106046	
4			4			4	0,109621		4	0,104519		4	0,105772	
5			5			5	0,108251		5	0,107482		5	0,106163	
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	0,1108818		7	0,1054812		7	0,1065194	

EPSILON =0,000001										Iteraciones 500	
FUNCION : THREE_HUMP_CAMEL_BACK [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	cae		1	0,0520232		1	cae		1	cae	
2			2			2	0,0539858		2			2		
3			3			3	0,0521271		3			3		
4			4			4	0,0537391		4			4		
5			5			5	0,0521111		5			5		
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	0,05279726		7	#DIV/0!		7	#DIV/0!	

EPSILON =0,01										Iteraciones 10	
FUNCION : TRECCANI [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	cae		1	cae		1	cae		1	cae	
2			2			2			2			2		
3			3			3			3			3		
4			4			4			4			4		
5			5			5			5			5		
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!	

EPSILON =0,01										Iteraciones 50	
FUNCION : TRECCANI [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	cae		1	0,034313		1	0,031929		1	0,025336	
2			2			2	0,03317		2	0,0319769		2	0,0251701	
3			3			3	0,0324311		3	0,0331109		3	0,025254	
4			4			4	0,032449		4	0,0315909		4	0,0251169	
5			5			5	0,0327141		5	0,0321782		5	0,0260129	
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	0,03301544		7	0,03215718		7	0,02537798	

EPSILON =0,01										Iteraciones 100	
FUNCION : TRECCANI [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1			1			1	0,058841		1			1	0,079798	
2			2			2	0,058789		2			2	0,0800521	
3			3			3	0,0587509		3			3	0,0814168	
4			4			4	0,0587881		4			4	0,0852151	
5			5			5	0,059052		5			5	0,083879	
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	0,0588442		7	#DIV/0!		7	0,0820722	

EPSILON =0,01										Iteraciones 500	
FUNCION : TRECCANI [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1			1			1	0,058079		1	0,0667982		1	0,07724	
2			2			2	0,0575349		2	0,0665021		2	0,074609	
3			3			3	0,0573199		3	0,0677178		3	0,0765738	
4			4			4	0,058743		4	0,0680962		4	0,0768969	
5			5			5	0,0587351		5	0,066968		5	0,0755351	
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	0,05808238		7	0,06721646		7	0,07617096	

EPSILON =0,0001										Iteraciones 10	
FUNCION : TRECCANI [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	cae		1	caec		1	cae		1	cae	
2			2			2			2			2		
3			3			3			3			3		
4			4			4			4			4		
5			5			5			5			5		
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!	

EPSILON =0,0001										Iteraciones 50	
FUNCION : TRECCANI [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	cae		1	0,0582681		1	0,054215		1	0,0462589	
2			2			2	0,05547		2	0,0542951		2	0,047539	
3			3			3	0,0567729		3	0,0535781		3	0,046824	
4			4			4	0,055809		4	0,055105		4	0,0483551	
5			5			5	0,0567551		5	0,054496		5	0,0475268	
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	0,05661502		7	0,05433784		7	0,04730076	

EPSILON =0,0001										Iteraciones 100	
FUNCION : TRECCANI [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	cae		1	0,105214		1	cae		1	cae	
2			2			2	0,105736		2			2		
3			3			3	0,101589		3			3		
4			4			4	0,107622		4			4		
5			5			5	0,103184		5			5		
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	0,104669		7	#DIV/0!		7	#DIV/0!	

EPSILON =0,0001										Iteraciones 500	
FUNCION : TRECCANI [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	cae		1	0,108649		1	cae		1		
2			2			2	0,105607		2			2	cae	
3			3			3	0,106709		3			3		
4			4			4	0,106701		4			4		
5			5			5	0,106312		5			5		
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	0,1067956		7	#DIV/0!		7	#DIV/0!	

EPSILON =0,000001										Iteraciones 10	
FUNCION : TRECCANI [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	cae		1	cae		1	cae		1	cae	
2			2			2			2			2		
3			3			3			3			3		
4			4			4			4			4		
5			5			5			5			5		
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!	

EPSILON =0,000001										Iteraciones 50	
FUNCION : TRECCANI [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	cae		1	0,075242		1	0,0715089		1	0,066839	
2			2			2	0,077302		2	0,0745091		2	0,064836	
3			3			3	0,077379		3	0,071579		3	0,0633709	
4			4			4	0,076607		4	0,074254		4	0,06424	
5			5			5	0,0790641		5	0,0745101		5	0,0665028	
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	0,07711882		7	0,07327222		7	0,06515774	

EPSILON =0,000001										Iteraciones 100	
FUNCION : TRECCANI [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	cae		1	0,14034		1	cae		1	cae	
2			2			2	0,135773		2			2		
3			3			3	0,135519		3			3		
4			4			4	0,137115		4			4		
5			5			5	0,137015		5			5		
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	0,1371524		7	#DIV/0!		7	#DIV/0!	

EPSILON =0,000001										Iteraciones 500	
FUNCION : TRECCANI [-5,5][-5,5]											
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	cae		1	cae		1	0,141688		1	cae		1	cae	
2			2			2	0,141777		2			2		
3			3			3	0,139955		3			3		
4			4			4	0,143906		4			4		
5			5			5	0,144589		5			5		
6			6			6			6			6		
7	#DIV/0!		7	#DIV/0!		7	0,142383		7	#DIV/0!		7	#DIV/0!	

EPSILON =0,01										Iteraciones 10					
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]															
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6	
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		
1	cae		1	cae		1	8,07002		1	9,81181		1	11,68350		
2			2			2	7,98327		2	9,79569		2	11,61720		
3			3			3	8,02462		3	10,08400		3	11,75840		
4			4			4	7,81053		4	10,09680		4	11,64040		
5			5			5	8,13566		5	9,81481		5	11,92190		
6			6			6			6			6			
7	#DIV/0!		7	#DIV/0!		7	8,0048		7	9,9206		7	11,7243		
8			8			8			8			8			
9			9			9			9			9			
10			10			10			10			10			

EPSILON =0,01										Iteraciones 50					
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]															
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6	
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		
1	cae		1	cae		1	cae		1	cae		1	6,59608		
2			2			2			2			2	6,68922		
3			3			3			3			3	6,71369		
4			4			4			4			4	6,51187		
5			5			5			5			5	6,64923		
6			6			6			6			6			
7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	6,6320		

EPSILON =0,01										Iteraciones 100					
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]															
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6	
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		
1	cae		1	cae		1	cae		1	cae		1	cae		
2			2			2			2			2			
3			3			3			3			3			
4			4			4			4			4			
5			5			5			5			5			
6			6			6			6			6			
7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		

EPSILON =0,01										Iteraciones 500					
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]															
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6	
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		
1	cae		1	cae		1	cae		1	cae		1	cae		
2			2			2			2			2			
3			3			3			3			3			
4			4			4			4			4			
5			5			5			5			5			
6			6			6			6			6			
7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		

EPSILON =0,0001										Iteraciones 10					
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]															
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6	
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		
1	cae		1	11,5229		1	20,1309		1	20,7305		1	21,8129		
2			2	11,4254		2	20,7526		2	20,6391		2	22,1520		
3			3	11,3328		3	20,7308		3	20,4019		3	22,2794		
4			4	11,7250		4	20,1822		4	20,5593		4	21,9208		
5			5	11,4555		5	20,1654		5	20,6795		5	22,2235		
6			6			6			6			6			
7	#DIV/0!		7	11,4923		7	20,3924		7	20,6021		7	22,0777		
8			8			8			8			8			
9			9			9			9			9			
10			10			10			10			10			

EPSILON =0,0001										Iteraciones 50					
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]															
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6	
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		
1	cae		1	13,7713		1	cae		1	cae		1	cae		
2			2	13,3689		2			2			2			
3			3	13,2867		3			3			3			
4			4	13,8529		4			4			4			
5			5	13,4304		5			5			5			
6			6			6			6			6			
7	#DIV/0!		7	13,5420		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		

EPSILON =0,0001										Iteraciones 100					
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]															
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6	
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		
1	cae		1	cae		1	13,1939		1	cae		1	cae		
2			2			2	13,1397		2			2			
3			3			3	13,5544		3			3			
4			4			4	13,2465		4			4			
5			5			5	13,5537		5			5			
6			6			6			6			6			
7	#DIV/0!		7	#DIV/0!		7	13,3376		7	#DIV/0!		7	#DIV/0!		

EPSILON =0,0001										Iteraciones 500					
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]															
	N PRO,	2		N PRO,	3		N PRO,	4		N PRO,	5		N PRO,	6	
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		
1	cae		1	13,0846		1	cae		1	cae		1	cae		
2			2	13,6543		2			2			2			
3			3	13,0185		3			3			3			
4			4	12,9987		4			4			4			
5			5	13,5503		5			5			5			
6			6			6			6			6			
7	#DIV/0!		7	13,2613		7	#DIV/0!		7	#DIV/0!		7	#DIV/0!		

EPSILON =0,000001										Iteraciones 10	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	cae		1	11,6156	1	34,0798	1	33,9075	1	37,0983	
2			2	11,5084	2	34,1090	2	34,7714	2	36,8576	
3			3	11,6271	3	34,8628	3	34,0704	3	36,5352	
4			4	11,6913	4	33,6291	4	34,8637	4	38,1339	
5			5	11,8459	5	33,7161	5	35,1440	5	37,0395	
6			6		6		6		6		
7	#DIV/0!		7	11,6577	7	34,0794	7	34,5514	7	37,1329	
8			8		8		8		8		
9			9		9		9		9		
10			10		10		10		10		

EPSILON =0,000001										Iteraciones 50	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	cae		1	13,3273	1	cae	1	cae	1	cae	
2			2	14,0345	2		2		2		
3			3	13,4201	3		3		3		
4			4	13,5324	4		4		4		
5			5	13,3155	5		5		5		
6			6		6		6		6		
7	#DIV/0!		7	13,5260	7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	

EPSILON =0,000001										Iteraciones 100	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	cae		1	cae	1	13,2742	1	cae	1	cae	
2			2		2	13,6163	2		2		
3			3		3	13,3615	3		3		
4			4		4	13,1756	4		4		
5			5		5	13,6619	5		5		
6			6		6		6		6		
7	#DIV/0!		7	#DIV/0!	7	13,4179	7	#DIV/0!	7	#DIV/0!	

EPSILON =0,000001										Iteraciones 500	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2]											
	N PRO,	2	N PRO,	3	N PRO,	4	N PRO,	5	N PRO,	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	cae		1	13,3085	1	cae	1	cae	1	cae	
2			2	13,5009	2		2		2		
3			3	13,3116	3		3		3		
4			4	13,3199	4		4		4		
5			5	13,7016	5		5		5		
6			6		6		6		6		
7	#DIV/0!		7	13,4285	7	#DIV/0!	7	#DIV/0!	7	#DIV/0!	

DATOS DEL ALGORITMO CAJAS RESULTADO CON UF COMPARTIDO EVALUADOS EN DISTINTAS FRECUENCIAS DE ACTUALIZACION

EPSILON =0,000001										Iteraciones 500	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2] A.3 5 actualizaciones											
	N PRO.	2	N PRO.	3	N PRO.	4	N PRO.	5	N PRO.	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	19,0156		1	10,7737	1	8,0969	1	5,8887	1	5,9014	
2	19,2074		2	11,1017	2	7,1358	2	6,0020	2	5,8559	
3	19,3704		3	10,8164	3	7,2293	3	5,9913	3	5,8512	
4	19,0998		4	10,7589	4	7,1754	4	5,9936	4	5,8774	
5	19,6536		5	10,7065	5	7,1583	5	5,8277	5	6,0014	
6			6		6		6		6		
7	19,2694		7	10,8314	7	7,3591	7	5,9407	7	5,8974	
8			8		8		8		8		
9			9		9		9		9		
10			10		10		10		10		

EPSILON =0,000001										Iteraciones 500	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2] A.3 15 actualizaciones											
	N PRO.	2	N PRO.	3	N PRO.	4	N PRO.	5	N PRO.	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	20,8791		1	10,9885	1	7,3515	1	6,0036	1	5,8899	
2	19,2652		2	10,8481	2	7,1910	2	5,8867	2	6,0478	
3	19,0168		3	10,8391	3	7,1379	3	6,0363	3	5,8441	
4	19,3665		4	11,0929	4	7,1806	4	5,8754	4	6,0006	
5	18,9615		5	10,8198	5	7,1089	5	5,9209	5	6,0494	
6			6		6		6		6		
7	19,4978		7	10,9177	7	7,1940	7	5,9446	7	5,9663	
8			8		8		8		8		
9			9		9		9		9		
10			10		10		10		10		

EPSILON =0,000001										Iteraciones 500	
FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2] A.3 15 actualizaciones											
	N PRO.	2	N PRO.	3	N PRO.	4	N PRO.	5	N PRO.	6	
N°	TIEMPO		N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	N°	TIEMPO	
1	19,6101		1	10,7617	1	7,3251	1	6,0376	1	5,8748	
2	19,6533		2	10,7350	2	7,3204	2	5,8721	2	6,0139	
3	18,9408		3	10,7605	3	7,1206	3	6,0232	3	5,8839	
4	19,2197		4	11,1056	4	7,1132	4	6,1849	4	5,8913	
5	19,1844		5	10,8810	5	7,1790	5	5,9071	5	5,8693	
6			6		6		6		6		
7	19,3217		7	10,8488	7	7,2117	7	6,0050	7	5,9066	
8			8		8		8		8		
9			9		9		9		9		
10			10		10		10		10		

DATOS DEL ALGORITMO CAJAS RESULTADO CON UF COMPARTIDO Y BALANCEO DE CARGA DINAMICO ENFOCADO A LA DERECHA EVALUADOS EN DISTINTAS FRECUENCIAS DE ACTUALIZACION

EPSILON =0,000001 Iteraciones 500

FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2] A.4 5 actualizaciones

N PRO.		2	N PRO.		3	N PRO.		4	N PRO.		5	N PRO.		6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	19,3357		1	10,3941		1	7,1156		1	5,6417		1	5,6019	
2	19,3651		2	10,0474		2	7,1516		2	5,6506		2	4,9940	
3	19,3853		3	10,3171		3	7,1702		3	5,5488		3	5,5690	
4	19,7178		4	10,0330		4	7,2012		4	5,5776		4	5,5639	
5	19,1519		5	10,0769		5	7,1594		5	5,5854		5	5,6020	
6			6			6			6			6		
7	19,3912		7	10,1737		7	7,1596		7	5,6008		7	5,4662	
8			8			8			8			8		
9			9			9			9			9		
10			10			10			10			10		

EPSILON =0,000001 Iteraciones 500

FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2] A.4 15 actualizaciones

N PRO.		2	N PRO.		3	N PRO.		4	N PRO.		5	N PRO.		6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	19,3332		1	10,0123		1	5,5853		1	5,5829		1	5,6495	
2	19,5386		2	10,1237		2	5,6040		2	5,5896		2	5,6128	
3	19,3025		3	10,1802		3	5,5870		3	5,5582		3	5,6127	
4	19,4084		4	10,1122		4	5,5750		4	5,5809		4	5,6195	
5	19,1369		5	10,0643		5	5,5688		5	5,5962		5	5,5258	
6			6			6			6			6		
7	19,3439		7	10,0985		7	5,5840		7	5,5815		7	5,6041	
8			8			8			8			8		
9			9			9			9			9		
10			10			10			10			10		

EPSILON =0,000001 Iteraciones 500

FUNCION : GOLDSTEIN_PRICE [-2,2][-2,2] A.4 15 actualizaciones

N PRO.		2	N PRO.		3	N PRO.		4	N PRO.		5	N PRO.		6
N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO		N°	TIEMPO	
1	19,2994		1	10,1258		1	7,1150		1	5,6157		1	5,5670	
2	19,2892		2	10,2150		2	7,1843		2	5,6543		2	5,5891	
3	19,3751		3	10,2864		3	7,2699		3	5,6012		3	5,6073	
4	19,9285		4	10,0821		4	7,1952		4	5,6813		4	5,5992	
5	19,1637		5	10,1193		5	7,1456		5	5,5881		5	5,6224	
6			6			6			6			6		
7	19,4112		7	10,1657		7	7,1820		7	5,6281		7	5,5970	
8			8			8			8			8		
9			9			9			9			9		
10			10			10			10			10		

BIBLIOGRAFÍA

(Moore, 1962) R. E. Moore. Interval Arithmetic and Automatic Error Analysis in Digital Computin. Published as Applied Mathematics and Statistics Laboratories Technical Report No. 25. 1962.

(Hansen, 1992) E. R. Hansen. Global optimization using interval analysis. Marcel Dekker, Inc., Ney York, 1992.

(Leclerc, 1992) A. P. Leclerc. Efficient and reliable global optimization. PhD thesis, The Ohio State University, 1992.

(Moore, 1966) R. E. Moore. Interval analysis. Prentice-Hll, Englewoods Cliffs, N. J., 1966.

(Van Iwaarden, 1996) R. J. Van Iwaarden. An improved unconstrained global optimization algorithm. PhD thesis, University of Colorado at Denver, 1996.

(Jaulin et.al., 2001) Jaulin, L., N. Kieffer, O. Didrit and E. Walter, Applied Interval Analysis, Springer-Verlag, London, 2001.

(Hales, 2005) T.C. Hales. A proof of the Kepler conjecture, Ann.Math., vol. 162., 2005.

(Lee y Mavroidis, 2002) E. Lee and C. Mavroidis. Solving the geometric design problem of spatial 3R robot manipulators using polynomial homotopy continuation, J. Mech. Design, Trans, 2002.

(Balakrishnan y Boyd, 1992) V. Balakrishnan and S. Boyd, Global Optimization in Control System Analysis and De-sign, Control and Dynamic Systems: AdvancesinTheoryandApplications, Academic Press, New York, 1992.

(Moore, 1991) R. E. Moore. Global optimization to prescribed accuracy. Computers

Math. Applic., 21:25-39, 1991.

(Pardalos y Vavasis, 1991) P. M. Pardalos y S. A. Vavasis. Quadratic programming with one negative eigenvalue is np-hard. Journal of Global Optimization, 1991.

(Luenberg, 1989) D. E. Luenberger. Programación lineal y no lineal. Addison-Wesley Iberoamericana, 1989.

(Andersen, Jennings y Ryan, 1972) L. S. Andersen, R. S. Jennings y D. M. Ryan. Global optimization. En R. S. Andersen, editor, Optimización. University of Queensland Press, 1972.

(Dekkers y Aarts, 1982) A. Dekkers y E. Aarts. Global optimization and simulated annealing. Mathematical programming, 1982.

(Rinnooy Kan y Timmer, 1984) H. G. Rinnooy Kan y G. T. Timmer. Stochastic methods for global optimization. American Journal of Methematical and Management Sciences, 1984.

(Skelboe, 1974) S. Skelboe. Computation of rational interval functions, 1974.

(Quinn, 2004) M. J. Quinn. Parallel programming in C with MPI and OpenMP. McGraw-Hill 2004

(Wilkinson, 2005) B. Wilkinson y M. Allen. Parallel programming. Segunda edición. Prentice Hall, 2005.

(García, 2003) Oscar Rafael García Regis, Enrique Cruz Martinez, Humberto Carrillo Calvet. Computación Paralela, 2003

(Rodríguez, 2004) Pedro Rodríguez, Dino Risso Rocco. Computación Paralela. Universidad del Bio-Bío, 9 de enero de 2004.

(Jorge, 2007) Javier Jorge, Alberto Villafañe. Introducción al Clustering con MPI. 1er. Ciclo de Conferencias Abiertas de Ingeniería en Computación Universidad Nacional de Córdova, 2007

(Flores, 2004) Antonio Flores Gil. Introducción a las arquitecturas paralelas. Universidad de valencia, 2004

(Cardinale, 2002) Yudith Cardinale, Mariela Curiel. Memoria Compartida Distribuida. Universidad simón bolívar, 2002

(Acebrón, 2005) Juan Antonio Acebrón de Torres. Multicomputadores. Universidad de Alcalá, 2005

(IBM, 1994) SP2 High Performance Switch Overview. IBM Corporation © Copyright, 1994

(Guillén, 2006)Dr. Pablo Guillén. Introducción a la Computación Paralela. CESIMO. Universidad de Los Andes, 2006

(Llorente, 2002) Ignacio martín Llorente. Tipos de aplicaciones paralelas. Universidad Complutense de Madrid, 2002

(Foster, 1995) Ian Foster. Designing and building parallel programs: Concepts and Tools for Parallel Software Engineering Addison-Wesley, 1995

(Badía, 2006) José Manuel Badía. Metodología de diseño de algoritmos paralelos. Escuela Superior de Tecnología y Ciencias Experimentales, 2006

(Rodríguez, 2007) Pedro Rodríguez Moreno Departamento de Sistemas de Información, Dino Risso Rocco Departamento de Física. Asignatura: Programación paralela. Universidad del Bio-Bio, 2007

(Medina, 2007) Andrés C. Medina. Instalación y Configuración de un cluster. Facultad de Ciencias, Departamento de Matemáticas Ingeniería Estadística. Univ.del Bio-Bio, 2007.

(Bernal, 2005) Iván Bernal C., David Mejía N. y Diego Fernández A. Desarrollo de Aplicaciones Paralelas para clusters utilizando MPI (Message Passing Interface). Escuela Politécnica Nacional Quito-Ecuador, 2005

(Ríos, 2007) Genghis Ríos. Clusters Linux, Grids Computacionales y el proyecto EELA. Dirección de Informática Académica. Pontificia Universidad Católica del Perú, 2007

(Sánchez, 2008) Alberto Sánchez Campos, Marcos Novalbos Mendiguchía. Clusters de computadores personales. Ampliación de Arquitectura de Computadores. Universidad Rey Juan Carlos España, 2008

(Foster, 1999) Ian Foster ,Carl Kesselmann. The Grid, Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers, 1999

(Fernández, 2006) Rocio Fernández Soto, Álvaro García Treviño. Super Computadores y

Computación Grid. Departamento de Lenguajes y Ciencias de la Computación. Universidad de Malaga, 2006.

(Escudero, 2003) José Joaquín Escudero Garzás. Grid e IPv6. Departamento Teoría de la Señal y Comunicaciones. Universidad CARLOS III de Madrid, 2003

(Decaer, 2001)Thomas Decaer. Dynamic Load Balancing and Scheduling.University of Paderborn Germany 2001

(Flynn,1972) M Flynn. Some Computer Organizations and Their Effectiveness. IEEE Trans. Comput., Vol. C-21. 1972.

(Duncan,1990) Ralph Duncan. A Survey of Parallel Computer Architectures. IEEE Computer. 1990

LINKOGRAFIA

(Barney, 2007) Blaise Barney. Introduction to Parallel Computing, 2007

https://computing.llnl.gov/tutorials/parallel_comp/

(Balum, 2008) Definición de toroide

http://es.wikipedia.org/wiki/Balun#Balun_toroidal

(Panadés, 1991)Jose Panádes Martínez, Violeta Migallón Gomis, Josep Arnal Garcia, Maria Jesús Castel de Haro, Sergio Ramon Ferry.Grupo de Computación de Altas Prestaciones y Paralelismo. Universidad de Alicante, 1991

http://www.dccia.ua.es/cp/

(Ardenghi, 2007) Jorge Ardenghi. Introducción: Computación Paralela y Distribuida. Universidad Nacional del Sur, 2007

http://cs.uns.edu.ar/

(AT&T, 2008)Definición de AT&T

http://es.wikipedia.org/wiki/ATT

(Guerrero, 2004) Fernando Guerrero Tala. Curso: Lenguaje C++. Guía para programadores, 2004

http://www.mailxmail.com/curso/informatica/cplusplus2/capitulo1.htm

Blaise Barney. Message Passing Interface (MPI), 2007

(Barney2, 2007)

https://computing.llnl.gov/tutorials/mpi/

(Rocks, 2008)

http://www.rocksclusters.org

(Oscar, 2008)

http://oscar.openclustergroup.org/

(Globus, 2008)

http://www.globus.org/

http://www-unix.globus.org/toolkit/

(Quiñónez, 2005)Marco Antonio Salas Quiñónez. Sistemas Operativos Distribuidos. Unidad VI Administración de recursos. Instituto Tecnológico de Los Mochis 2005

http://mx.geocities.com/lyly_sak/sod.html