Predicción de mínimos en algoritmos paralelos de Optimización Global Intervalar

Jose Luis Berenguel Gómez

PROYECTO FIN DE CARRERA
dirigido por Leocadio González Casado

Universidad de Almería. Septiembre de 2007

Índice de contenidos

Optimización Global

El problema

$S \subset \mathbb{R}^n$ *: una caja en la región de búsqueda.*

$f(x) : \mathbb{R}^n \rightarrow \mathbb{R}$ *: la función objetivo.*

Encontrar:

$$f^* = \min_{x \in S}\{f(x)\}$$

$$y$$

$$x^* = \{x \in S : \ f(x) = f^*\},$$

Optimización Global

El problema

$S \subset \mathbb{R}^n$ *: una caja en la región de búsqueda.*

$f(x) : \mathbb{R}^n \rightarrow \mathbb{R}$ *: la función objetivo.*

Encontrar:

$$f^* = \min_{x \in S}\{f(x)\}$$

$$y$$

$$x^* = \{x \in S : \ f(x) = f^*\},$$

Reglas básicas de los algoritmos de Ramificación y Acotación

Los algoritmos de Ramificación y Acotación se caracterizan por cinco reglas:

Acotación: *Calcula un límite inferior de la solución óptima para cada subproblema en problemas de minimización.*

Extensión natural a intervalos de f

Selección: *Establece qué nodo o subproblema será el siguiente en dividirse.*

Seleccionar primero la caja con menor límite inferior

Eliminación: *Establece cómo reconocer y eliminar subproblemas donde no se encuentra la solución óptima del problema original.*

Test del Punto Medio y Test de Corte

División: *Establece cómo se subdivide el problema original en problemas más pequeños.*

Bisección

Terminación: *Determina cuándo un nodo pertenece a la solución final.* $w(X) < \epsilon$

Reglas básicas de los algoritmos de Ramificación y Acotación

Los algoritmos de Ramificación y Acotación se caracterizan por cinco reglas:

Acotación: *Calcula un límite inferior de la solución óptima para cada subproblema en problemas de minimización.*

Extensión natural a intervalos de f

Selección: *Establece qué nodo o subproblema será el siguiente en dividirse.*

Seleccionar primero la caja con menor límite inferior

Eliminación: *Establece cómo reconocer y eliminar subproblemas donde no se encuentra la solución óptima del problema original.*

Test del Punto Medio y Test de Corte

División: *Establece cómo se subdivide el problema original en problemas más pequeños.*

Bisección

Terminación: *Determina cuándo un nodo pertenece a la solución final.* $w(X) < \epsilon$

Reglas básicas de los algoritmos de Ramificación y Acotación

Los algoritmos de Ramificación y Acotación se caracterizan por cinco reglas:

Acotación: *Calcula un límite inferior de la solución óptima para cada subproblema en problemas de minimización.*

Extensión natural a intervalos de f

Selección: *Establece qué nodo o subproblema será el siguiente en dividirse.*

Seleccionar primero la caja con menor límite inferior

Eliminación: *Establece cómo reconocer y eliminar subproblemas donde no se encuentra la solución óptima del problema original.*

Test del Punto Medio y Test de Corte

División: *Establece cómo se subdivide el problema original en problemas más pequeños.*

Bisección

Terminación: *Determina cuándo un nodo pertenece a la solución final.* $w(X) < \epsilon$

Reglas básicas de los algoritmos de Ramificación y Acotación

Los algoritmos de Ramificación y Acotación se caracterizan por cinco reglas:

Acotación: *Calcula un límite inferior de la solución óptima para cada subproblema en problemas de minimización.*

Extensión natural a intervalos de f

Selección: *Establece qué nodo o subproblema será el siguiente en dividirse.*

Seleccionar primero la caja con menor límite inferior

Eliminación: *Establece cómo reconocer y eliminar subproblemas donde no se encuentra la solución óptima del problema original.*

Test del Punto Medio y Test de Corte

División: *Establece cómo se subdivide el problema original en problemas más pequeños.*

Bisección

Terminación: *Determina cuándo un nodo pertenece a la solución final.* $w(X) < \epsilon$

Reglas básicas de los algoritmos de Ramificación y Acotación

Los algoritmos de Ramificación y Acotación se caracterizan por cinco reglas:

Acotación: *Calcula un límite inferior de la solución óptima para cada subproblema en problemas de minimización.*

Extensión natural a intervalos de f

Selección: *Establece qué nodo o subproblema será el siguiente en dividirse.*

Seleccionar primero la caja con menor límite inferior

Eliminación: *Establece cómo reconocer y eliminar subproblemas donde no se encuentra la solución óptima del problema original.*

Test del Punto Medio y Test de Corte

División: *Establece cómo se subdivide el problema original en problemas más pequeños.*

Bisección

Terminación: *Determina cuándo un nodo pertenece a la solución final.* $w(X) < \epsilon$

Reglas básicas de los algoritmos de Ramificación y Acotación

Los algoritmos de Ramificación y Acotación se caracterizan por cinco reglas:

Acotación: *Calcula un límite inferior de la solución óptima para cada subproblema en problemas de minimización.*

Extensión natural a intervalos de f

Selección: *Establece qué nodo o subproblema será el siguiente en dividirse.*

Seleccionar primero la caja con menor límite inferior

Eliminación: *Establece cómo reconocer y eliminar subproblemas donde no se encuentra la solución óptima del problema original.*

Test del Punto Medio y Test de Corte

División: *Establece cómo se subdivide el problema original en problemas más pequeños.*

Bisección

Terminación: *Determina cuándo un nodo pertenece a la solución final.* $w(X) < \epsilon$

Reglas básicas de los algoritmos de Ramificación y Acotación

Los algoritmos de Ramificación y Acotación se caracterizan por cinco reglas:

Acotación: *Calcula un límite inferior de la solución óptima para cada subproblema en problemas de minimización.*

Extensión natural a intervalos de f

Selección: *Establece qué nodo o subproblema será el siguiente en dividirse.*

Seleccionar primero la caja con menor límite inferior

Eliminación: *Establece cómo reconocer y eliminar subproblemas donde no se encuentra la solución óptima del problema original.*

Test del Punto Medio y Test de Corte

División: *Establece cómo se subdivide el problema original en problemas más pequeños.*

Bisección

Terminación: *Determina cuándo un nodo pertenece a la solución final.* $w(X) < \epsilon$

Reglas básicas de los algoritmos de Ramificación y Acotación

Los algoritmos de Ramificación y Acotación se caracterizan por cinco reglas:

Acotación: *Calcula un límite inferior de la solución óptima para cada subproblema en problemas de minimización.*

Extensión natural a intervalos de f

Selección: *Establece qué nodo o subproblema será el siguiente en dividirse.*

Seleccionar primero la caja con menor límite inferior

Eliminación: *Establece cómo reconocer y eliminar subproblemas donde no se encuentra la solución óptima del problema original.*

Test del Punto Medio y Test de Corte

División: *Establece cómo se subdivide el problema original en problemas más pequeños.*

Bisección

Terminación: *Determina cuándo un nodo pertenece a la solución final.* $w(X) < \epsilon$

Reglas básicas de los algoritmos de Ramificación y Acotación

Los algoritmos de Ramificación y Acotación se caracterizan por cinco reglas:

Acotación: *Calcula un límite inferior de la solución óptima para cada subproblema en problemas de minimización.*

Extensión natural a intervalos de f

Selección: *Establece qué nodo o subproblema será el siguiente en dividirse.*

Seleccionar primero la caja con menor límite inferior

Eliminación: *Establece cómo reconocer y eliminar subproblemas donde no se encuentra la solución óptima del problema original.*

Test del Punto Medio y Test de Corte

División: *Establece cómo se subdivide el problema original en problemas más pequeños.*

Bisección

Terminación: *Determina cuándo un nodo pertenece a la solución final.* $w(X) < \epsilon$

Reglas básicas de los algoritmos de Ramificación y Acotación

Los algoritmos de Ramificación y Acotación se caracterizan por cinco reglas:

Acotación: *Calcula un límite inferior de la solución óptima para cada subproblema en problemas de minimización.*

Extensión natural a intervalos de f

Selección: *Establece qué nodo o subproblema será el siguiente en dividirse.*

Seleccionar primero la caja con menor límite inferior

Eliminación: *Establece cómo reconocer y eliminar subproblemas donde no se encuentra la solución óptima del problema original.*

Test del Punto Medio y Test de Corte

División: *Establece cómo se subdivide el problema original en problemas más pequeños.*

Bisección

Terminación: *Determina cuándo un nodo pertenece a la solución final.* $w(X) < \epsilon$

Reglas básicas de los algoritmos de Ramificación y Acotación

Los algoritmos de Ramificación y Acotación se caracterizan por cinco reglas:

Acotación: *Calcula un límite inferior de la solución óptima para cada subproblema en problemas de minimización.*

Extensión natural a intervalos de f

Selección: *Establece qué nodo o subproblema será el siguiente en dividirse.*

Seleccionar primero la caja con menor límite inferior

Eliminación: *Establece cómo reconocer y eliminar subproblemas donde no se encuentra la solución óptima del problema original.*

Test del Punto Medio y Test de Corte

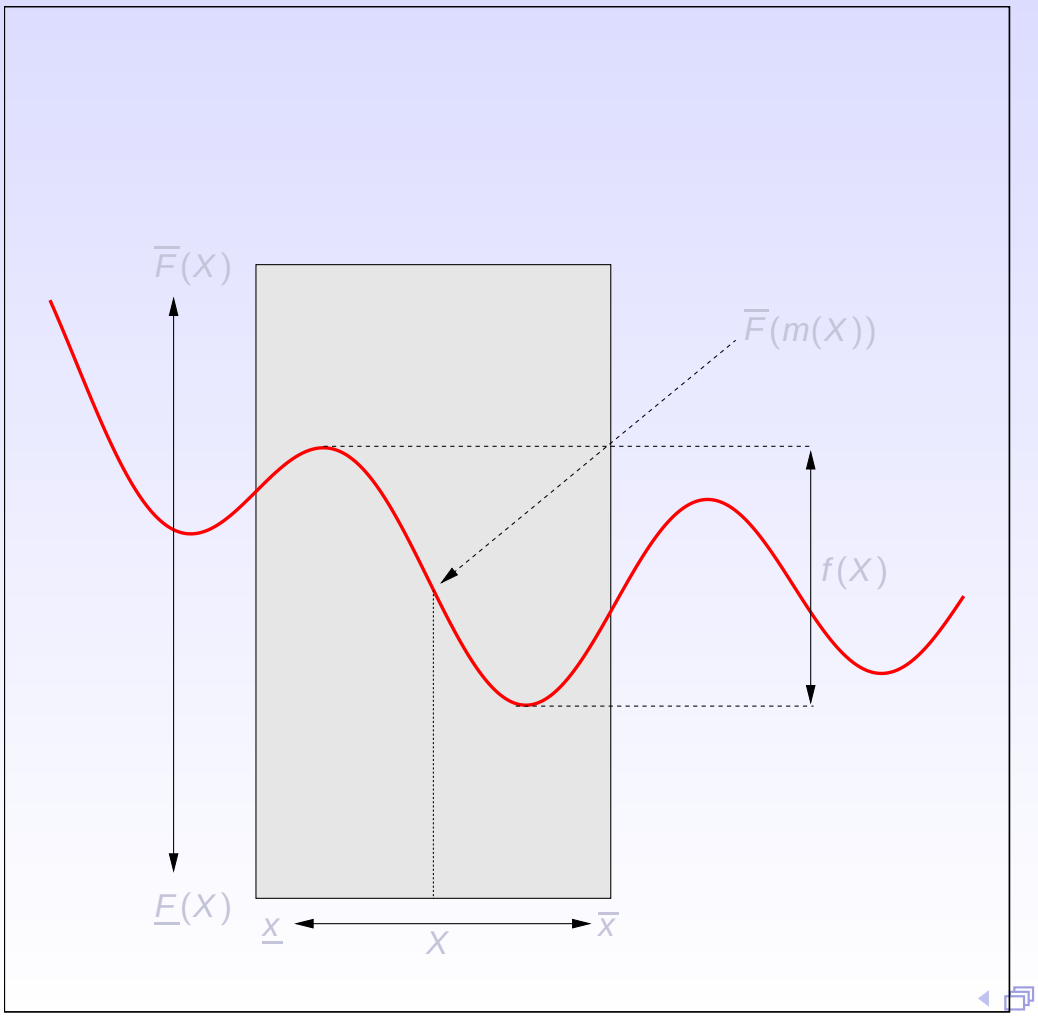
División: *Establece cómo se subdivide el problema original en problemas más pequeños.*

Bisección

Terminación: *Determina cuándo un nodo pertenece a la solución final.* $w(X) < \epsilon$

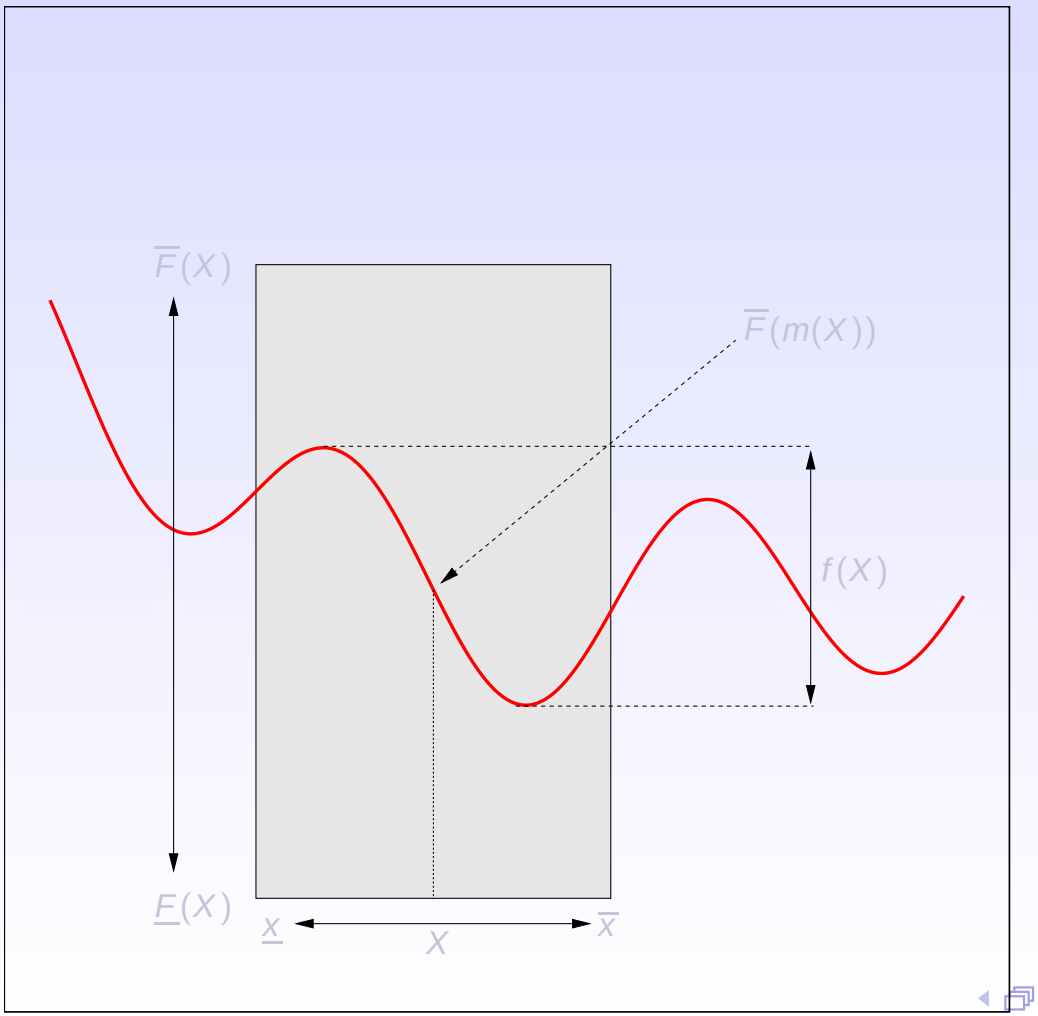
Breve introducción a la Aritmética de Intervalos

- La Aritmética de Intervalos surge para evitar errores de redondeo.
- Estos errores se producen debido a que la representación de números reales en un ordenador es finita (IEEE 754).
- La Aritmética de Intervalos se atribuye a Moore en 1966.



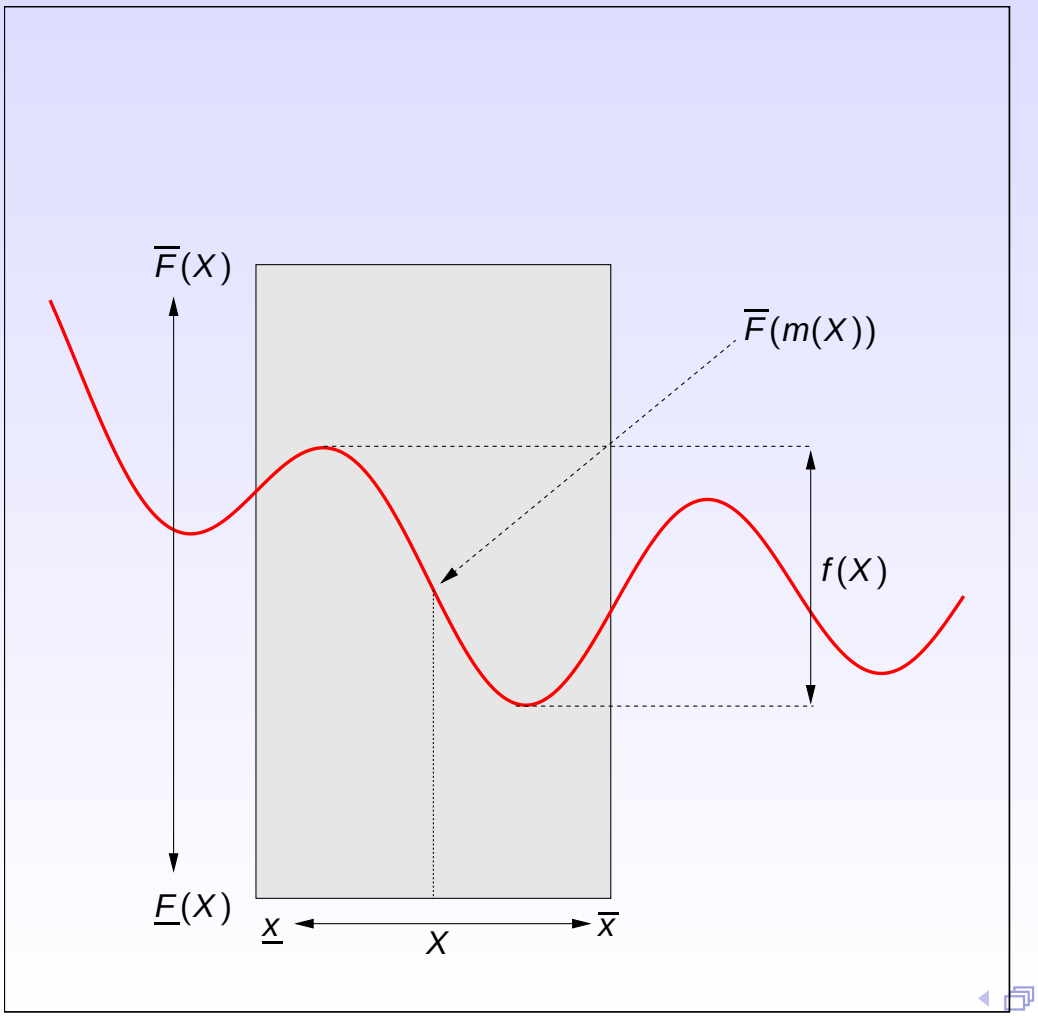
Breve introducción a la Aritmética de Intervalos

- La Aritmética de Intervalos surge para evitar errores de redondeo.
- Estos errores se producen debido a que la representación de números reales en un ordenador es finita (IEEE 754).
- La Aritmética de Intervalos se atribuye a Moore en 1966.



Breve introducción a la Aritmética de Intervalos

- La Aritmética de Intervalos surge para evitar errores de redondeo.
- Estos errores se producen debido a que la representación de números reales en un ordenador es finita (IEEE 754).
- La Aritmética de Intervalos se atribuye a Moore en 1966.



Inexistencia de mínimos globales

Test del Punto Medio

Supongamos que hemos obtenido un límite superior del mínimo global (f^*) al que llamaremos $\overline{f^*}$.
Para una caja $X \subset S$, $\overline{f^*} < \underline{F}(X)$, entonces no existe un mínimo en X.
El valor de $\overline{f^*}$ se obtiene evaluando el punto medio de las cajas generadas.

Test de Corte

$\overline{f^*}$ obtenido mediante el Test del Punto Medio
Todas las cajas $X^i \mid \overline{f^*} < \underline{F}(X)$ son eliminadas.

Test de Monotonía

Si $0 \notin F_i'(X)$ para algún $i \in 1, \ldots, n$, entonces no hay ningún punto estacionario en X.

Inexistencia de mínimos globales

Test del Punto Medio

Supongamos que hemos obtenido un límite superior del mínimo global (f^*) al que llamaremos $\overline{f^*}$.
Para una caja $X \subset S$, $\overline{f^*} < \underline{F}(X)$, entonces no existe un mínimo en X.
El valor de $\overline{f^*}$ se obtiene evaluando el punto medio de las cajas generadas.

Test de Corte

$\overline{f^*}$ obtenido mediante el Test del Punto Medio
Todas las cajas $X^i \mid \overline{f^*} < \underline{F}(X)$ son eliminadas.

Test de Monotonía

Si $0 \notin F_i'(X)$ para algún $i \in 1, \ldots, n$, entonces no hay ningún punto estacionario en X.

Inexistencia de mínimos globales

Test del Punto Medio

Supongamos que hemos obtenido un límite superior del mínimo global (f^*) al que llamaremos $\overline{f^*}$.
Para una caja $X \subset S$, $\overline{f^*} < \underline{F}(X)$, entonces no existe un mínimo en X.
El valor de $\overline{f^*}$ se obtiene evaluando el punto medio de las cajas generadas.

Test de Corte

$\overline{f^*}$ obtenido mediante el Test del Punto Medio
Todas las cajas $X^i \mid \overline{f^*} < \underline{F}(X)$ son eliminadas.

Test de Monotonía

Si $0 \notin F_i'(X)$ para algún $i \in 1, \ldots, n$, entonces no hay ningún punto estacionario en X.

Algoritmo básico de Optimización Global

Funct IGO(S, f)

1. Inicializamos la lista de trabajo $L := \{S\}$
2. Inicializamos la lista final $Q := \{\}$
3. **while** ($L \neq \{\}$)
4. Seleccionamos un intervalo X en L — *Regla de Selección*
5. **if** X no puede eliminarse — *Regla de eliminación*
6. Dividimos X generando X^i, $i = 1 \ldots s$ — *Regla de división*
7. **if** X^i satisface el criterio de terminación — *Regla de terminación*
8. Almacenar X^i en Q
9. **else**
10. Almacenar X^i en L
11. **return** Q

Descripición y motivaciones

- El proyecto presenta un nuevo algoritmo de balanceo de la carga basado en la predicción de mínimos.
- Se pretende crear regiones alrededor de los mínimos, cuya localización no se conoce a priori.
- Las regiones serán enviadas a procesadores libres.
- Para ello, hemos construido un modelo paralelo simulado.

Descripición y motivaciones

- El proyecto presenta un nuevo algoritmo de balanceo de la carga basado en la predicción de mínimos.
- Se pretende crear regiones alrededor de los mínimos, cuya localización no se conoce a priori.
- Las regiones serán enviadas a procesadores libres.
- Para ello, hemos construido un modelo paralelo simulado.

Descripición y motivaciones

- El proyecto presenta un nuevo algoritmo de balanceo de la carga basado en la predicción de mínimos.
- Se pretende crear regiones alrededor de los mínimos, cuya localización no se conoce a priori.
- Las regiones serán enviadas a procesadores libres.
- Para ello, hemos construido un modelo paralelo simulado.

Descripición y motivaciones

- El proyecto presenta un nuevo algoritmo de balanceo de la carga basado en la predicción de mínimos.
- Se pretende crear regiones alrededor de los mínimos, cuya localización no se conoce a priori.
- Las regiones serán enviadas a procesadores libres.
- Para ello, hemos construido un modelo paralelo simulado.

Descripción y motivaciones

La creación de regiones nos puede permitir:

- Estimar mejor la carga computacional del procesador.
- Disponer de procesadores libres para otras tareas.
- Mejorar las técnicas de optimización basándonos en la proximidad de las cajas.

Objetivo general:

Mejorar la eficiencia de los algoritmos paralelos de Optimización Global

Descripción y motivaciones

La creación de regiones nos puede permitir:

- Estimar mejor la carga computacional del procesador.
- Disponer de procesadores libres para otras tareas.
- Mejorar las técnicas de optimización basándonos en la proximidad de las cajas.

Objetivo general:

Mejorar la eficiencia de los algoritmos paralelos de Optimización Global

Descripción y motivaciones

La creación de regiones nos puede permitir:

- Estimar mejor la carga computacional del procesador.
- Disponer de procesadores libres para otras tareas.
- Mejorar las técnicas de optimización basándonos en la proximidad de las cajas.

Objetivo general:

Mejorar la eficiencia de los algoritmos paralelos de Optimización Global

Descripción y motivaciones

La creación de regiones nos puede permitir:

- Estimar mejor la carga computacional del procesador.
- Disponer de procesadores libres para otras tareas.
- Mejorar las técnicas de optimización basándonos en la proximidad de las cajas.

Objetivo general:

Mejorar la eficiencia de los algoritmos paralelos de Optimización Global

Paralelización

- La paralelización de los algoritmos pretende reducir el tiempo de ejecución de la versión secuencial y resolver instancias de los problemas que no pueden resolverse en tiempo razonable.

¿Por qué se ha simulado el entorno paralelo?

- Debemos disponer de los recursos hardware necesarios.
- La implementación debe tener en cuenta las diferencias entre arquitecturas.
- Es necesario utilizar un lenguaje de programación paralelo.

Solución: Simular la ejecución paralela en un procesador

- Porque simplifica enormemente la implementación y las pruebas.
- Porque nos permite tener una visión global durante la ejecución.
- Porque es un entorno más controlado, lo que facilita el estudio de los resultados.

Paralelización

- La paralelización de los algoritmos pretende reducir el tiempo de ejecución de la versión secuencial y resolver instancias de los problemas que no pueden resolverse en tiempo razonable.

¿Por qué se ha simulado el entorno paralelo?

- Debemos disponer de los recursos hardware necesarios.
- La implementación debe tener en cuenta las diferencias entre arquitecturas.
- Es necesario utilizar un lenguaje de programación paralelo.

Solución: Simular la ejecución paralela en un procesador

- Porque simplifica enormemente la implementación y las pruebas.
- Porque nos permite tener una visión global durante la ejecución.
- Porque es un entorno más controlado, lo que facilita el estudio de los resultados.

Paralelización

- La paralelización de los algoritmos pretende reducir el tiempo de ejecución de la versión secuencial y resolver instancias de los problemas que no pueden resolverse en tiempo razonable.

¿Por qué se ha simulado el entorno paralelo?

- Debemos disponer de los recursos hardware necesarios.
- La implementación debe tener en cuenta las diferencias entre arquitecturas.
- Es necesario utilizar un lenguaje de programación paralelo.

Solución: Simular la ejecución paralela en un procesador

- Porque simplifica enormemente la implementación y las pruebas.
- Porque nos permite tener una visión global durante la ejecución.
- Porque es un entorno más controlado, lo que facilita el estudio de los resultados.

Paralelización

- La paralelización de los algoritmos pretende reducir el tiempo de ejecución de la versión secuencial y resolver instancias de los problemas que no pueden resolverse en tiempo razonable.

¿Por qué se ha simulado el entorno paralelo?

- Debemos disponer de los recursos hardware necesarios.
- La implementación debe tener en cuenta las diferencias entre arquitecturas.
- Es necesario utilizar un lenguaje de programación paralelo.

Solución: Simular la ejecución paralela en un procesador

- Porque simplifica enormemente la implementación y las pruebas.
- Porque nos permite tener una visión global durante la ejecución.
- Porque es un entorno más controlado, lo que facilita el estudio de los resultados.

Paralelización

- La paralelización de los algoritmos pretende reducir el tiempo de ejecución de la versión secuencial y resolver instancias de los problemas que no pueden resolverse en tiempo razonable.

¿Por qué se ha simulado el entorno paralelo?

- Debemos disponer de los recursos hardware necesarios.
- La implementación debe tener en cuenta las diferencias entre arquitecturas.
- Es necesario utilizar un lenguaje de programación paralelo.

Solución: Simular la ejecución paralela en un procesador

- Porque simplifica enormemente la implementación y las pruebas.
- Porque nos permite tener una visión global durante la ejecución.
- Porque es un entorno más controlado, lo que facilita el estudio de los resultados.

Características del programa ogsp

- El programa ogsp (*Optimización Global mediante Simulación Paralela*) simula la ejecución de un algoritmo paralelo de OG en un procesador.

Parámetros configurables

- N^o de procesadores
- Tasa de Transferencia de la Red
- Latencia
- Algoritmo de balanceo

Características del programa ogsp

- El programa ogsp (*Optimización Global mediante Simulación Paralela*) simula la ejecución de un algoritmo paralelo de OG en un procesador.

Parámetros configurables

- N^o de procesadores
- Tasa de Transferencia de la Red
- Latencia
- Algoritmo de balanceo

Características del programa ogsp

- El programa ogsp (*Optimización Global mediante Simulación Paralela*) simula la ejecución de un algoritmo paralelo de OG en un procesador.

Parámetros configurables

- N^o de procesadores
- Tasa de Transferencia de la Red
- Latencia
- Algoritmo de balanceo

Características del programa ogsp

- El programa ogsp (*Optimización Global mediante Simulación Paralela*) simula la ejecución de un algoritmo paralelo de OG en un procesador.

Parámetros configurables

- N^o de procesadores
- Tasa de Transferencia de la Red
- Latencia
- Algoritmo de balanceo

Características del programa ogsp

- El programa ogsp (*Optimización Global mediante Simulación Paralela*) simula la ejecución de un algoritmo paralelo de OG en un procesador.

Parámetros configurables

- N^o de procesadores
- Tasa de Transferencia de la Red
- Latencia
- Algoritmo de balanceo

Selección del procesador que debe entrar en ejecución

El simulador permite a cada procesador una iteración del bucle del Algoritmo de OG secuencial

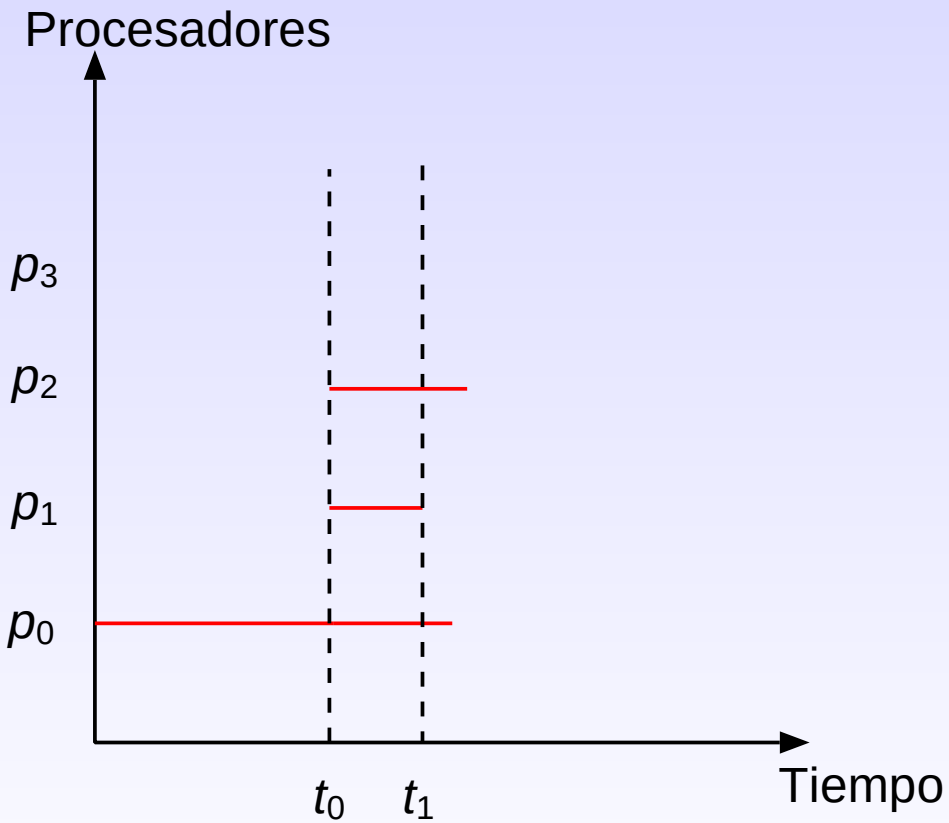


Figura: Selección del procesador que debe entrar en ejecución

Cálculo de tiempos

El simulador evita conflictos ya que no habrá dos procesadores balanceando a la vez

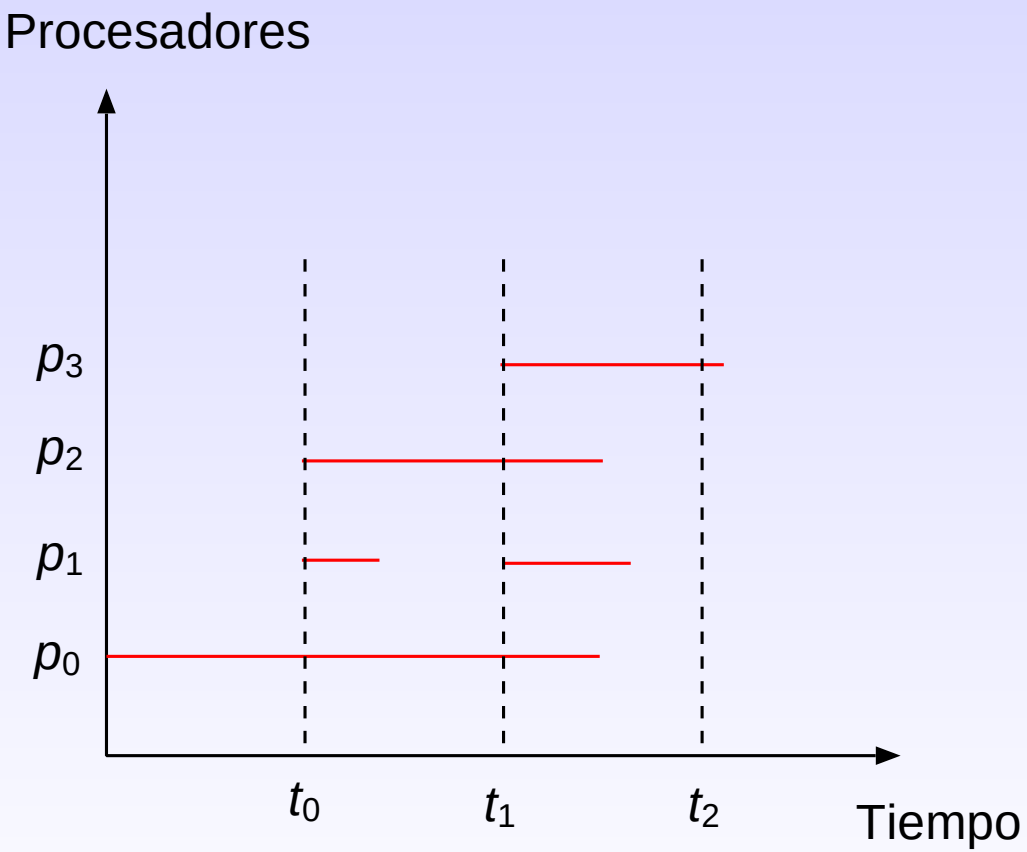


Figura: Cálculo del tiempo de ejecución global de la simulación

Cálculo de tiempos

- Tiempo de creación de un proceso. Opción `-tc` (100ms).
- Tiempo de proceso de una caja.
  $t_{pc} = \frac{tiempo\_en\_procesar\_n\_cajas}{nb}$. Opción `-nb` (200 cajas).
- Tiempo de envío de n cajas a un procesador.
  $t_{ec}(n) = latencia + \frac{n \cdot sizeof(caja)}{ttr}$.
- Tiempo de ejecución total de un procesador.
- Tiempo de ejecución global de la simulación.

Cálculo de tiempos

- Tiempo de creación de un proceso. Opción `-tc` (100ms).
- Tiempo de proceso de una caja.
  $t_{pc} = \frac{tiempo\_en\_procesar\_n\_cajas}{nb}$. Opción `-nb` (200 cajas).
- Tiempo de envío de n cajas a un procesador.
  $t_{ec}(n) = latencia + \frac{n \cdot sizeof(caja)}{ttr}$.
- Tiempo de ejecución total de un procesador.
- Tiempo de ejecución global de la simulación.

Cálculo de tiempos

- Tiempo de creación de un proceso. Opción `-tc` (100ms).
- Tiempo de proceso de una caja.
  $t_{pc} = \frac{tiempo\_en\_procesar\_n\_cajas}{nb}$. Opción `-nb` (200 cajas).
- Tiempo de envío de n cajas a un procesador.
  $t_{ec}(n) = latencia + \frac{n \cdot sizeof(caja)}{ttr}$.
- Tiempo de ejecución total de un procesador.
- Tiempo de ejecución global de la simulación.

Cálculo de tiempos

- Tiempo de creación de un proceso. Opción `-tc` (100ms).
- Tiempo de proceso de una caja.
  $t_{pc} = \frac{tiempo\_en\_procesar\_n\_cajas}{nb}$. Opción `-nb` (200 cajas).
- Tiempo de envío de n cajas a un procesador.
  $t_{ec}(n) = latencia + \frac{n \cdot sizeof(caja)}{ttr}$.
- Tiempo de ejecución total de un procesador.
- Tiempo de ejecución global de la simulación.

Cálculo de tiempos

- Tiempo de creación de un proceso. Opción `-tc` (100ms).
- Tiempo de proceso de una caja.
  $t_{pc} = \frac{tiempo\_en\_procesar\_n\_cajas}{nb}$. Opción `-nb` (200 cajas).
- Tiempo de envío de n cajas a un procesador.
  $t_{ec}(n) = latencia + \frac{n \cdot sizeof(caja)}{ttr}$.
- Tiempo de ejecución total de un procesador.
- Tiempo de ejecución global de la simulación.

Algoritmos de balanceo usados en los resultados

Balanceo de la carga en Baraja (BAR)

Consiste en repartir las cajas de la lista de trabajo de forma alternativa cuando cualquier procesador detecta que hay un procesador libre.

Balanceo de la carga basado en regiones (MPR)

Un procesador genera regiones de cajas cuando detecta que hay otro libre. Deben existir almenos dos regiones para realizar el balanceo.

Decisión sobre el balanceo

Hemos incluido una toma de decisión a los algoritmos de balanceo para establecer cuándo es adecuado balancear la carga, basándonos en una estimación de la carga computacional presente en el procesador. A estos algoritmos los hemos llamado BARD y MPRD.

Algoritmos de balanceo usados en los resultados

Balanceo de la carga en Baraja (BAR)

Consiste en repartir las cajas de la lista de trabajo de forma alternativa cuando cualquier procesador detecta que hay un procesador libre.

Balanceo de la carga basado en regiones (MPR)

Un procesador genera regiones de cajas cuando detecta que hay otro libre. Deben existir almenos dos regiones para realizar el balanceo.

Decisión sobre el balanceo

Hemos incluido una toma de decisión a los algoritmos de balanceo para establecer cuándo es adecuado balancear la carga, basándonos en una estimación de la carga computacional presente en el procesador. A estos algoritmos los hemos llamado BARD y MPRD.

Algoritmos de balanceo usados en los resultados

Balanceo de la carga en Baraja (BAR)

Consiste en repartir las cajas de la lista de trabajo de forma alternativa cuando cualquier procesador detecta que hay un procesador libre.

Balanceo de la carga basado en regiones (MPR)

Un procesador genera regiones de cajas cuando detecta que hay otro libre. Deben existir almenos dos regiones para realizar el balanceo.

Decisión sobre el balanceo

Hemos incluido una toma de decisión a los algoritmos de balanceo para establecer cuándo es adecuado balancear la carga, basándonos en una estimación de la carga computacional presente en el procesador. A estos algoritmos los hemos llamado BARD y MPRD.

Algoritmo de balanceo de la carga en Baraja

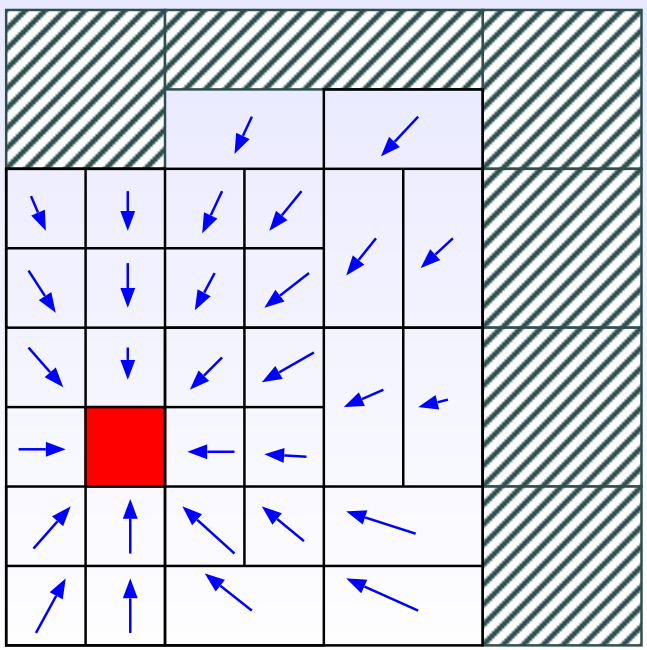
Funct Baraja(P_L,L,L_1,L_2)

1. $contador := \emptyset$
2. **if** ($numeroElementos(L) < 2$) — *No hay cajas suficientes para repartir*
3. **return** 0
4. **while** ($L \neq \{\emptyset\}$) — *Repartimos las cajas*
5. $X := seleccionaCaja(L)$ — $\underline{F}(X) = min(\underline{F}(X^i), \forall X^i \in L$
6. $L := L - \{X\}$
7. **if** ($contador \%2 = 0$)
8. $L_1 := L_1 + \{X\}$ — *X a la lista* L_1
9. **else**
10. $L_2 := L_2 + \{X\}$ — *X a la lista* L_2
11. $contador := contador + 1$
12. $MoverCajas(P_L, L_2)$ — *L_2 se envía al procesador libre*
13. $L := L_1$ — *L_1 se queda en el procesador que ejecuta el balanceo*
14. **return** $numeroElementos(L_2)$

Descripción de la predicción de mínimos

Se pretende generar regiones de cajas próximas entre sí, cuyos centros de atracción tratarán de estar localizados sobre los mínimos de la función objetivo.

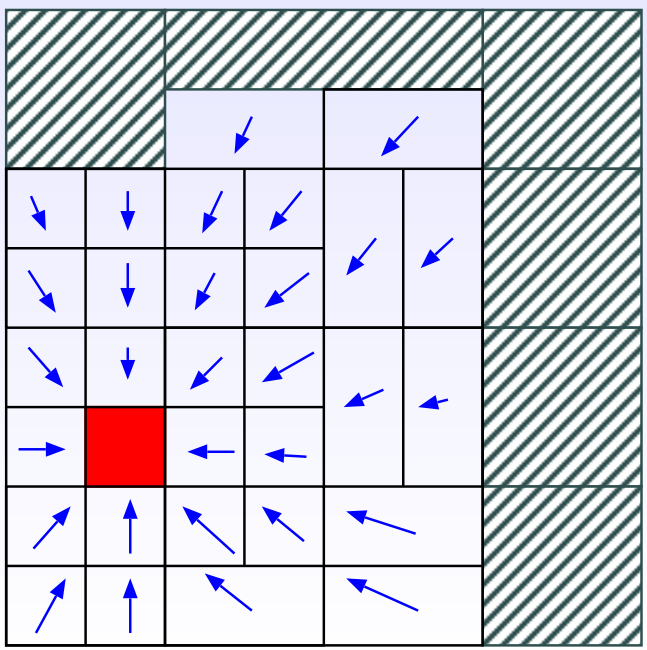
- Disponer de cajas con localidad parecida incrementa la información acerca de donde se encuentra el mínimo, ya que podríamos analizar la dirección que seguimos en el proceso de evaluación de las cajas.
- Presentaremos el algoritmo de balanceo MPR: *Minimus Prediction by Regions* (Predicción de Mínimos mediante Regiones).



Descripción de la predicción de mínimos

Se pretende generar regiones de cajas próximas entre sí, cuyos centros de atracción tratarán de estar localizados sobre los mínimos de la función objetivo.

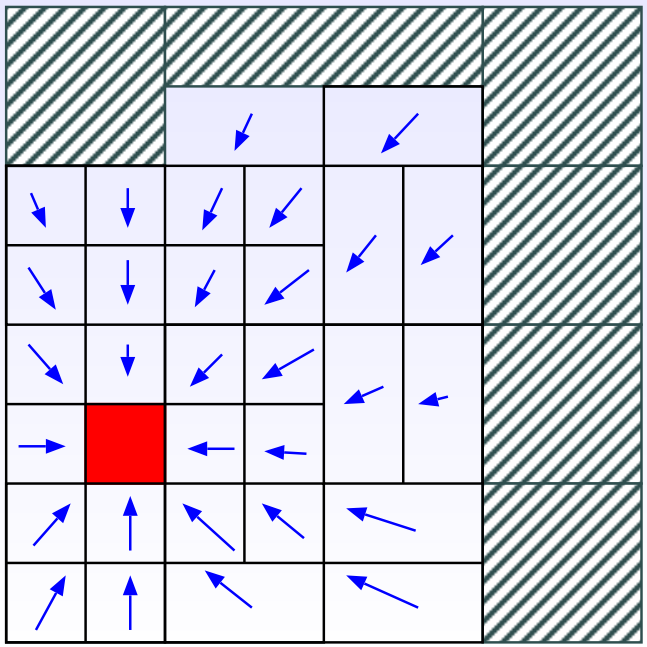
- Disponer de cajas con localidad parecida incrementa la información acerca de donde se encuentra el mínimo, ya que podríamos analizar la dirección que seguimos en el proceso de evaluación de las cajas.
- Presentaremos el algoritmo de balanceo MPR: *Minimus Prediction by Regions* (Predicción de Mínimos mediante Regiones).



Descripción de la predicción de mínimos

Se pretende generar regiones de cajas próximas entre sí, cuyos centros de atracción tratarán de estar localizados sobre los mínimos de la función objetivo.

- Disponer de cajas con localidad parecida incrementa la información acerca de donde se encuentra el mínimo, ya que podríamos analizar la dirección que seguimos en el proceso de evaluación de las cajas.
- Presentaremos el algoritmo de balanceo MPR: *Minimus Prediction by Regions* (Predicción de Mínimos mediante Regiones).



Estructura de las regiones

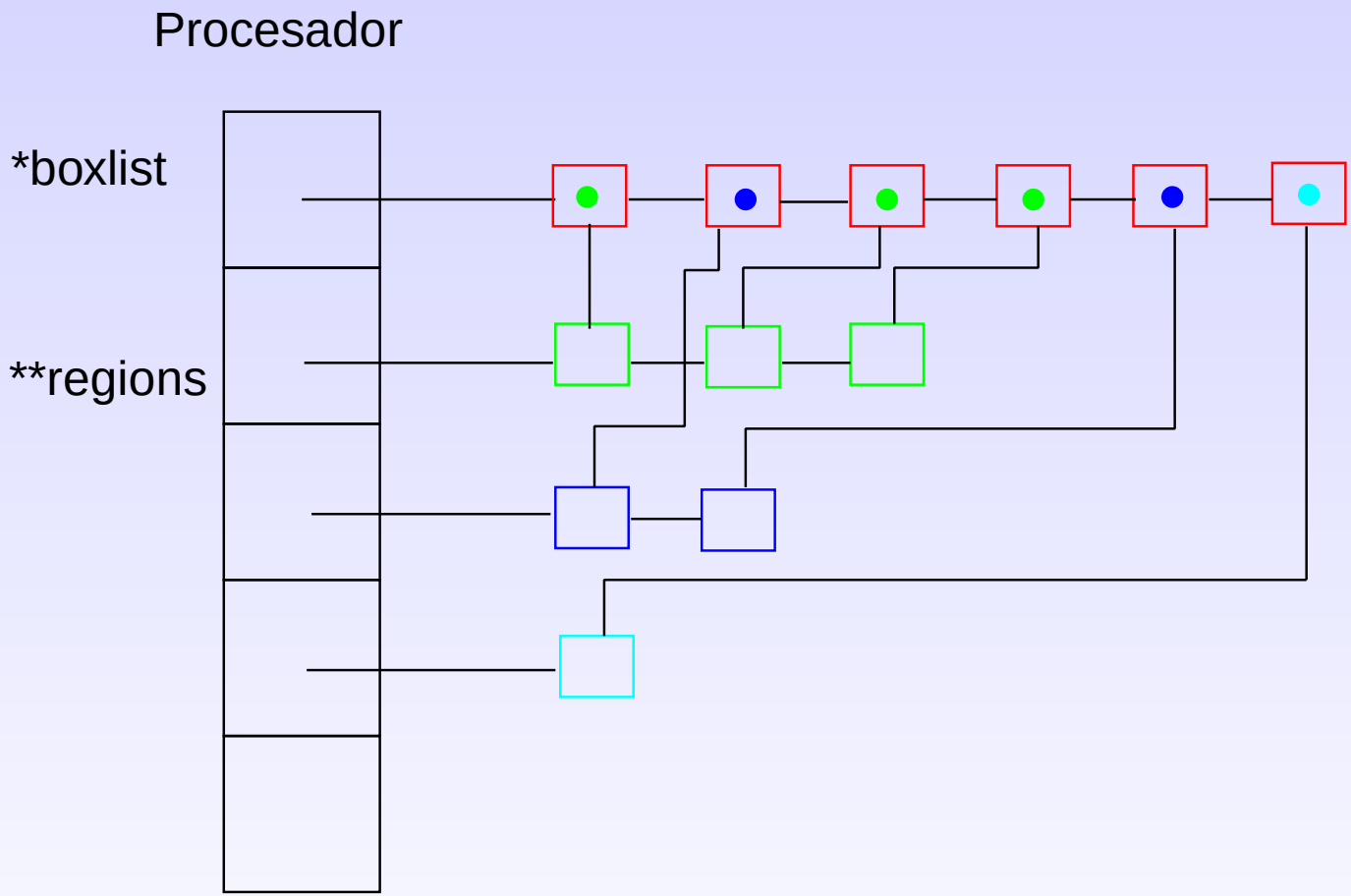


Figura: Estructura de las regiones y lista de cajas

Creación de regiones

La primera caja líder será la primera de la lista.
Las cajas candidatas a líderes serán aquellas cuyo $\underline{F}(X)$ esté por debajo del umbral.

Umbral de candidatos a líderes de una región

$$umbral = Lmin + \frac{Lmax - Lmin}{max\_proc}$$

Cercanía de cajas y elección de líderes

$$f_c = \frac{w(S')}{w(X_L)}$$

$$d_{min} = \frac{f_c}{2} \cdot w(X_C)$$

Sean dos cajas X_1 y X_2, y d_e la distancia euclídea entre los centros de ambas cajas. La caja X_1 es cercana a X_2 cuando se cumple que: $d_e < d_{min}$

Creación de regiones

La primera caja líder será la primera de la lista.
Las cajas candidatas a líderes serán aquellas cuyo $\underline{F}(X)$ esté por debajo del umbral.

Umbral de candidatos a líderes de una región

$$umbral = Lmin + \frac{Lmax - Lmin}{max\_proc}$$

Cercanía de cajas y elección de líderes

$$f_c = \frac{w(S')}{w(X_L)}$$

$$d_{min} = \frac{f_c}{2} \cdot w(X_C)$$

Sean dos cajas X_1 y X_2, y d_e la distancia euclídea entre los centros de ambas cajas. La caja X_1 es cercana a X_2 cuando se cumple que: $d_e < d_{min}$

Creación de regiones

La primera caja líder será la primera de la lista.
Las cajas candidatas a líderes serán aquellas cuyo $\underline{F}(X)$ esté por debajo del umbral.

Umbral de candidatos a líderes de una región

$$umbral = Lmin + \frac{Lmax - Lmin}{max\_proc}$$

Cercanía de cajas y elección de líderes

$$f_c = \frac{w(S')}{w(X_L)}$$

$$d_{min} = \frac{f_c}{2} \cdot w(X_C)$$

Sean dos cajas X_1 y X_2, y d_e la distancia euclídea entre los centros de ambas cajas. La caja X_1 es cercana a X_2 cuando se cumple que: $d_e < d_{min}$

Creación de regiones

La primera caja líder será la primera de la lista.
Las cajas candidatas a líderes serán aquellas cuyo $\underline{F}(X)$ esté por debajo del umbral.

Umbral de candidatos a líderes de una región

$$umbral = Lmin + \frac{Lmax - Lmin}{max\_proc}$$

Cercanía de cajas y elección de líderes

$$f_c = \frac{w(S')}{w(X_L)}$$

$$d_{min} = \frac{f_c}{2} \cdot w(X_C)$$

Sean dos cajas X_1 y X_2, y d_e la distancia euclídea entre los centros de ambas cajas. La caja X_1 es cercana a X_2 cuando se cumple que: $d_e < d_{min}$

Creación de regiones

La primera caja líder será la primera de la lista.
Las cajas candidatas a líderes serán aquellas cuyo $\underline{F}(X)$ esté por debajo del umbral.

Umbral de candidatos a líderes de una región

$$umbral = Lmin + \frac{Lmax - Lmin}{max\_proc}$$

Cercanía de cajas y elección de líderes

$$f_c = \frac{w(S')}{w(X_L)}$$

$$d_{min} = \frac{f_c}{2} \cdot w(X_C)$$

Sean dos cajas X_1 y X_2, y d_e la distancia euclídea entre los centros de ambas cajas. La caja X_1 es cercana a X_2 cuando se cumple que: $d_e < d_{min}$

Creación de regiones

La primera caja líder será la primera de la lista.
Las cajas candidatas a líderes serán aquellas cuyo $\underline{F}(X)$ esté por debajo del umbral.

Umbral de candidatos a líderes de una región

$$umbral = Lmin + \frac{Lmax - Lmin}{max\_proc}$$

Cercanía de cajas y elección de líderes

$$f_c = \frac{w(S')}{w(X_L)}$$

$$d_{min} = \frac{f_c}{2} \cdot w(X_C)$$

Sean dos cajas X_1 y X_2, y d_e la distancia euclídea entre los centros de ambas cajas. La caja X_1 es cercana a X_2 cuando se cumple que: $d_e < d_{min}$

Creación de regiones

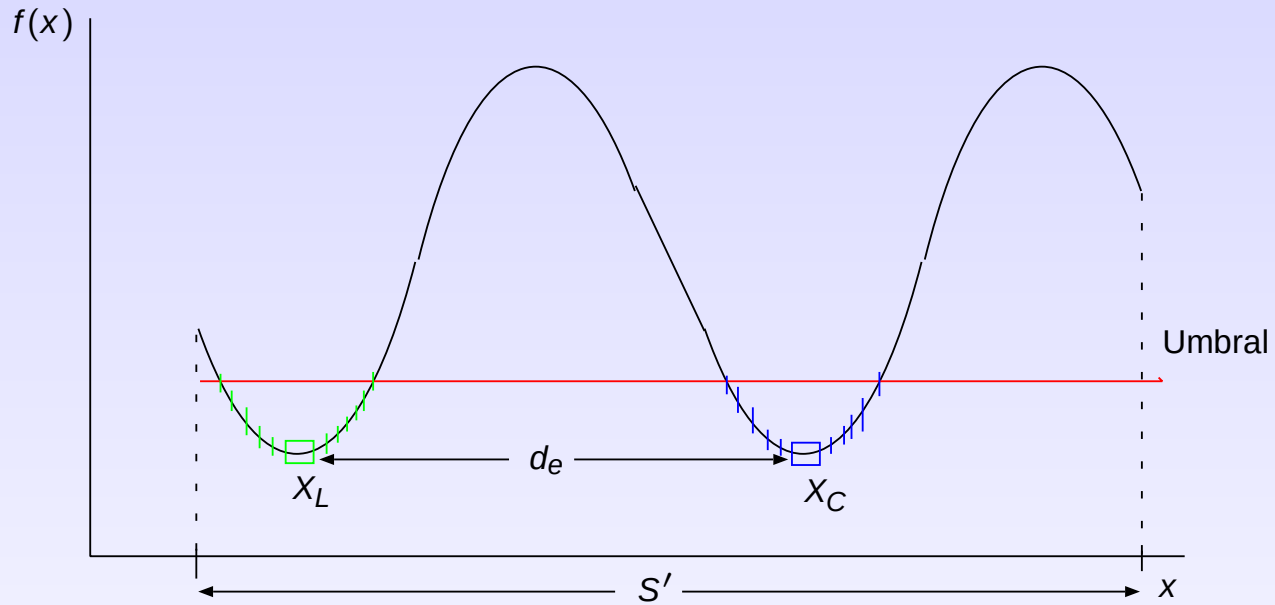


Figura: Determinación de líderes de una región y cercanía entre cajas

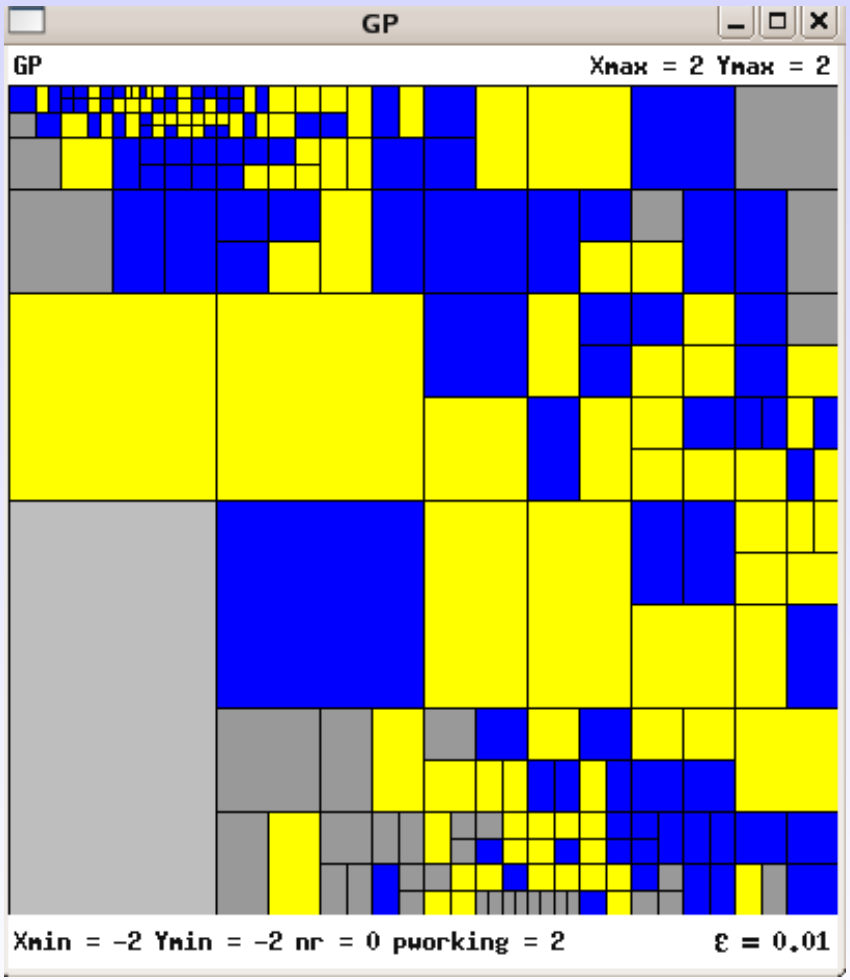
Algoritmo de balanceo MPR

Funct MPR($L, Q, Z, R, lideres, P$)

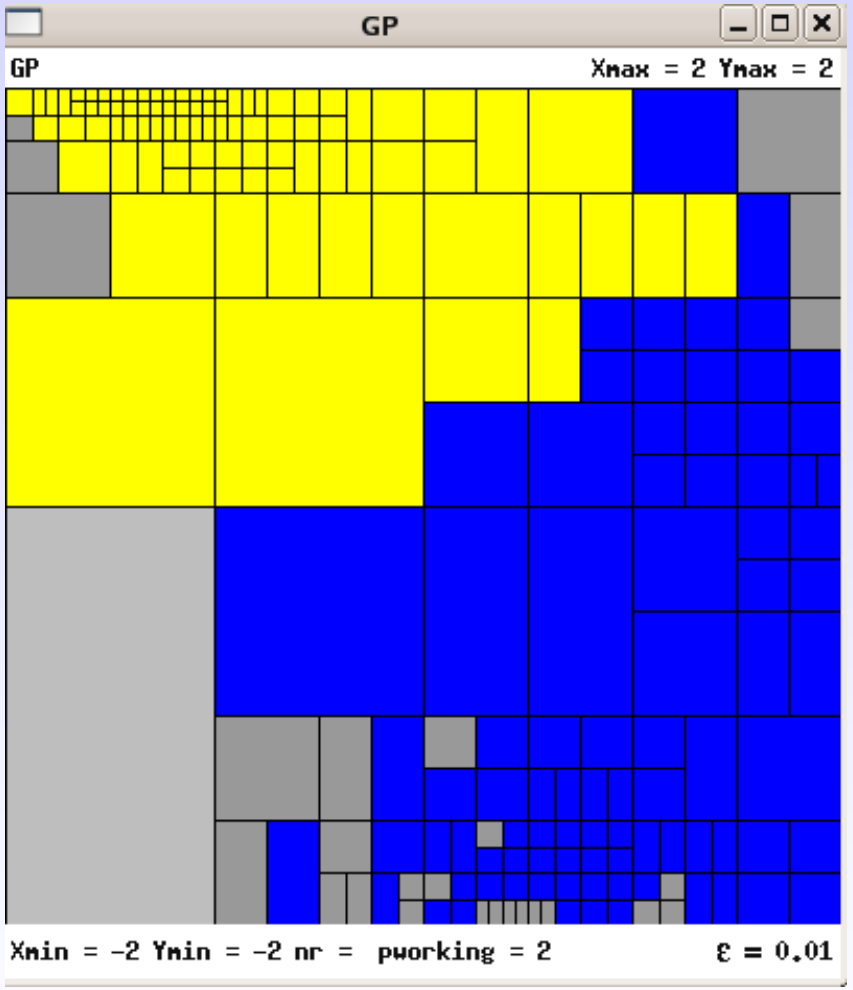
1. $umbral := calcularUmbral(L)$
2. $Z := ElegirLideres(L, Z, umbral, lideres)$
3. **if** $(numeroElementos(Z) < 2)$
4. **return** *Si no hay al menos dos regiones*
5. $R := AsignarCajasRegiones(L, Z, R)$
6. $L := R_{Z^1}$ *La primera región se queda en el procesador*
7. **for** $i := 2$ **to** $lideres$
8. $MoverCajas(P^{i-1}, R_{Z^i})$
9. **return** L

Coloreado de las regiones

GP con balanceo BAR



GP con balanceo MPR



Decisión de balanceo

Utilizado en las versiones BARD y MPRD

Tiempo de trabajo en potencia de un procesador

Se define el tiempo de trabajo en potencia de un procesador, y se notará por t_p, al tiempo de evaluación pendiente en el procesador.

$$t_p = t_{pc} \cdot wp(L)$$

Tiempo de envío a otro procesador

Se define el tiempo de envío, y se notará como t_{pe}, al tiempo que tardaría en enviarse la mitad de las cajas a otro procesador.

$$t_{pe} = t_{cp} + l + \frac{\frac{n}{2} \cdot size(X)}{ttr}$$

El trabajo de una caja

Definición de trabajo de una caja

Se define el trabajo de una caja X, y se notará por $wk(X)$, como el número de cajas estimado que se evaluarán por el algoritmo de OG siendo el número de descendientes de X, $desc(X)$, una cota superior de este trabajo.

Cálculo del trabajo de una caja

$$desc(X) = 2^{i+1} - 1$$

$$i = \frac{2 \cdot \log \frac{w(X)}{\epsilon}}{\log 2}$$

$$wk(X) = \left(2^{\frac{2 \cdot \log \frac{w(X)}{\epsilon}}{\log 2} + 1} - 1 \right) \cdot \overline{pf^*}$$

El trabajo de una caja

Definición de trabajo de una caja

Se define el trabajo de una caja X, y se notará por $wk(X)$, como el número de cajas estimado que se evaluarán por el algoritmo de OG siendo el número de descendientes de X, $desc(X)$, una cota superior de este trabajo.

Cálculo del trabajo de una caja

$$desc(X) = 2^{i+1} - 1$$

$$i = \frac{2 \cdot \log \frac{w(X)}{\epsilon}}{\log 2}$$

$$wk(X) = \left(2^{\frac{2 \cdot \log \frac{w(X)}{\epsilon}}{\log 2} + 1} - 1 \right) \cdot \overline{pf^*}$$

El trabajo de una caja

Definición de trabajo de una caja

Se define el trabajo de una caja X, y se notará por $wk(X)$, como el número de cajas estimado que se evaluarán por el algoritmo de OG siendo el número de descendientes de X, $desc(X)$, una cota superior de este trabajo.

Cálculo del trabajo de una caja

$$desc(X) = 2^{i+1} - 1$$

$$i = \frac{2 \cdot \log \frac{w(X)}{\epsilon}}{\log 2}$$

$$wk(X) = \left(2^{\frac{2 \cdot \log \frac{w(X)}{\epsilon}}{\log 2} + 1} - 1 \right) \cdot \overline{pf^*}$$

El trabajo de una caja

Definición de trabajo de una caja

Se define el trabajo de una caja X, y se notará por $wk(X)$, como el número de cajas estimado que se evaluarán por el algoritmo de OG siendo el número de descendientes de X, $desc(X)$, una cota superior de este trabajo.

Cálculo del trabajo de una caja

$$desc(X) = 2^{i+1} - 1$$

$$i = \frac{2 \cdot \log \frac{w(X)}{\epsilon}}{\log 2}$$

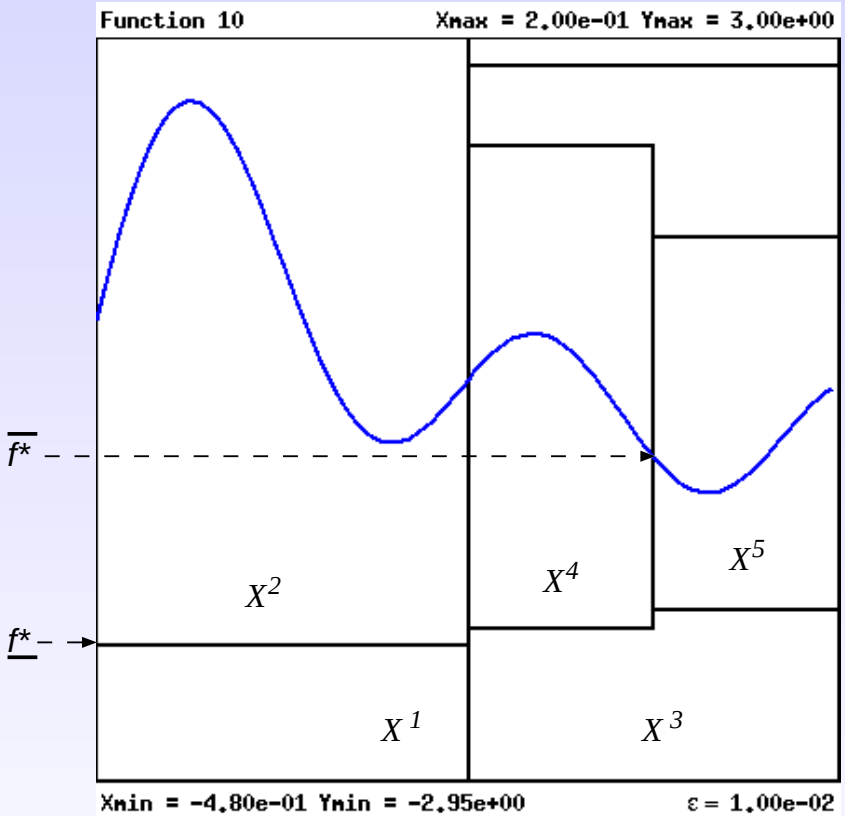
$$wk(X) = \left(2^{\frac{2 \cdot \log \frac{w(X)}{\epsilon}}{\log 2} + 1} - 1 \right) \cdot \overline{pf^*}$$

El parámetro $p\overline{f^*}(X)$

- Estimador de la cercanía a un mínimo.

$$p\overline{f^*}(X) = \frac{\overline{f^*} - \underline{F}(X)}{w(F(X))} \in [0,1]$$

- Autores lo han utilizado en la regla de selección.
- También como estimador de la carga de trabajo.

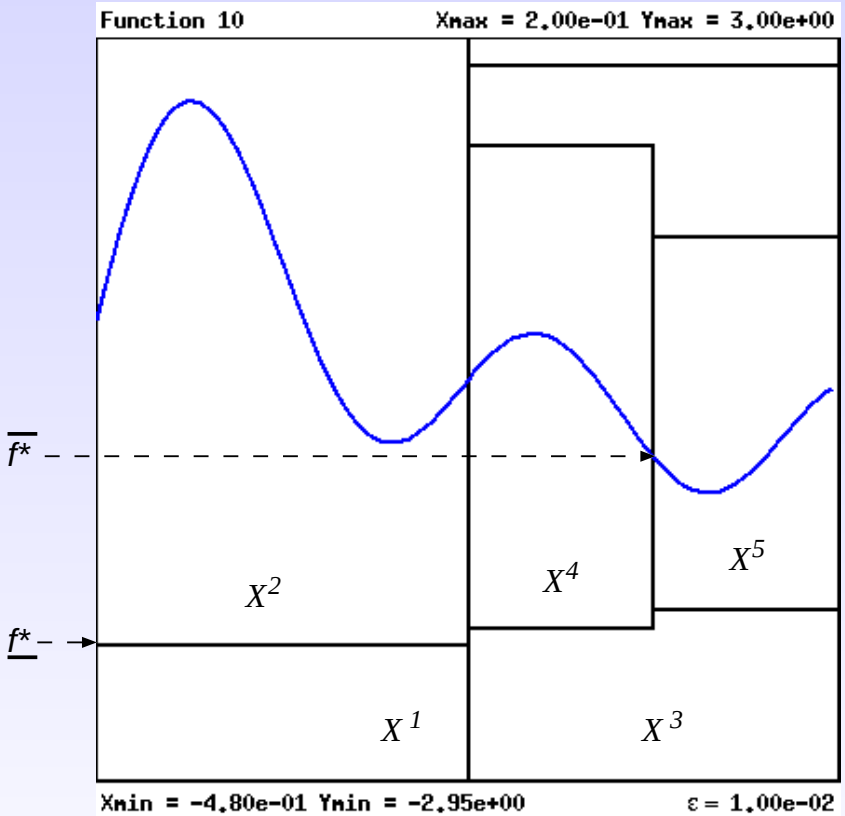


El parámetro $p\overline{f^*}(X)$

- Estimador de la cercanía a un mínimo.

$$p\overline{f^*}(X) = \frac{\overline{f^*} - \underline{F}(X)}{w(F(X))} \in [0,1]$$

- Autores lo han utilizado en la regla de selección.
- También como estimador de la carga de trabajo.

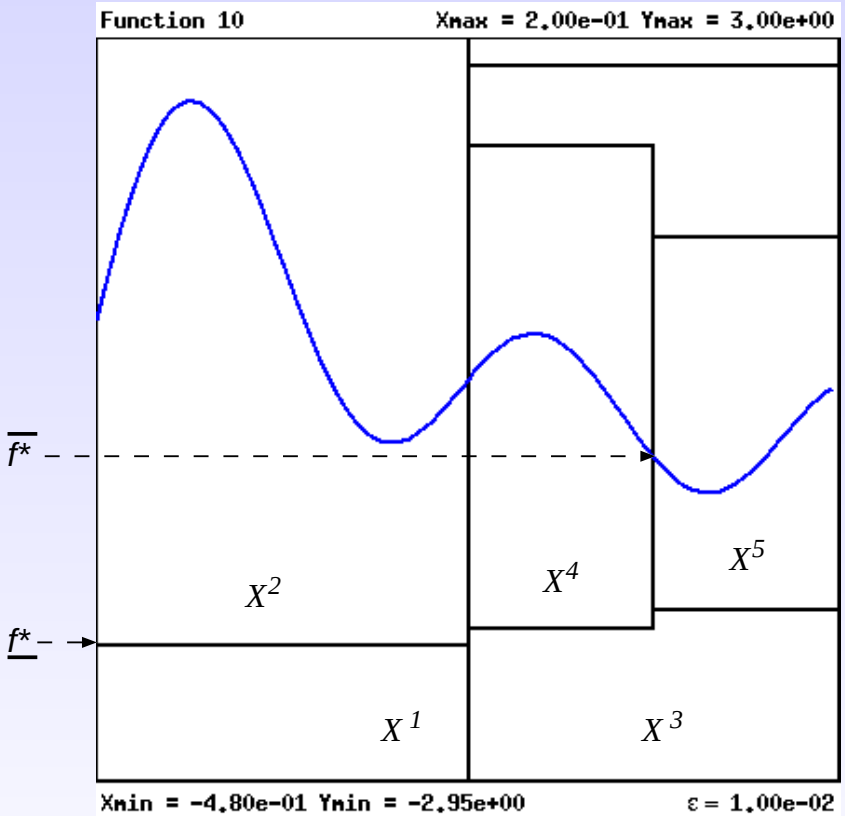


El parámetro $p\overline{f^*}(X)$

- Estimador de la cercanía a un mínimo.

$$p\overline{f^*}(X) = \frac{\overline{f^*} - \underline{F}(X)}{w(F(X))} \in [0,1]$$

- Autores lo han utilizado en la regla de selección.
- También como estimador de la carga de trabajo.

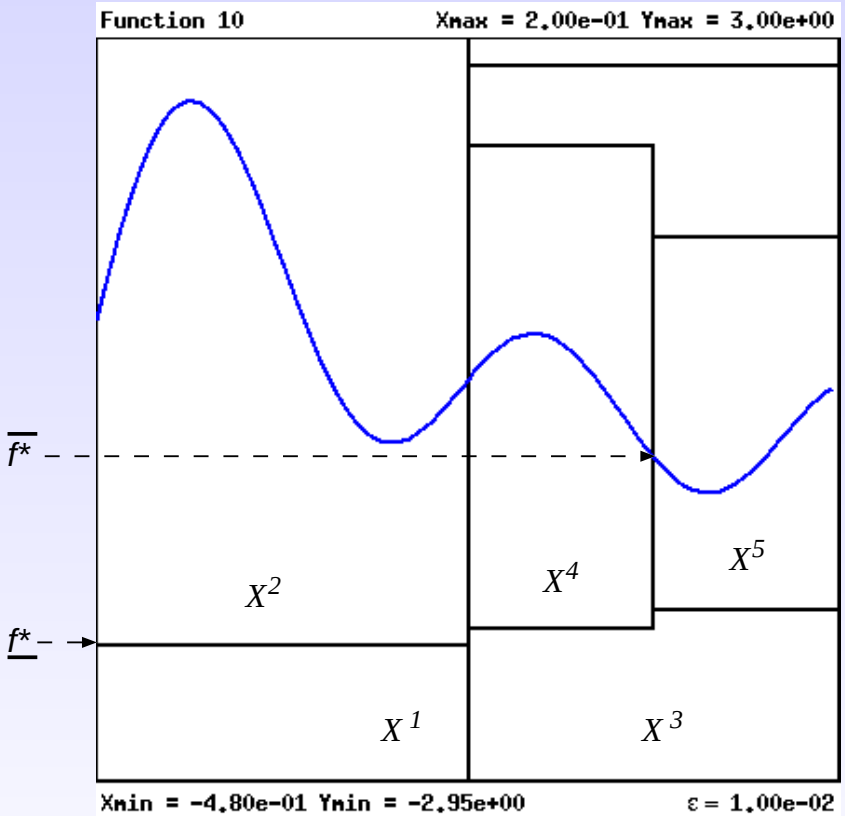


El parámetro $p\overline{f^*}(X)$

- Estimador de la cercanía a un mínimo.

$$p\overline{f^*}(X) = \frac{\overline{f^*} - \underline{F}(X)}{w(F(X))} \in [0,1]$$

- Autores lo han utilizado en la regla de selección.
- También como estimador de la carga de trabajo.



Algoritmo de decisión de balanceo

Funct DecidePredecir($L, t_{pc}, t_{cp}, t_p, t_{pe}, n, t_{ec}$)

1. $n := numeroElementos(L)$
2. $t_p := calcularTiempoTrabajo(t_{pc}, L)$
3. $t_{pe} := calcularTEnvio(t_{cp}, t_p, n, t_{ec})$
4. **if** $(\frac{t_p}{2} > t_{pe})$
5. **return** PREDECIR
6. **else**
7. **return** NO_PREDECIR

Características de las pruebas

Los resultados medios se han obtenido haciendo cinco ejecuciones de las pruebas y rechazando los de mayor y menor valor.

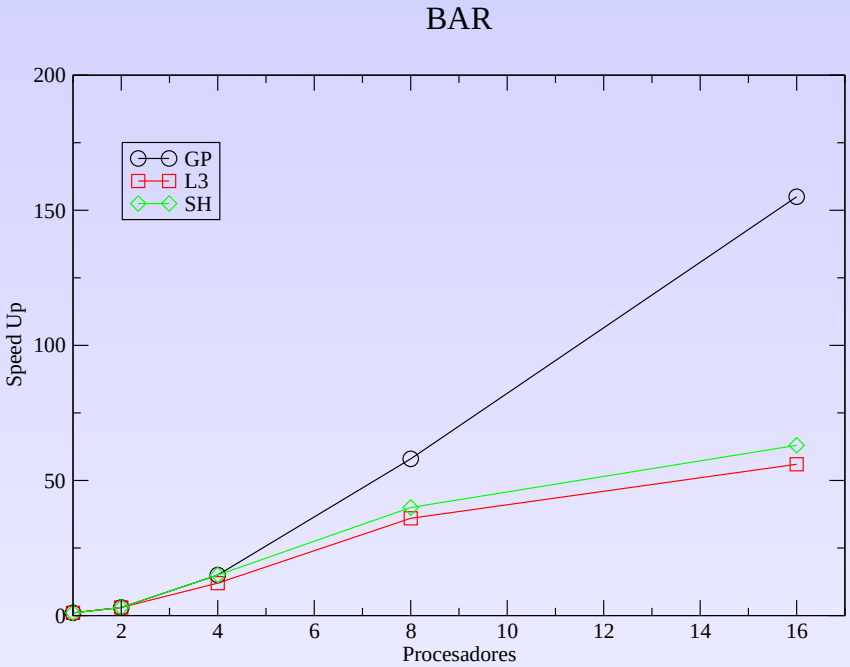
Funciones test utilizadas

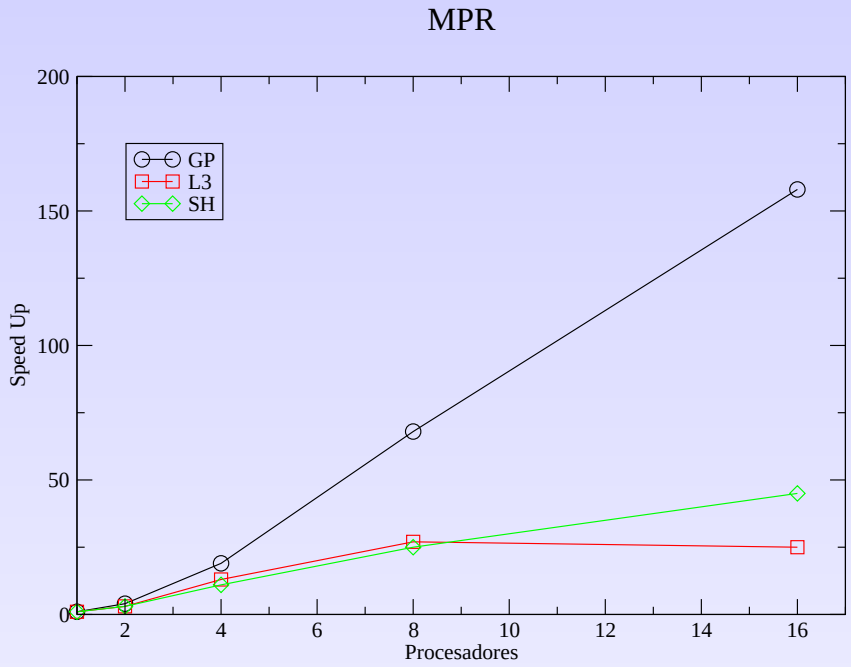
No.	Función	Dimensiones	Precisión (ϵ)
0	Goldstein-Price	2	10^{-4}
1	Levy 3	2	10^{-4}
2	Six Hump Camel Back	2	10^{-4}

Haremos las siguientes comparaciones

- BAR – MPR
- BARD – MPRD

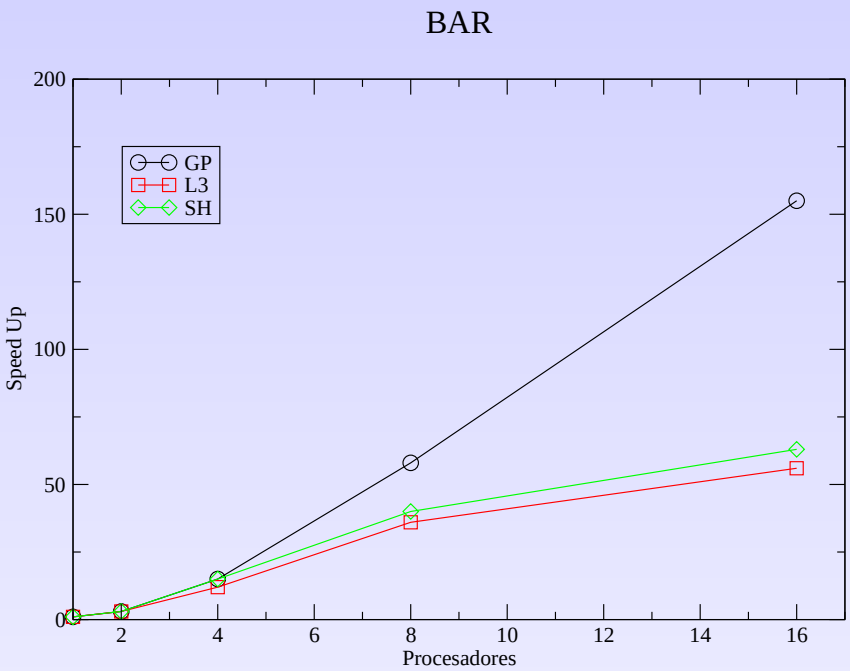
SpeedUp BAR – MPR

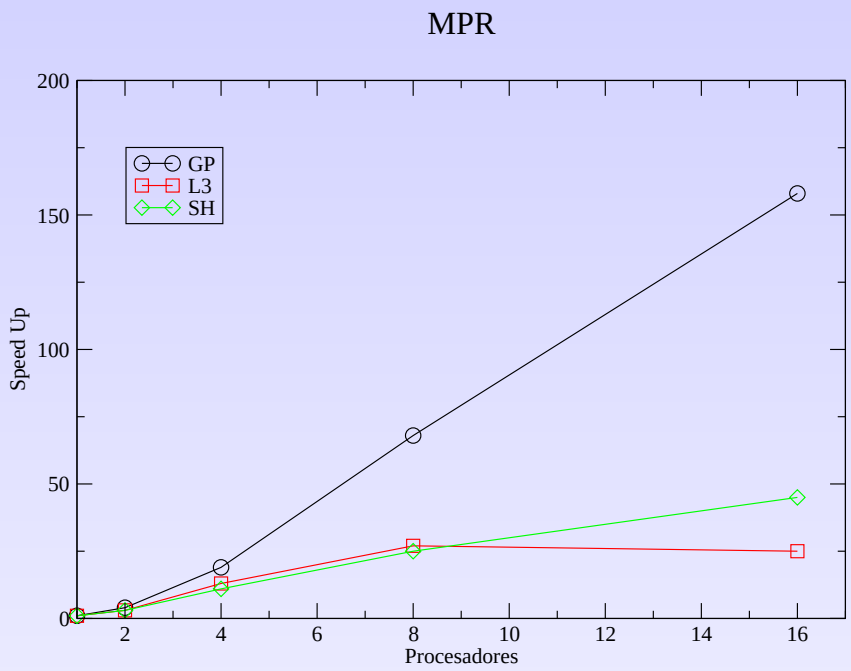




Modelo	BAR			MPR		
2 procesadores	GP	L3	SH	GP	L3	SH
Speed-up	3,93	3,72	3,68	4,26	3,84	3,65
T. Proceso	1068,97	181,86	250,05	983,67	175,43	250,21
4 procesador	GP	L3	SH	GP	L3	SH
Speed-up	15,75	12,87	15,51	19,31	13,36	11,89
T. Proceso	526,05	99,31	112,37	420,37	94,4	137
8 procesadores	GP	L3	SH	GP	L3	SH
Speed-up	58,57	36,40	40,76	68,87	27,77	25,19
T. Proceso	261,66	60,4	70,99	205,42	71,76	104,73
16 procesadores	GP	L3	SH	GP	L3	SH
Speed-up	155,67	56,31	63,06	158,48	25,59	45,14
T. Proceso	164,68	60,64	69,32	146,04	92,28	93,55

SpeedUp BAR – MPR





Modelo	BAR			MPR		
2 procesadores	GP	L3	SH	GP	L3	SH
Speed-up	3,93	3,72	3,68	4,26	3,84	3,65
T. Proceso	1068,97	181,86	250,05	983,67	175,43	250,21
4 procesador	GP	L3	SH	GP	L3	SH
Speed-up	15,75	12,87	15,51	19,31	13,36	11,89
T. Proceso	526,05	99,31	112,37	420,37	94,4	137
8 procesadores	GP	L3	SH	GP	L3	SH
Speed-up	58,57	36,40	40,76	68,87	27,77	25,19
T. Proceso	261,66	60,4	70,99	205,42	71,76	104,73
16 procesadores	GP	L3	SH	GP	L3	SH
Speed-up	155,67	56,31	63,06	158,48	25,59	45,14
T. Proceso	164,68	60,64	69,32	146,04	92,28	93,55

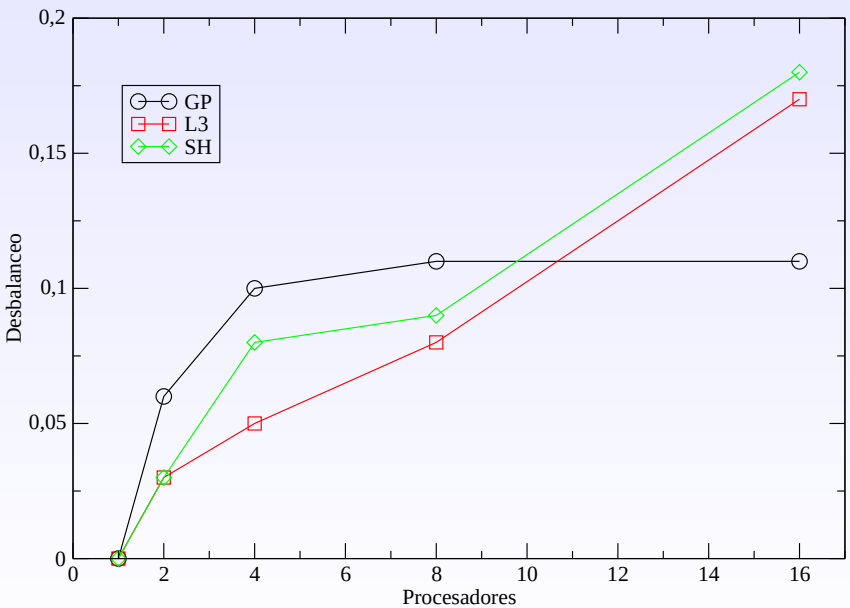
Desbalanceo BAR – MPR

Cálculo del desbalanceo

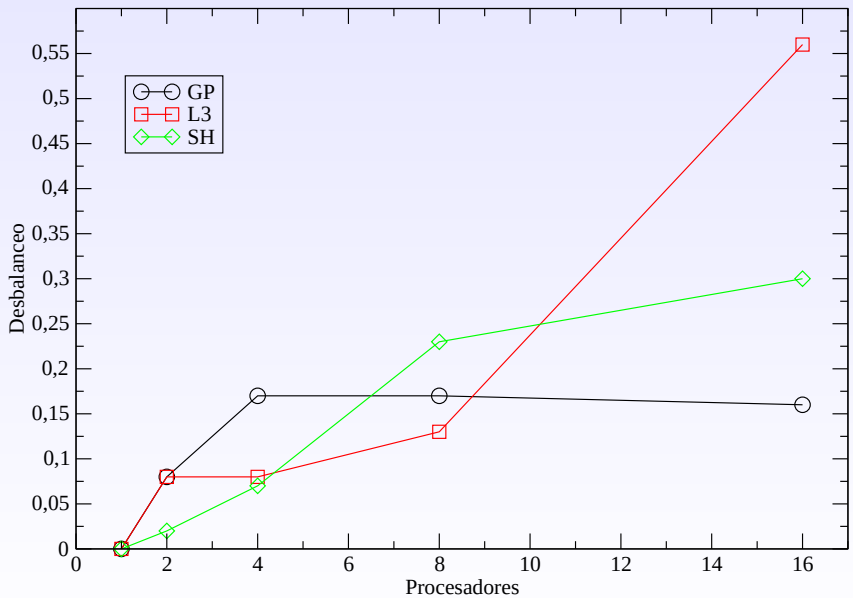
$$Desbalanceo = \frac{\sigma(Esf)}{\overline{Esf}}$$

$$\sigma(Esf) = \sqrt{\frac{1}{p}\sum_{i=1}^{p}(Esf(p_i) - \overline{Esf})^2}$$

BAR



MPR



Nº de balanceos BAR – MPR

Modelo	BAR			MPR		
2 procesadores	GP	L3	SH	GP	L3	SH
Repartos	7,33	7	4	15	10,67	8,33
Predicciones				1742	46,33	342,67
4 procesador	GP	L3	SH	GP	L3	SH
Repartos	35	26,67	25	64,67	28,33	80
Predicciones				3233,67	805,67	367,67
8 procesadores	GP	L3	SH	GP	L3	SH
Repartos	104,33	49	76	141	103	174,33
Predicciones				5617,67	1185	768,67
16 procesadores	GP	L3	SH	GP	L3	SH
Repartos	193,67	119	174,67	262,67	204,33	270
Predicciones				4372,33	7757,33	1074,33

Nº de balanceos BAR – MPR

Modelo	BAR			MPR		
2 procesadores	GP	L3	SH	GP	L3	SH
Repartos	7,33	7	4	15	10,67	8,33
Predicciones				1742	46,33	342,67
4 procesador	GP	L3	SH	GP	L3	SH
Repartos	35	26,67	25	64,67	28,33	80
Predicciones				3233,67	805,67	367,67
8 procesadores	GP	L3	SH	GP	L3	SH
Repartos	104,33	49	76	141	103	174,33
Predicciones				5617,67	1185	768,67
16 procesadores	GP	L3	SH	GP	L3	SH
Repartos	193,67	119	174,67	262,67	204,33	270
Predicciones				4372,33	7757,33	1074,33

Evolución de cajas en los procesadores – BAR

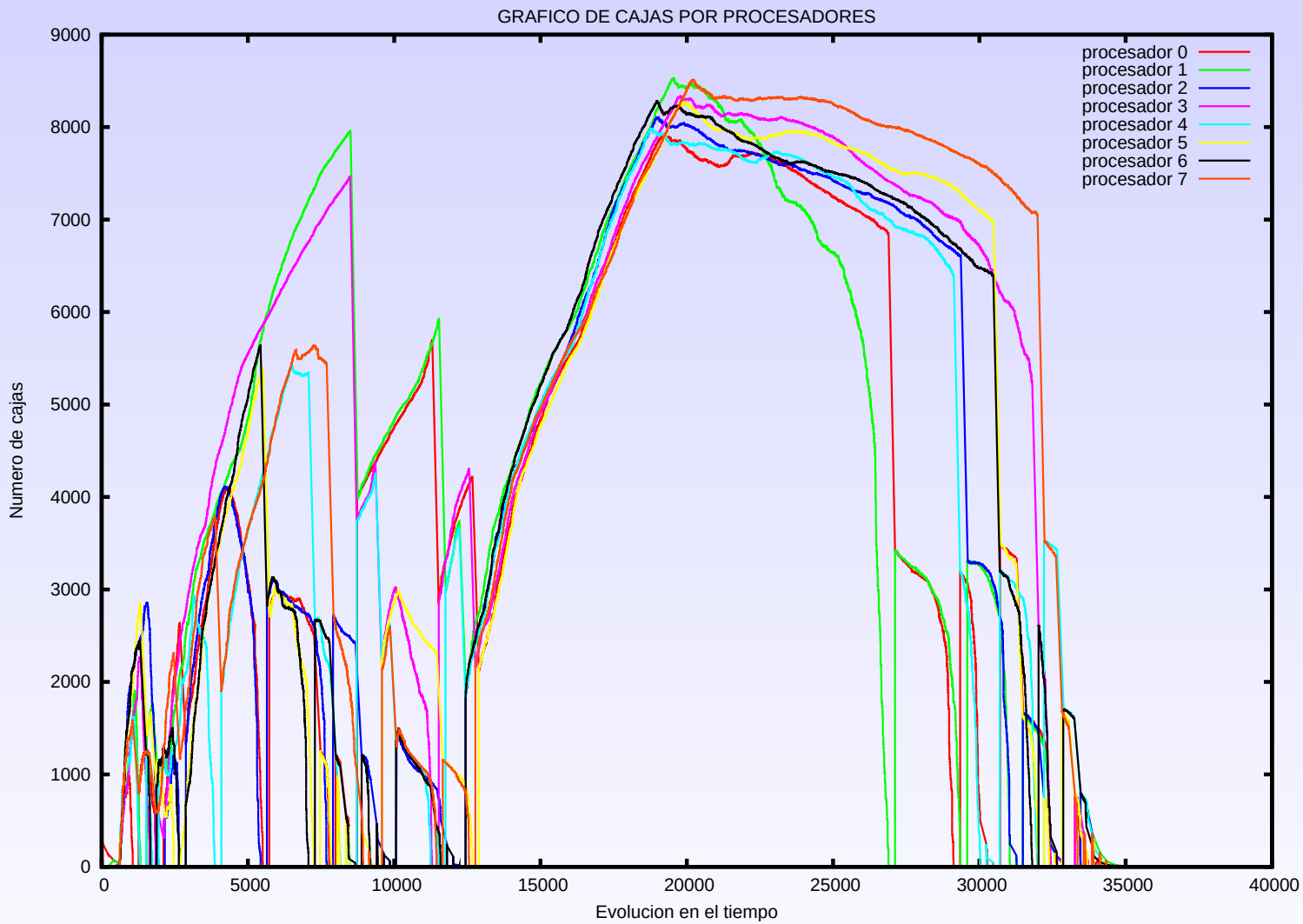


Figura: Evolución de cajas en los procesadores para el problema GP con 8 procesadores, balanceador BAR y $\epsilon = 10^{-4}$.

Evolución de cajas en los procesadores – MPR

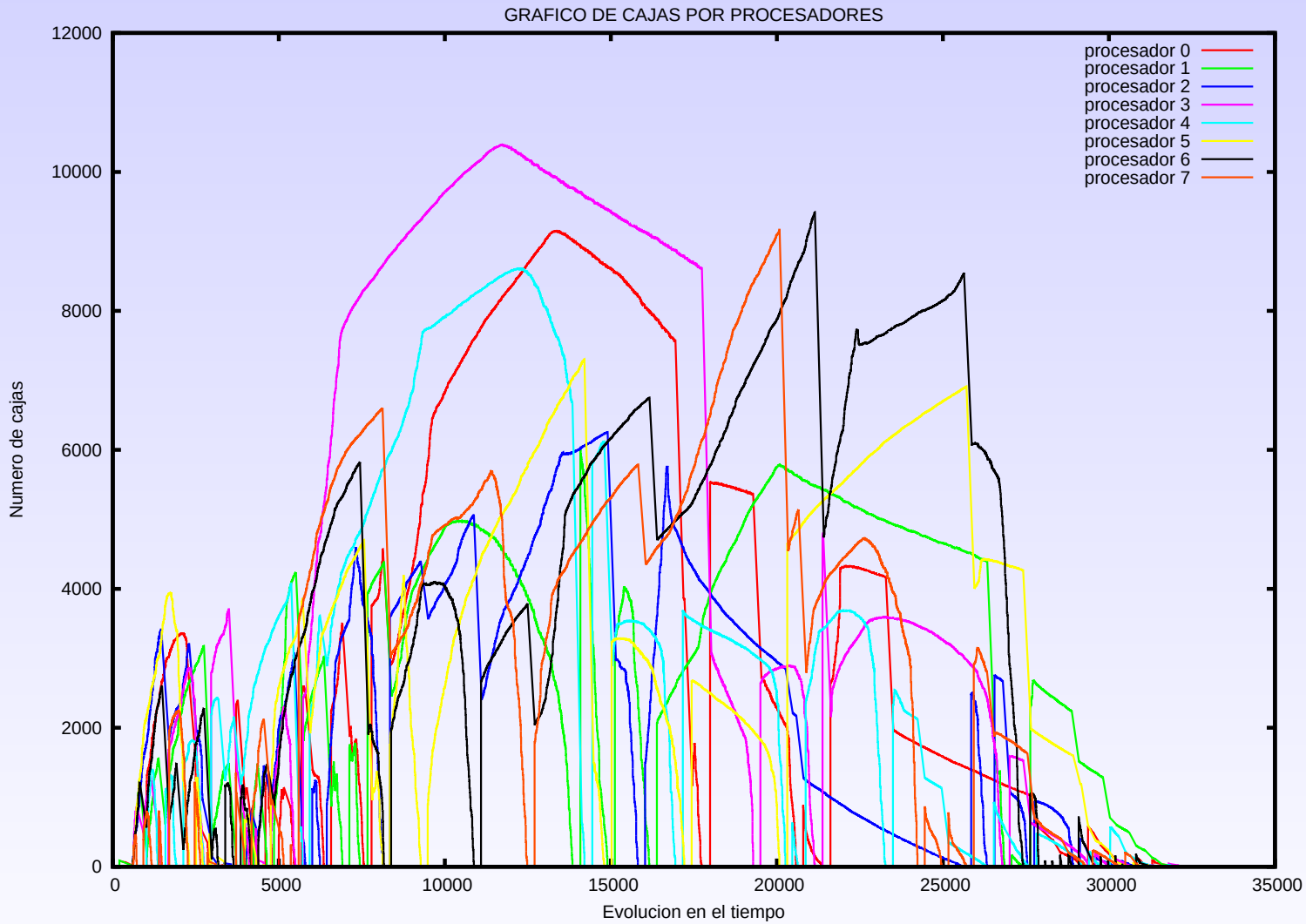
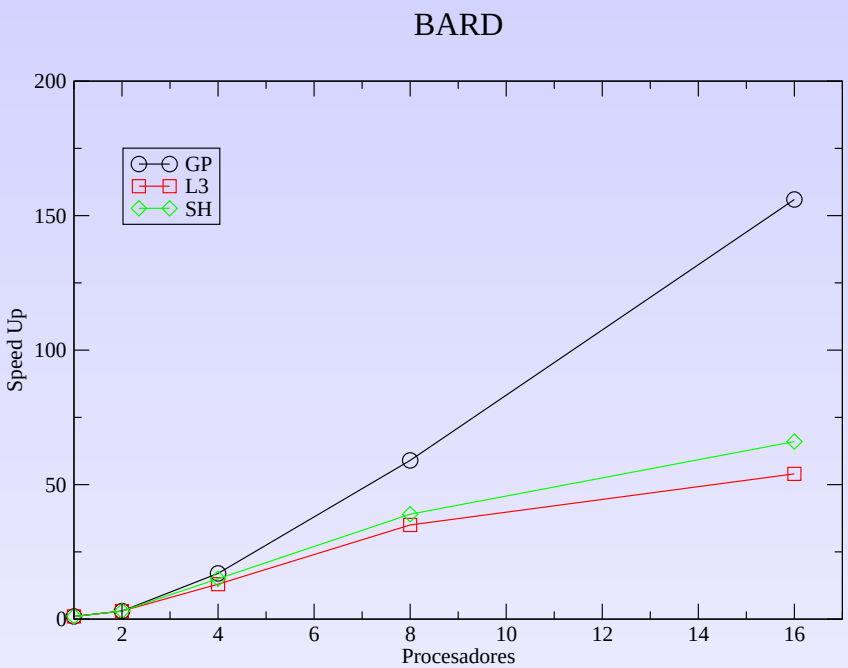
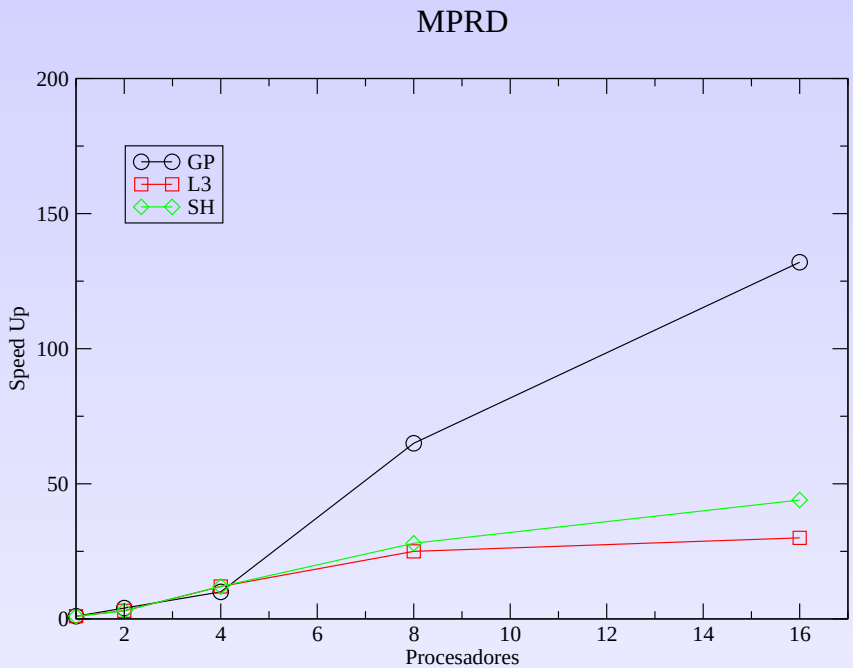


Figura: Evolución de cajas en los procesadores para el problema GP con 8 procesadores, balanceador MPR y $\epsilon = 10^{-4}$.

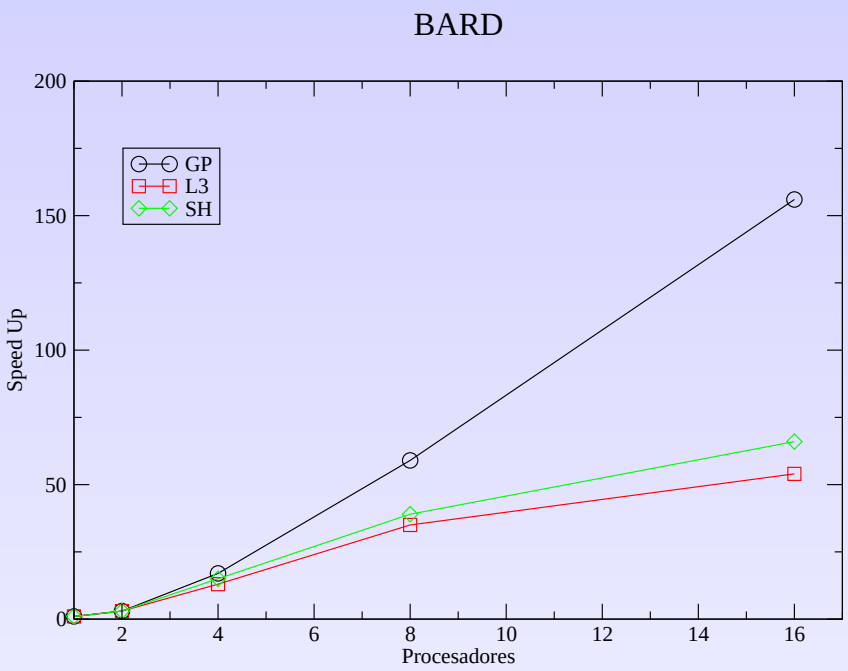
SpeedUp BARD – MPRD

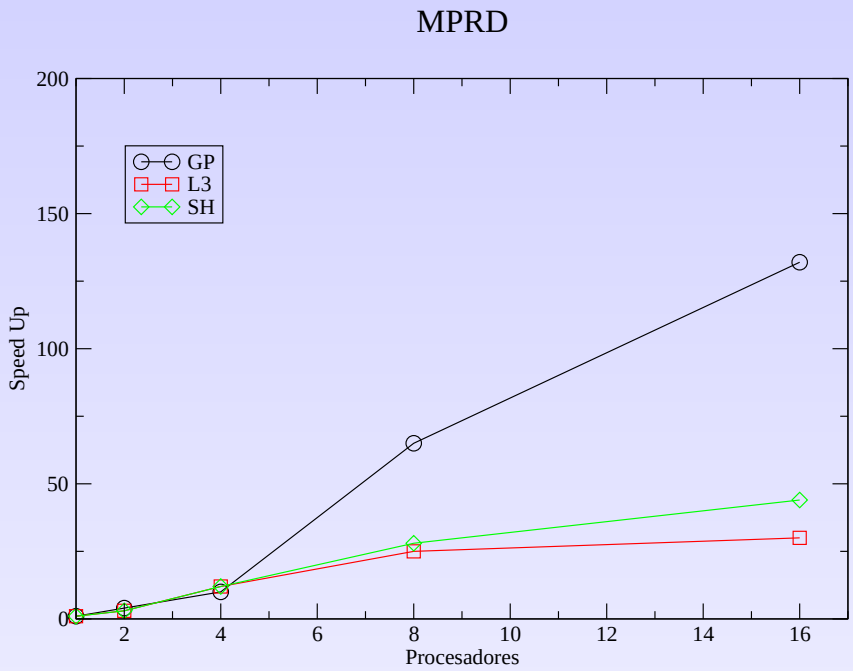




Modelo	BARD			MPRD		
2 procesador	GP	L3	SH	GP	L3	SH
Speed-up	3,99	3,81	3,72	4,22	3,80	3,67
T. Proceso	1052,09	178,33	248,31	992,94	177,63	249,59
4 procesador	GP	L3	SH	GP	L3	SH
Speed-up	17,39	13,59	15,19	10,75	12,71	12,07
T. Proceso	476,46	93,52	116,27	527,14	98,29	133,49
8 procesador	GP	L3	SH	GP	L3	SH
Speed-up	59,70	35,91	39,95	65,51	25,08	28,51
T. Proceso	257,95	60,79	72,91	218,48	72,57	90,8
16 procesador	GP	L3	SH	GP	L3	SH
Speed-up	156,71	54,09	66,31	132,63	30,49	44,74
T. Proceso	157,59	62,27	67,31	168,62	91,67	94,3

SpeedUp BARD – MPRD

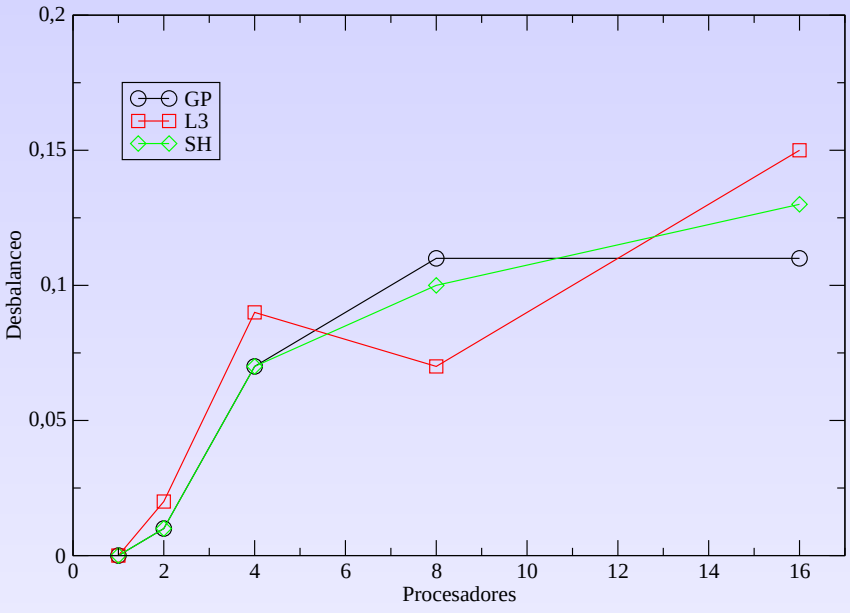




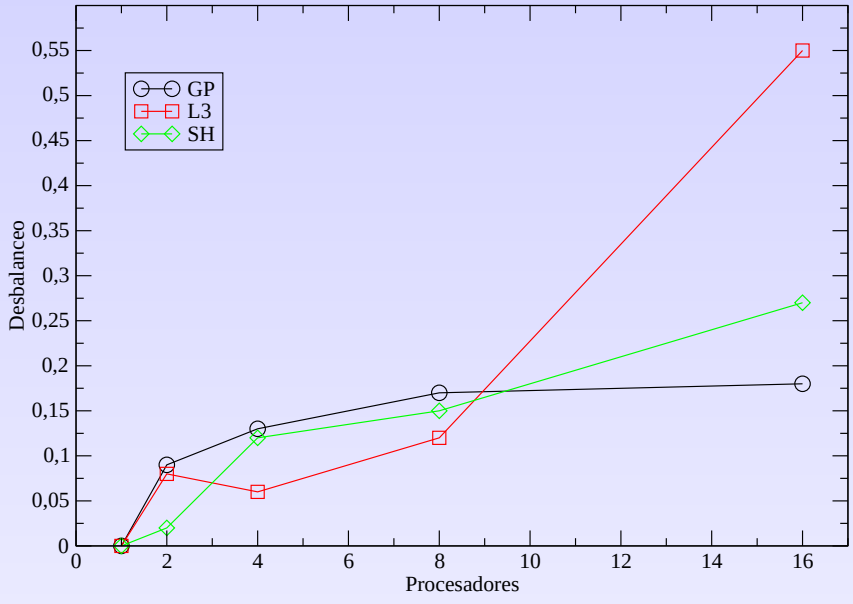
Modelo	BARD			MPRD		
2 procesador	GP	L3	SH	GP	L3	SH
Speed-up	3,99	3,81	3,72	4,22	3,80	3,67
T. Proceso	1052,09	178,33	248,31	992,94	177,63	249,59
4 procesador	GP	L3	SH	GP	L3	SH
Speed-up	17,39	13,59	15,19	10,75	12,71	12,07
T. Proceso	476,46	93,52	116,27	527,14	98,29	133,49
8 procesador	GP	L3	SH	GP	L3	SH
Speed-up	59,70	35,91	39,95	65,51	25,08	28,51
T. Proceso	257,95	60,79	72,91	218,48	72,57	90,8
16 procesador	GP	L3	SH	GP	L3	SH
Speed-up	156,71	54,09	66,31	132,63	30,49	44,74
T. Proceso	157,59	62,27	67,31	168,62	91,67	94,3

Desbalanceo BARD – MPRD

BARD



MPRD



Modelo	BARD			MPRD		
2 procesadores	GP	L3	SH	GP	L3	SH
Repartos	7,33	5,67	2	16	11,33	4,33
Predicciones				489	31,33	397,33
4 procesador	GP	L3	SH	GP	L3	SH
Repartos	32,33	27,33	21	60	38,33	76,67
Predicciones				18977,67	204,67	414,67
8 procesadores	GP	L3	SH	GP	L3	SH
Repartos	93,67	53,33	77,67	167,67	95,67	158,33
Predicciones				1742	2318	668,33
16 procesadores	GP	L3	SH	GP	L3	SH
Repartos	224,67	128,67	162,67	326,67	199,33	277
Predicciones				4992	8634	833,67

Un nuevo algoritmo de balanceo basado en regiones

- Nos interesa comparar entre sí dos algoritmos de balanceo basados en regiones para estudiar diferencias en el rendimiento.
- Nuestro algoritmo de balanceo MPR tiene un proceso de creación de regiones bastante complejo.
- El balanceador MPR no garantiza la existencia de regiones.
- Vamos a simplificar el proceso de creación de regiones.

Algoritmo de balanceo ERP: *Exclusive Regions Prediction*

- El nuevo balanceador predicirá exclusivamente regiones. Se elimina por tanto la predicción de mínimos.
- Se generan dos regiones. Los líderes serán las cajas con menor $\underline{F}(X)$.

Un nuevo algoritmo de balanceo basado en regiones

- Nos interesa comparar entre sí dos algoritmos de balanceo basados en regiones para estudiar diferencias en el rendimiento.
- Nuestro algoritmo de balanceo MPR tiene un proceso de creación de regiones bastante complejo.
- El balanceador MPR no garantiza la existencia de regiones.
- Vamos a simplificar el proceso de creación de regiones.

Algoritmo de balanceo ERP: *Exclusive Regions Prediction*

- El nuevo balanceador predicirá exclusivamente regiones. Se elimina por tanto la predicción de mínimos.
- Se generan dos regiones. Los líderes serán las cajas con menor $\underline{F}(X)$.

Un nuevo algoritmo de balanceo basado en regiones

- Nos interesa comparar entre sí dos algoritmos de balanceo basados en regiones para estudiar diferencias en el rendimiento.
- Nuestro algoritmo de balanceo MPR tiene un proceso de creación de regiones bastante complejo.
- El balanceador MPR no garantiza la existencia de regiones.
- Vamos a simplificar el proceso de creación de regiones.

Algoritmo de balanceo ERP: *Exclusive Regions Prediction*

- El nuevo balanceador predicirá exclusivamente regiones. Se elimina por tanto la predicción de mínimos.
- Se generan dos regiones. Los líderes serán las cajas con menor $\underline{F}(X)$.

Un nuevo algoritmo de balanceo basado en regiones

- Nos interesa comparar entre sí dos algoritmos de balanceo basados en regiones para estudiar diferencias en el rendimiento.
- Nuestro algoritmo de balanceo MPR tiene un proceso de creación de regiones bastante complejo.
- El balanceador MPR no garantiza la existencia de regiones.
- Vamos a simplificar el proceso de creación de regiones.

Algoritmo de balanceo ERP: *Exclusive Regions Prediction*

- El nuevo balanceador predicirá exclusivamente regiones. Se elimina por tanto la predicción de mínimos.
- Se generan dos regiones. Los líderes serán las cajas con menor $\underline{F}(X)$.

Un nuevo algoritmo de balanceo basado en regiones

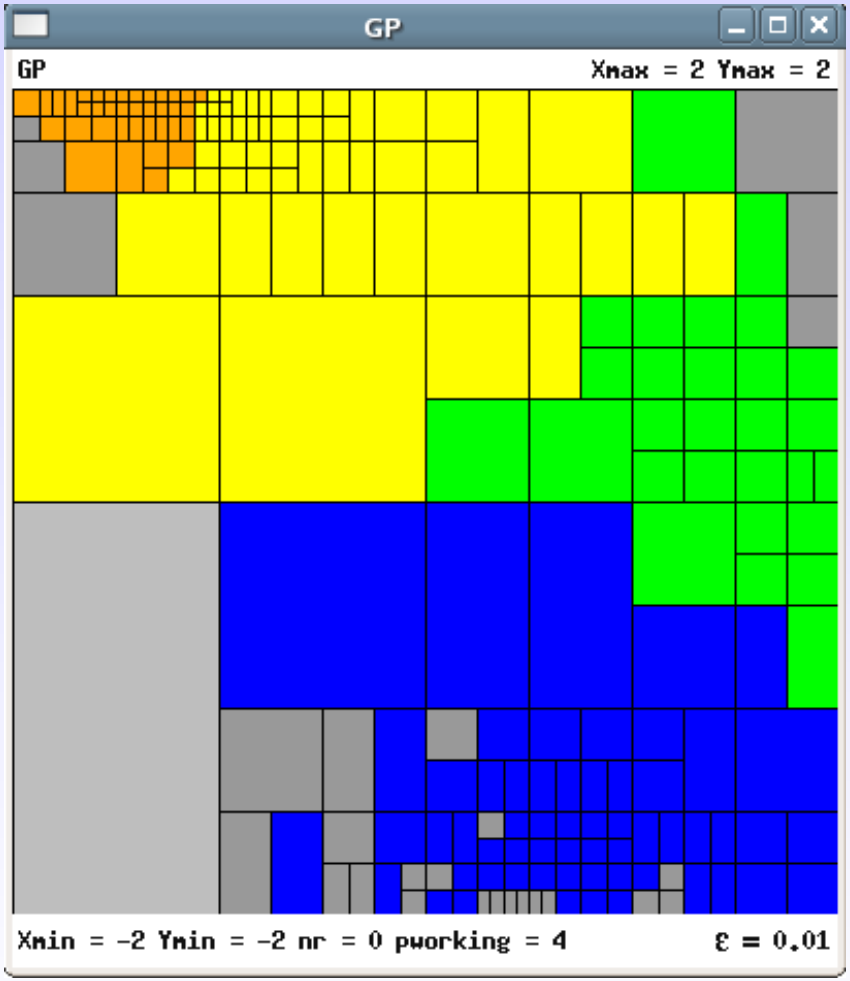
- Nos interesa comparar entre sí dos algoritmos de balanceo basados en regiones para estudiar diferencias en el rendimiento.
- Nuestro algoritmo de balanceo MPR tiene un proceso de creación de regiones bastante complejo.
- El balanceador MPR no garantiza la existencia de regiones.
- Vamos a simplificar el proceso de creación de regiones.

Algoritmo de balanceo ERP: *Exclusive Regions Prediction*

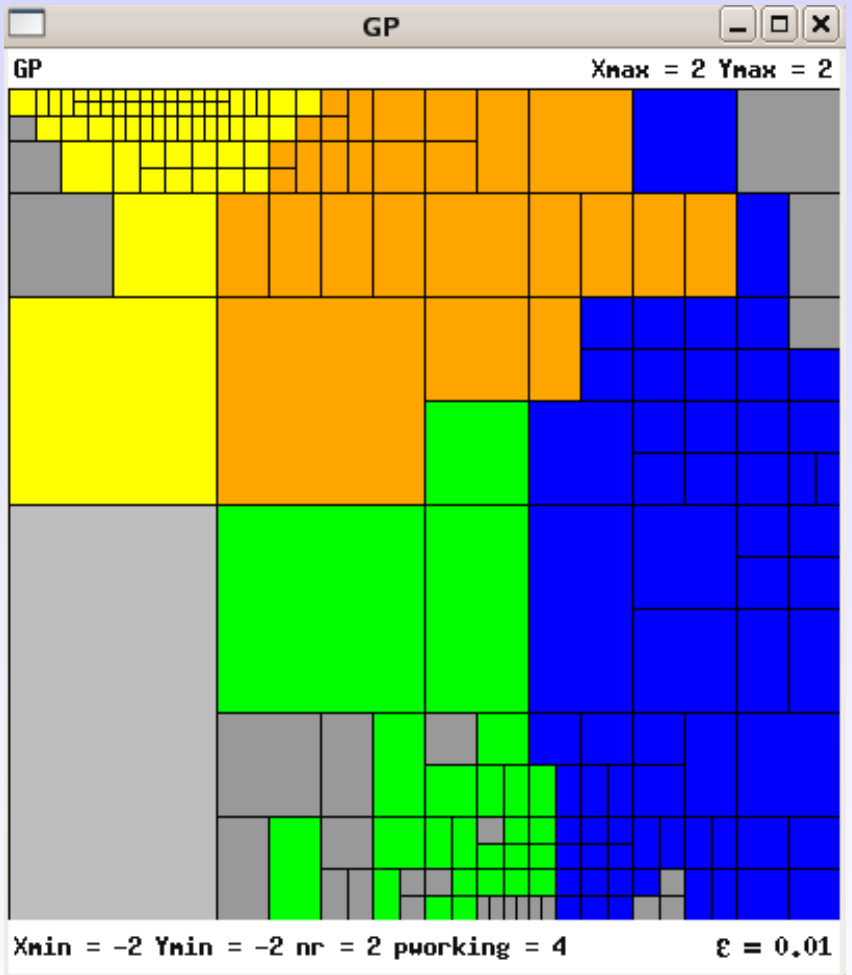
- El nuevo balanceador predicirá exclusivamente regiones. Se elimina por tanto la predicción de mínimos.
- Se generan dos regiones. Los líderes serán las cajas con menor $\underline{F}(X)$.

Diferencias entre MPR y ERP

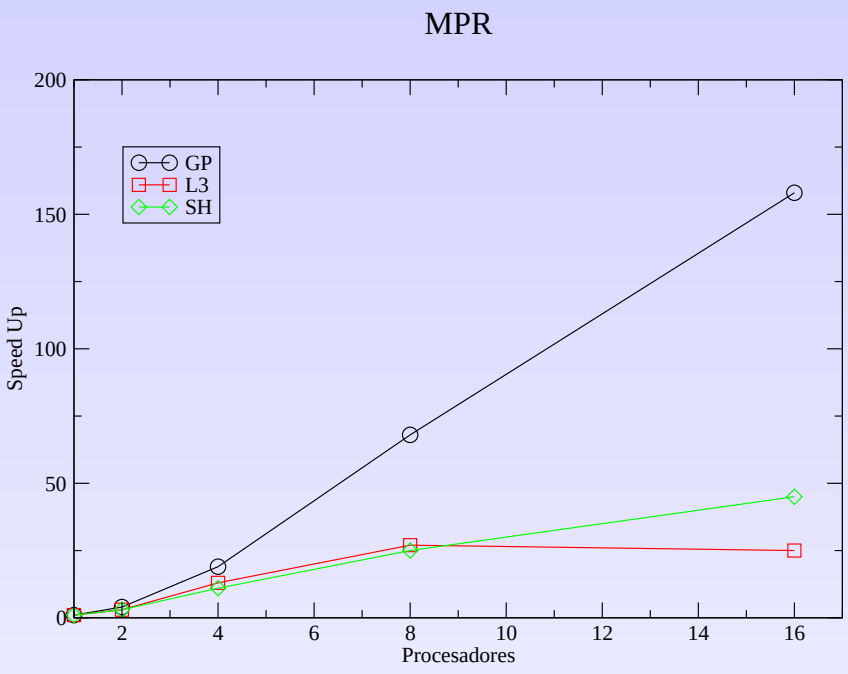
GP con balanceo MPR

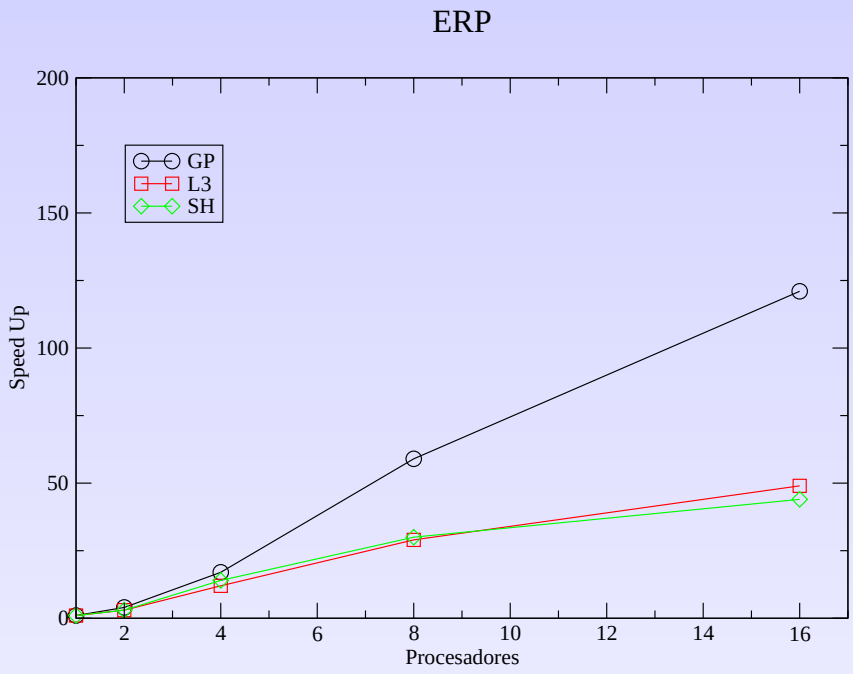


GP con balanceo ERP



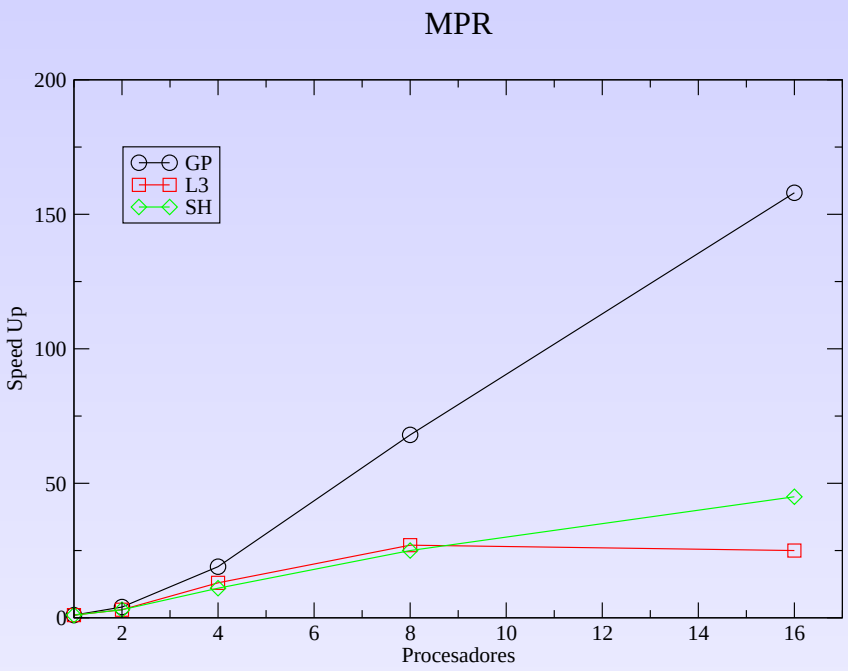
SpeedUp MPR – ERP

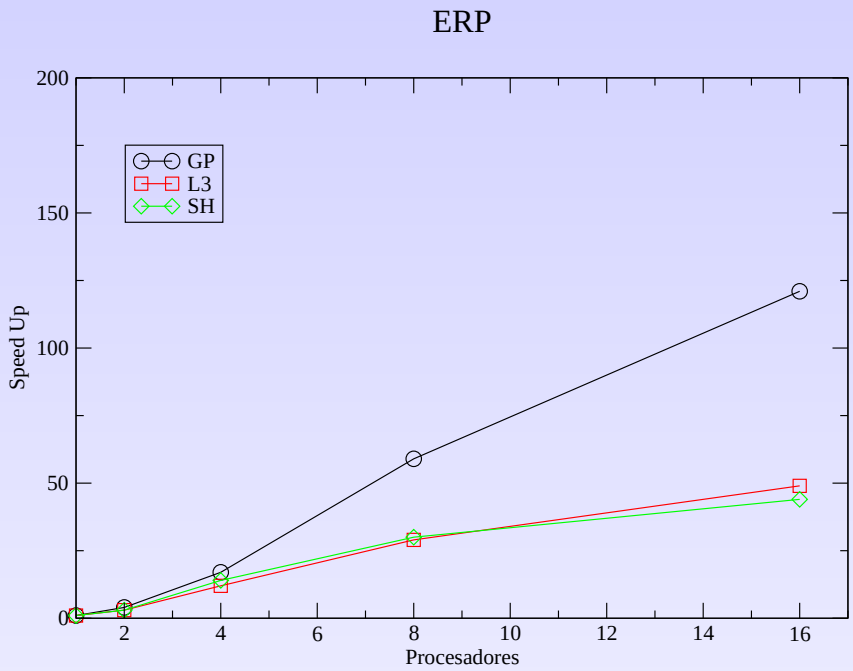




Modelo	MPR			ERP		
2 procesadores	GP	L3	SH	GP	L3	SH
Speed-up	4,26	3,84	3,65	4,20	3,80	3,69
T. Proceso	983,67	175,43	250,21	997,22	177,67	248,97
4 procesador	GP	L3	SH	GP	L3	SH
Speed-up	19,31	13,36	11,89	17,82	12,74	14,59
T. Proceso	420,37	94,4	137	459,25	98,8	115,49
8 procesadores	GP	L3	SH	GP	L3	SH
Speed-up	68,87	27,77	25,19	59,73	29,42	30,94
T. Proceso	205,42	71,76	104,73	241,48	69,98	87,52
16 procesadores	GP	L3	SH	GP	L3	SH
Speed-up	158,48	25,59	45,14	121,32	49,04	44,06
T. Proceso	146,04	92,28	93,55	179,16	68,31	95,48

SpeedUp MPR – ERP

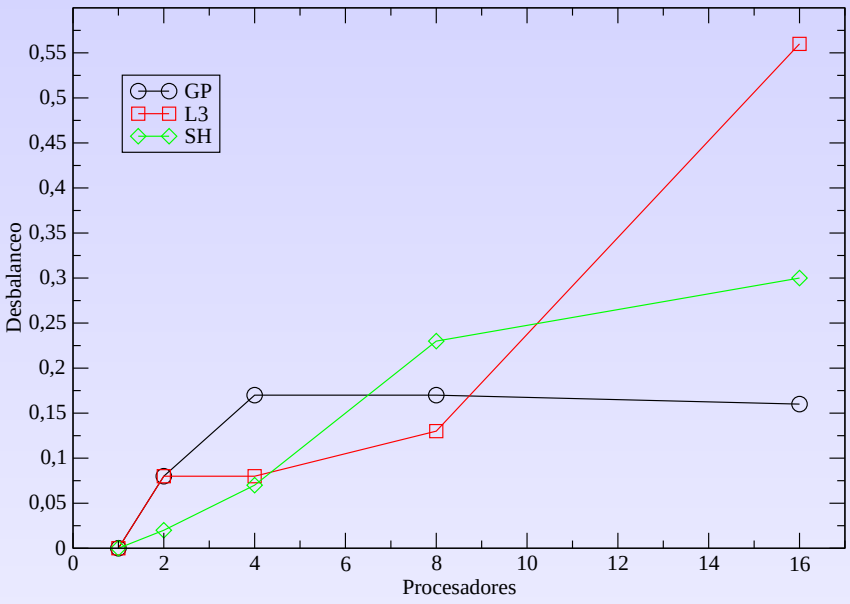




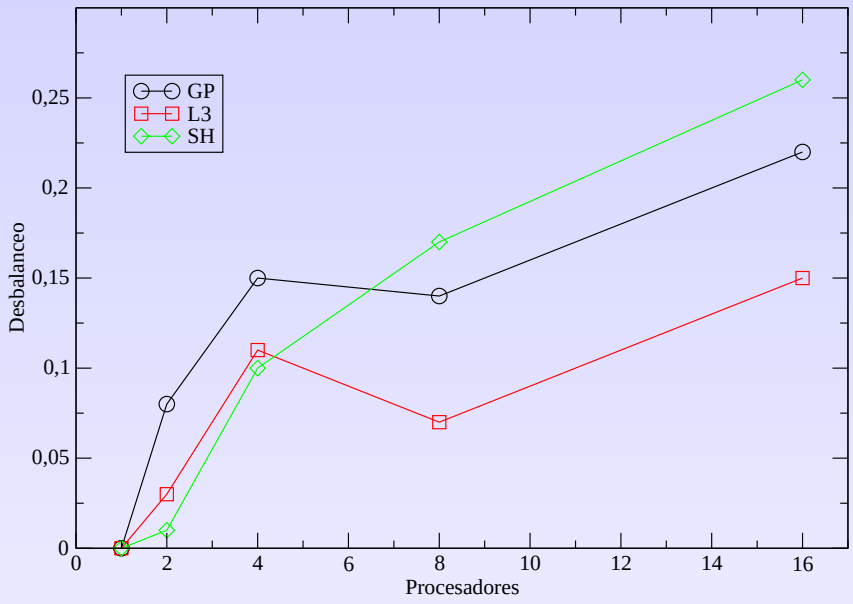
Modelo	MPR			ERP		
2 procesadores	GP	L3	SH	GP	L3	SH
Speed-up	4,26	3,84	3,65	4,20	3,80	3,69
T. Proceso	983,67	175,43	250,21	997,22	177,67	248,97
4 procesador	GP	L3	SH	GP	L3	SH
Speed-up	19,31	13,36	11,89	17,82	12,74	14,59
T. Proceso	420,37	94,4	137	459,25	98,8	115,49
8 procesadores	GP	L3	SH	GP	L3	SH
Speed-up	68,87	27,77	25,19	59,73	29,42	30,94
T. Proceso	205,42	71,76	104,73	241,48	69,98	87,52
16 procesadores	GP	L3	SH	GP	L3	SH
Speed-up	158,48	25,59	45,14	121,32	49,04	44,06
T. Proceso	146,04	92,28	93,55	179,16	68,31	95,48

Desbalanceo MPR – ERP

MPR



ERP



Modelo	MPR			ERP		
2 procesadores	GP	L3	SH	GP	L3	SH
Repartos	15	10,67	8,33	16	11	7,33
Predicciones	1742	46,33	342,67			
4 procesador	GP	L3	SH	GP	L3	SH
Repartos	64,67	28,33	80	68	35	49,33
Predicciones	3233,67	805,67	367,67			
8 procesadores	GP	L3	SH	GP	L3	SH
Repartos	141	103	174,33	174,33	91	132
Predicciones	5617,67	1185	768,67			
16 procesadores	GP	L3	SH	GP	L3	SH
Repartos	262,67	204,33	270	367	155,33	281,67
Predicciones	4372,33	7757,33	1074,33			

Evolución de cajas en los procesadores – ERP

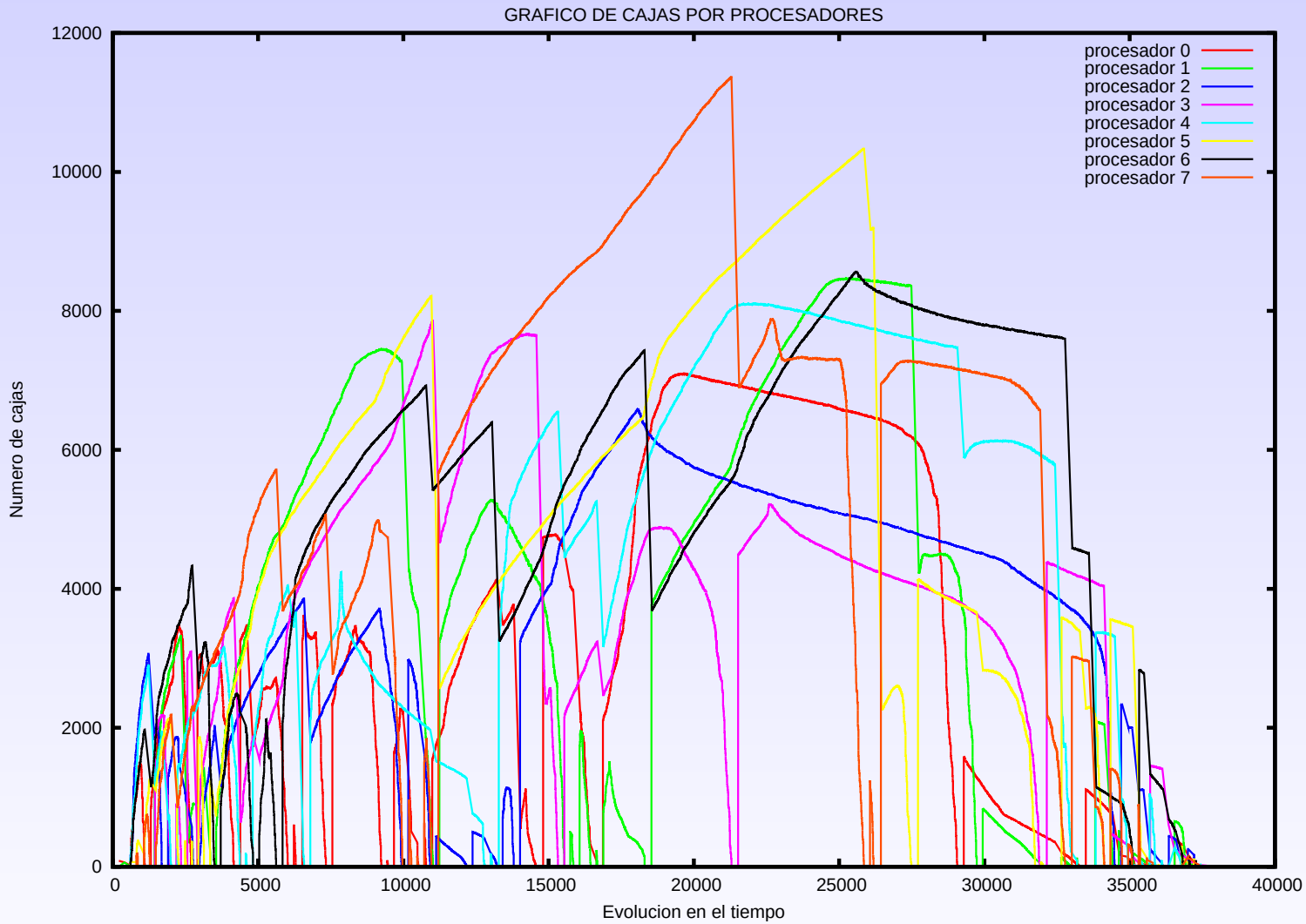


Figura: Evolución de cajas en los procesadores para el problema GP con 8 procesadores, balanceador ERP y $\epsilon = 10^{-4}$.

Conclusiones generales

- La implementación del programa de simulación paralela ogsp nos permitirá ensayar con nuevos algoritmos de balanceo paralelos de manera más sencilla.
- No encontramos anomalías en los algoritmos paralelos en cuanto al número de cajas evaluadas, con el simple hecho de utilizar regiones en el balanceo.
- BAR tiene mejor Speed-Up porque mantiene todos los procesadores ocupados.
- MPR tiene peor Speed-Up que BAR porque permite tener procesadores libres y las regiones generadas no tienen una carga computacional balanceada.
- ERP mantiene la idea de regiones haciendo que todos los procesadores estén ocupados (como hace BAR), pero penaliza el número de balanceos al no estar las regiones balanceadas, y el tiempo de creación de esas regiones.

Trabajo para el futuro

- Desarrollar los algoritmos paralelos de predicción de mínimos en arquitecturas reales.
- Utilizar árboles binarios balanceados en lugar de listas.
- Usar más funciones de prueba
- Añadir nuevos test de rechazo a los algoritmos
- Creación de regiones balanceadas, y mantener las regiones en el procesador en tiempo de ejecución.
- Investigar nuevos métodos basados en las regiones, que permitan mejorar la eficiencia de los algoritmos de Optimización Global paralelos,
- Actualizar en tiempo de ejecución el cálculo del tiempo de proceso de una caja.
- Factor adaptativo del trabajo de un procesador.
- Investigar nuevas fórmulas del cálculo del umbral de candidatos y de los líderes de las regiones.
- Nuevos factores de cercanía de cajas.

Trabajo para el futuro

- Desarrollar los algoritmos paralelos de predicción de mínimos en arquitecturas reales.
- Utilizar árboles binarios balanceados en lugar de listas.
- Usar más funciones de prueba
- Añadir nuevos test de rechazo a los algoritmos
- Creación de regiones balanceadas, y mantener las regiones en el procesador en tiempo de ejecución.
- Investigar nuevos métodos basados en las regiones, que permitan mejorar la eficiencia de los algoritmos de Optimización Global paralelos,
- Actualizar en tiempo de ejecución el cálculo del tiempo de proceso de una caja.
- Factor adaptativo del trabajo de un procesador.
- Investigar nuevas fórmulas del cálculo del umbral de candidatos y de los líderes de las regiones.
- Nuevos factores de cercanía de cajas.

Trabajo para el futuro

- Desarrollar los algoritmos paralelos de predicción de mínimos en arquitecturas reales.
- Utilizar árboles binarios balanceados en lugar de listas.
- Usar más funciones de prueba
- Añadir nuevos test de rechazo a los algoritmos
- Creación de regiones balanceadas, y mantener las regiones en el procesador en tiempo de ejecución.
- Investigar nuevos métodos basados en las regiones, que permitan mejorar la eficiencia de los algoritmos de Optimización Global paralelos,
- Actualizar en tiempo de ejecución el cálculo del tiempo de proceso de una caja.
- Factor adaptativo del trabajo de un procesador.
- Investigar nuevas fórmulas del cálculo del umbral de candidatos y de los líderes de las regiones.
- Nuevos factores de cercanía de cajas.

Trabajo para el futuro

- Desarrollar los algoritmos paralelos de predicción de mínimos en arquitecturas reales.
- Utilizar árboles binarios balanceados en lugar de listas.
- Usar más funciones de prueba
- Añadir nuevos test de rechazo a los algoritmos
- Creación de regiones balanceadas, y mantener las regiones en el procesador en tiempo de ejecución.
- Investigar nuevos métodos basados en las regiones, que permitan mejorar la eficiencia de los algoritmos de Optimización Global paralelos,
- Actualizar en tiempo de ejecución el cálculo del tiempo de proceso de una caja.
- Factor adaptativo del trabajo de un procesador.
- Investigar nuevas fórmulas del cálculo del umbral de candidatos y de los líderes de las regiones.
- Nuevos factores de cercanía de cajas.

Trabajo para el futuro

- Desarrollar los algoritmos paralelos de predicción de mínimos en arquitecturas reales.
- Utilizar árboles binarios balanceados en lugar de listas.
- Usar más funciones de prueba
- Añadir nuevos test de rechazo a los algoritmos
- Creación de regiones balanceadas, y mantener las regiones en el procesador en tiempo de ejecución.
- Investigar nuevos métodos basados en las regiones, que permitan mejorar la eficiencia de los algoritmos de Optimización Global paralelos,
- Actualizar en tiempo de ejecución el cálculo del tiempo de proceso de una caja.
- Factor adaptativo del trabajo de un procesador.
- Investigar nuevas fórmulas del cálculo del umbral de candidatos y de los líderes de las regiones.
- Nuevos factores de cercanía de cajas.

Trabajo para el futuro

- Desarrollar los algoritmos paralelos de predicción de mínimos en arquitecturas reales.
- Utilizar árboles binarios balanceados en lugar de listas.
- Usar más funciones de prueba
- Añadir nuevos test de rechazo a los algoritmos
- Creación de regiones balanceadas, y mantener las regiones en el procesador en tiempo de ejecución.
- Investigar nuevos métodos basados en las regiones, que permitan mejorar la eficiencia de los algoritmos de Optimización Global paralelos,
- Actualizar en tiempo de ejecución el cálculo del tiempo de proceso de una caja.
- Factor adaptativo del trabajo de un procesador.
- Investigar nuevas fórmulas del cálculo del umbral de candidatos y de los líderes de las regiones.
- Nuevos factores de cercanía de cajas.

Trabajo para el futuro

- Desarrollar los algoritmos paralelos de predicción de mínimos en arquitecturas reales.
- Utilizar árboles binarios balanceados en lugar de listas.
- Usar más funciones de prueba
- Añadir nuevos test de rechazo a los algoritmos
- Creación de regiones balanceadas, y mantener las regiones en el procesador en tiempo de ejecución.
- Investigar nuevos métodos basados en las regiones, que permitan mejorar la eficiencia de los algoritmos de Optimización Global paralelos,
- Actualizar en tiempo de ejecución el cálculo del tiempo de proceso de una caja.
- Factor adaptativo del trabajo de un procesador.
- Investigar nuevas fórmulas del cálculo del umbral de candidatos y de los líderes de las regiones.
- Nuevos factores de cercanía de cajas.

Trabajo para el futuro

- Desarrollar los algoritmos paralelos de predicción de mínimos en arquitecturas reales.
- Utilizar árboles binarios balanceados en lugar de listas.
- Usar más funciones de prueba
- Añadir nuevos test de rechazo a los algoritmos
- Creación de regiones balanceadas, y mantener las regiones en el procesador en tiempo de ejecución.
- Investigar nuevos métodos basados en las regiones, que permitan mejorar la eficiencia de los algoritmos de Optimización Global paralelos,
- Actualizar en tiempo de ejecución el cálculo del tiempo de proceso de una caja.
- Factor adaptativo del trabajo de un procesador.
- Investigar nuevas fórmulas del cálculo del umbral de candidatos y de los líderes de las regiones.
- Nuevos factores de cercanía de cajas.

Trabajo para el futuro

- Desarrollar los algoritmos paralelos de predicción de mínimos en arquitecturas reales.
- Utilizar árboles binarios balanceados en lugar de listas.
- Usar más funciones de prueba
- Añadir nuevos test de rechazo a los algoritmos
- Creación de regiones balanceadas, y mantener las regiones en el procesador en tiempo de ejecución.
- Investigar nuevos métodos basados en las regiones, que permitan mejorar la eficiencia de los algoritmos de Optimización Global paralelos,
- Actualizar en tiempo de ejecución el cálculo del tiempo de proceso de una caja.
- Factor adaptativo del trabajo de un procesador.
- Investigar nuevas fórmulas del cálculo del umbral de candidatos y de los líderes de las regiones.
- Nuevos factores de cercanía de cajas.

Trabajo para el futuro

- Desarrollar los algoritmos paralelos de predicción de mínimos en arquitecturas reales.
- Utilizar árboles binarios balanceados en lugar de listas.
- Usar más funciones de prueba
- Añadir nuevos test de rechazo a los algoritmos
- Creación de regiones balanceadas, y mantener las regiones en el procesador en tiempo de ejecución.
- Investigar nuevos métodos basados en las regiones, que permitan mejorar la eficiencia de los algoritmos de Optimización Global paralelos,
- Actualizar en tiempo de ejecución el cálculo del tiempo de proceso de una caja.
- Factor adaptativo del trabajo de un procesador.
- Investigar nuevas fórmulas del cálculo del umbral de candidatos y de los líderes de las regiones.
- Nuevos factores de cercanía de cajas.

Preguntas

MUCHAS GRACIAS