Algoritmos paralelos para la determinación de modelos de series temporales multivariantes

TRABAJO FIN DE GRADO
2018/2019

Autor: Juan Antonio Rodríguez Lorente

Trabajo dirigido por Antonio Javier Cuenca Muñoz y Domingo Giménez Cánovas

Índice

1. Introducción.

1. Introducción.

- **Big Data:** economía, medicina…
- **Series temporales:** análisis de los datos
- **Series temporales multivariantes**

1. Introducción.

$$Y(i+1:t,:) \approx$$
$$[Y(i:t-1,:)|\ldots|Y(1:t-i,:)|Z(i:t-1,:)|\ldots|Z(i-k+1:t-k,:)|1] \quad * \begin{bmatrix} A_1 \\ \vdots \\ A_i \\ B_1 \\ \vdots \\ B_k \\ a \end{bmatrix}$$

- El problema consiste en:
  - Dadas las matrices **Y** (*parámetros internos*), **Z** (*parámetros externos*)
  - Encontrar un **modelo**

- **Dos tipos** de problema

2. Implementaciones de un Algoritmo Genético.

2. Implementaciones de un Algoritmo Genético.

Solución secuencial

```
1  Inicializar (S,Sol)
2  while (not CondicionFin()) do
3      for tamPoblacion/2 do
4          I1 = SeleccionTorneo (S)
5          I2 = SeleccionTorneo (S)
6          Cruzar (I1,I2)
7          Mutar (I1)
8          Mutar (I2)
9          Fitness (I1)
10         Fitness (I2)
11         ActualizarSolucion (I1,Sol)
12         ActualizarSolucion (I2,Sol)
13         SS = Incluir (I1,I2)
14     end
15     S = SS
16     SS = ∅
17 end
```

- **Inicializar:** aleatorio
- **Nueva población:** se generan n individuos **de dos en dos**
- Cada **par de individuos** se genera de la siguiente forma:
  - *Selección Torneo*
  - *Cruzar*
  - *Mutar*
  - *Calcular nuevos fitness*
  - *Actualizar solución global*
  - *Incluir*

2. Implementaciones de un Algoritmo Genético.

Soluciones paralelas

- **Esquema de islas:** cada hilo/proceso trabaja en una *población reducida*
- Endogamia -> **Migraciones**
- **Speed-Up:** utilizando p hilos/procesos, **casi de un factor p**

2. Implementaciones de un Algoritmo Genético.

Solución OpenMP

```
1  GenerarIndividuo (Sol)
2  EN PARALELO en cada hilo P_i (i = 0, ..., n − 1) HACER
3  Inicializar (S_i,Sol_i,tamPoblacion/n)
4  for numGeneraciones do
5      for numIteraciones do
6          EsquemaSecuencial (S_i,Sol_i,tamPoblacion/n)
7      end
8      Barrera
9      Copiar Sol_i en el array mejores
10     Barrera
11     Integrar cada individuo (excepto Sol_i) del array mejores en S_i
12 end
13 Sincronización para actualizar Sol con Sol_i si es necesario
14 FIN PARALELO
```

- Se crea una **solución global** aleatoria
- Se **lanzan n hilos**
- **Inicializar:** poblaciones reducidas y soluciones locales
- **Migraciones:** agrupar iteraciones en generaciones
- **Sincronización:** actualizar solución global

2. Implementaciones de un Algoritmo Genético.

Solución MPI

```
1  EN PARALELO en cada proceso P_i (i = 0, ..., p − 1) HACER
2  P_0 envía los datos necesarios al resto de procesos
3  if nodo == P_0 then
4      GenerarIndividuo (Sol)
5  end
6  Inicializar (S_i,Sol_i,tamPoblacion/p)
7  for numGeneraciones do
8      for numIteraciones do
9          EsquemaSecuencial (S_i,Sol_i,tamPoblacion/p)
10     end
11     MPI_Allgather (Sol_i,mejores)
12     Integrar cada individuo (excepto Sol_i) del array mejores en S_i
13 end
14 MPI_Gather (Sol_i,solucionesLocales)
15 P_0 actualiza Sol con los individuos del array solucionesLocales
16 FIN PARALELO
```

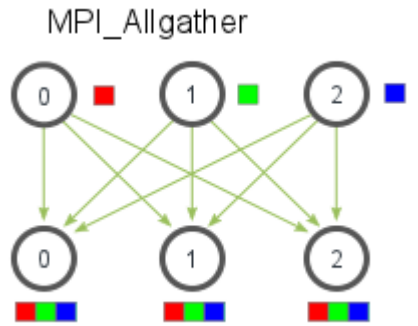
- Se **lanzan p procesos**
- **Proceso 0:** *envía los datos necesarios a los demás* procesos y crea una *solución global* aleatoria
- **Inicializar:** poblaciones reducidas y soluciones locales
- **Migraciones :** agrupar iteraciones en generaciones

2. Implementaciones de un Algoritmo Genético.

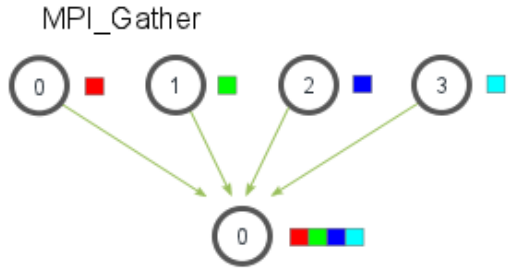
Solución MPI

```
1  EN PARALELO en cada proceso P_i (i = 0, ..., p − 1) HACER
2  P_0 envía los datos necesarios al resto de procesos
3  if nodo == P_0 then
4      GenerarIndividuo (Sol)
5  end
6  Inicializar (S_i,Sol_i,tamPoblacion/p)
7  for numGeneraciones do
8      for numIteraciones do
9          EsquemaSecuencial (S_i,Sol_i,tamPoblacion/p)
10     end
11     MPI_Allgather (Sol_i,mejores)
12     Integrar cada individuo (excepto Sol_i) del array mejores en S_i
13 end
14 MPI_Gather (Sol_i,solucionesLocales)
15 P_0 actualiza Sol con los individuos del array solucionesLocales
16 FIN PARALELO
```

- **MPI_Allgather:** comunicar soluciones locales



- **Proceso 0:** actualiza la solución global *al final* utilizando el método **MPI_Gather**



2. Implementaciones de un Algoritmo Genético.

Solución híbrida (MPI+OpenMP)

```
1  EN PARALELO en cada proceso P_i (i = 0, ..., p − 1) HACER
2  P_0 envía los datos necesarios al resto de procesos
3  if nodo == P_0 then
4      GenerarIndividuo (Sol)
5  end
6  Inicializar (S_i,Sol_i,tamPoblacion/p)
7  for numGeneraciones do
8      for numIteraciones do
9          EsquemaSecuencial (S_i,Sol_i,tamPoblacion/p)
10     end
11     MPI_Allgather (Sol_i,mejores)
12     Integrar cada individuo (excepto Sol_i) del array mejores en S_i
13 end
14 MPI_Gather (Sol_i,solucionesLocales)
15 P_0 actualiza Sol con los individuos del array solucionesLocales
16 FIN PARALELO
```

Cada hilo inicializa una subpoblación de tamaño tamPoblacion/p*n y una solución

Se lanzan n hilos y con una sincronización se actualiza la solución local del proceso

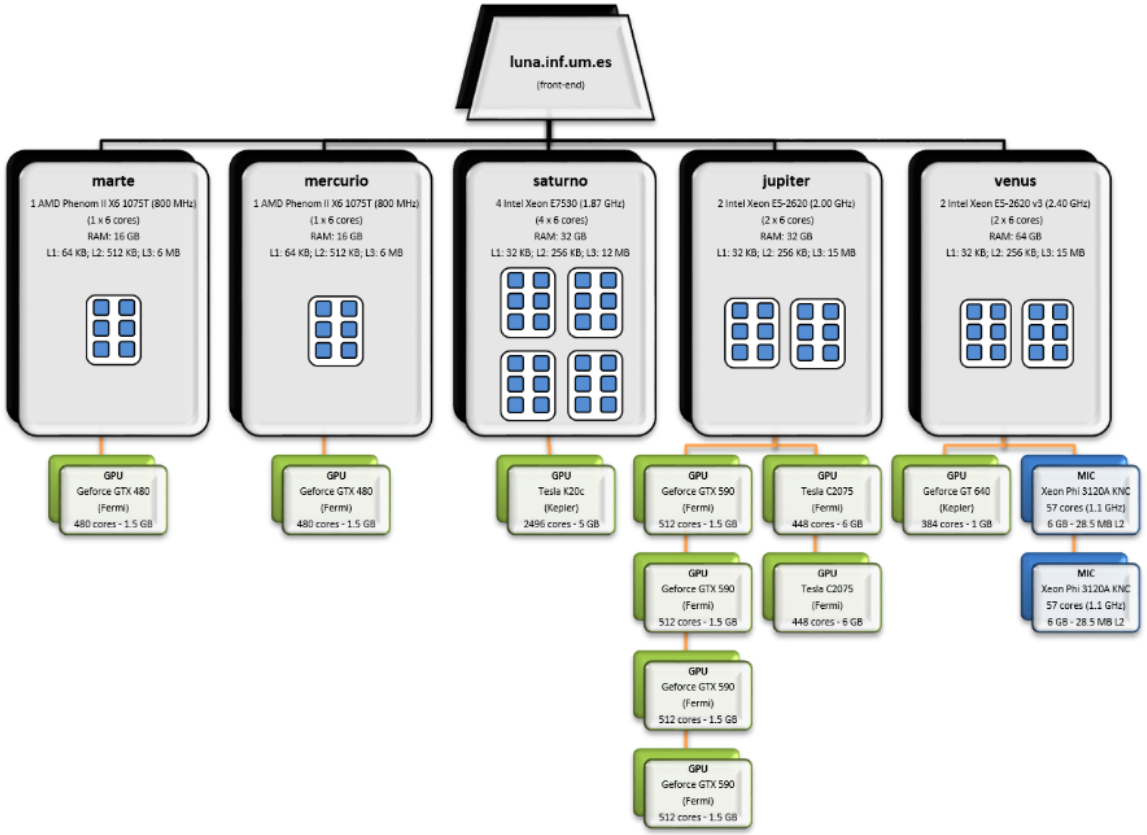
Cada hilo introduce las soluciones de los demás procesos en su subpoblación

- Idea: combinar **MPI y OpenMP**
- **Similar al esquema MPI**

3. Resultados experimentales.

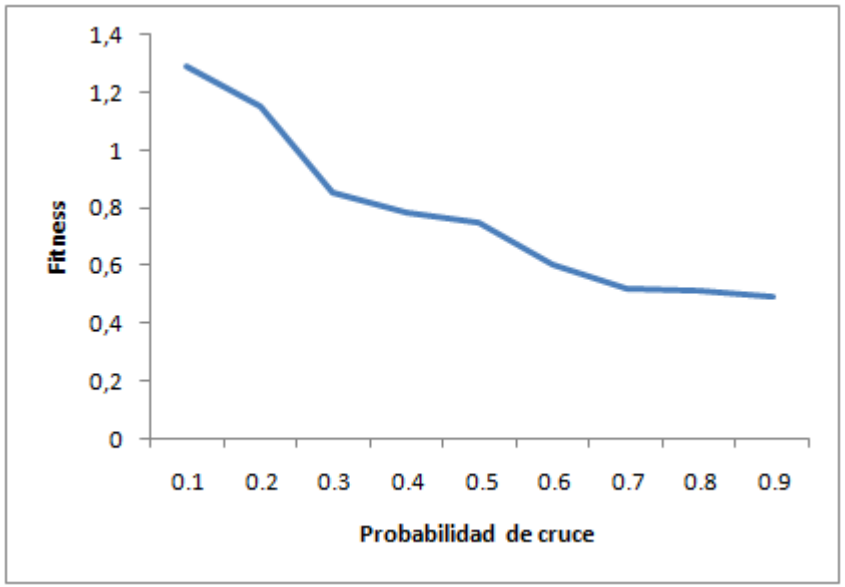
3. Resultados experimentales.

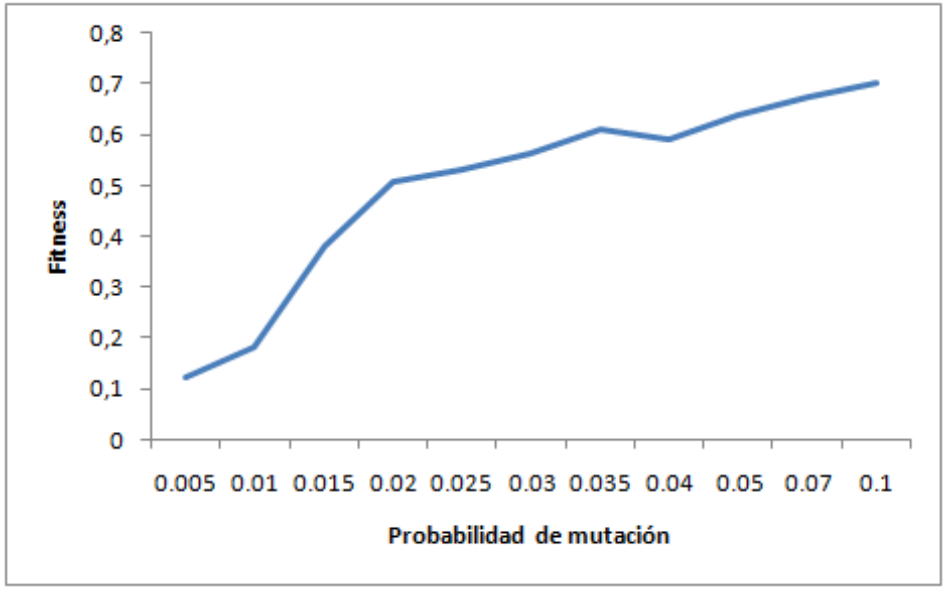
- **Problema fijo:** mejor análisis de las soluciones
- **Resultados fiables:** se han tomado promedios
- **Clúster Heterosolar**



3. Resultados experimentales.

- Tuneado de parámetros



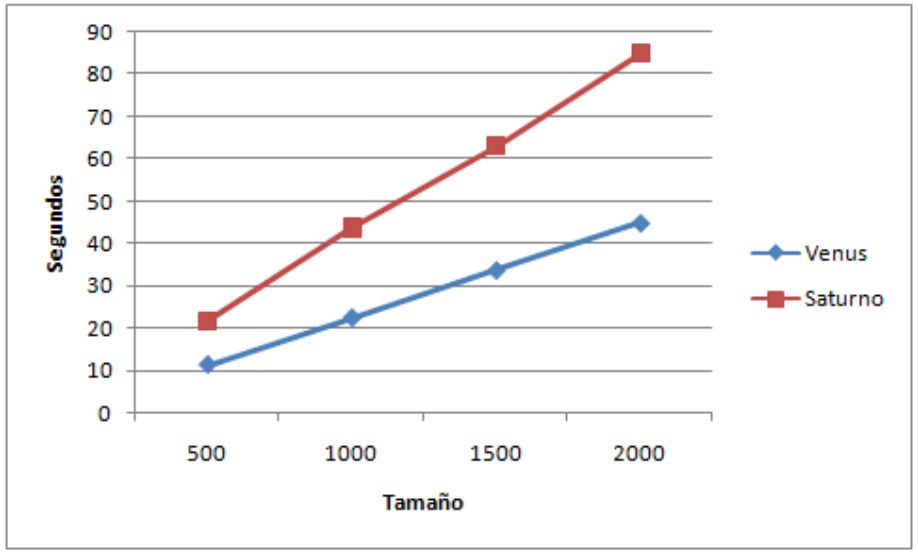


+ Cruces, - Mutaciones -> Mejores soluciones
(para un tiempo fijo)

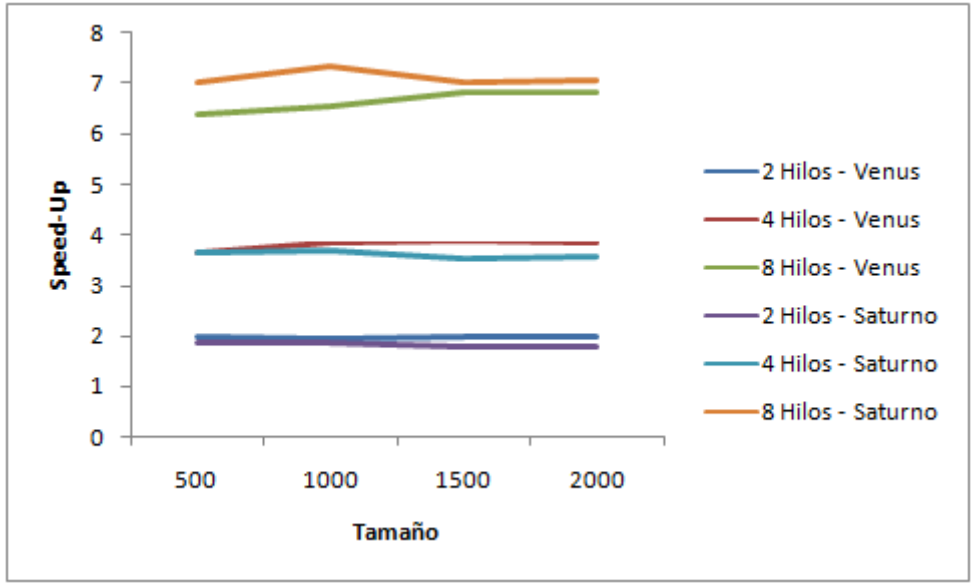
3. Resultados experimentales.

Experimentos de tiempo

- Con la solución **secuencial**:
  - Comportamiento lineal
  - Venus + rápido



- Con la solución **OpenMP**:
  - Speed-Up: casi lineal hasta 8 hilos, fluctúa por las sincronizaciones
  - Saturno + Speed-Up

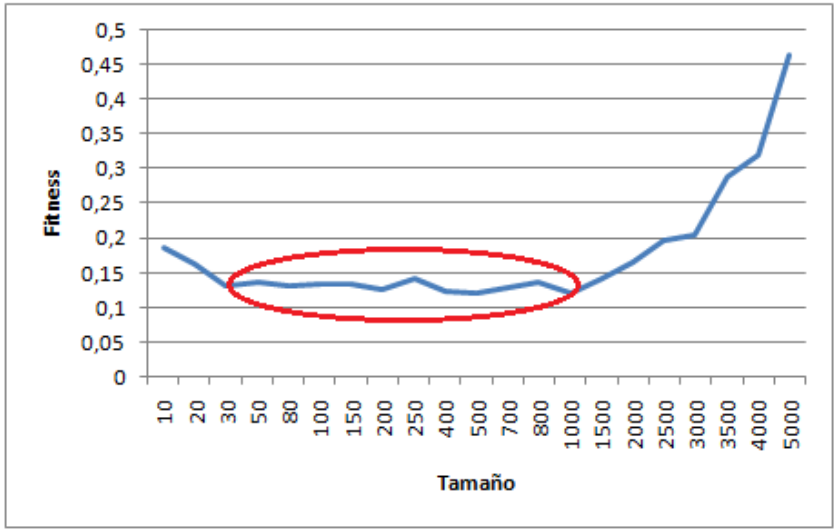


Mismas conclusiones con MPI y OpenMP+MPI

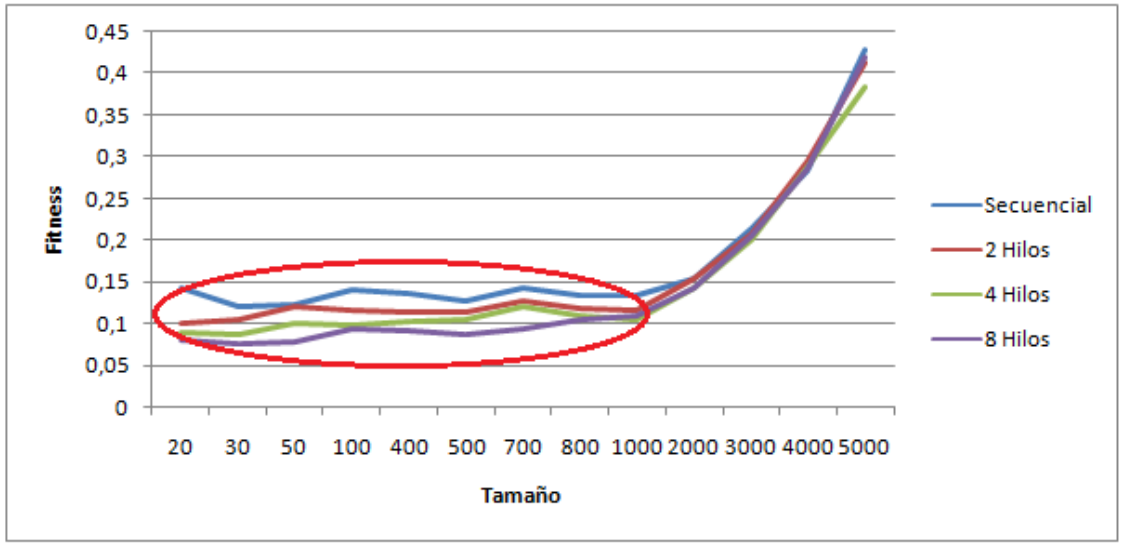
3. Resultados experimentales.

Experimentos de fitness

- Con la solución **secuencial**:
  - Tamaño de población entre 30 y 1000



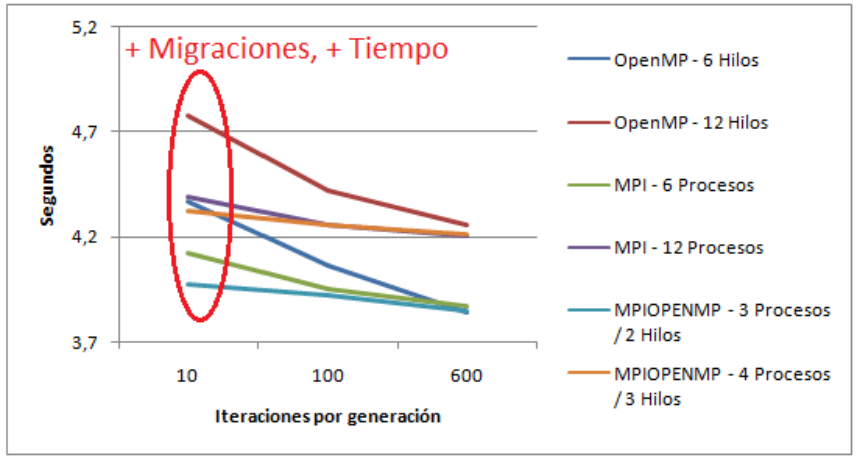
- Con la solución **OpenMP**:
  - OpenMP > Secuencial
  - + Hilos, Mejores soluciones

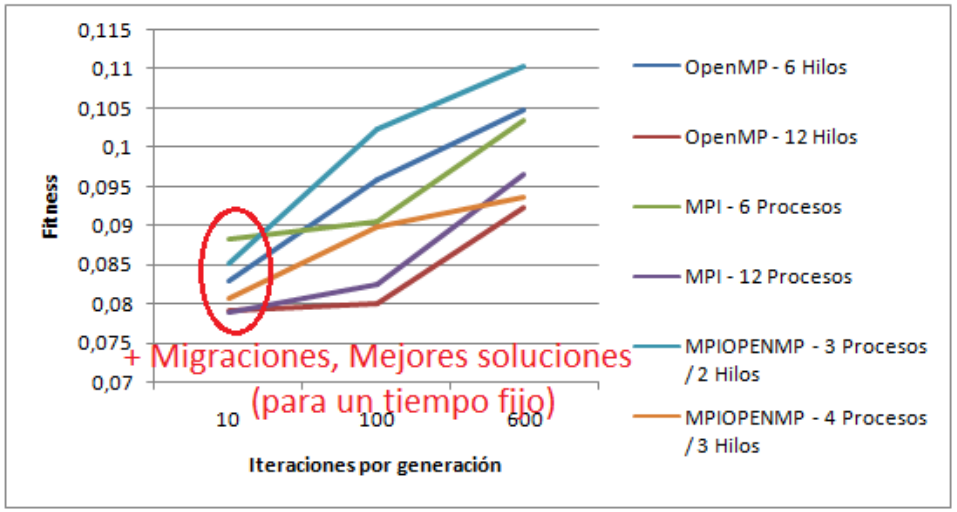


Mismas conclusiones con MPI y OpenMP+MPI

3. Resultados experimentales.

Experimentos de migraciones





+ Migraciones -> Mejores soluciones
(para un tiempo fijo)

4. Mejoras.

4. Mejoras.

- Se consideran **mejoras de distintos tipos**:

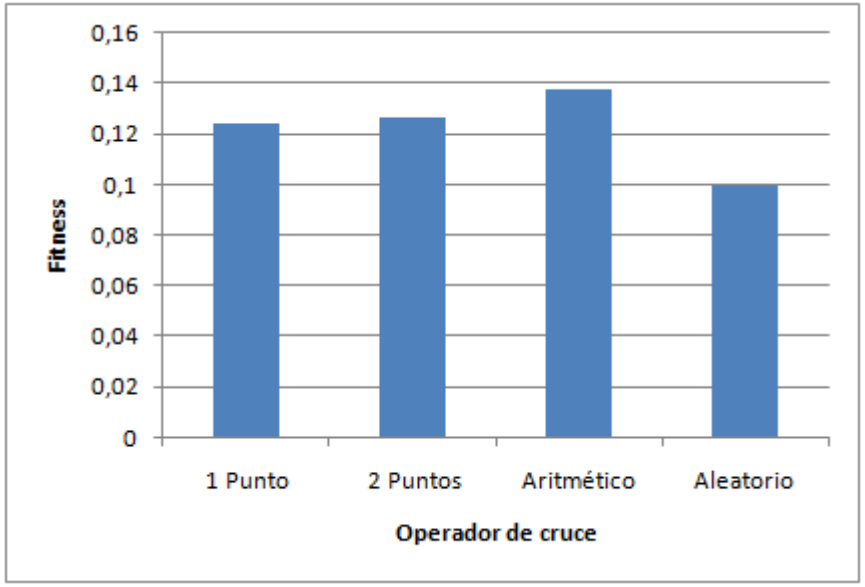
- **Mejoras en las funciones del algoritmo genético:**
  - ❑ Nuevos operadores de cruce
  - ❑ Nuevas funciones de selección
  - ❑ Elitismo
  - ❑ Hibridación con Búsqueda Local

- **Mejoras en el cálculo del fitness:**
  - ❑ Optimización del cálculo del fitness en la mejora de Búsqueda Local
  - ❑ Optimización del cálculo general del fitness

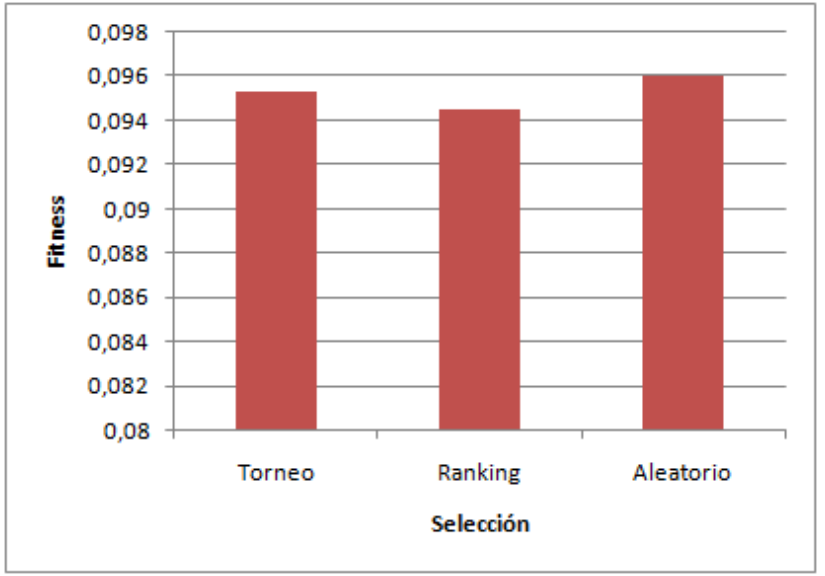
- **Explotación de la Heterogeneidad**

4. Mejoras.

- Mejoras en las funciones del algoritmo genético:
  - ❑ Nuevos operadores de cruce: el mejor es el **Cruce Aleatorio**



  - ❑ Nuevas funciones de selección: no hay mejoras significativas. Se sigue utilizando **Selección por Torneo**



4. Mejoras.

- Mejoras en las funciones del algoritmo genético:
  - ❑ Elitismo: incluir **solo a los mejores** individuos
  - ❑ Hibridación con Búsqueda Local
    - Idea: **Algoritmo Genético + Búsqueda Local**
    - **Vecindad de dos**
    - Análisis de la vecindad: **número determinado de pasos**

```
1  Inicializar (S,Sol)        ⇦ Aplicar Búsqueda Local
2  while (not CondicionFin()) do
3      for tamPoblacion/2 do
4          I1 = SeleccionTorneo (S)
5          I2 = SeleccionTorneo (S)
6          Cruzar (I1,I2)
7          Mutar (I1)
8          Mutar (I2)
9          Fitness (I1)
10         Fitness (I2)
11         ActualizarSolucion (I1,Sol)
12         ActualizarSolucion (I2,Sol)
13         SS = Incluir (I1,I2)
14     end
15     S = SS                 ⇦ Aplicar Búsqueda Local
16     SS = ∅
17 end
```

4. Mejoras.

- Mejoras en el cálculo del fitness:
  - ❑ Optimización del cálculo del fitness en la mejora de Búsqueda Local
    - Problema: la función de **calcular fitness es muy costosa**
    - Idea: **modelos parecidos**
    - Impacto de la mejora en el tiempo: **$O(n^3)$ -> $O(n)$**

$$\begin{pmatrix} y_1^{(3)} & y_2^{(3)} & y_3^{(3)} \\ y_1^{(4)} & y_2^{(4)} & y_3^{(4)} \\ & \dots & \\ y_1^{(t)} & y_2^{(t)} & y_3^{(t)} \end{pmatrix} \approx \begin{pmatrix} y_1^{(2)} & y_2^{(2)} & y_3^{(2)} & y_1^{(1)} & y_2^{(1)} & y_3^{(1)} \\ y_1^{(3)} & y_2^{(3)} & y_3^{(3)} & y_1^{(2)} & y_2^{(2)} & y_3^{(2)} \\ & \dots & & & & \\ y_1^{(t-1)} & y_2^{(t-1)} & y_3^{(t-1)} & y_1^{(t-2)} & y_2^{(t-2)} & y_3^{(t-2)} \end{pmatrix} * \begin{pmatrix} A_{11}^{(1)} & A_{12}^{(1)} & A_{13}^{(1)} \\ A_{21}^{(1)} & A_{22}^{(1)} & A_{23}^{(1)} \\ A_{31}^{(1)} & A_{32}^{(1)} & A_{33}^{(1)} \\ A_{11}^{(2)} & A_{12}^{(2)} & A_{13}^{(2)} \\ A_{21}^{(2)} & A_{22}^{(2)} & A_{23}^{(2)} \\ A_{31}^{(2)} & A_{32}^{(2)} & A_{33}^{(2)} \end{pmatrix}$$

4. Mejoras.

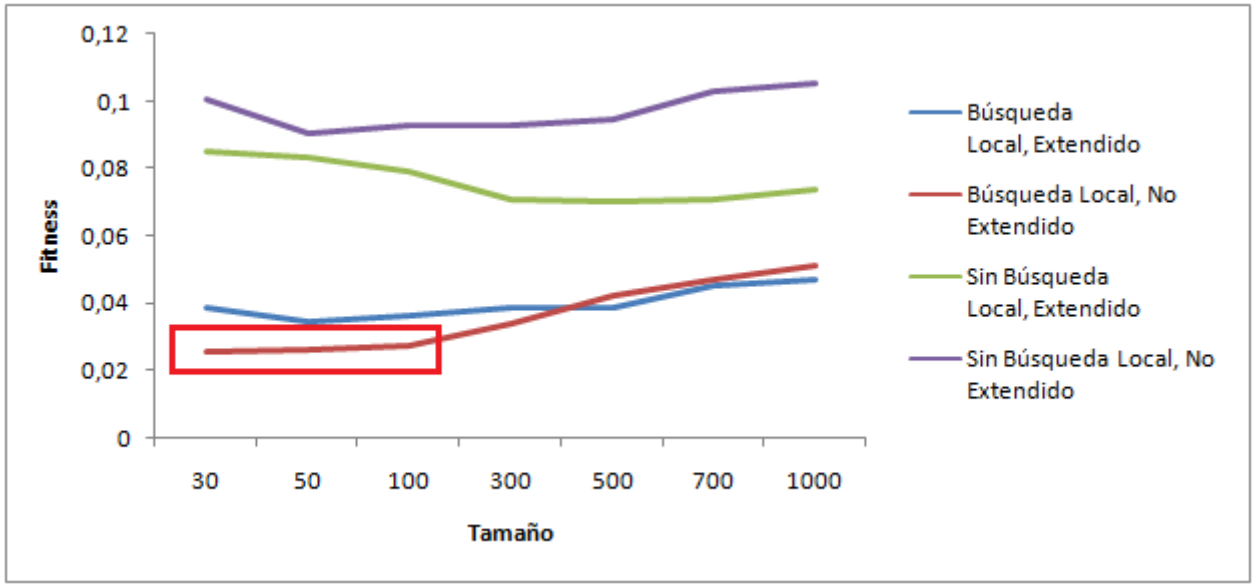
- Mejoras en el cálculo del fitness:
  - ❑ Optimización del cálculo general del fitness
    - Librería BLAS optimizada: OpenBLAS
    - Sustituir bucles: **rutina dgemm**
    - Impacto de la mejora en el tiempo:
      + Tamaño Problema, + Ganancia

Tamaño:	20	200	1000	5000
Venus	1.397	2.021	2.104	3.066
Saturno	1.514	2.014	2.389	3.585

Ganancia al utilizar OpenBLAS
en la función calcular fitness

4. Mejoras.

- Resultados:
  - Mejor combinación: **Búsqueda Local sin elitismo con tamaños de población entre 30 y 100**

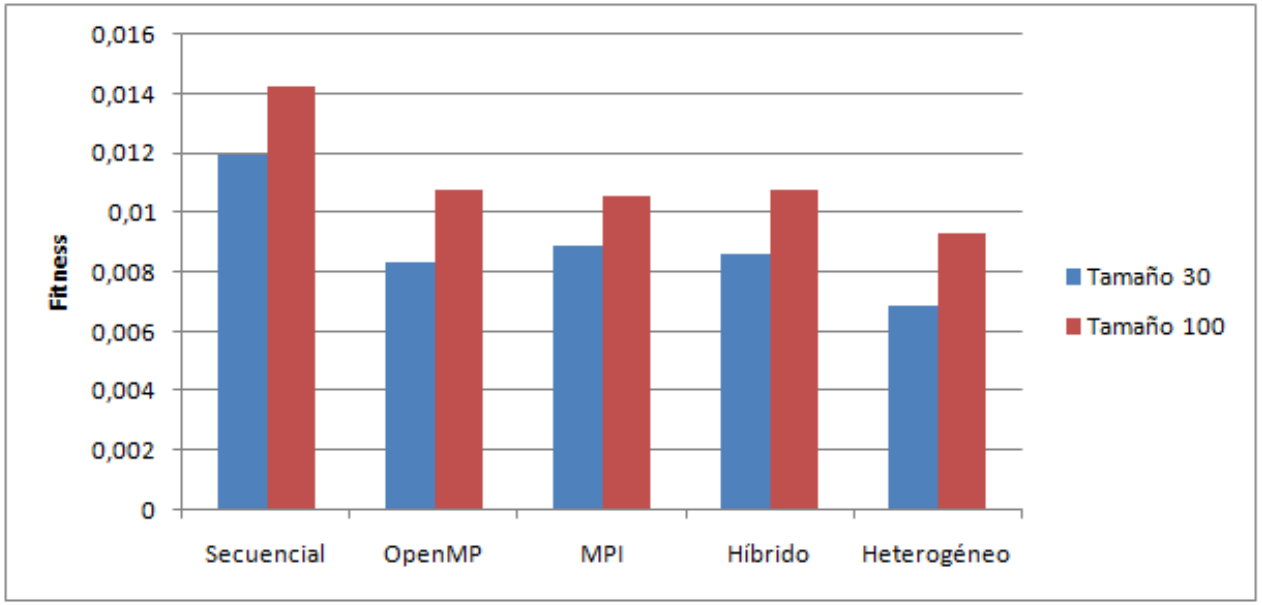


4. Mejoras.

- Explotación de la Heterogeneidad:
  - ❑ Motivación: **versión híbrida + clústers heterogéneos**
  - ❑ Nodos diferentes: **distribución de trabajo**
  - ❑ Buena distribución de trabajo: **- Impacto de las migraciones en el tiempo**

4. Mejoras.

- Resultados:
  - Mejor opción: **Versión Heterogénea**



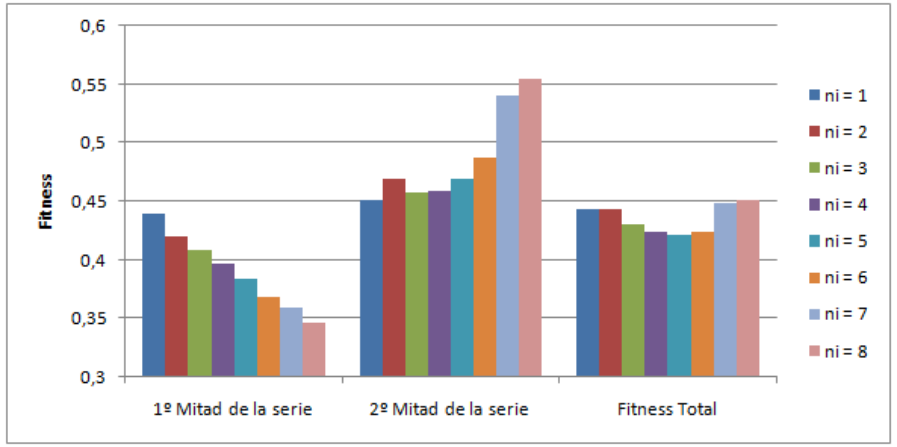
5. Resolución de problemas reales.

5. Resolución de problemas reales.

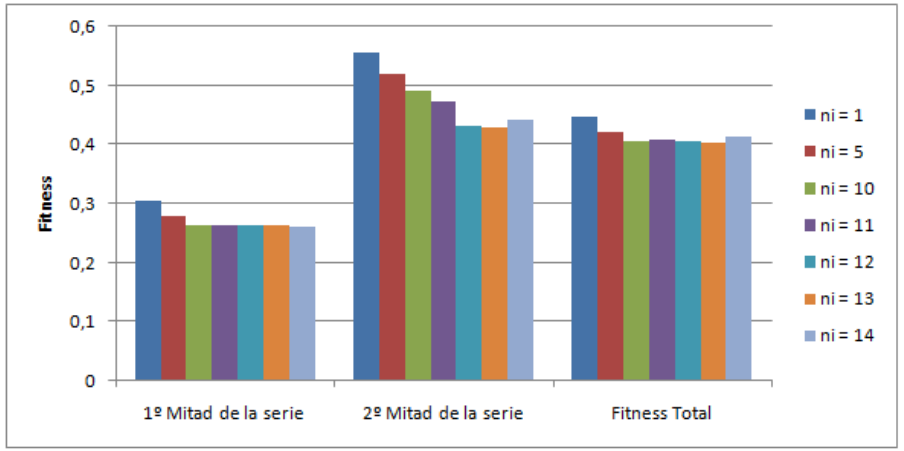
- Cuatro problemas reales: los **tres primeros de economía** y **el último de un registro cerebral**
- Estos problemas solo contienen **parámetros internos**
- Longitud de las **dependencias** temporales: se han obtenido **de forma experimental**

5. Resolución de problemas reales.

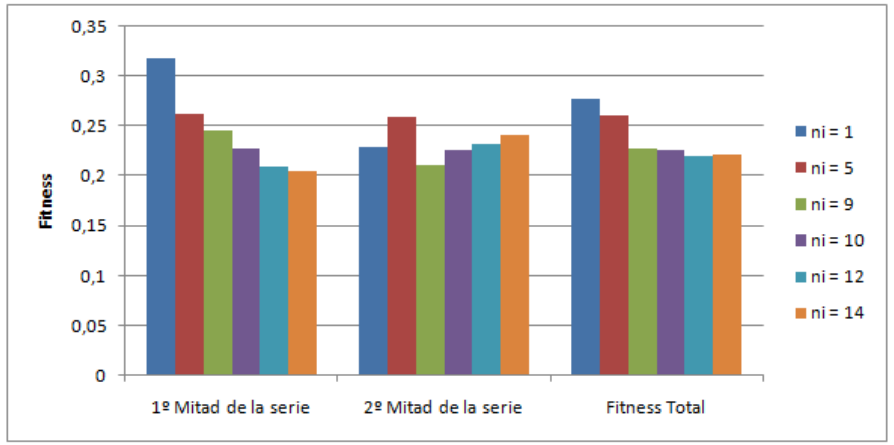
- Problema 1



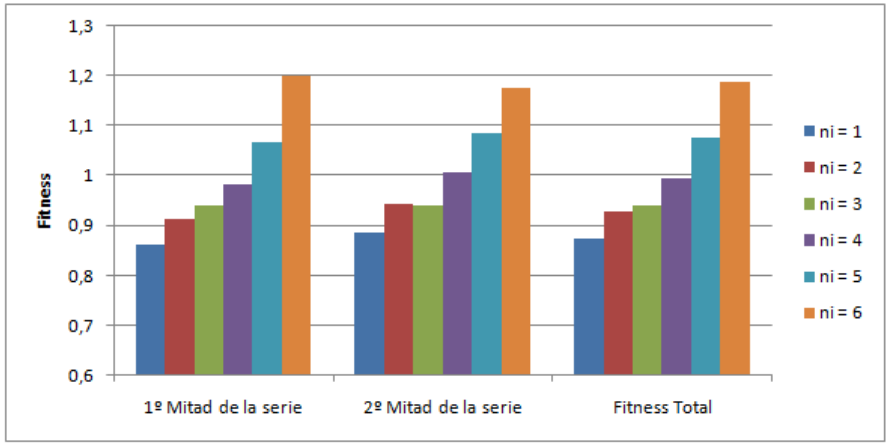
- Problema 2



- Problema 3



- Problema 4



6. Conclusiones y Trabajo Futuro.

6. Conclusiones y Trabajo Futuro.

Conclusiones generales

- El software se puede **aplicar con éxito a problemas reales**
- El **uso del paralelismo** ha sido muy beneficioso

6. Conclusiones y Trabajo Futuro.

Conclusiones de los experimentos

- Utilizar la versión híbrida utilizando todos los nodos disponibles del clúster con una frecuencia de migración alta
- Búsqueda Local y su optimización sin elitismo
- Utilizar la librería OpenBLAS o similar
- Cruces aleatorios y selección por torneo
- Tamaños de población entre 30 y 100
- Probabilidad de cruce de 0.9. Probabilidad de mutación de 0.005

6. Conclusiones y Trabajo Futuro.

Trabajo Futuro

- Desarrollar o utilizar **implementaciones GPU o Xeon Phi**
- **Otras metaheurísticas**
- Analizar con expertos el uso de este software en sus problemas

¿Preguntas?