Algoritmos basados en cúmulos de partículas para el análisis de microarrays de ADN

E. Alba, J. García-Nieto y G. Luque

15 de febrero de 2007

UNIVERSIDAD
DE MÁLAGA

1. Introducción
2. Análisis de microarrays de ADN
3. Algoritmos basados en cúmulos de partículas (PSO)
4. PSO para la reordenación de genes en MAs
   - Esquema de paralelización
5. Experimentos y resultados computacionales
   - Experimentos: resultados (I)
   - Experimentos: resultados (II)
   - Experimentos: resultados (III)
6. Conclusiones y trabajo futuro

Introducción

- Mediante la técnica de microarrays de ADN (MAs) es posible manejar desde cientos hasta decenas de miles de genes

Introducción

- Mediante la técnica de microarrays de ADN (MAs) es posible manejar desde cientos hasta decenas de miles de genes
- Por este motivo, son necesarias técnicas de reducción que permitan agrupar genes con patrones de expresión relacionados

Introducción

- Mediante la técnica de microarrays de ADN (MAs) es posible manejar desde cientos hasta decenas de miles de genes
- Por este motivo, son necesarias técnicas de reducción que permitan agrupar genes con patrones de expresión relacionados
- Para este propósito se vienen utilizando técnicas de clustering como *K-means* o métodos aglomerativos

Introducción

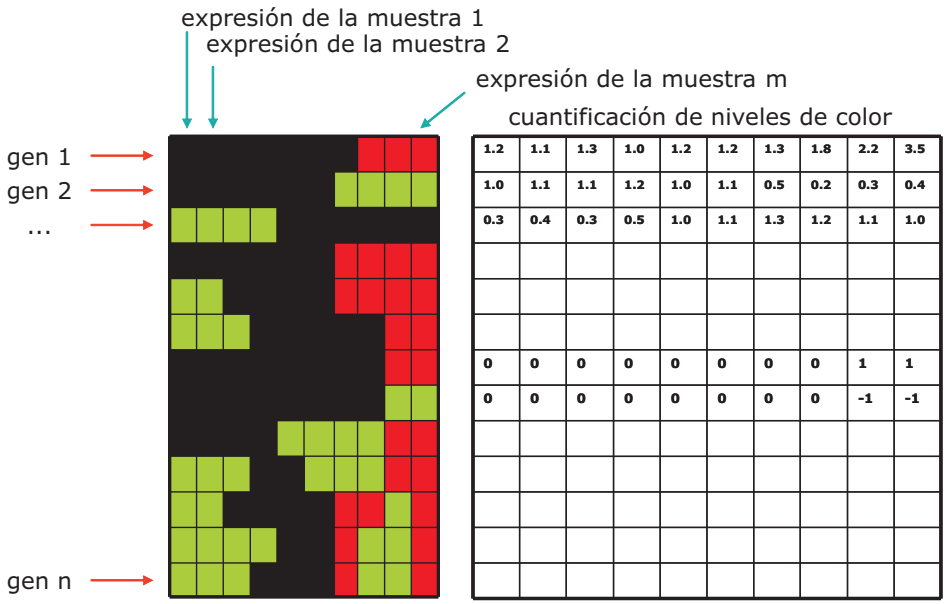
- Mediante la técnica de microarrays de ADN (MAs) es posible manejar desde cientos hasta decenas de miles de genes
- Por este motivo, son necesarias técnicas de reducción que permitan agrupar genes con patrones de expresión relacionados
- Para este propósito se vienen utilizando técnicas de clustering como *K-means* o métodos aglomerativos
- Es posible mejorar más aún la calidad de las soluciones que se obtienen con estos métodos

Introducción

- Mediante la técnica de microarrays de ADN (MAs) es posible manejar desde cientos hasta decenas de miles de genes
- Por este motivo, son necesarias técnicas de reducción que permitan agrupar genes con patrones de expresión relacionados
- Para este propósito se vienen utilizando técnicas de clustering como *K-means* o métodos aglomerativos
- Es posible mejorar más aún la calidad de las soluciones que se obtienen con estos métodos
- En este trabajo se propone la utilización de algoritmos basados en cúmulos de partículas (PSO) para realizar una reordenación de los genes muestreados en un MA

Microarrays

- Un microarray o "chip" de ADN es una plantilla de vidrio (o un sustrado sólido) en la cual se disponen de manera sistemática cientos o miles de muestras de moléculas de ADN
- Tras un proceso de hibridación y escaneo de imagen



- El objetivo es encontrar un orden óptimo de los genes en el MA

Optimización basada en cúmulos de partículas (PSO)

Partícula

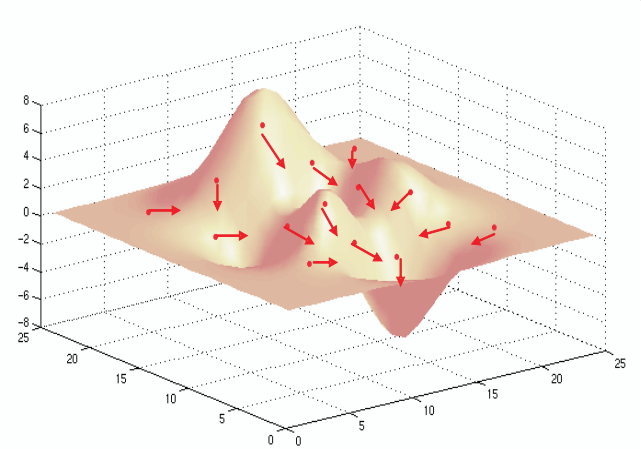
- Vector Solución (posición) (0.2, -1.4, 3.5)
- Vector Velocidad (dirección) (1.0, 10.3, 7.2)

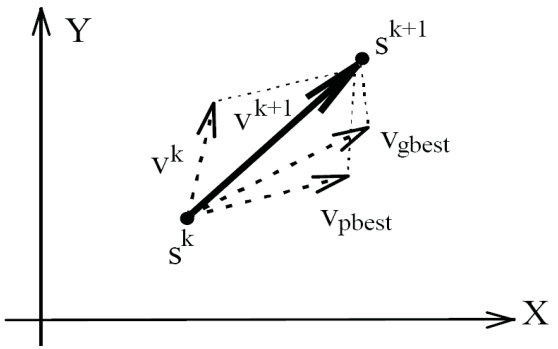
Actualización

$v_i \leftarrow \omega \cdot v_i + \varphi_1 \cdot rand_1 \cdot (pBest_i - x_i) + \varphi_2 \cdot rand_2 \cdot (g_i - x_i)$

$x_i \leftarrow x_i + v_i$

Gráficamente





PSO para la reordenación de genes en MAs

Representación

- Microarray $M = \{g_{ij}\}_{j=1\ldots m}^{i=1\ldots n}$, donde n es el número de genes y m es el número de muestras por gen
- Solución (Posición): $x = (2, 4, 3, 6, 1, 5)$
- Velocidad: $v = [(3 \rightarrow 1), (6 \rightarrow 6), (1 \rightarrow 5)]$
- Movimiento: $x = (2, 4, 3, 6, 1, 5)$

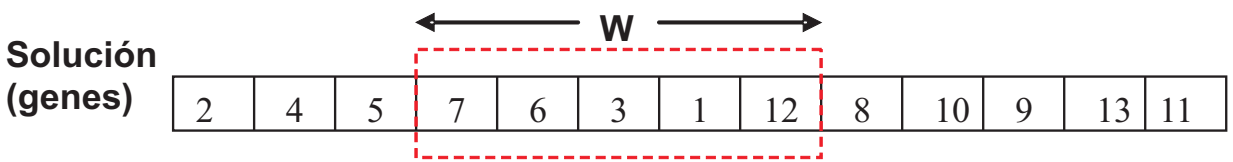
PSO para la reordenación de genes en MAs

Representación

- Microarray $M = \{g_{ij}\}_{j=1\ldots m}^{i=1\ldots n}$, donde n es el número de genes y m es el número de muestras por gen
- Solución (Posición): $x = (2, 4, 3, 6, 1, 5)$
- Velocidad: $v = [(3 \rightarrow 1), (6 \rightarrow 6), (1 \rightarrow 5)]$
- Movimiento: $x = (2, 4, 1, 6, 3, 5)$

Función de evaluación: distancia euclídea con ventana deslizante

- $DT(\pi) = \sum_{l=1}^{n} \sum_{i=max(l-s_w,1)}^{min(l+s_w,n)} w(i,l) D[\pi_l, \pi_i]$



Pseudocódigo

PSO para permutaciones de enteros

1: $S \leftarrow InicializarCumulo()$
2: **while** no se alcance la condición de parada **do**
3: **for** $i = 1$ to $size(S)$ **do**
4: evaluar cada partícula x_i del cúmulo S
5: **if** $fitness(x_i)$ es mejor que $fitness(pBest_i)$ **then**
6: $pBest_i \leftarrow x_i$; $fitness(pBest_i) \leftarrow fitness(x_i)$
7: **end if**
8: **if** $fitness(pBest_i)$ es mejor que $fitness(g_i)$ **then**
9: $g_i \leftarrow pBest_i$; $fitness(g_i) \leftarrow fitness(pBest_i)$
10: **end if**
11: **end for**
12: **for** $i = 1$ to $size(S)$ **do**
13: $v_i \leftarrow v_i \circ \varphi_1 \otimes (pBest_i \ominus x_i) \circ \varphi_2 \otimes (g_i \ominus x_i)$
14: $x_i \leftarrow x_i \oplus v_i$
15: **end for**
16: **end while**
17: **Salida:** la mejor solución encontrada

Operadores

- **Diferencia de posiciones ($\ominus$)**:
  genera una nueva velocidad (vector de pares $(i \longrightarrow j)$)

Operadores

- **Diferencia de posiciones ($\ominus$)**:
  genera una nueva velocidad (vector de pares $(i \longrightarrow j)$)
- **Suma de velocidades ($\circ$)**:
  genera una nueva velocidad (trasposición de velocidades)

Operadores

- **Diferencia de posiciones ($\ominus$)**:
  genera una nueva velocidad (vector de pares $(i \longrightarrow j)$)
- **Suma de velocidades ($\circ$)**:
  genera una nueva velocidad (trasposición de velocidades)
- **Producto coeficiente velocidad ($\otimes$)**:
  genera una nueva velocidad (reduce pares)

Operadores

- **Diferencia de posiciones ($\ominus$)**:
  genera una nueva velocidad (vector de pares $(i \longrightarrow j)$)
- **Suma de velocidades ($\circ$)**:
  genera una nueva velocidad (trasposición de velocidades)
- **Producto coeficiente velocidad ($\otimes$)**:
  genera una nueva velocidad (reduce pares)
- **Suma de posición con velocidad ($\oplus$)**:
  genera una nueva posición (movimiento)

Operadores

- **Diferencia de posiciones ($\ominus$)**:
  genera una nueva velocidad (vector de pares $(i \longrightarrow j)$)
- **Suma de velocidades ($\circ$)**:
  genera una nueva velocidad (trasposición de velocidades)
- **Producto coeficiente velocidad ($\otimes$)**:
  genera una nueva velocidad (reduce pares)
- **Suma de posición con velocidad ($\oplus$)**:
  genera una nueva posición (movimiento)

Búsqueda Local

Sucesivas operaciones de intercambio (swap de la posición) sobre la mejor partícula del cúmulo.

Operadores

- **Diferencia de posiciones ($\ominus$)**:
  genera una nueva velocidad (vector de pares $(i \longrightarrow j)$)
- **Suma de velocidades ($\circ$)**:
  genera una nueva velocidad (trasposición de velocidades)
- **Producto coeficiente velocidad ($\otimes$)**:
  genera una nueva velocidad (reduce pares)
- **Suma de posición con velocidad ($\oplus$)**:
  genera una nueva posición (movimiento)

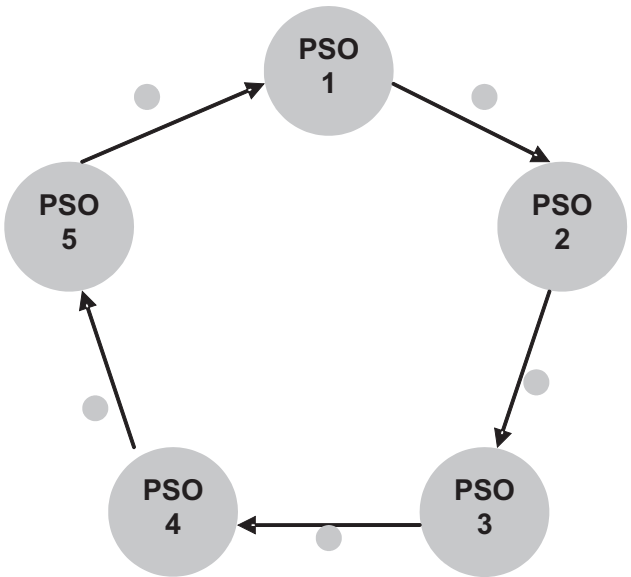
Búsqueda Local

Sucesivas operaciones de intercambio (swap de la posición) sobre la mejor partícula del cúmulo.

- Estos operadores "mantienen las permutaciones"

Esquema de paralelización

- Arquitectura de esqueletos de código de la biblioteca MALLBA (versiones paralelas LAN/WAN) disponible en la URL `http://neo.lcc.uma.es/mallba/easy-mallba/index.html`
- Modelo descentralizado grano grueso
- Topología de anillo unidireccional



- Intercambio de soluciones: versión síncrona *vs.* versión asíncrona

Experimentos

Instancias

- **HERPES**: *herpes de Kaposi*, 106 genes (21 muestra por gen)
- **FIBROBLAST**: *fibroblasto*, 517 genes (19 muestras por gen)
- **DIAUXIC**: *salto diauxico*, 210 genes (7 muestras por gen)

Parámetros

Parámetro	Secuencial	Paralelo
Tamaño del Cúmulo	100	20
Tamaño del Vecindario	8	8
Límites de Inercia	(-10,10)	(-10,10)
Velocidad (Mín,Máx)	(0.4,1.4)	(0.4,1.4)
Frecuencia de Migración	-	8
Número de Procesos	1	5

- 30 ejecuciones independientes para conseguir confianza estadística
- Cada ejecución realiza 500 iteraciones + búsqueda local en cada iteración

Experimentos: resultados I

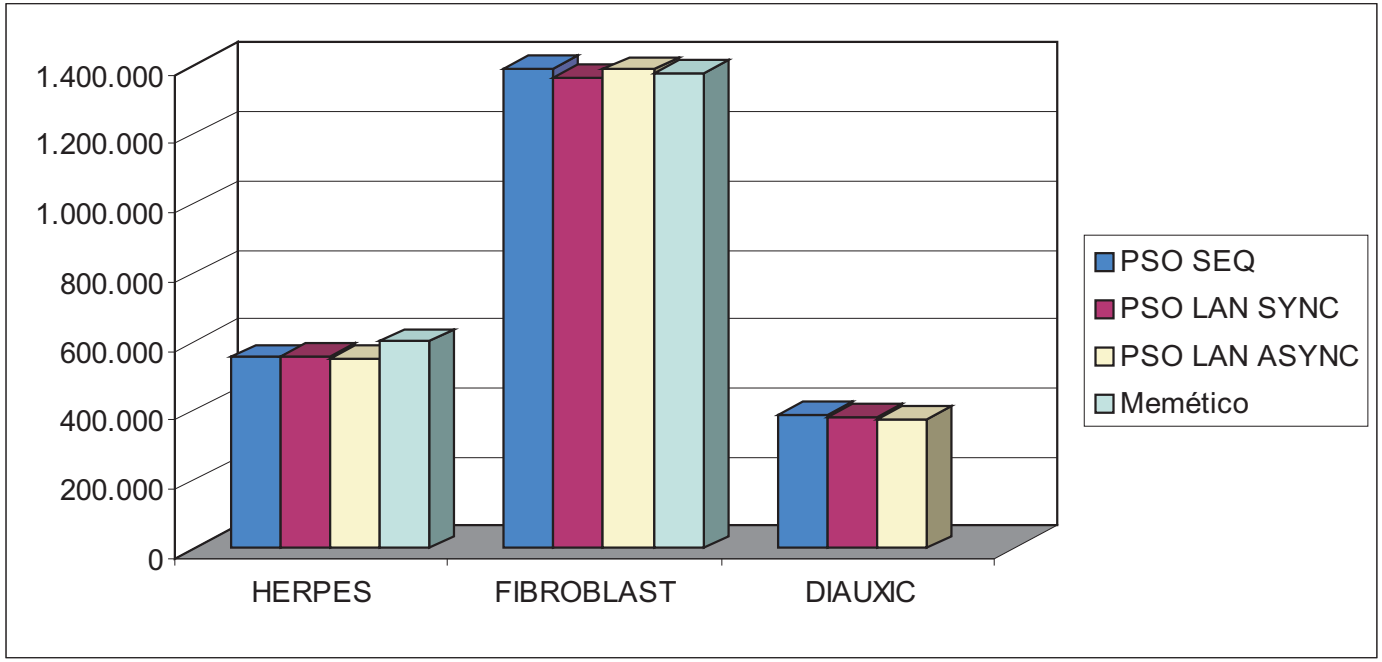
Distancias

Instancia	Algoritmo	M	MV	PE	E	T
HERPES	PSO SEQ	500.478	554.202	0.00000 %	39800	707.54 *s*
	PSO LAN SYNC	497.459	559.080	0.00000 %	50000	1586.71 *s*
	PSO LAN ASYNC	496.263	552.345	0.00000 %	41900	375.03 *s*
	Memético	-	598.798	-	-	-
FIBROBLAST	PSO SEQ	1359.680	1391.060	0.01035 %	50000	21436.40 *s*
	PSO LAN SYNC	1359.680	1362.670	0.00000 %	47000	16652.60 *s*
	PSO LAN ASYNC	1362.980	1386.700	0.00718 %	50000	26972.40 *s*
	Memético	-	1376.804	-	-	-
DIAUXIC	PSO SEQ	355.047	388.323	0.00000 %	49700	797.55 *s*
	PSO LAN SYNC	349.232	381.630	0.00000 %	49900	909.14 *s*
	PSO LAN ASYNC	345.796	374.734	0.00000 %	50000	2206.16 *s*
	Memético	-	-	-	-	-

El PSO en su versión paralela asíncrona es la que ofrece mejores resultados

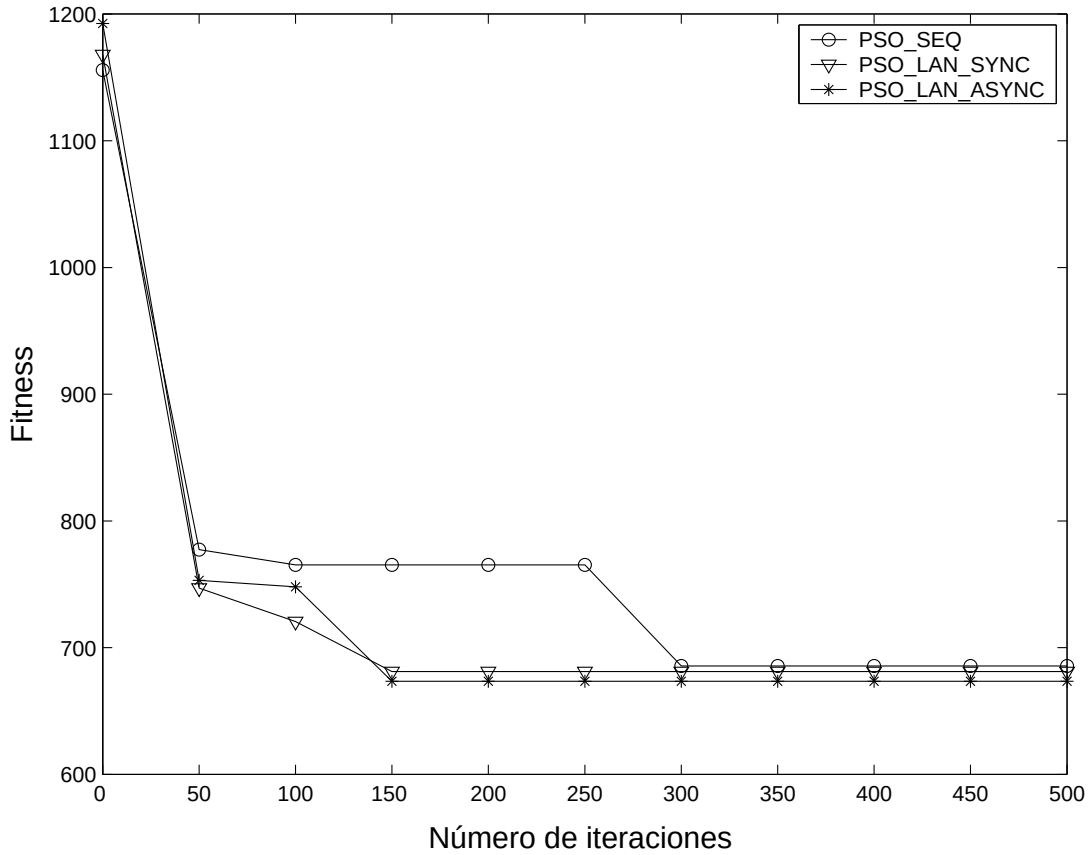
Gráficamente

Distancias



Gráficamente

Evolución



Experimentos: resultados II

Speedup/Eficiencia

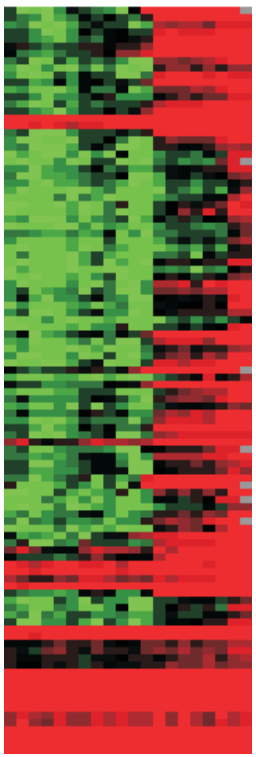
Instancia	$Speedup_{s1,5}$	$Speedup_{a1,5}$	$Eficiencia_{s1,5}$	$Eficiencia_{a1,5}$	**Final**
HERPES	2.299	3.952	46 %	70 %	630
FIBROBLAST	1.444	1.992	28 %	40 %	1600
DIAUXIC	2.856	2.621	57 %	52 %	500

Eficiencia alrededor del 50 %

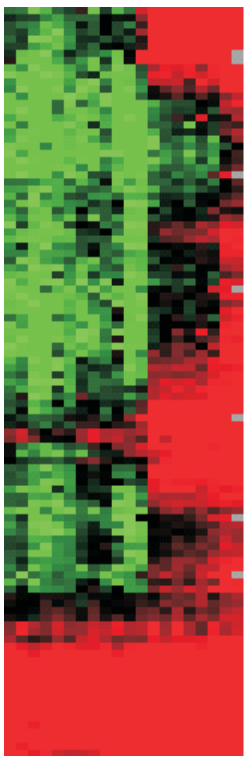
La versión paralela asíncrona reporta más eficiencia

Experimentos: microarrays iniciales vs. óptimamente ordenados

HERPES



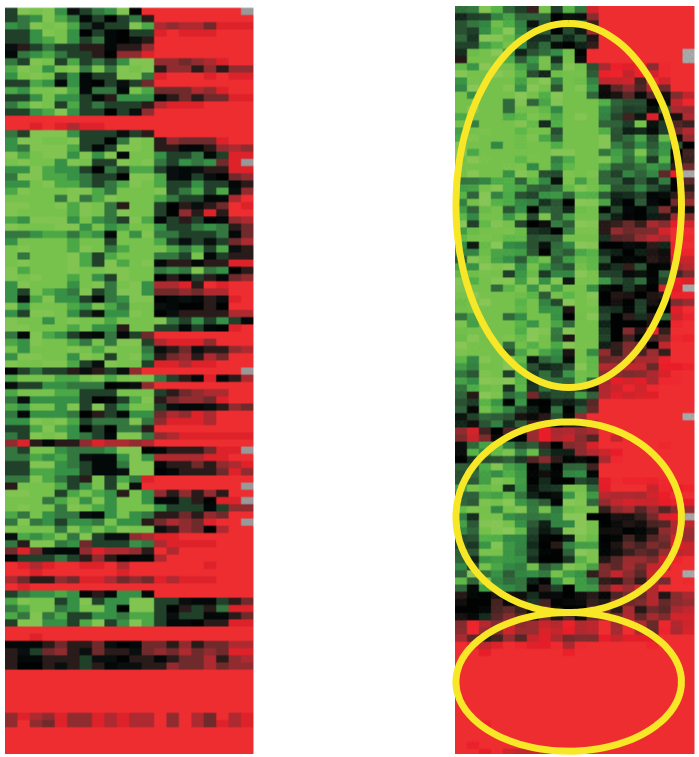
Inicial



Resultado

Experimentos: microarrays iniciales vs. óptimamente ordenados

HERPES



Inicial **Resultado**

Conclusiones y trabajo futuro

Conclusiones

- Hemos aplicado PSO para la reordenación eficiente de genes en microarrays de ADN
- Hemos utilizado tres instancias reales de microarrays
- Además, evaluamos el PSO para permutaciones de enteros tanto en versión secuencial como paralela
- Se mejoran las soluciones obtenidas por otros algortimos que encontramos en la literatura

Trabajo futuro

- Estudiar nuevas versiones de PSO para permutaciones de enteros
- Evaluar nuevas instancias de microarray y realizar nuevas comparaciones
- Incorporar funcionalidades de clustering jerárquico

Preguntas

¡¡¡Gracias por su Atención!!!