



Tema 2

Modelos de programación paralela adaptados a la arquitectura

Nicolás Calvo Cruz

Dpto. de Arquitectura y Tecnología de los Computadores

@ncalvocruz

ncalvocruz@ugr.es

Objetivos

- Aprender a **encontrar** puntos para paralelizar un algoritmo
- Aprender a **descomponer** un algoritmo en tareas
- Aprender a identificar y respetar **dependencias** entre tareas
- **Agrupar** las tareas que se pueden realizar en paralelo
- Aplicar revisiones de **diseño**



Motivación

- 1 Abordar **problemas** cada vez más grandes.
- 2 Obtener (mejores) soluciones en un **tiempo razonable**.
- 3 Ilustrar diferentes **enfoques** de la bibliografía para **parallelizar** algoritmos tradicionales.

Índice

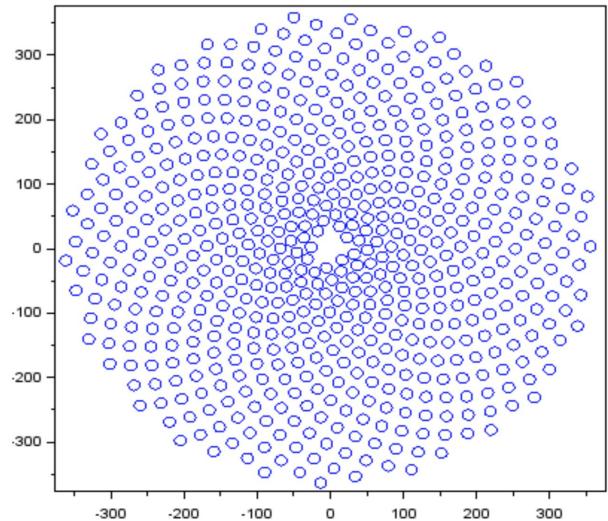
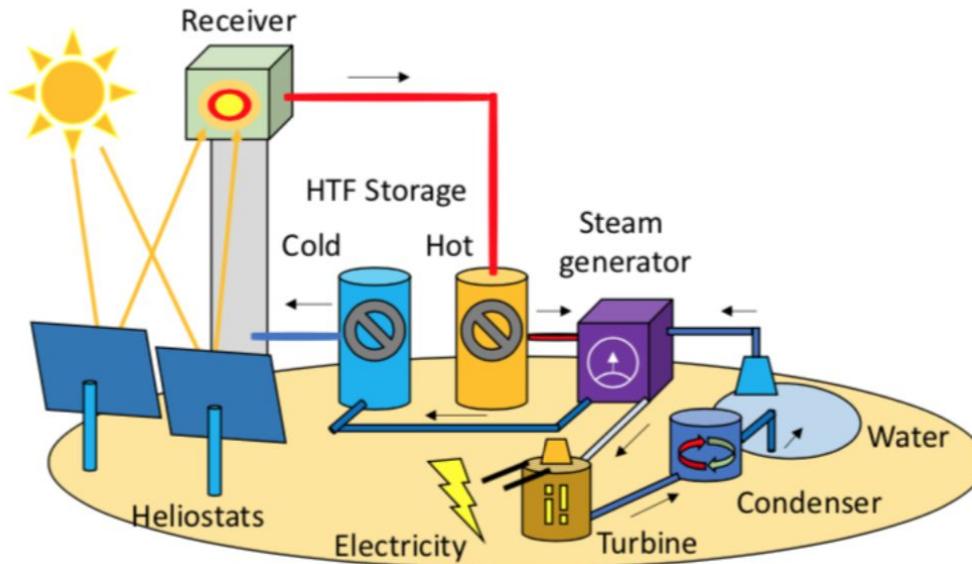
1. Introducción
2. ¿Cómo encontrar concurrencia?
 - a. Encontrar tareas
 - b. Agrupar tareas
 - c. Buscar patrones de comunicación
3. Patrones de los principales algoritmos paralelos
4. Estructuras de algoritmos más comunes
5. Qué hacer con las estructuras de datos
6. Ejemplos prácticos de algoritmos paralelos

6. Ejemplos prácticos de algoritmos paralelos

6. Ejemplos prácticos: Simular un campo de heliostatos

Simular un campo de heliostatos (I)

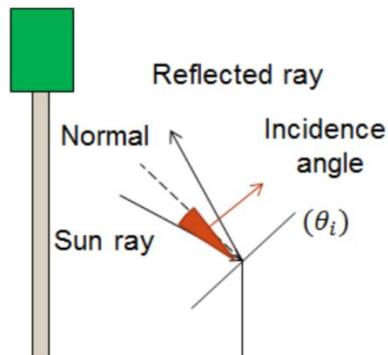
- 50% de coste
- 40% de pérdida energética



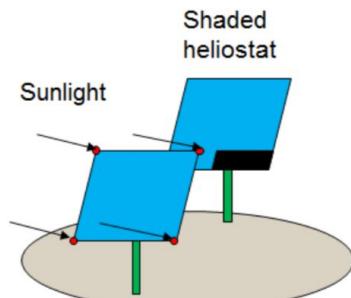
6. Ejemplos prácticos: Simular un campo de helióstatos

Simular un campo de helióstatos (II)

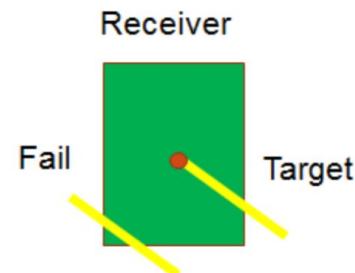
$$\eta = \eta_{cos} \cdot \eta_{sb} \cdot \eta_{itc} \cdot \eta_{aa} \cdot \eta_{ref}$$



Factor coseno



Bloqueo y sombreado



Factor de intercepción

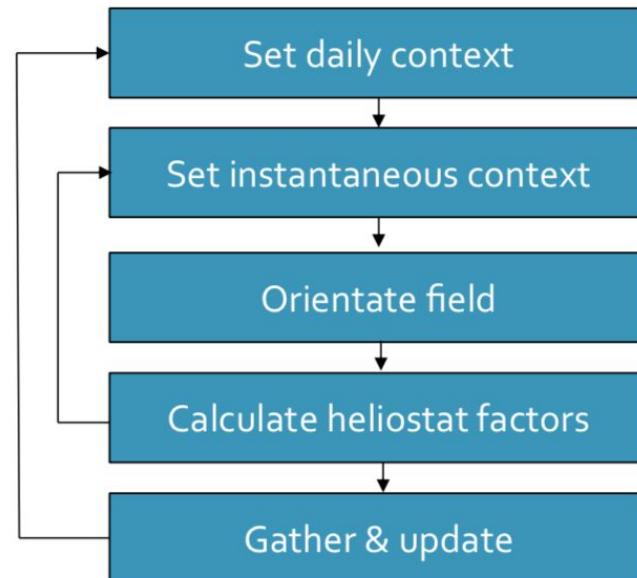


Atenuación atmosférica

6. Ejemplos prácticos: Simular un campo de helióstatos

Simular un campo de helióstatos (III)

El proceso de simulación se puede resumir así:



6. Ejemplos prácticos: Simular un campo de helióstatos

Simular un campo de helióstatos (IV)

Este proceso se puede implementar con éxito bajo un modelo **fork-join** en el que un conjunto de hilos se encarga de realizar el cálculo **supeditado** a donde se necesite (optimizador?).

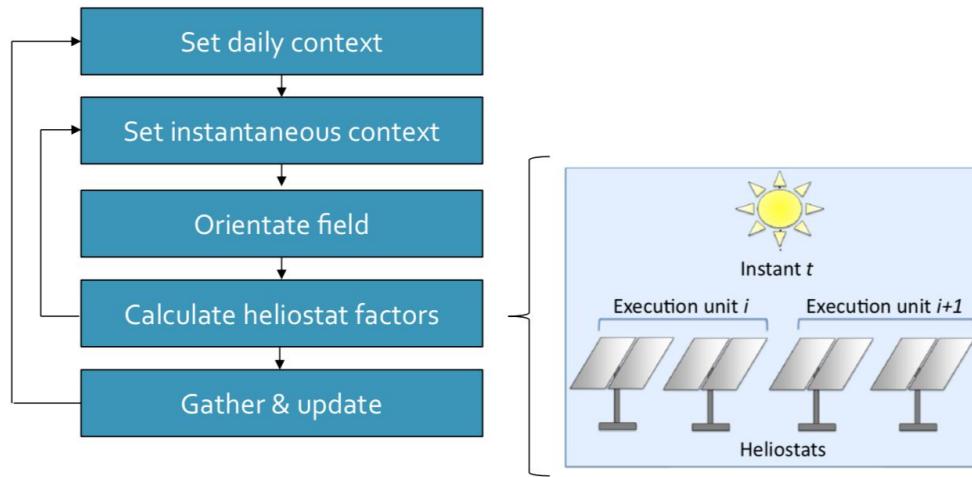
Encontramos 3 “entidades”:

- Helióstatos
- Instantes
- Conjuntos de instantes (p.ej. días)

6. Ejemplos prácticos: Simular un campo de helióstatos

Simular un campo de helióstatos (V)

División por helióstatos:

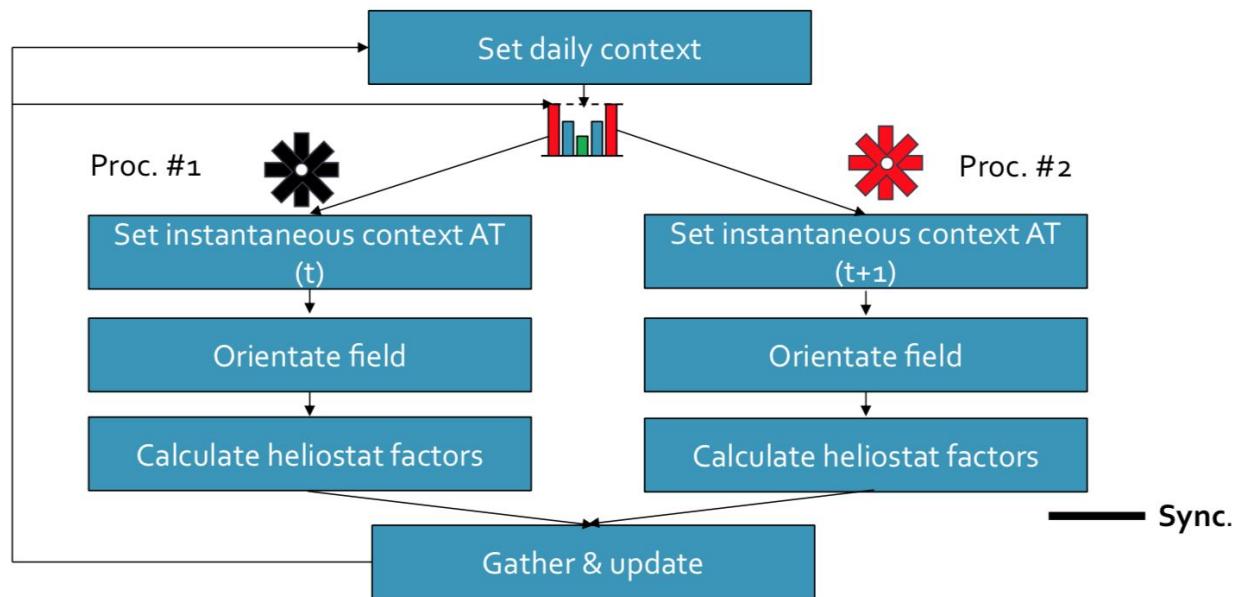


Más simple...
y más overhead

6. Ejemplos prácticos: Simular un campo de helióstatos

Simular un campo de helióstatos (VI)

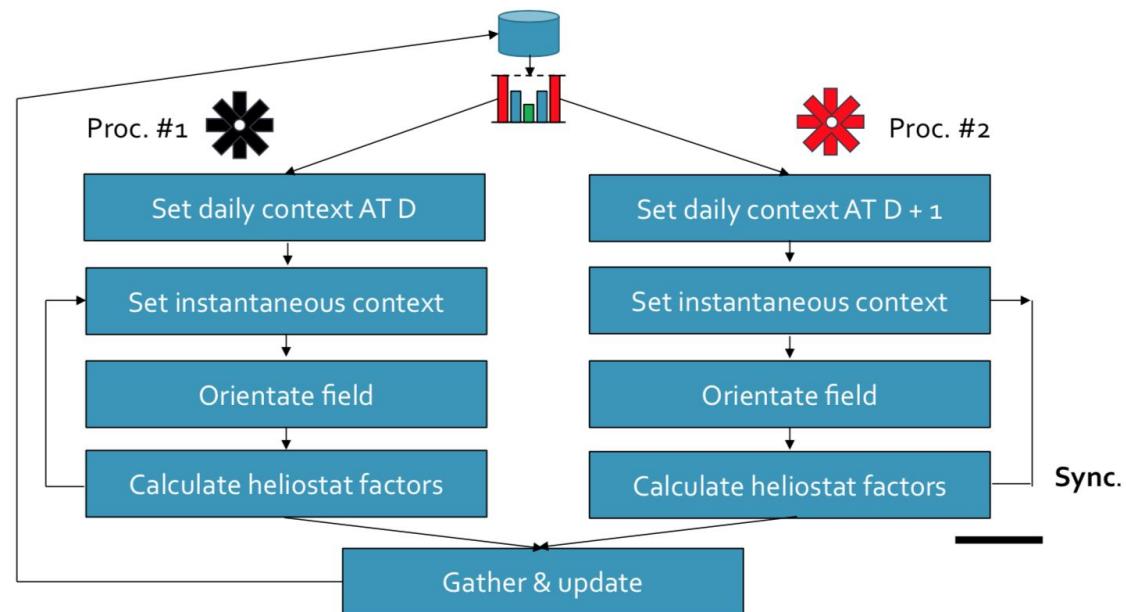
División por instantes:



6. Ejemplos prácticos: Simular un campo de helióstatos

Simular un campo de helióstatos (VII)

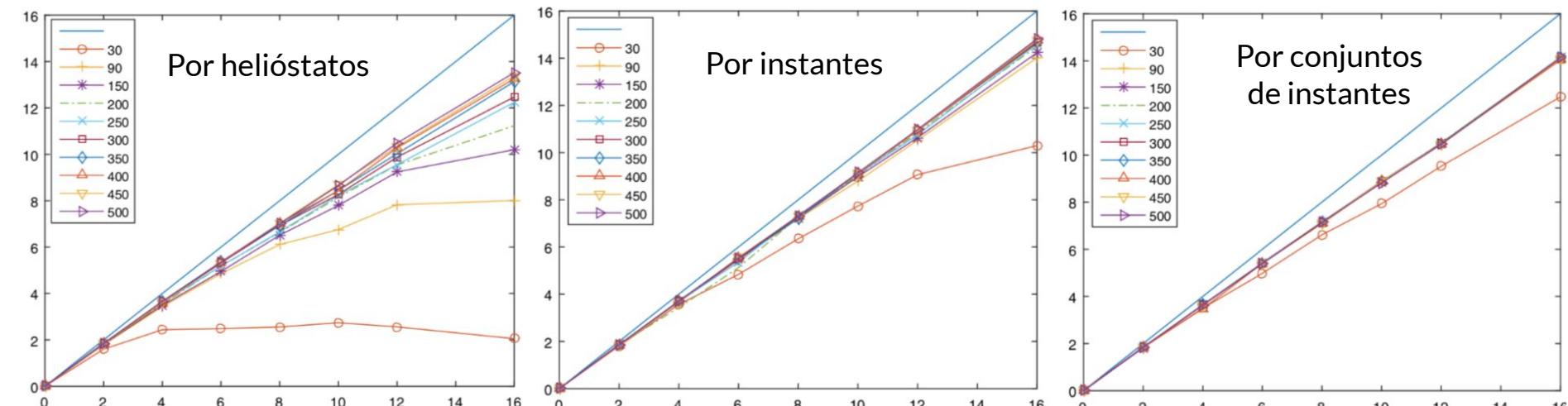
División por conjuntos de instantes:



Más complejo
y más tamaño de grano

6. Ejemplos prácticos: Simular un campo de helióstatos

Simular un campo de helióstatos (VIII)



6. Ejemplos prácticos: Optimizadores paralelos

Optimización continua (I)

Búsqueda de los extremos de una función continua. Esta función f , conocida como “función objetivo”, modela algún aspecto de la realidad (coste de distribución, eficiencia de producción...) y depende de una serie (N) de variables:

$$f : \mathbb{R}^N \rightarrow \mathbb{R}$$

Si f es un coste, nos interesan sus mínimos (**minimización**). Si es una eficiencia, nos interesan sus máximos (**maximización**)... Realmente cambia poco, pues minimizar f es igual que maximizar $-f$.

(Este tipo de problemas también se denominan **Programación Matemática**)

6. Ejemplos prácticos: Optimizadores paralelos

Optimización continua (II)

Podemos tener además una serie de **restricciones** que cumplir, y que nos acotan las soluciones válidas.

Las más restricciones más simples, conocidas como “de caja”, limita el rango de cada variable entre un límite inferior, L, y uno superior, U:

$$f : \mathbb{R}^N \rightarrow \mathbb{R}$$

queda limitada al espacio vectorial definido por $[L_1, U_1] \times \dots \times [L_N, U_N]$

6. Ejemplos prácticos: Optimizadores paralelos

Optimización continua (III)

Formalmente, podríamos definir así un problema de optimización (minimización) continua con restricciones de caja:

$$\underset{x}{\text{minimize}} \quad f(x)$$

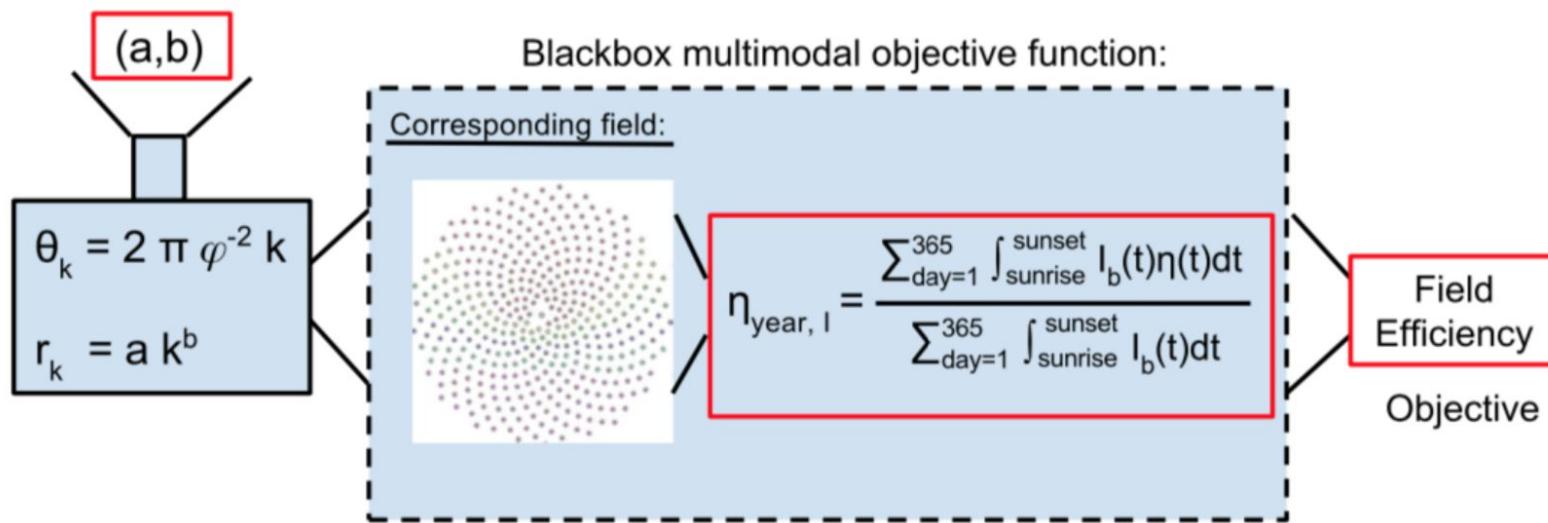
$$\text{subject to} \quad L_i \leq x_i \leq U_i, \quad i = 1, \dots, N$$

Si f tiene una forma analítica concreta y propiedades como linealidad y convexidad, estos problemas se pueden resolver fácilmente por métodos analíticos exactos. Sin embargo, esto no siempre ocurre: a veces sólo podemos evaluar (black-box), y no siempre es rápido.

6. Ejemplos prácticos: Optimizadores paralelos

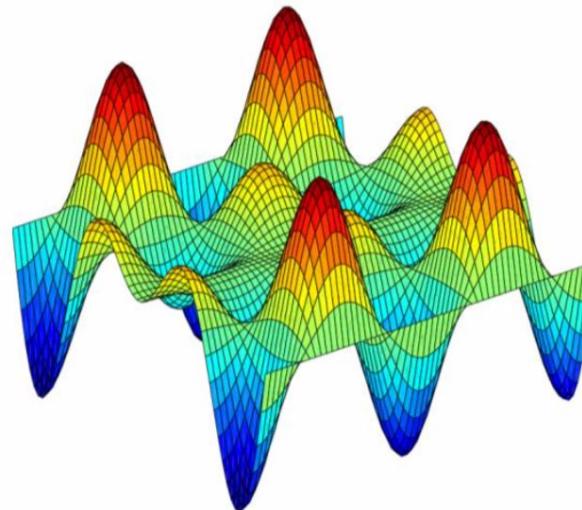
Optimización continua (IV)

Under optimization



6. Ejemplos prácticos: Optimizadores paralelos

Optimización continua (V)



Sin propiedades matemáticas que explotar, se aplican **heurísticas y meta-heurísticas**, que son **estrategias de resolución que** se sabe que **obtienen buenos resultados**, aunque no siempre se pueda demostrar su optimalidad.

6. Ejemplos prácticos: Optimizadores paralelos

Optimización continua (VI)

- Muchos de estos métodos se apoyan fuertemente en el uso de aleatoriedad (métodos estocásticos / Monte-Carlo) y en maximizar la capacidad de evaluación de soluciones por unidad de tiempo:
 - El **HPC** puede jugar un papel clave en este sentido, encajando muy bien con el modelo de la Ley de Gustafson (tiempo fijo).
- En este contexto destacan los métodos basados en poblaciones, como algoritmos genéticos (**evolutivos**) y los de optimización de conjunto de partículas (**inteligencia colectiva**).

6. Ejemplos prácticos: Optimizadores paralelos

Búsqueda Aleatoria Pura

- Sencilla
- **Muy paralelizable**
- Eficaz para problemas de pocas dimensiones

(Versión modificada para registrar múltiples puntos)

```
function [solutions, cycles] = ParExplorer(func, nVars, mode, killSwitchName)
    solutions = []; % As many rows as solutions
    cycles = 0;

    pool = gcp;
    numThreads = pool.NumWorkers; disp(['Using: ', num2str(numThreads), ' threads']);

    buffer = zeros(numThreads, nVars+1); % Where to store partial results

    while ~exist(killSwitchName, 'file') % Iterate indefinitely
        parfor i = 1:numThreads
            point = rand(1, nVars);
            val = func(point);
            buffer(i,:) = [point, val];
        end

        solutions = [solutions; buffer]; %#ok<AGROW>
        cycles = cycles + numThreads;
    end

    solutions = unique(solutions, 'rows');
    [~, indices] = sort(solutions(:, end), mode);
    solutions = solutions(indices, :); % Sort the output

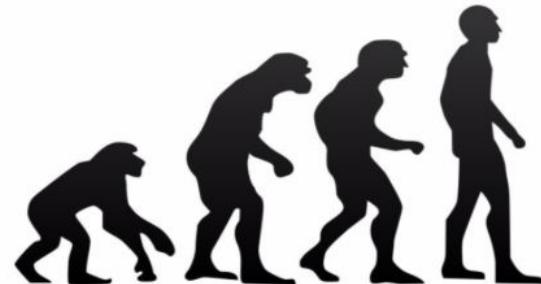
    delete(pool);
end
```

6. Ejemplos prácticos: Optimizadores paralelos

Algoritmos evolutivos (I)

Gestionan un conjunto (población) de soluciones candidatas (individuos) y simulan su interacción y evolución siguiendo el modelo de Darwin.

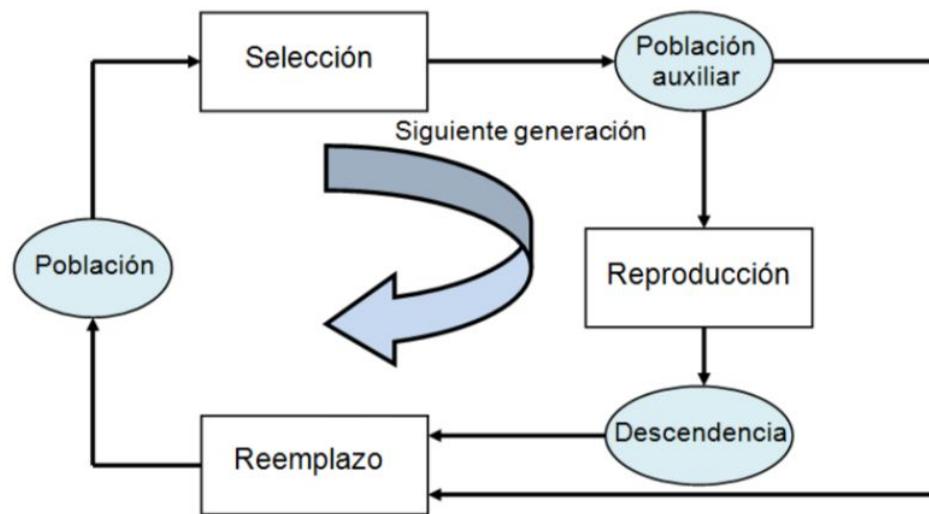
Punto	Valor
(0.1, ..., 0.75)	19
(...)	(...)
(0.9, ..., 1.0)	10



6. Ejemplos prácticos: Optimizadores paralelos

Algoritmos evolutivos (II)

Ciclo básico de un algoritmo evolutivo (aunque realmente son muy modificables):

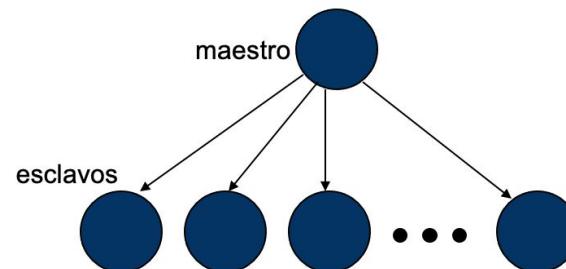


6. Ejemplos prácticos: Optimizadores paralelos

Algoritmos evolutivos (III)

El método más simple de parallelizar un algoritmo evolutivo es la “**parallelización global**”:

- Sólo hay una población, bajo responsabilidad final del hilo/proceso principal, pero la evaluación de individuos se paralleliza explícitamente, lo que sigue un **modelo maestro-esclavo**:



6. Ejemplos prácticos: Optimizadores paralelos

Algoritmos evolutivos (IV)

El método más simple de paralelizar un algoritmo evolutivo es la “**paralelización global**”:

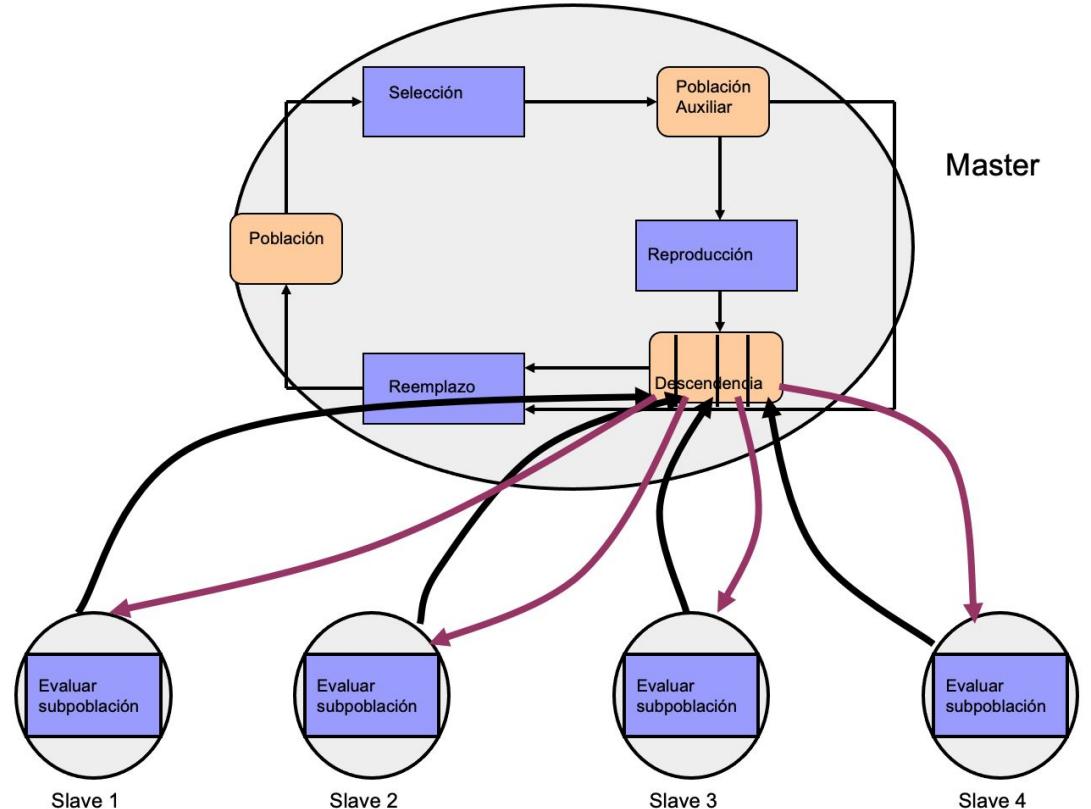
- En memoria distribuida: La comunicación sólo ocurre cuando el procesador **recibe** los individuos a evaluar y cuando **devuelve** los resultados.
- En memoria compartida: No hay comunicación explícita, pero sí necesidad de **puntos de sincronismo** (barreras y mutex).

6. Ejemplos prácticos: Optimizadores paralelos

Algoritmos evolutivos (V)

- En memoria distribuida:
 - Más escalabilidad

Grano Grueso:
Menor ratio trabajo / comunicación



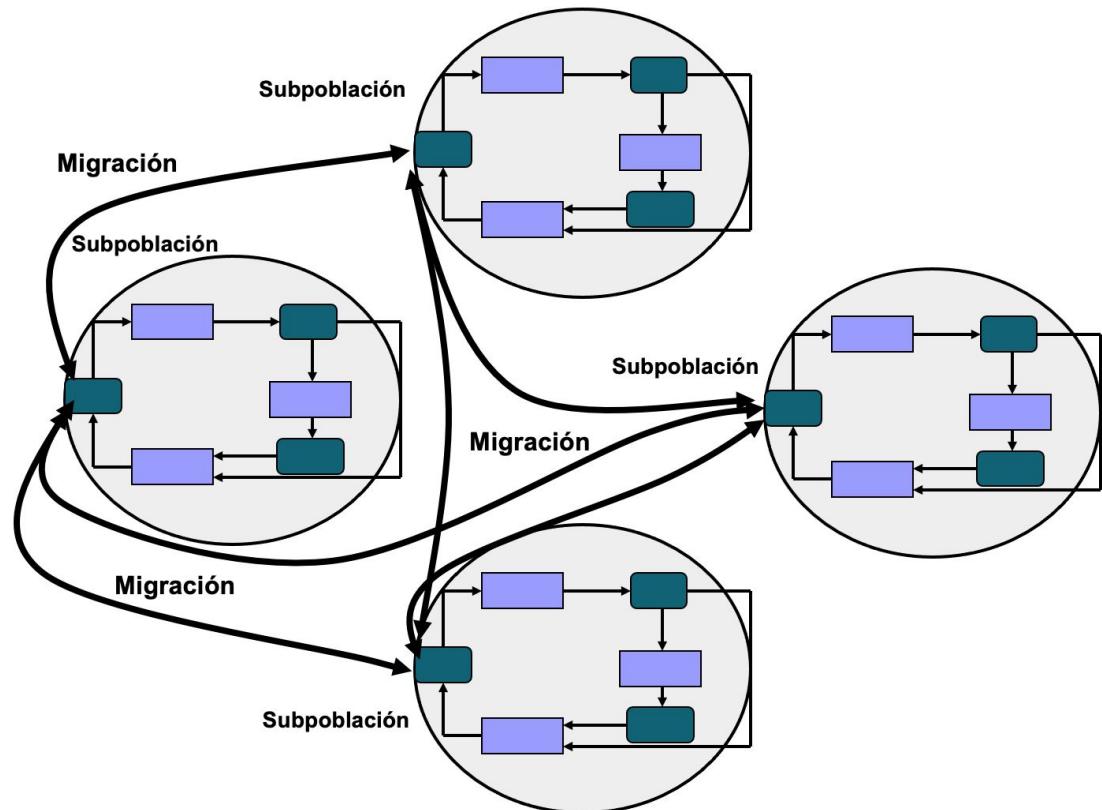
6. Ejemplos prácticos: Optimizadores paralelos

Algoritmos evolutivos (VI)

- En memoria distribuida:
 - Más escalabilidad

Grano Grueso:

Menos interacción... y
aún más escalabilidad
(para el algoritmo, no la
máquina)



6. Ejemplos prácticos: Optimizadores paralelos

Algoritmos evolutivos (VII)

- En memoria compartida:
 - Menos overhead



- 1.- Create pool of **threads**
- 2.- Create **population** of solutions (**Knowledge?**)
- 3.- For $i = 1$ to **cycles**
- 4.- **Select** progenitors
- 5.- **Generate** descendants
- 6.- **Mutate** descendants
- 7.- **Replace** individuals
- 8.- **Return** the best solution

6. Ejemplos prácticos: Optimizadores paralelos

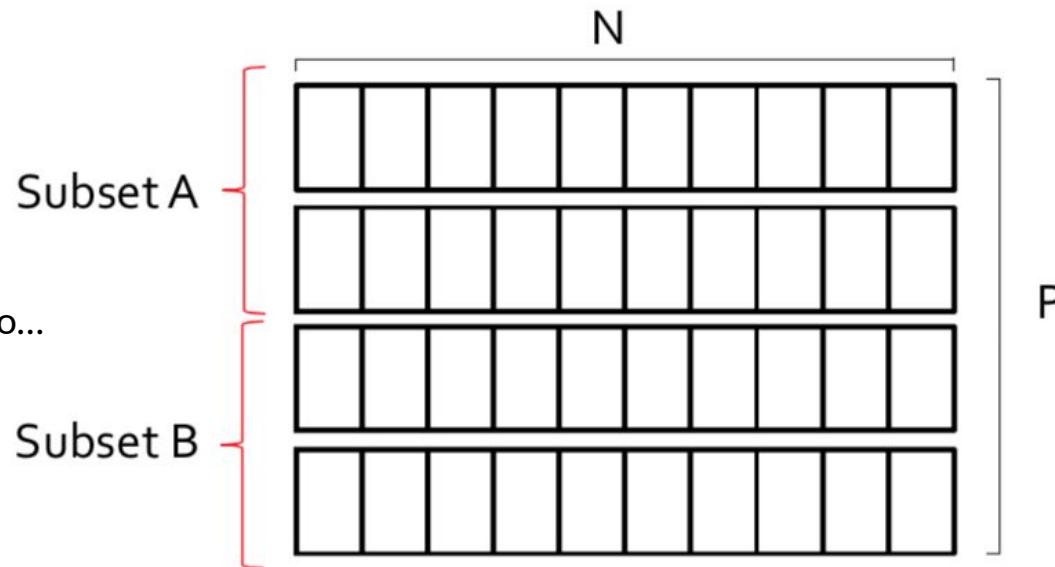
Algoritmos evolutivos (VIII)

- En memoria compartida:

- Menos overhead

Balanceo de carga estático...

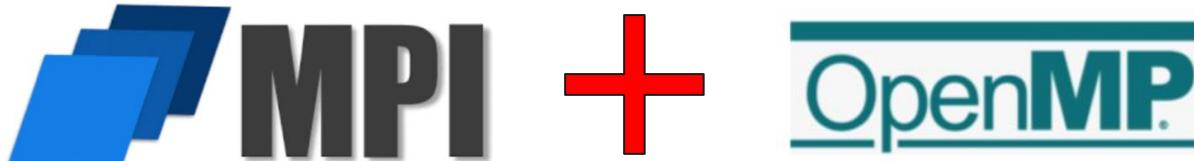
O dinámico:



6. Ejemplos prácticos: Optimizadores paralelos

Algoritmos evolutivos (IX)

- También podemos combinar ambas estrategias:
 - Memoria Distribuida (gestión principal) & Memoria Compartida (auxiliar)



6. Ejemplos prácticos: Optimizadores paralelos

Algoritmo Genético (I)

- Veamos un ejemplo de algoritmo genético en memoria compartida:

```
T* population = 0; T* new_population = 0; T* descendants = 0;//NULL  
this->getMemory(popSize, numPairs, &population, &new_population, &descendants);  
T best_solution;//DEFAULT->power = -inf <minimum possible>  
  
if(threads>numPairs)  
    threads = numPairs;//IT WOULD NOT MAKE SENSE TO LAUNCH MORE THREADS THAN PAIRS TO BE PROCESSED  
MersenneTwister RNG(seed);
```

6. Ejemplos prácticos: Optimizadores paralelos

Algoritmo Genético (II) (Balanceo de carga estático)

```
#pragma omp parallel num_threads(threads)//Every variable created inside the parallel region is thread-local
{
    T local_result;//DEFAULT->power = -inf <minimum possible> To update it as a thread without critical

    uint32_t mySeed = 0;//Where to save the seed for my new local RNG
    #pragma omp critical(RAND) //rand() is not thread-safe, but we will force every thread to get its own RNG at t
    {
        mySeed = RNG(); //Using the initial MersenneTwister to create a seed for every local one
    }
    MersenneTwister randGen(mySeed);
    #ifdef _OPENMP //This is simply to allow non-OpenMP compilation(! -fopenmp)
        int rank = omp_get_thread_num(); //I would like to know what my number of thread is
    #else
        int rank = 0; threads = 1; //I am the only thread
    #endif
    int start = 0, end = 0, pairStart = 0, pairEnd = 0; //Let's compute my region of activity as a thread <POPULATI
    computeThreadLoad(popSize, threads, rank, start, end); //Population
    computeThreadLoad(numPairs, threads, rank, pairStart, pairEnd); //Descendants

    int local_best = initializePopulation(context, randGen, population, start, end);
    if(population[local_best].isBetterThan(local_result))
        local_result = population[local_best];
}
```



6. Ejemplos prácticos: Optimizadores paralelos

Algoritmo Genético (III) (Balanceo de carga estático)

```
for(int i = 2*pairStart; i<(2*pairEnd); i=i+2){//Generating every pair:  
    this->selectProgenitors(randGen, popSize, population, tournSize, prog_A, prog_B);
```

```
#pragma omp barrier // We need to know that the whole population has been initialized before trying to operate with it (s  
for(int i = 0; i<numCycles; i++){//Evolutionary loop:  
    local_best = pairAndReproduce(context, randGen, population, popSize, pairStart, pairEnd, tournSize, descendants);  
    if(descendants[local_best].isBetterThan(local_result))  
        local_result = descendants[local_best];  
  
    local_best = mutate(context, randGen, descendants, pairStart, pairEnd, mutProb, perBitMutProb);  
    if(descendants[local_best].isBetterThan(local_result))  
        local_result = descendants[local_best];  
  
#pragma omp barrier //Before replacing and swapping, all threads must achieve this point (and single does not pro  
#pragma omp single  
{  
    this->replace(randGen, population, popSize, descendants, numPairs, tournSize, elite, new_population);// T  
    this->swap(&population, &new_population);  
} //Implicit BARRIER at the end of the single block -> Keep all threads at the same cycle  
}
```

6. Ejemplos prácticos: Optimizadores paralelos

Algoritmo Genético (IV) (Balanceo de carga dinámico)

```
#pragma omp parallel num_threads(threads)//Every variable created inside the parallel region is thread-local
{
    T local_result;//DEFAULT->power = -inf <minimum possible> To update it as a thread without critical

    uint32_t mySeed = 0;//Where to save the seed for my new local RNG
    #pragma omp critical(RAND) //rand() is not thread-safe, but we will force every thread to get its own RNG
    {
        mySeed = RNG();//Using the initial MersenneTwister to create a seed for every local one
    }
    MersenneTwister randGen(mySeed);
    #ifdef _OPENMP //This is simply to allow non-OpenMP compilation(! -fopenmp)
        int rank = omp_get_thread_num(); //I would like to know what my number of thread is
    #else
        int rank = 0; threads = 1;//I am the only thread
                                //Include here knowledge-based solutions distribution)
    #endif

    int local_best = initializePopulation(context, randGen, population, popSize);//Parallel no-wait
    if(population[local_best].isBetterThan(local_result))
        local_result = population[local_best];
}

#pragma omp for schedule(dynamic) nowait//Think in this
for(int i = 0; i<popSize; i++){//If we inject the first
    population[i].randomSolution(randGen, context);}
```



6. Ejemplos prácticos: Optimizadores paralelos

Algoritmo Genético (V) (Balanceo de carga dinámico)

```
#pragma omp for schedule(dynamic) //DO NOT ADD nowait (we cannot try to mutate descendants in advance
for(int i = 0; i<(2*numPairs); i=i+2){//Generating every pair:
    this->selectProgenitors(randGen, popSize, population, tournSize, prog_A, prog_B);
```

```
#pragma omp barrier // We need to know that the whole population has been initialized before trying to operate with it
for(int i = 0; i<numCycles; i++){//Evolutionary loop:
    local_best = this->pairAndReproduce(context, randGen, population, popSize, tournSize, descendants, numPairs);
    if(descendants[local_best].isBetterThan(local_result))
        local_result = descendants[local_best];
    //There is a barrier at the end of pairAndReproduce because we cannot know if descendants re-assigned for mutation
    local_best = mutate(context, randGen, descendants, numPairs, mutProb, perBitMutProb);//Parallel no-wait
    if(descendants[local_best].isBetterThan(local_result))
        local_result = descendants[local_best];

#pragma omp barrier //Before replacing and swapping, all threads must achieve this point (and single does not
#pragma omp single
{
    this->replace(randGen, population, popSize, descendants, numPairs, tournSize, elite, new_population);
    this->swap(&population, &new_population);
}//Implicit BARRIER at the end of the single block -> Keep all threads at the same cycle
}
```

6. Ejemplos prácticos: Optimizadores paralelos

Algoritmo Genético (VI)

```
#pragma omp critical
{
    if(local_result.isBetterThan(best_solution))
        best_solution = local_result;
}
//Implicit barrier at the end of a parallel region <JOIN>

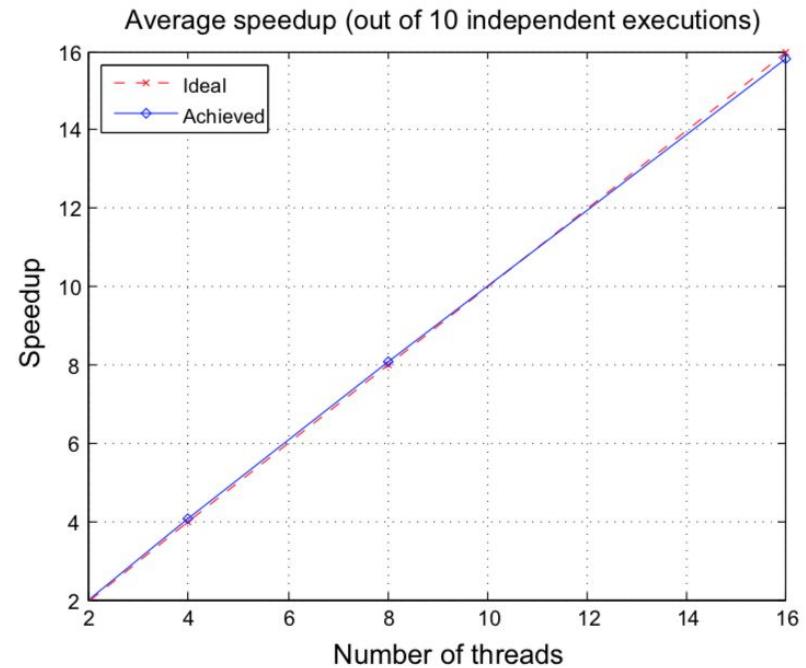
this->freeMemory(population, new_population, descendants);
return best_solution;
}
```

6. Ejemplos prácticos: Optimizadores paralelos

Algoritmo Genético (VII)

Estático:

El dinámico quedaba algo por debajo pues el desbalanceo solución buena/mala duraba poco



6. Ejemplos prácticos: Optimizadores paralelos

Algoritmo Memético (I)

Un **algoritmo memético** es un optimizador basado en **poblaciones** en el que los individuos adquieren también un rol activo de auto-mejora (**búsqueda local**).

Un ejemplo es **UEGO**, “*Universal Evolutionary Global Optimizer*”, que define un procedimiento general de gestión de soluciones candidatas, y uno seleccionable de búsqueda local.

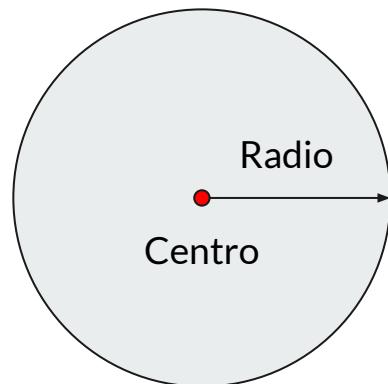
Es **multimodal**: Compatible con la identificación de múltiples óptimos

Se trata de un método **aplicado con éxito** en la búsqueda de fármacos, plegado de proteínas, diseño de campos de helióstatos, ajuste de modelos neuronales...

6. Ejemplos prácticos: Optimizadores paralelos

Algoritmo Memético (II)

Cada solución candidata o individuo es una “especie” para UEGO, y representa a una región del espacio de búsqueda:

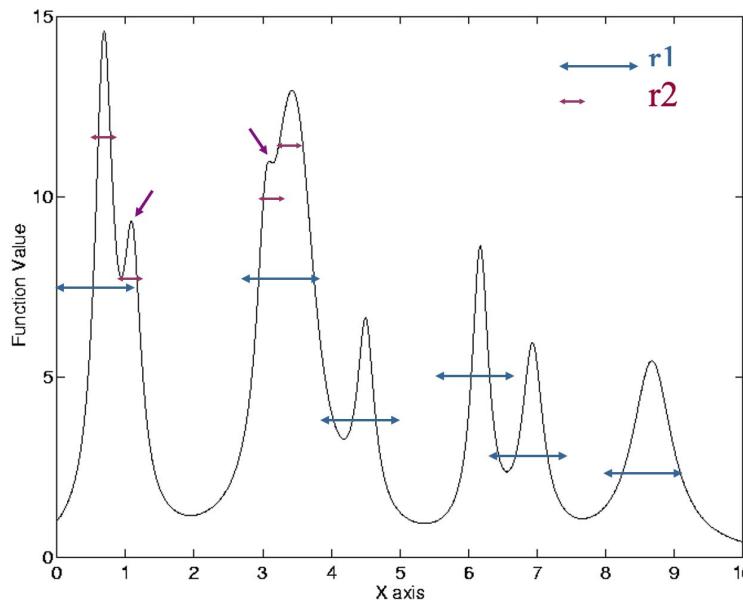


Cada especie se define por su centro y su radio de acción.

6. Ejemplos prácticos: Optimizadores paralelos

Algoritmo Memético (III)

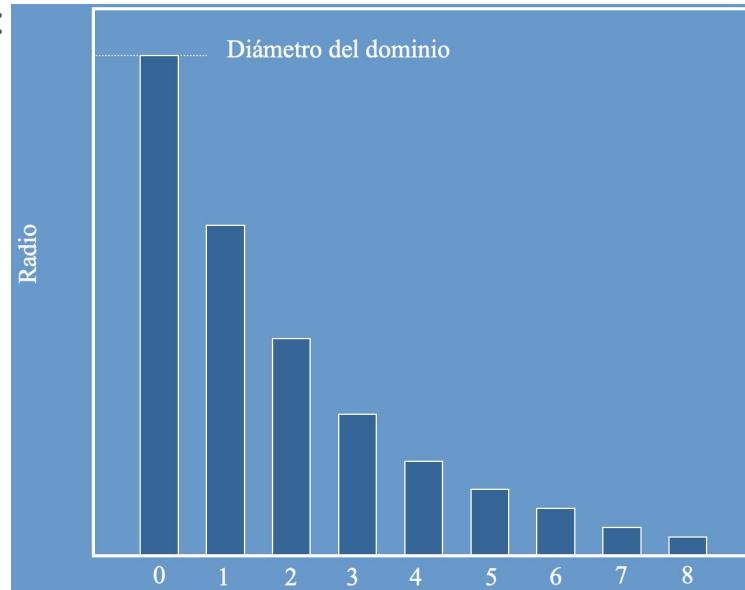
Las especies se distribuyen por el espacio, y los radios de las nuevas se van reduciendo (*cooling* o “enfriamiento”):



6. Ejemplos prácticos: Optimizadores paralelos

Algoritmo Memético (III)

Las especies se distribuyen por el espacio, y los radios de las nuevas se van reduciendo (*cooling* o “enfriamiento”):



6. Ejemplos prácticos: Optimizadores paralelos

Algoritmo Memético (IV)

Parámetros de **entrada**:

- N: Número máximo de evaluaciones
- I: Número de niveles (iteraciones)
- M: Número máximo de especies
- r: Radio mínimo (último nivel)

Información calculada para el nivel i:

- R_i : Radio del nivel i
- new_i : Máximo de evaluaciones para crear especies
- n_i : Máximo de evaluaciones para optimización local

6. Ejemplos prácticos: Optimizadores paralelos

Algoritmo Memético (V)

1. Inicializar lista de especies
2. Optimizar especies (n_1)
3. Desde $i=2$ hasta l
 - a. Determinar R_i , new_i , n_i
 - b. Crear especies (new_i) % Presupuesto por especie: $new_i / size(pop)$
 - c. Fundir especies (R_i)
 - d. Eliminar especies (M)
 - e. Optimizar especies (n_i) % Presupuesto por especie: n_i / M
 - f. Fundir especies (R_i)

6. Ejemplos prácticos: Optimizadores paralelos

Algoritmo Memético (VI)

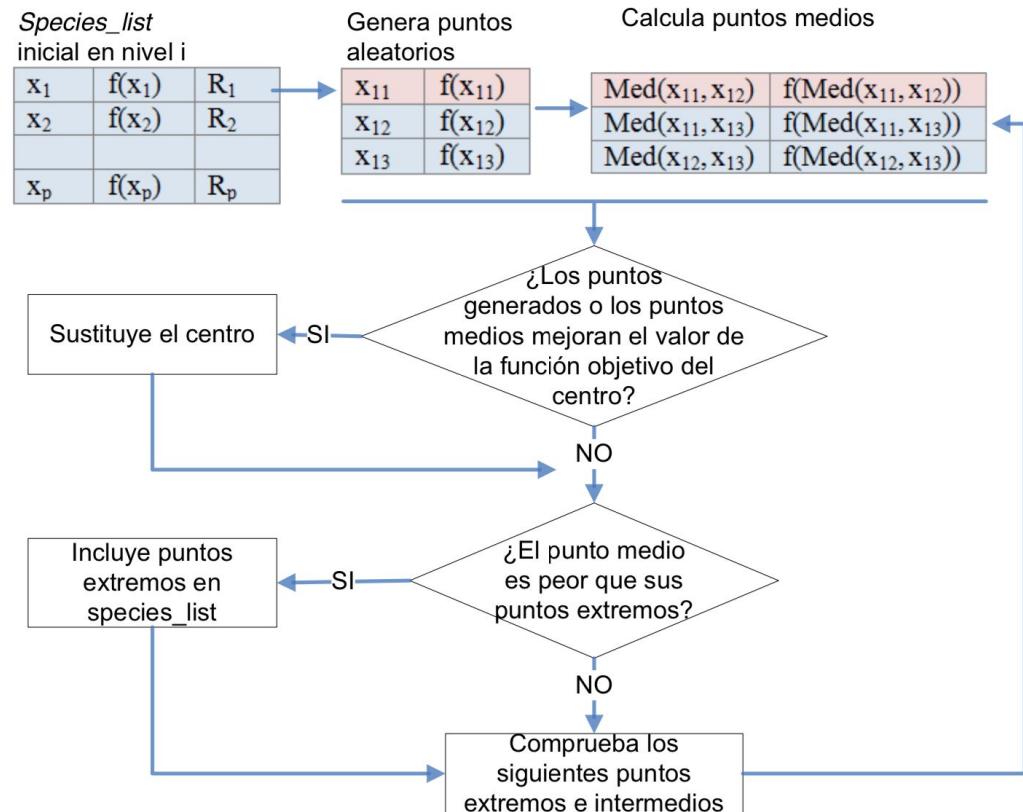
1. Inicializar lista de especies
2. Optimizar especies (n_1)
3. Desde $i=2$ hasta l
 - a. Determinar R_i , new_i , n_i
 - b. Crear especies (new_i)
 - c. Fundir especies (R_i)
 - d. Eliminar especies (M)
 - e. Optimizar especies (n_i)
 - f. Fundir especies (R_i)



6. Ejemplos prácticos: Optimizadores paralelos

Algoritmo Memético (VII)

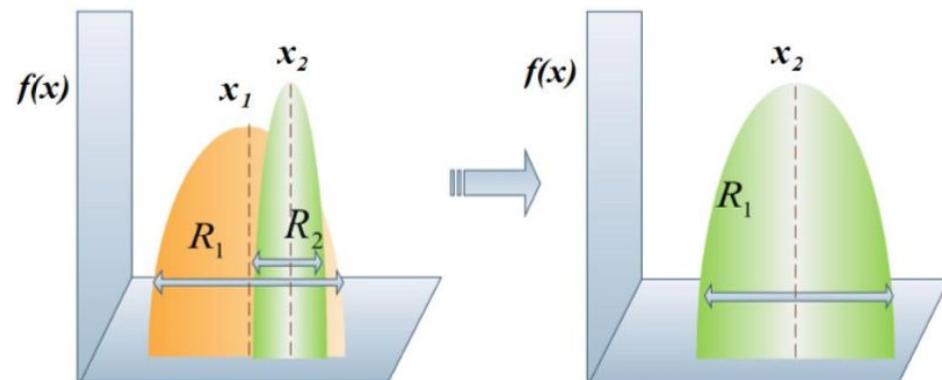
1. Inicializar lista de especies
2. Optimizar especies (n_1)
3. Desde $i=2$ hasta l
 - a. Determinar R_i , new_i , n_i
 - b. **Crear especies** (new_i)
 - c. Fundir especies (R_i)
 - d. Eliminar especies (M)
 - e. Optimizar especies (n_i)
 - f. Fundir especies (R_i)



6. Ejemplos prácticos: Optimizadores paralelos

Algoritmo Memético (VIII)

1. Inicializar lista de especies
2. Optimizar especies (n_1)
3. Desde $i=2$ hasta l
 - a. Determinar R_i , new_i , n_i
 - b. Crear especies (new_i)
 - c. **Fundir especies (R_i)**
 - d. Eliminar especies (M)
 - e. Optimizar especies (n_i)
 - f. **Fundir especies (R_i)**



6. Ejemplos prácticos: Optimizadores paralelos

Algoritmo Memético (IX)

1. Inicializar lista de especies
2. Optimizar especies (n_1)
3. Desde $i=2$ hasta l
 - a. Determinar R_i , new_i , n_i
 - b. Crear especies (new_i)
 - c. Fundir especies (R_i)
 - d. **Eliminar especies (M)**
 - e. Optimizar especies (n_i)
 - f. Fundir especies (R_i)



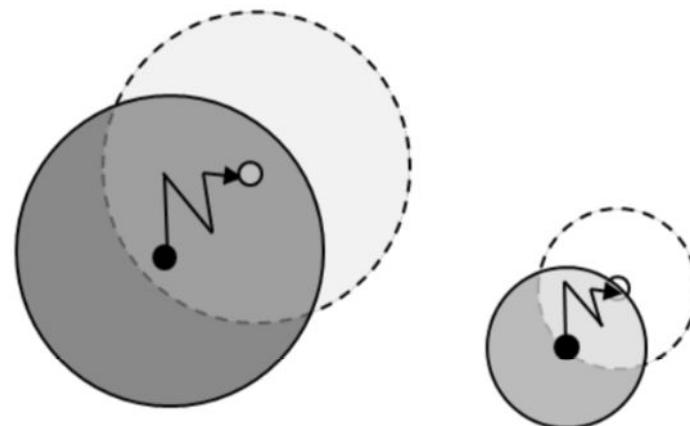
Si la lista es más grande de lo permitido, se elimina el excedente (empezando por los de mayor nivel (o menor radio)).

6. Ejemplos prácticos: Optimizadores paralelos

Algoritmo Memético (IX)

1. Inicializar lista de especies
2. **Optimizar especies (n_1)**
3. Desde $i=2$ hasta l
 - a. Determinar R_i , new_i , n_i
 - b. Crear especies (new_i)
 - c. Fundir especies (R_i)
 - d. Eliminar especies (M)
 - e. **Optimizar especies (n_i)**
 - f. Fundir especies (R_i)

Se lanza un optimizador local dentro de cada especie. Cuando encuentra un punto mejor que el centro, éste pasa a ser el nuevo centro y la especie se mueve.
Se suele usar **SASS**.



6. Ejemplos prácticos: Optimizadores paralelos

Algoritmo Memético (X)

Se suele usar **SASS**:

```

Parámetros de entrada: Scnt, Fcnt, ex, ct,  $\sigma_{ub}$ ,  $\sigma_{lb}$ , MaxItc
Seleccionar aleatoriamente  $x_0$  #inicializar vector búsqueda
 $b_0=0$  #inicializar vector parcial
 $k=1$ , scnt=0, fcnt=0,  $\sigma_0=1$ 
MIENTRAS  $k < \text{MaxItc}$  HACER:
    SI scnt> Scnt ENTONCES  $\sigma_k=ex \cdot \sigma_{k-1}$ 
    SI fcnt> Fcnt ENTONCES  $\sigma_k=ct \cdot \sigma_{k-1}$ 
    SI  $\sigma_{k-1} < \sigma_{lb}$  ENTONCES  $\sigma_k= \sigma_{lb}$ 
         $\xi_k=N(b_k, \sigma_k I)$  # generar vector aleatorio
        # gaussiano multivariado
    SI  $\Phi(x_k + \xi_k) < \Phi(x_k)$  ENTONCES
         $x_{k+1}=x_k + \xi_k$ 
         $b_{k+1}=0.4\xi_k+0.2b_k$ 
        scnt=scnt+1, fcnt=0
    SI NO
        SI  $\Phi(x_k - \xi_k) < \Phi(x_k) < \Phi(x_k + \xi_k)$  ENTONCES
             $x_{k+1}=x_k - \xi_k$ 
             $b_{k+1}=b_k-0.4\xi_k$ 
            scnt=scnt+1, fcnt=0
        SI NO
             $x_{k+1}=x_k$ 
             $b_{k+1}=0.5b_k$ 
            fcnt=fcnt+1, scnt=0
    k=k+1

```

Ascenso de colinas
estocástico adaptativo

6. Ejemplos prácticos: Optimizadores paralelos

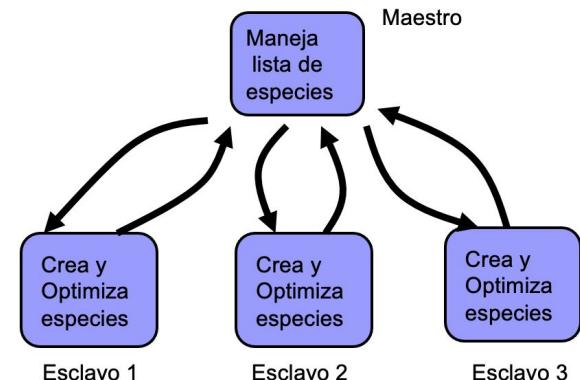
Algoritmo Memético (XI)

Paralelizando UEGO:

- **Estrategia Maestro-Esclavo con modelo global**
- El maestro maneja la lista de especies que distribuye entre los esclavos
- Los esclavos crean y optimizan las especies recibidas

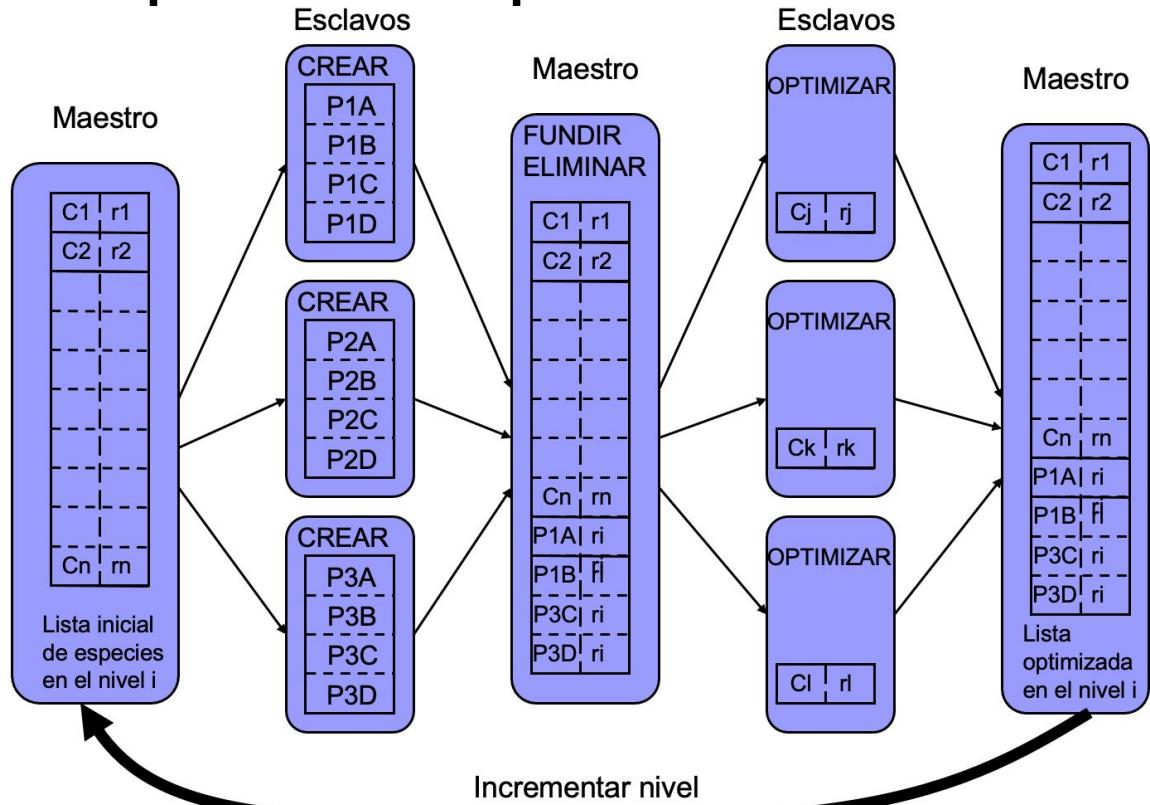
Dos implementaciones:

- Síncrona (PSUEGO)
- Asíncrona (PAUEGO)



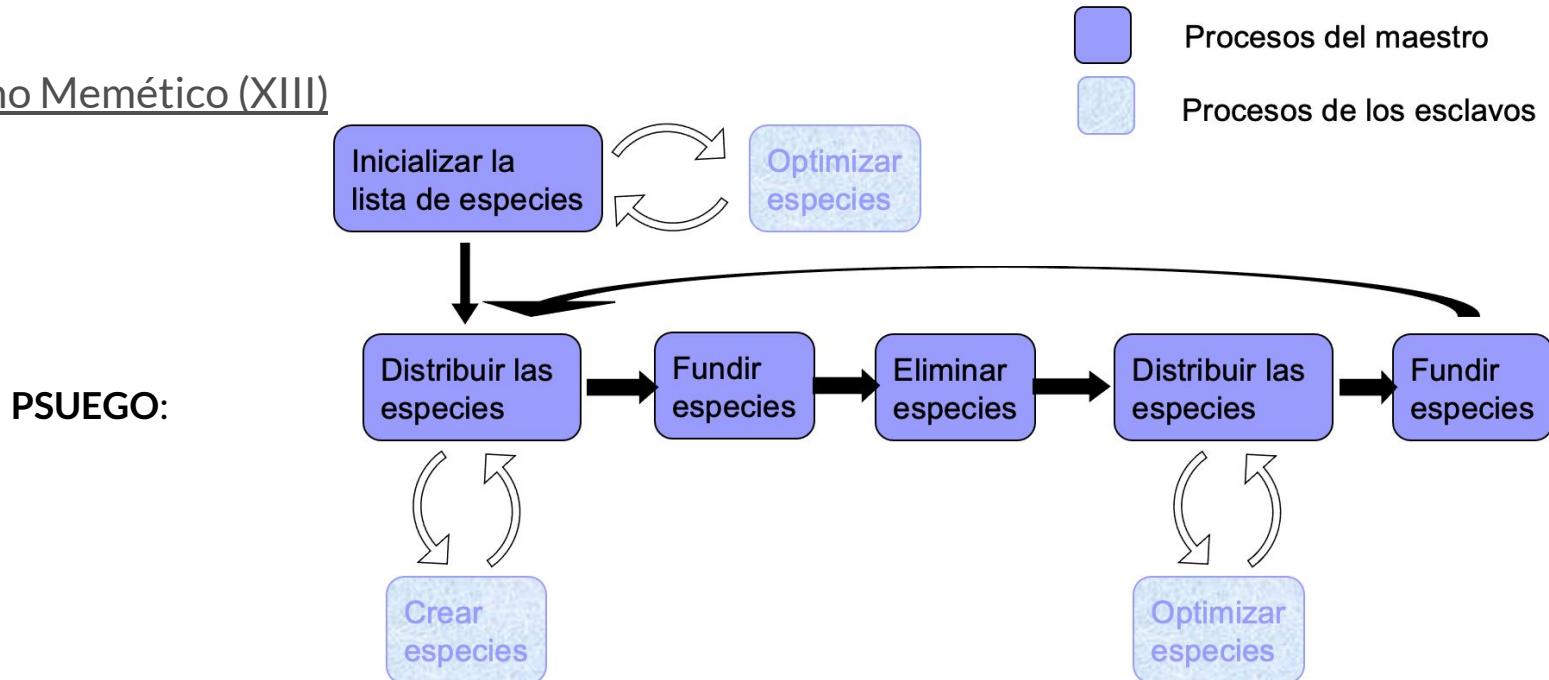
6. Ejemplos prácticos: Optimizadores paralelos

Algoritmo Memético (XII)



6. Ejemplos prácticos: Optimizadores paralelos

Algoritmo Memético (XIII)



6. Ejemplos prácticos: Optimizadores paralelos

Algoritmo Memético (XIV)

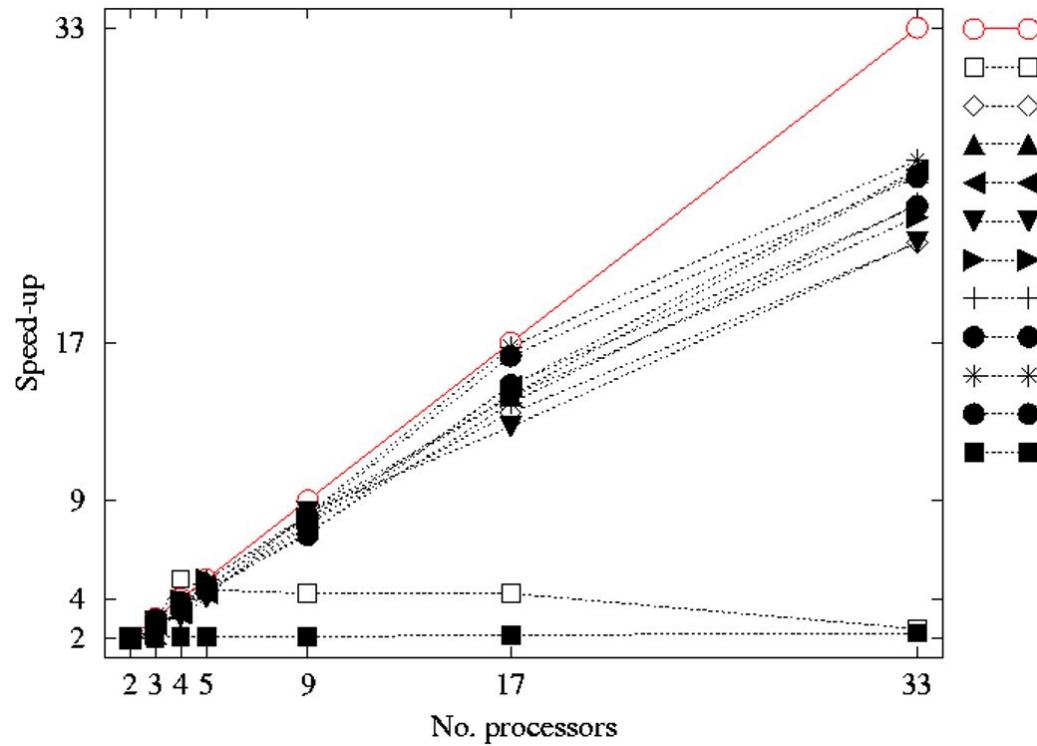
PSUEGO:

Función	% Tiempo en espera							
	NP=2	NP=3	NP=4	NP=5	NP=9	NP=17	NP=33	
F1	49	34	26	20	18	23	36	
F2	49	34	25	20	20	28	26	
F3	49	36	27	21	18	21	38	
F4	49	34	26	21	19	23	36	
F5	49	35	28	21	18	21	36	
F6	49	38	25	16	19	28	38	
F7	49	36	27	21	18	17	35	
F8	49	34	26	20	20	29	36	
F9	49	33	26	20	19	16	35	
F10	50	34	28	21	18	26	36	
F11	49	34	26	21	18	21	33	

6. Ejemplos prácticos: Optimizadores paralelos

Algoritmo Memético (XV)

PSUEGO:



6. Ejemplos prácticos: Optimizadores paralelos

Algoritmo Memético (XVI)

Problemas de PSUEGO:

- Puntos de sincronismo tras las etapas de creación y optimización
- El maestro permanece prácticamente parado durante la creación y la optimización
- La carga computacional no está bien balanceada

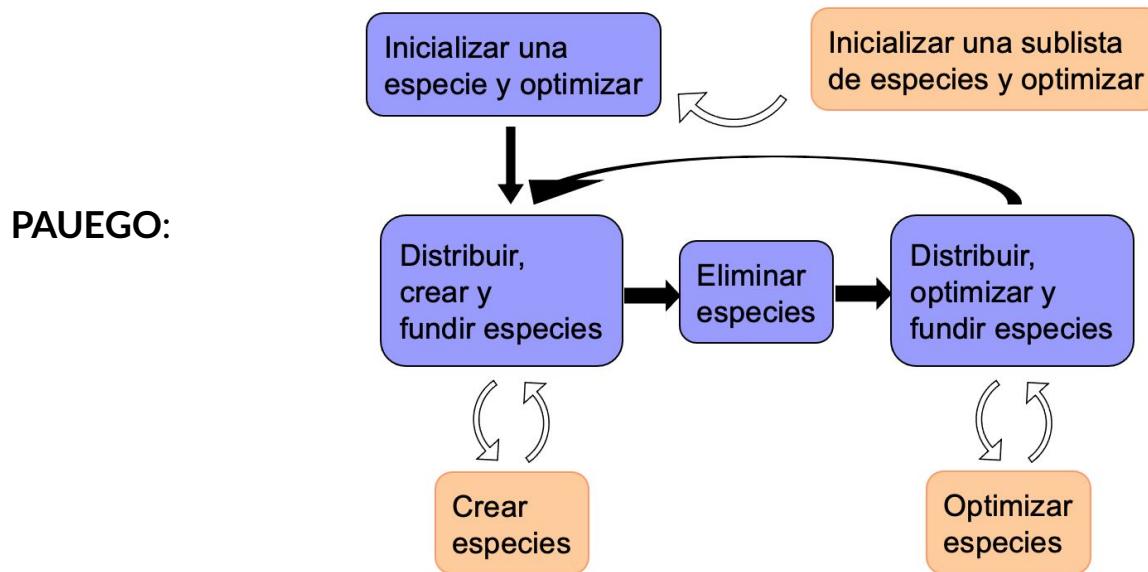
PAUEGO busca:

- Balancear mejor la carga: el maestro también participa en la creación y optimización
- Reducir esperas: el maestro comienza a fundir especies en cuanto empieza a recibirlas

6. Ejemplos prácticos: Optimizadores paralelos

Algoritmo Memético (XVII)

- Procesos del maestro
- Procesos de los esclavos



6. Ejemplos prácticos: Optimizadores paralelos

Algoritmo Memético (XVIII)

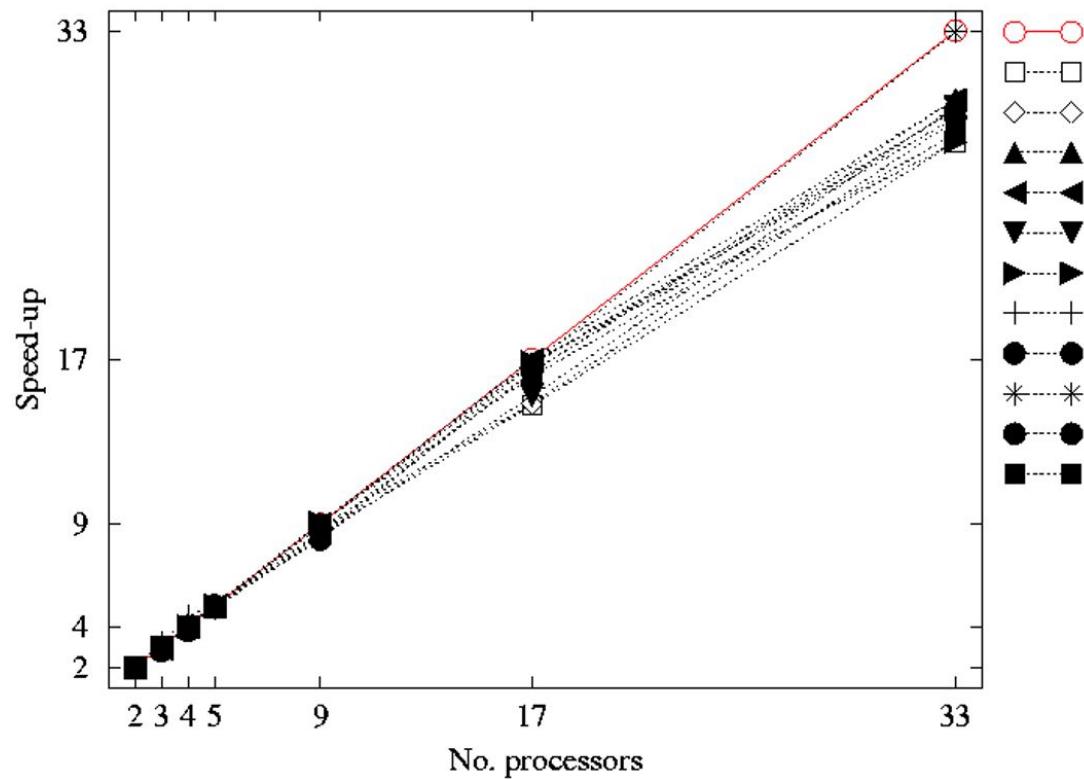
PAUEGO:

Función	% Tiempo en espera						
	NP=2	NP=3	NP=4	NP=5	NP=9	NP=17	NP=33
F1	0	1	1	2	2	3	4
F2	0	0	1	1	1	1	2
F3	0	0	1	2	2	2	4
F4	0	0	0	1	1	2	2
F5	0	1	1	0	1	1	2
F6	1	1	1	2	2	3	4
F7	0	1	1	0	1	1	2
F8	0	0	1	1	0	1	2
F9	0	0	1	1	0	1	1
F10	0	0	1	1	2	2	3
F11	0	0	1	0	1	2	2

6. Ejemplos prácticos: Optimizadores paralelos

Algoritmo Memético (XIX)

PAUEGO:



6. Ejemplos prácticos: Optimizadores paralelos

Algoritmo Memético (XX)

- El número de evaluaciones de la función en el proceso de optimización varía para cada especie, de ahí que resulte muy difícil balancear la carga.
- En **PSUEGO**, el tiempo dedicado a las **comunicaciones** es de alrededor de un **40% del tiempo** de ejecución.
- En **PAUEGO**, este tiempo se reduce al **15%** del tiempo de ejecución.

¿Y una paralelización sobre GPU?

- Es complejo: las estructuras internas del algoritmo han de modificarse.

6. Ejemplos prácticos: Optimizadores paralelos

Algoritmo Memético (XXI)

UEGO sobre GPU:

- Prescindir de listas enlazadas y **apostar por matrices**
- **Desenrollar bucles** (aunque no demasiado: uso de registros, demasiadas tareas para el planificador...)

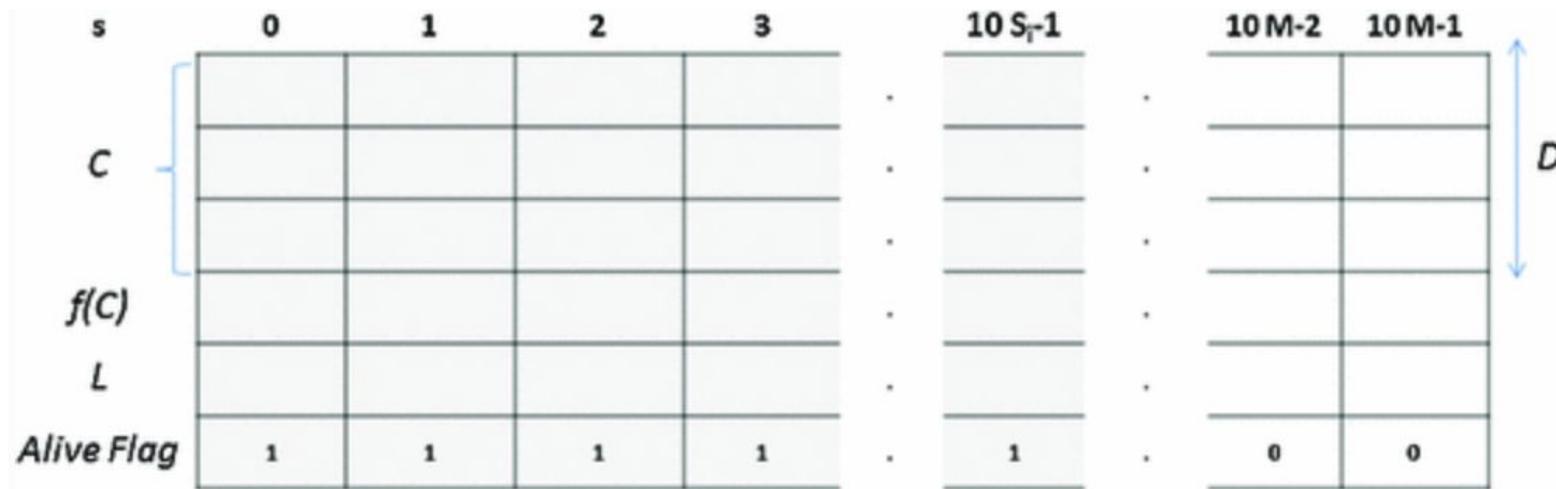
```
/* Antes del desenrollado */
for (i = 0; i < N; ++i) {
    c[i] = a[i] + b[i];
}
```

```
/* Despues del desenrollado */
for (i = 0; i < N - (4 - 1); i += 4) {
    c[i] = a[i] + b[i];
    c[i+1] = a[i+1] + b[i+1];
    c[i+2] = a[i+2] + b[i+2];
}
```

6. Ejemplos prácticos: Optimizadores paralelos

Algoritmo Memético (XXII)

Máximo reservado directamente
UEGO sobre GPU:

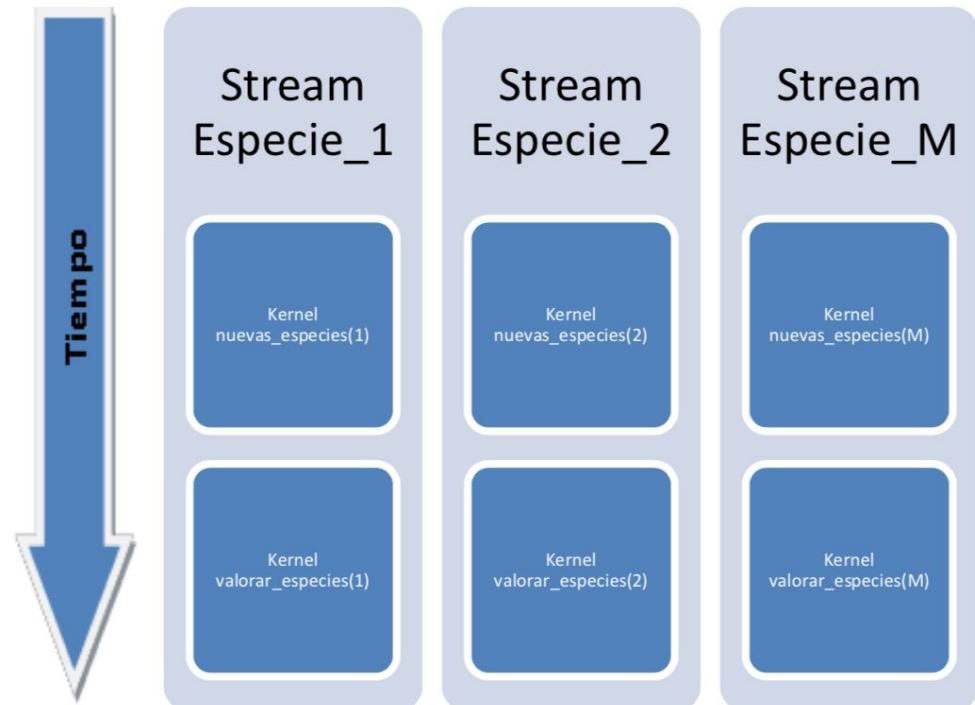


6. Ejemplos prácticos: Optimizadores paralelos

Algoritmo Memético (XXIII)

UEGO sobre GPU:

```
streams = (cudaStream_t*)  
malloc(sizeof(cudaStream_t)*longitud_lista);  
for (int st = 0; st < longitud_lista; ++st) {  
    cudaStreamCreate (&streams[st]);  
}  
for (int s = 0; s < longitud_lista; ++s) {  
    nuevas_especies<<<(budget+(hilos-1))/hilos,  
        hilos,0,streams[s]>>>  
    (devStates, d_puntos, s, i+1, budget, D, M, d_nivel);  
    valorar_especies<<<(budget+(hilos-1))/hilos,  
        hilos,0,streams[s]>>>  
    (s, budget, M, d_valor, d_nivel, d_orden);  
}
```



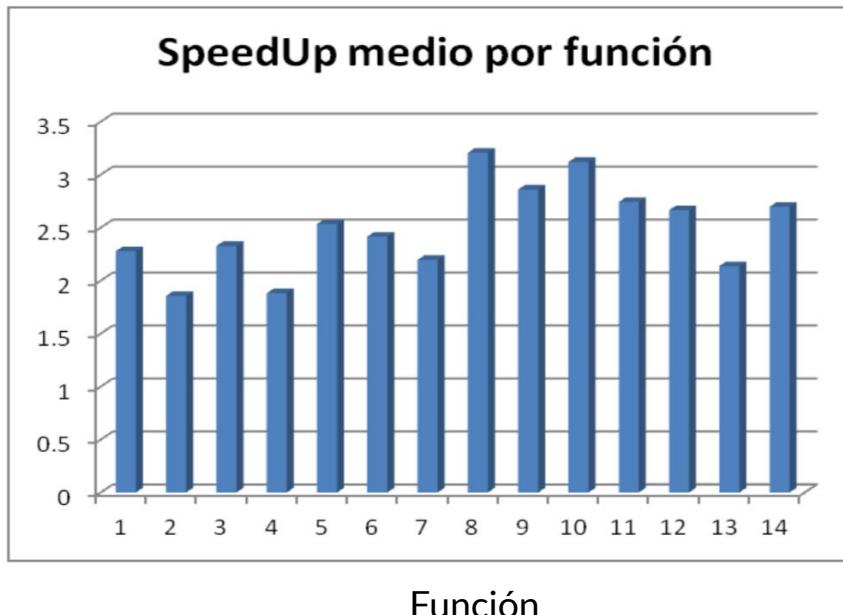
Y también un optimizador local más paralelizable que SASS!

6. Ejemplos prácticos: Optimizadores paralelos

Algoritmo Memético (XXIV)

UEGO sobre GPU:

Aceleración

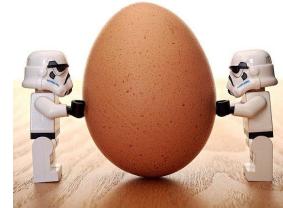


La aceleración es baja comparada con la multi-CPU:

- ¿Funciones objetivo más costosas?
- ¿Mayores poblaciones?

...

Trabajo para la próxima semana



Cada grupo debe buscar y exponer una aplicación/problema en el que la computación basada en GPU's resulte beneficiosa y superior al paralelismo sobre CPU.

=> Un miembro del grupo presentará su trabajo la próxima semana. Habrá que entregar tanto la presentación como una transcripción de lo que se dice (incluyendo las fuentes consultadas).

Importante: Poneos de acuerdo entre grupos para no escoger el mismo caso.



—

Gracias.

