Introducción a la Computación Evolutiva

Carlos A. Coello Coello

carlos.coellocoello@cinvestav.mx
CINVESTAV-IPN
Evolutionary Computation Group (EVOCINV)
Departamento de Computación
Av. IPN No. 2508, Col. San Pedro Zacatenco
México, D.F. 07360, MEXICO

Clase 13

Algoritmos Genéticos Paralelos

Paralelización Global

La paralelización global es un método relativamente fácil de implementar y puede obtenerse un incremento significativo de velocidad si los costos de comunicación no dominan los costos de procesamiento.

A la paralelización global también se le conoce como **algoritmo genético panmítico**, pues se cuenta con un solo depósito de material genético (*gene pool*), o sea con una sola población.

En el algoritmo genético panmítico no se requieren nuevos operadores ni nuevos parámetros y la solución encontrada será la misma que la producida con un algoritmo genético convencional (o sea, serial).

Paralelización Global

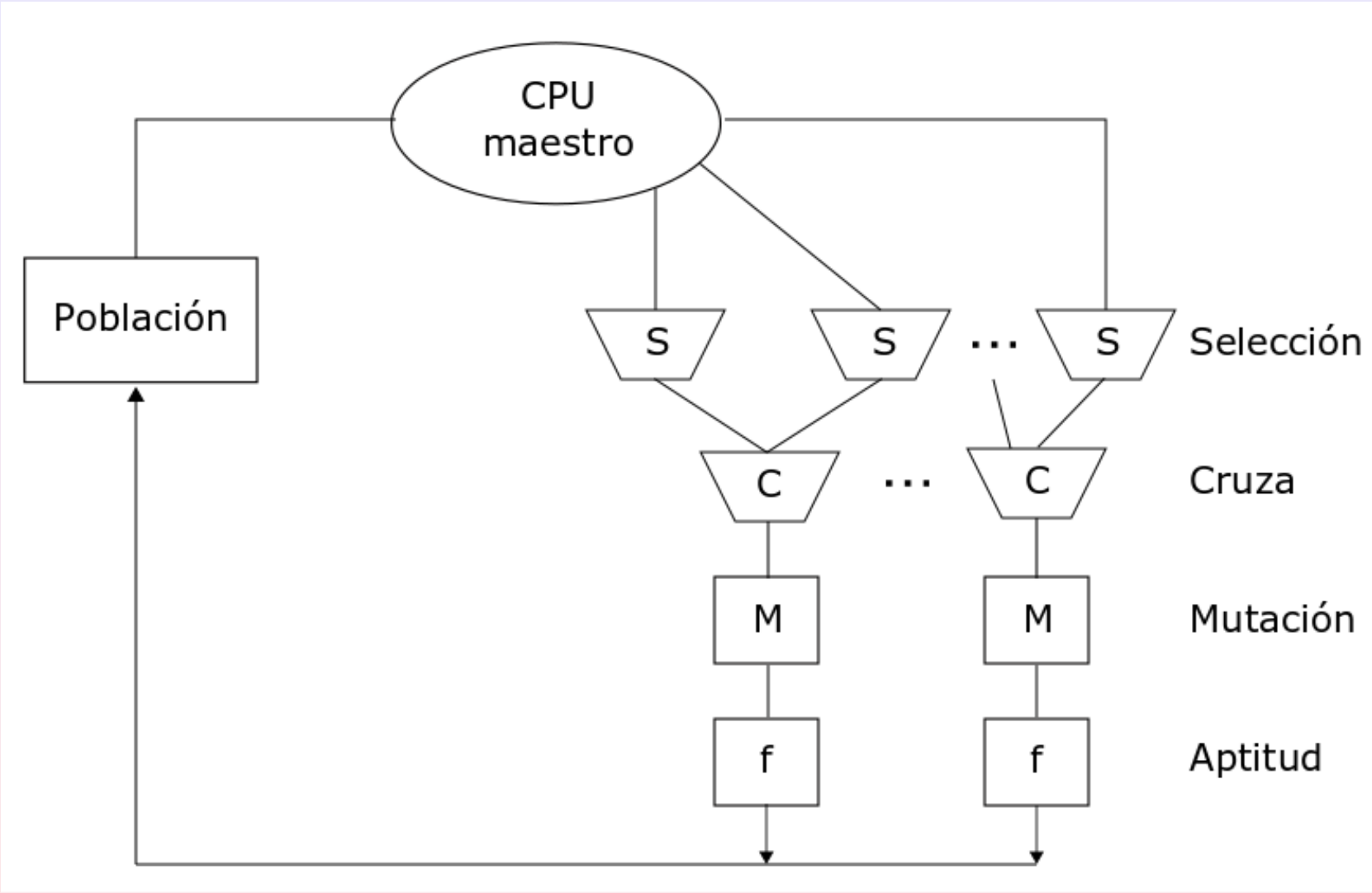
Los algoritmos genéticos panmíticos son útiles cuando el costo de evaluar la función de aptitud es elevado (p.ej., una simulación).

Es importante hacer notar que aunque el paralelismo global normalmente es **síncrono** (o sea, que el programa se detiene y espera a recibir los valores de aptitud de toda la población antes de proceder a producir la siguiente generación), puede implementarse también de forma **asíncrona**, aunque en ese caso, su funcionamiento ya no resultará equivalente al de un algoritmo genético convencional.

Además de paralelizarse la evaluación de la función de aptitud, en el paralelismo global es posible incluir también los operadores genéticos, pero dada la simplicidad de éstos, no suelen paralelizarse, pues los costos de comunicación disiparían cualquier mejora en el desempeño del programa.

Evidentemente, la paralelización global corresponde a un esquema **Amo-Esclavo** como se muestra en el acetato siguiente.

Algoritmos Genéticos Paralelos



Algoritmos Genéticos Paralelos

Aunque un esquema de paralelización global es muy fácil de implementar, el uso de ejecuciones independientes puede ser muy efectivo.

Consideremos un caso en el que una sola ejecución de un algoritmo tiene una *probabilidad de éxito p* dada (o sea, una probabilidad de encontrar una solución satisfactoria en un tiempo límite dado).

Si usamos varias ejecuciones independientes, se puede incrementar de manera significativa esta probabilidad de éxito [Sudholt, 2015]. A este enfoque se le conoce como *amplificación de la probabilidad*.

Dirk Sudholt, "**Parallel Evolutionary Algorithms**", in J. Kacprzyk and W. Pedrycz (Editors), *Springer Handbook of Computational Intelligence*, Chapter 46, pp. 929–959, Springer, Berlin, 2015.

Algoritmos Genéticos Paralelos

La probabilidad de que, en λ ejecuciones independientes, ninguna ejecución sea exitosa, es $(1-p)^{\lambda}$

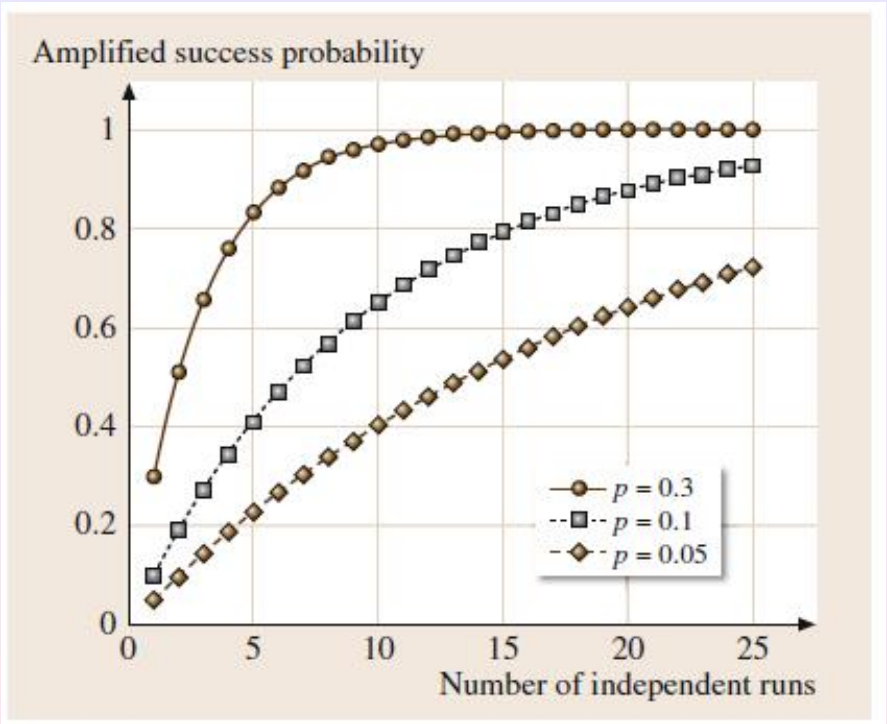
La probabilidad de que haya al menos una ejecución exitosa de entre las λ ejecutadas está dada por:

$$1-(1-p)^{\lambda} \tag{1}$$

En la gráfica del acetato siguiente, donde se muestra esta probabilidad de éxito para diferentes valores de λ y p.

Puede verse cómo, para un número pequeño de procesadores, la probabilidad de éxito se incrementa de manera casi lineal. Sin embargo, si el número de procesadores es grande, ocurre un efecto de saturación.

Algoritmos Genéticos Paralelos



Fuente de la figura

Dirk Sudholt, "**Parallel Evolutionary Algorithms**", in J. Kacprzyk and W. Pedrycz (Editors), *Springer Handbook of Computational Intelligence*, Chapter 46, pp. 929–959, Springer, Berlin, 2015.

Algoritmos Genéticos Paralelos

El beneficio de usar más procesadores decrece conforme aumenta el número de éstos.

El punto en el que ocurre la saturación depende de manera crucial de p.

Para probabilidades menores, la saturación ocurre sólo con un número muy grande de procesadores.

Cabe agregar que es posible efectuar las ejecuciones independientes con diferentes condiciones iniciales o diferentes parámetros.

Esto resulta útil para explorar de manera más efectiva el espacio de los parámetros y para obtener valores adecuados para dichos parámetros en un tiempo razonablemente corto.

Algoritmos Genéticos Paralelos

Las ejecuciones independientes sufren de una desventaja obvia: una vez que una ejecución alcanza una situación donde su población queda estancada en un óptimo local difícil, es muy posible que no pueda salirse de ahí.

Esto es muy desafortunado, porque otras ejecuciones podrían alcanzar regiones más promisorias del espacio de búsqueda al mismo tiempo.

Tiene más sentido establecer alguna forma de comunicación entre las diferentes ejecuciones para coordinar la búsqueda, de tal forma que las ejecuciones que hayan alcanzado soluciones de baja calidad puedan unirse a la búsqueda de regiones más prometedoras del espacio de búsqueda.

Algoritmos Genéticos Paralelos

Una alternativa es considerar cada ejecución como una **subpoblación** o una **isla**. Ese es precisamente el enfoque de los denominados **algoritmos genéticos de grano grueso**.

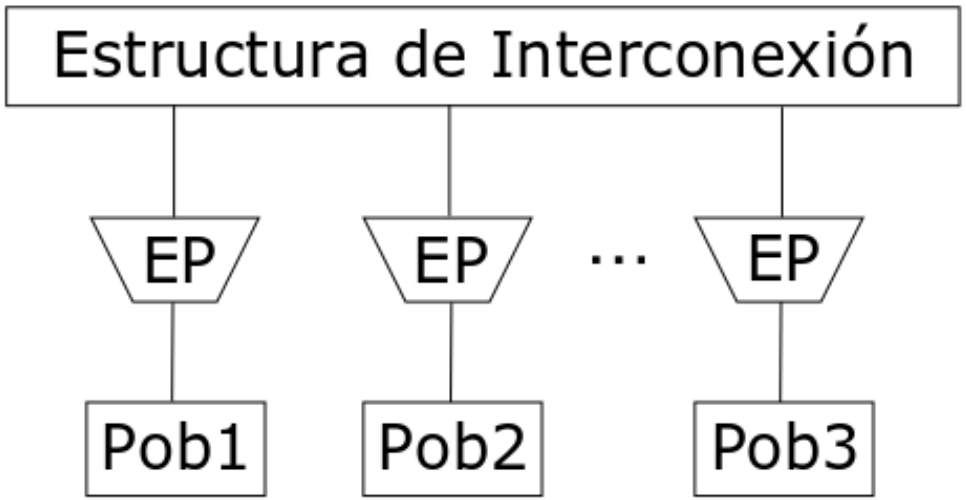
En este caso, la población del algoritmo genético se divide en múltiples subpoblaciones o **demes** que evolucionan de manera aislada la mayor parte del tiempo, aunque intercambian individuos ocasionalmente.

A este intercambio de individuos se le llama **migración**, y se considera como un nuevo operador genético.

Además de requerirse parámetros adicionales en este caso, el comportamiento de un algoritmo genético de grano grueso es diferente del de un algoritmo genético convencional (o sea, serial).

Algoritmos Genéticos Paralelos

Cada EP tiene una memoria local



Elementos de procesamiento

Poblaciones o demes

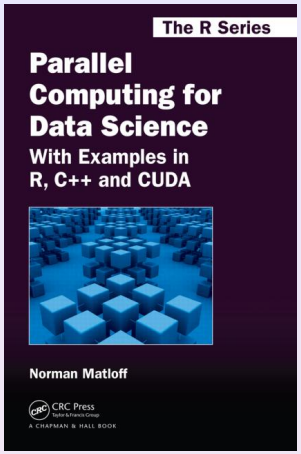
Algoritmos Genéticos Paralelos

A los **algoritmos genéticos de grano grueso** se les suele llamar también **algoritmos genéticos distribuidos**, porque suelen implementarse en computadoras MIMD con memoria distribuida.

Asimismo, algunos autores los llaman también **algoritmos genéticos de isla**, haciendo alusión a un modelo poblacional usado en genética en el cual se consideran **demes** relativamente aislados.

A éste se le conoce como **modelo de isla**, y es el modelo más popular de paralelización de un algoritmo genético.

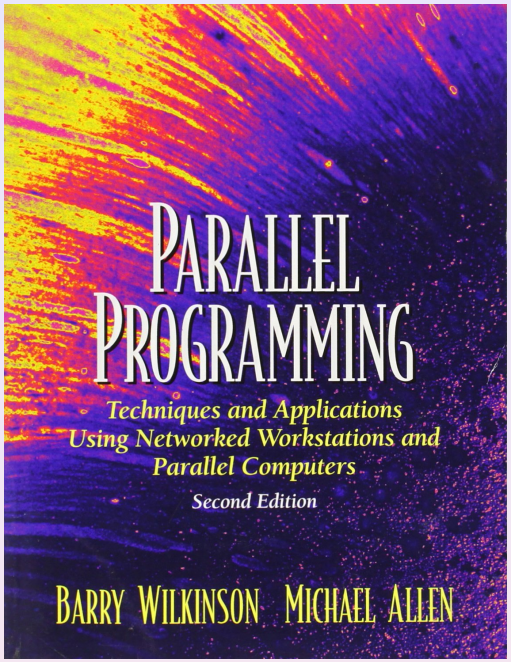
Algoritmos Genéticos Paralelos



Los algoritmos genéticos de grano grueso son muy populares debido a varias razones:

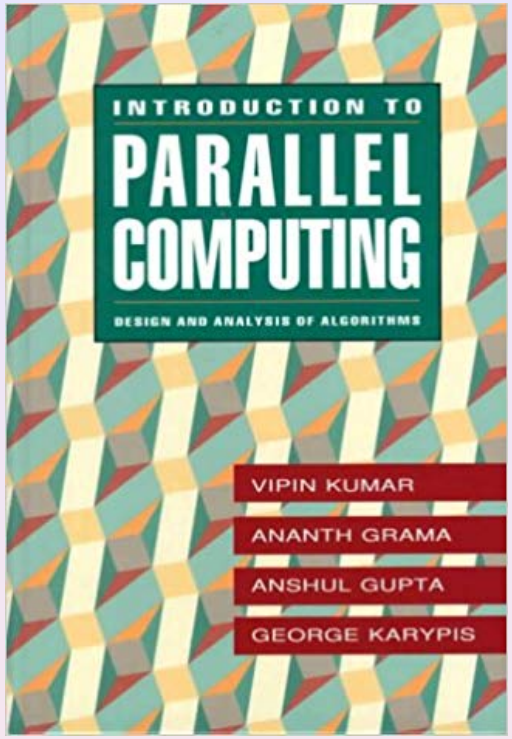
- Son una extensión muy simple de los algoritmos genéticos seriales. Simplemente se toman unos cuantos algoritmos genéticos convencionales (seriales), se ejecuta cada uno de ellos en un procesador diferente y, a ciertos intervalos de tiempo, se intercambian unos cuantos individuos entre ellos.

Algoritmos Genéticos Paralelos



- Aunque no se tenga acceso a una arquitectura paralela, puede implementarse un algoritmo genético de grano grueso a través de una simulación efectuada en una red de estaciones de trabajo, o incluso en una computadora con un solo procesador haciendo la simulación mediante software (usando por ejemplo MPI o PVM).

Algoritmos Genéticos Paralelos



- Se requiere relativamente de poco esfuerzo para convertir un algoritmo genético serial en un algoritmo genético de grano grueso. La mayor parte de la programación permanece igual, y sólo se requieren ciertas rutinas adicionales para implementar la migración.

Algoritmos Genéticos Paralelos

Los parámetros que requieren los algoritmos genéticos de grano grueso son:

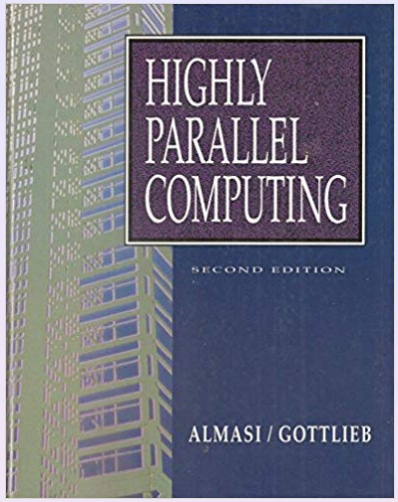
- Número de **demes** (esto puede estar determinado por el hardware disponible).
- Tamaño de cada **deme**.
- Estructura de la interconexión (o sea, la **topología** de migración).
- Intervalo de migración
- Número de migrantes
- Política de emigración
- Política de inmigración

Algoritmos Genéticos Paralelos

De entre estos parámetros, algunos como la **topología**, juegan un papel preponderante en el desempeño del algoritmo genético.

La topología determina qué tan rápido (o qué tan lentamente) se disemina una buena solución hacia los otros **demes**.

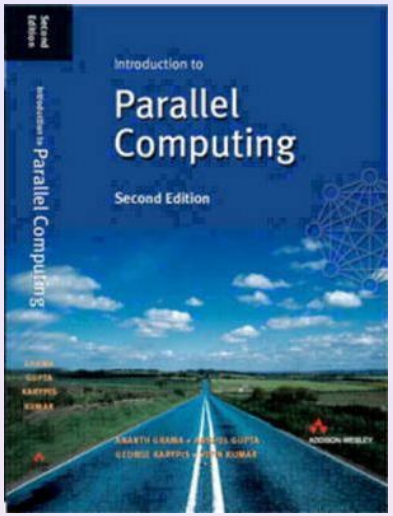
Algoritmos Genéticos Paralelos



Si se usa una topología dispersamente conectada (con un diámetro grande), las soluciones se diseminarán más lentamente y los **demes** estarán más aislados entre sí, permitiendo la aparición de soluciones diferentes, favoreciendo probablemente la diversidad.

La topología juega también un papel preponderante en el costo de las migraciones.

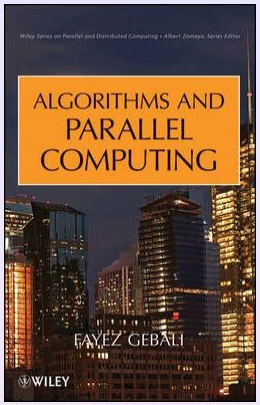
Algoritmos Genéticos Paralelos



Por ejemplo, una topología densamente conectada puede promover una mejor mezcla de individuos, pero a un costo computacional más alto.

Se sabe, por ejemplo, que una topología densa tiende a encontrar soluciones globales con un menor número de evaluaciones de la función de aptitud que si se usa una topología dispersa.

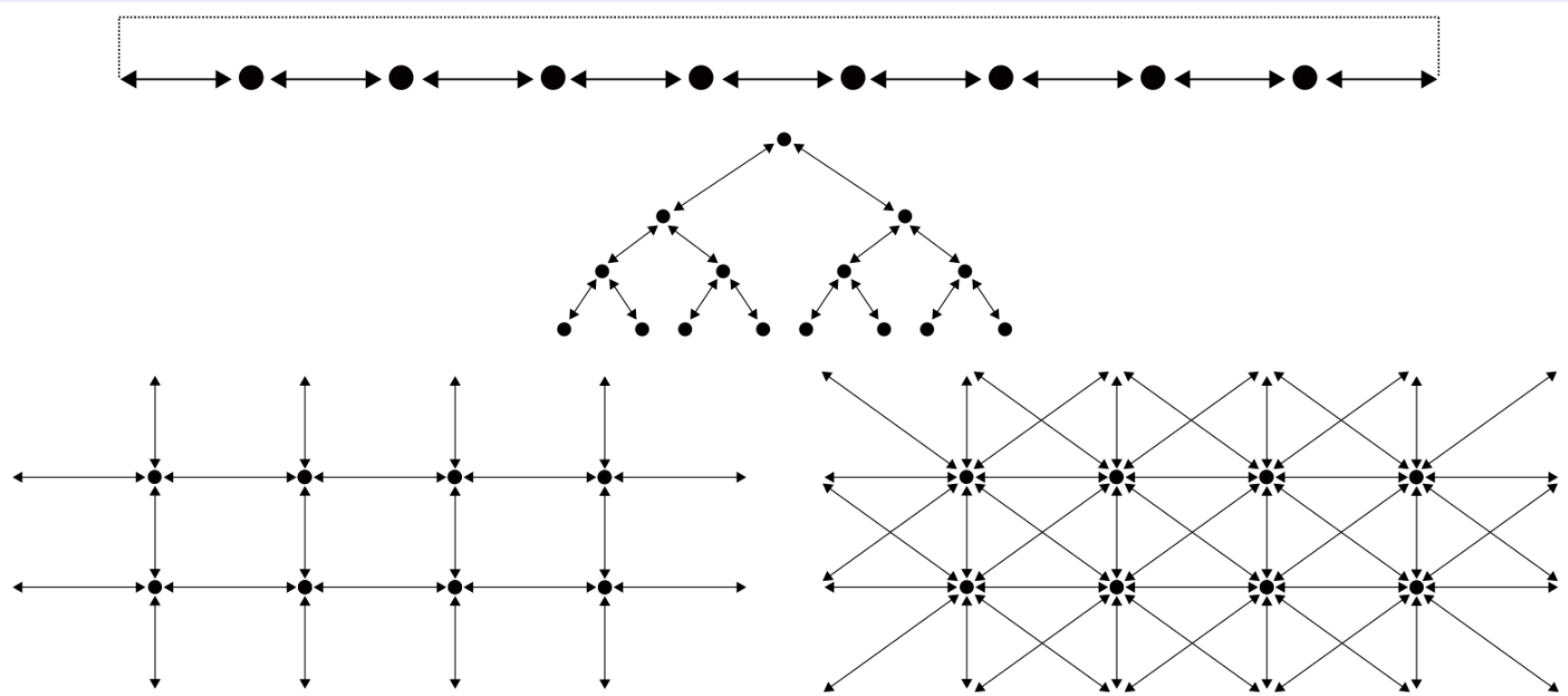
Algoritmos Genéticos Paralelos



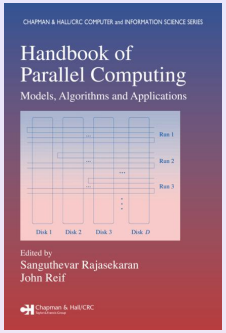
También es posible usar topologías **dinámicas**, en las que los **demes** no están limitados a poder comunicarse sólo con un cierto conjunto predefinido de **demes**, sino que envía sus migrantes a aquellos **demes** que satisfacen ciertos criterios.

La idea de este esquema es que puedan identificarse los **demes** donde los migrantes tienen mayores probabilidades de producir algún efecto. Usualmente, se usa la diversidad como el criterio principal para definir qué tan adecuado es un cierto **deme**.

Algoritmos Genéticos Paralelos
Ejemplos de Topologías



Algoritmos Genéticos Paralelos



Es importante mantener en mente la idea de que una topología es una estructura lógica que puede diferir de la estructura de hardware disponible.

Es decir, la topología de un algoritmo genético paralelo no necesariamente debe coincidir con la de nuestra computadora.

El problema de hacer esto, sin embargo, es que los costos de comunicación resultantes pueden ser muy elevados.

Algoritmos Genéticos Paralelos

Otro parámetro importante es la **política de emigración**.

Cuando se envían individuos a otra isla, éstos pueden ser removidos de la isla de origen. Otra alternativa, es enviar copias de ellos.

A este segundo esquema se le denomina **polinización**.

La selección de migrantes también es importante. Se puede seleccionar al mejor, al peor o a un individuo al azar.

Algoritmos Genéticos Paralelos

La **política de inmigración** también es importante.

Los inmigrantes puede reemplazar a los peores individuos, a individuos elegidos al azar o ser sujetos al mismo esquema de selección y/o de reemplazo utilizado dentro de la isla a la que arriban.

Se pueden usar mecanismos adicionales como reemplazar a los individuos más similares.

Adicionalmente, los inmigrantes pueden ser recombinados con los individuos presentes en la isla a la que llegan, antes de someterse al proceso de selección.

Algoritmos Genéticos Paralelos

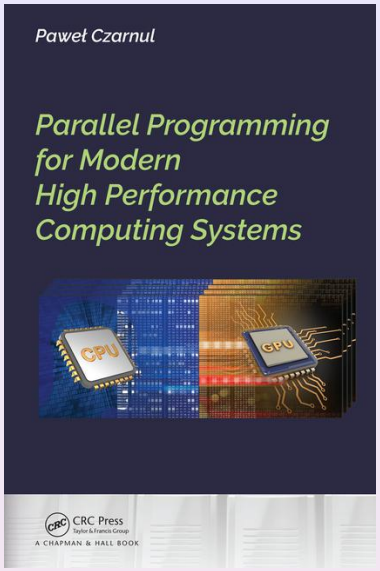
El **intervalo de migración** se refiere al intervalo de tiempo entre migraciones y es un parámetro que determina la velocidad a la que la información se disemina a través del modelo de isla.

Su recíproco es llamado frecuentemente **frecuencia de migración**.

Las migraciones frecuentes implican una diseminación rápida de información, mientras que una frecuencia baja de migración permite una mayor exploración.

Se hace notar que un intervalo de migración de ∞ produce ejecuciones totalmente independientes, como un caso especial.

Algoritmos Genéticos Paralelos



Otro parámetro relevante de los algoritmos genéticos de grano grueso es el **número de migrantes**.

A este parámetro también se le llama **tamaño de la migración** y determina qué tan rápidamente los inmigrantes toman el control de una isla.

Algoritmos Genéticos Paralelos

Si todas las islas ejecutan el mismo algoritmo bajo condiciones idénticas, decimos que tenemos un **modelo de isla homogéneo**.

Si no es así, tenemos entonces un **modelo de isla heterogéneo**.

En un modelo heterogéneo pueden usarse diferentes representaciones, funciones de aptitud, operadores y/o parámetros.

Este modelo tiene sentido cuando no sabemos qué configuración algorítmica es la más adecuada para resolver un problema.

Algoritmos Genéticos Paralelos

Skolicki [2000] propuso una visión de dos niveles de la dinámica de búsqueda de los modelos de isla.

El término **evolución intra-islas** describe el proceso que toma lugar dentro de cada isla.

A un nivel más elevado, la **evolución inter-islas** describe la interacción entre diferentes islas.

Z. Skolicki, "**An analysis of Island Models in Evolutionary Computation**", PhD thesis, George Mason University, Fairfax, Virginia, USA, 2000.

Algoritmos Genéticos Paralelos

Skolicki argumenta que las islas pueden verse como individuos en un nivel más elevado del proceso evolutivo. Las islas compiten entre sí y pueden tomar el control de otras islas, de manera análoga a cómo los individuos pueden reemplazar a otros en una población.

Una conclusión respecto a esta forma de ver a los algoritmos genéticos de isla es que un modelo de isla es más como una entidad compacta.

Evidentemente, los dos niveles de evolución a los que se refiere Skolicki interactúan entre sí. La importancia de cada nivel la determina el intervalo de migración y otros parámetros del sistema que afectan la diseminación de información.

Algoritmos Genéticos Paralelos

Otra forma de paralelizar un algoritmo genético es usando un esquema de **grano fino**. También se le conoce como **modelo de difusión**.

En este caso, la población de un algoritmo genético se divide en un gran número de subpoblaciones muy pequeñas.

De hecho, el caso ideal sería tener sólo un individuo por cada unidad de procesamiento disponible. Ese caso en particular se denomina **algoritmo genético celular**, pues en este caso, a cada isla se le llama **célula**.

En los 1990s, este modelo resultaba muy adecuado para arquitecturas masivas en paralelo.

Algoritmos Genéticos Paralelos

El problema del **paralelismo de grano fino** es que el costo de comunicación entre los procesadores puede hacer que el desempeño del algoritmo se degrade con relativa facilidad.

Es común implementar este tipo de paralelismo colocando los individuos en una malla bidimensional, debido a que ésta era la topología usada en hardware para muchas arquitecturas masivas en paralelo.

En los **algoritmos genéticos celulares**, las topologías más comunes son los anillos y los grafos toroidales bi-dimensionales. Cada individuo sólo puede aparearse con sus vecinos en la topología. Este tipo de interacción se produce a cada generación.

Algoritmos Genéticos Paralelos

En los algoritmos genéticos celulares, el concepto de **vecindario** es, por tanto, muy importante.

El **vecindario** se refiere al área dentro de la cual puede moverse un migrante de una cierta **célula**.

Asociado al vecindario se encuentra el concepto de **radio de selección**, que se refiere a la cantidad de vecinos entre los cuales se puede efectuar la selección.

En los algoritmos genéticos de grano grueso, es común usar un **radio de selección** de **cero**, o sea, efectuar la selección sólo dentro del mismo **deme**. Evidentemente, ese no es el caso en los **algoritmos genéticos celulares**.

Sin embargo, en general, se suelen usar vecindarios compactos en los algoritmos genéticos paralelos, bajo el argumento de que tales vecindarios compactos existen también en la naturaleza.

Algoritmos Genéticos Paralelos

En los **algoritmos genéticos celulares** no hay evolución intra-islas, puesto que las células consisten de un solo individuo.

En este caso, sólo pueden producirse mejoras a través de las interacciones de una célula con las demás.

Los modelos de grano fino resultan adecuados para investigar la dinámica inter-islas, aunque evidentemente, el dinámica intra-islas ha sido removida.

Es posible actualizar todas las células de manera síncrona (a esto se le llama **algoritmo genético celular síncrono**. Una forma común de implementar este tipo de esquema es crear una población temporal nueva. Posteriormente, se toman todos los padres de la población actual y se escriben los individuos nuevos en la población temporal. Al final de proceso, la población actual es reemplazada por la población temporal.

Algoritmos Genéticos Paralelos

Alternativamente, las células pueden actualizarse secuencialmente, dando pie a un **algoritmo genético celular asíncrono**.

Esto producirá en general un comportamiento diferente de búsqueda de acuerdo a la forma en que los individuos se apareen con sus vecinos.

Alba et al. [2002] definirieron las siguientes estrategias de actualización para algoritmos genéticos celulares (se suele presuponer el uso de mallas bi-dimensionales para la topología:

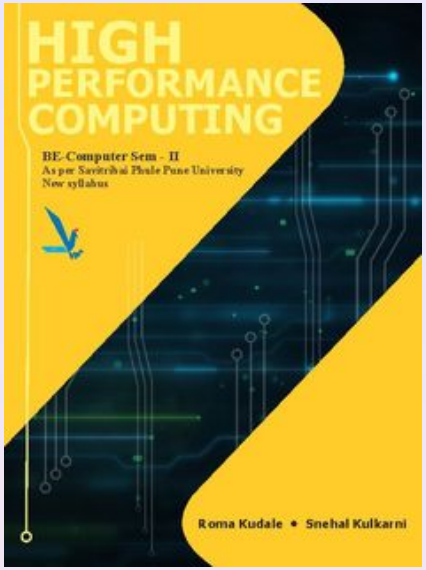
- **Elección Uniforme:** La siguiente célula a ser actualizada se elige de manera aleatoria (siguiendo una distribución uniforme).

E. Alba, M. Giacobini, M. Tomassini and S. Romero, "**Comparing Synchronous and Asynchronous Cullar Genetic Algorithms**", in *Parallel Problem Solving from Nature VII*, pp. 601–610, Springer, Berlin, 2002.

Algoritmos Genéticos Paralelos

- **Barrido Lineal Fijo:** Las células se actualizan secuencialmente, línea por línea.
- **Barrido Fijo Aleatorio:** Las células se actualizan secuencialmente, de acuerdo a algún orden fijo. Este orden se determina mediante la permutación de todas las células. Esta permutación se crea aleatoriamente (siguiendo una distribución uniforme) durante la inicialización y se mantienen durante toda la ejecución.
- **Nuevo Barrido Aleatorio:** Esta estrategia es similar al Barrido Fijo Aleatorio, pero en este caso, después de completar cada barrido, se crea una nueva permutación aleatoriamente (siguiendo una distribución aleatoria).

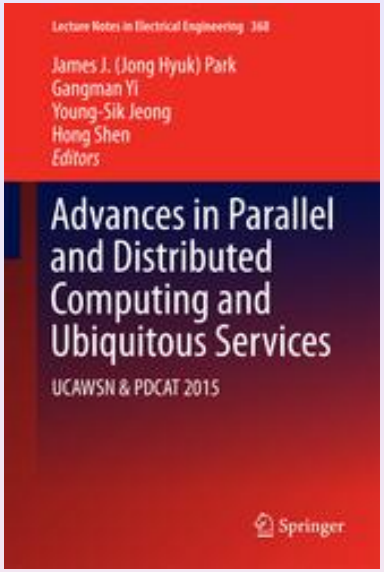
Algoritmos Genéticos Paralelos



Resulta difícil comparar de manera justa a un algoritmo genético paralelo de grano fino con uno de grano grueso, y los pocos estudios al respecto suelen enfatizar sólo una cierta métrica (por ejemplo, la calidad de las soluciones encontradas).

De tal forma, no hay un claro ganador entre estos dos esquemas.

Algoritmos Genéticos Paralelos

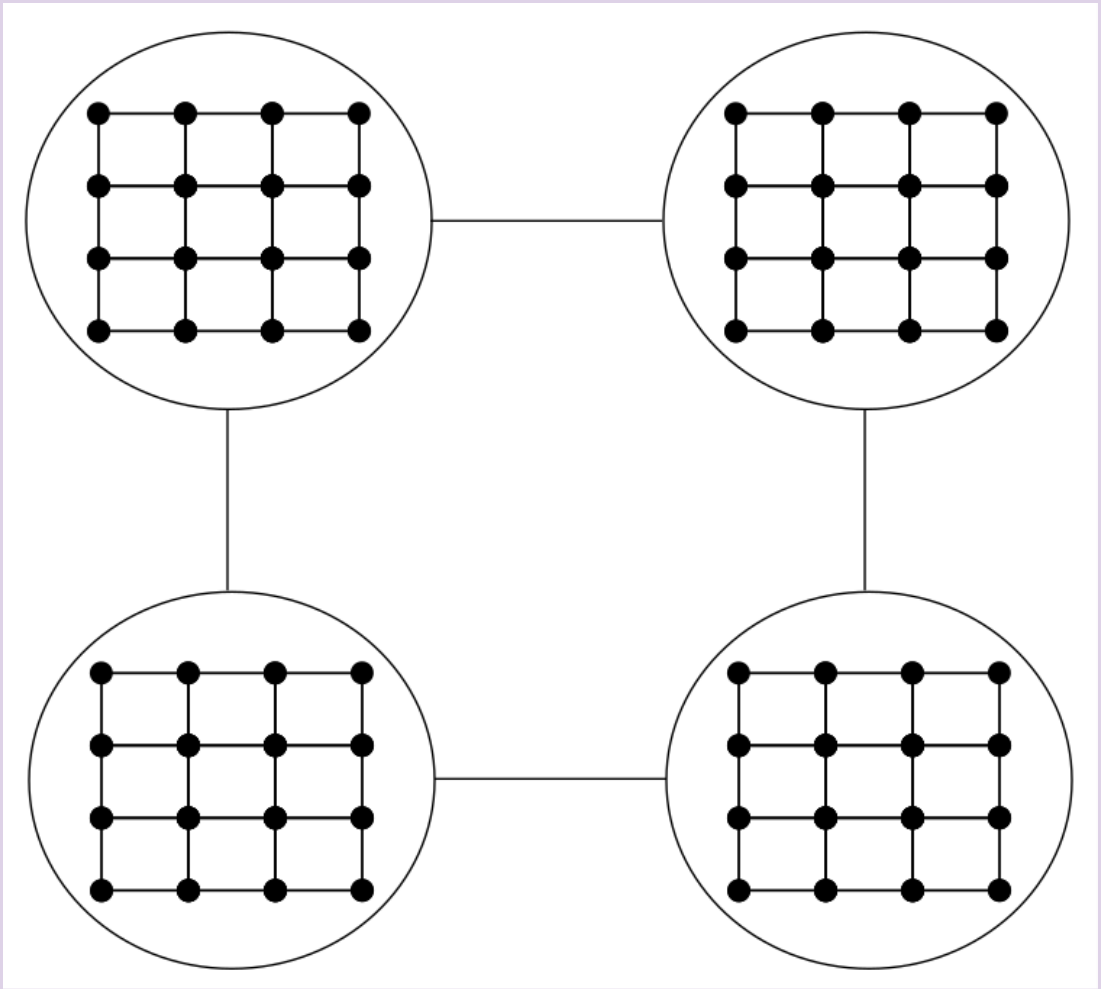


Otra posibilidad para implementar un algoritmo genético paralelo es combinar los esquemas descritos anteriormente. Debe cuidarse, sin embargo, de que el esquema resultante no sea más complejo que los esquemas originales.

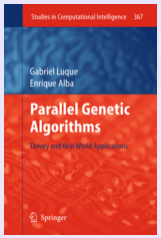
A continuación veremos varios híbridos posibles.

Algoritmos Genéticos Paralelos

Un posible híbrido consiste en usar un algoritmo genético de grano fino a bajo nivel y otro de grano grueso a alto nivel.



Algoritmos Genéticos Paralelos



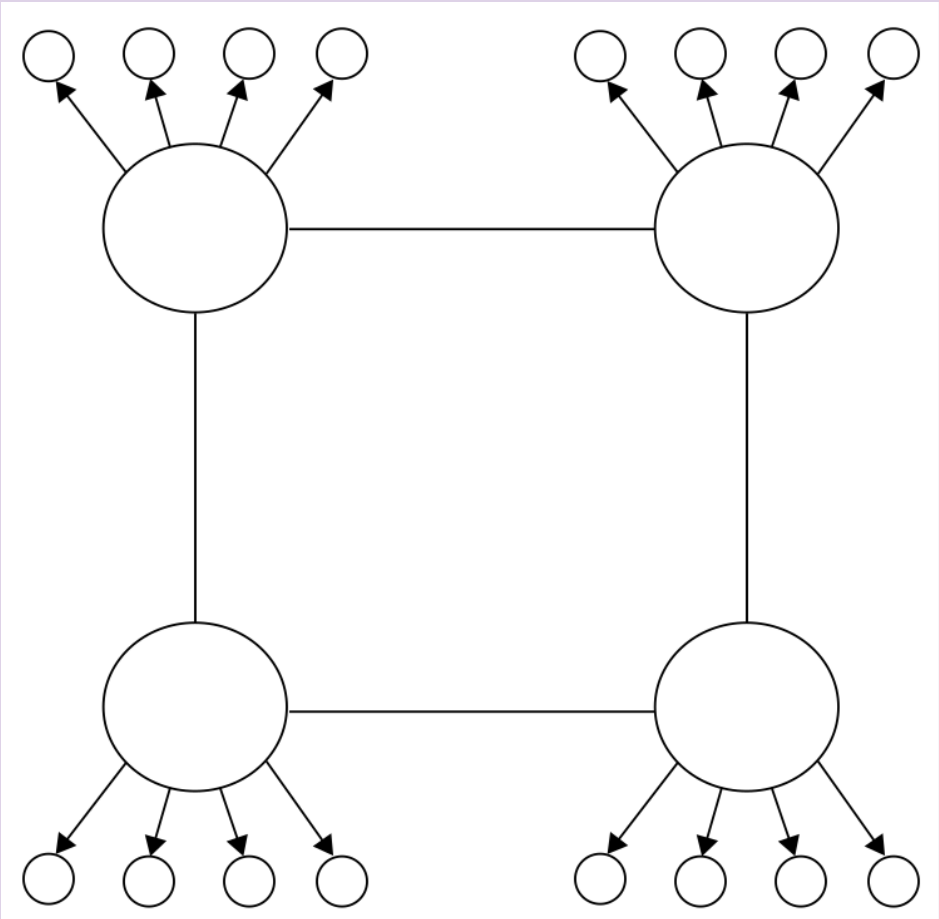
Un ejemplo de este tipo de híbrido es el algoritmo genético propuesto por Gruau [1994], en el cual la población de cada **deme** se coloca en una malla bidimensional y los **demes** se conectan entre sí en forma de toroide bidimensional.

La migración entre los **demes** vecinos ocurre a intervalos regulares.

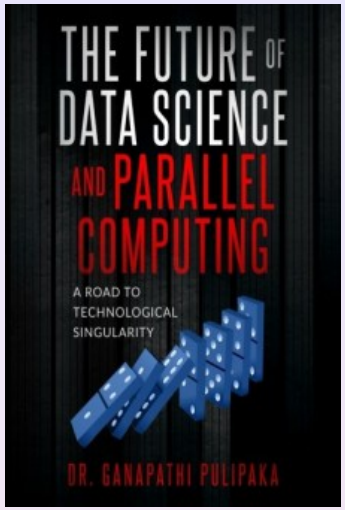
F. Gruau, "**Neural Network Synthesis Using Cellular Encoding and the Genetic Algorithm**", PhD thesis, L'Université Claude Bernard-Lyon I, France, 1994.

Algoritmos Genéticos Paralelos

Otro posible esquema híbrido consiste en usar una forma de paralelización global en cada uno de los **demes** de un algoritmo genético de grano grueso.



Algoritmos Genéticos Paralelos

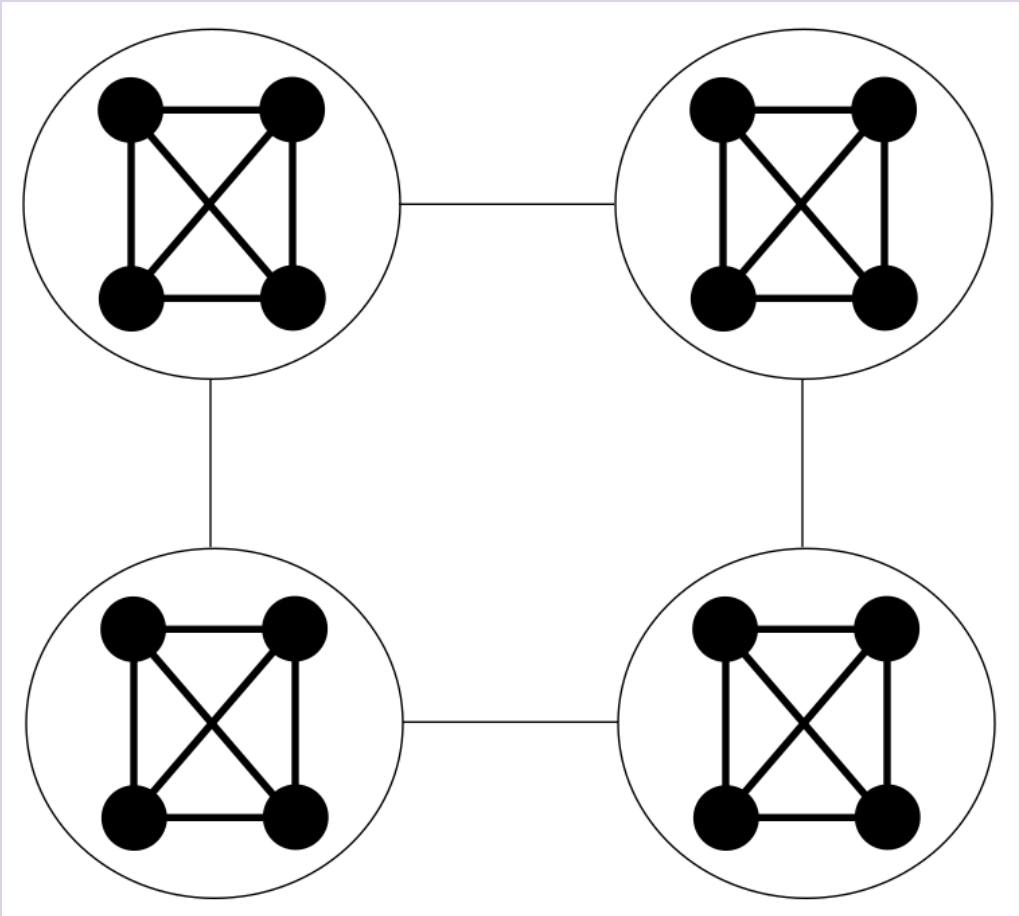


En este caso, la migración ocurre entre los **demes** de manera similar a un algoritmo genético de grano grueso, pero la evaluación de los individuos se maneja en paralelo.

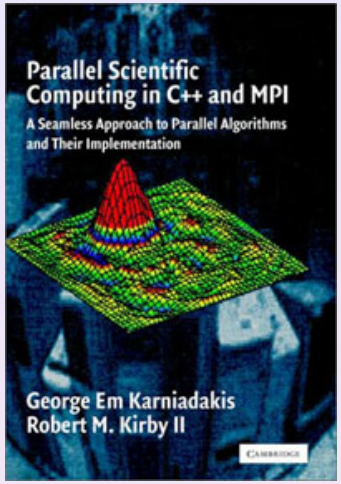
Esta técnica no introduce nuevos problemas analíticos, y puede ser muy útil cuando se trabaja con aplicaciones en las cuales la mayor parte del tiempo de procesamiento lo consume la evaluación de la función de aptitud.

Algoritmos Genéticos Paralelos

Un tercer método híbrido podría consistir en usar un algoritmo genético de grano grueso tanto a bajo como a alto nivel.



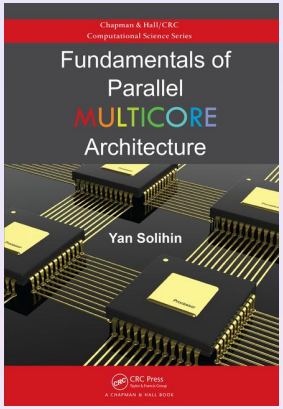
Algoritmos Genéticos Paralelos



En este caso, la idea es forzar el mezclado panmítico a bajo nivel usando una alta tasa de migración y una topología densa, y usar una baja tasa de migración a alto nivel.

Este híbrido sería equivalente en complejidad a un algoritmo genético de grano grueso, si consideramos a los grupos de subpoblaciones panmíticas como un solo **deme**.

Algoritmos Genéticos Paralelos

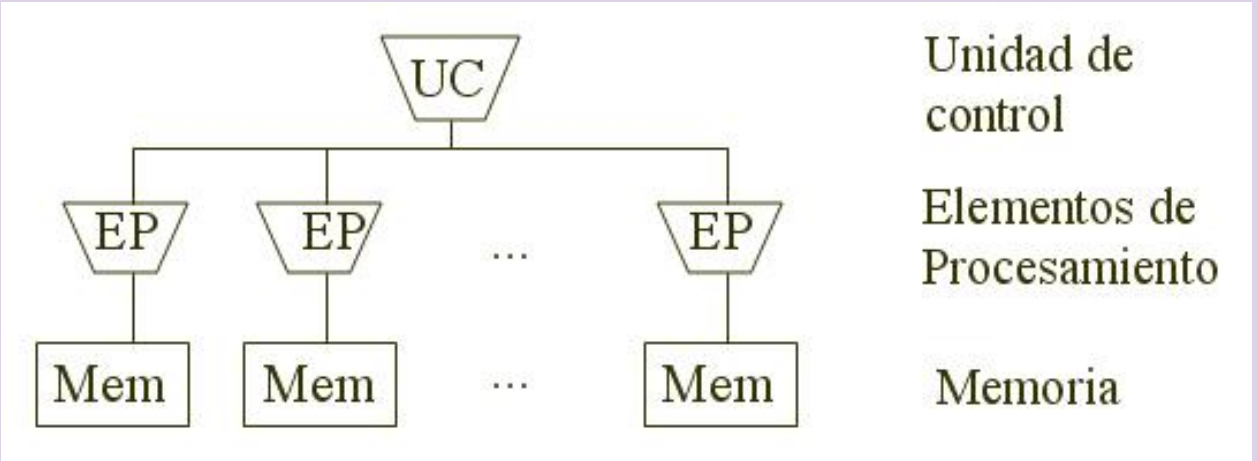


Otra forma de hablar sobre algoritmos genéticos paralelos, es desde la perspectiva del tipo de arquitectura computacional a utilizarse.

Desde este punto de vista, podemos hablar fundamentalmente de usar:

1. SIMD
2. MIMD

Un ejemplo de arquitectura SIMD es el siguiente:



En este caso, cada elemento de procesamiento (EP) tiene su propia memoria y controla su propio espacio de direccionamiento, aunque también podría haber una sola memoria global compartida por todos los elementos de procesamiento.

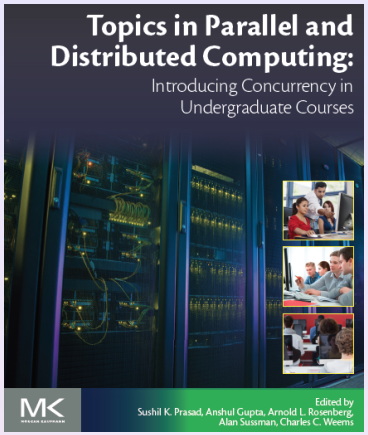
Algoritmos Genéticos Paralelos

Las arquitecturas SIMD normalmente tienen una forma de malla (*mesh*) o de toroide.

Esta arquitectura (SIMD) suele usarse para algoritmos genéticos de grano fino (o sea, para **demes** de tamaño reducido: 12 a 15 individuos cada uno).

Sin embargo, puede implementarse un AG de grano grueso con la misma arquitectura si se usan tamaños mayores de **demes** (p.ej. 50 ó 100 individuos).

Algoritmos Genéticos Paralelos

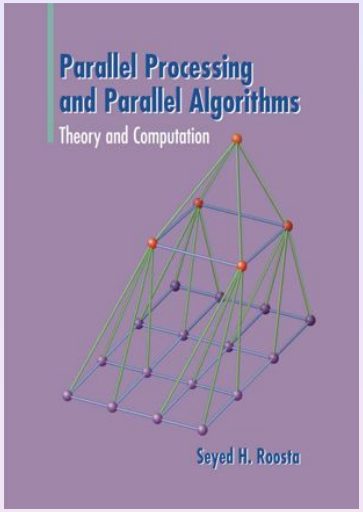


El grado de conectividad es mucho más importante que la estructura de las interconexiones en una arquitectura SIMD.

Se sabe que un grado de conectividad de alrededor de 6 es razonablemente bueno.

El uso de una arquitectura SIMD está asociado con el uso de **migración** y dicho operador puede complicarse bastante, dependiendo del grado de conectividad de la arquitectura

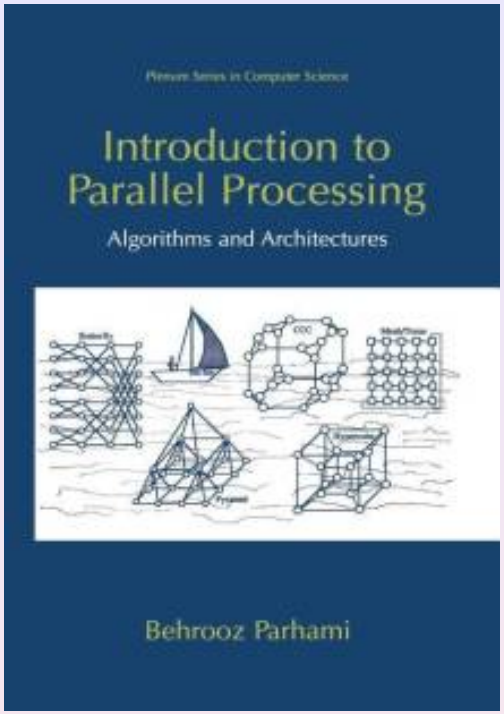
Algoritmos Genéticos Paralelos



Dos son los parámetros principales relacionados con la **migración**:

1. **Vecindario de migración**: hacia qué demes podemos migrar un individuo.
2. **Probabilidad de migración**: ¿cuál es la probabilidad de aceptar a un migrante en un cierto **deme** (suelen usarse valores altos, p.ej. 0.8)?

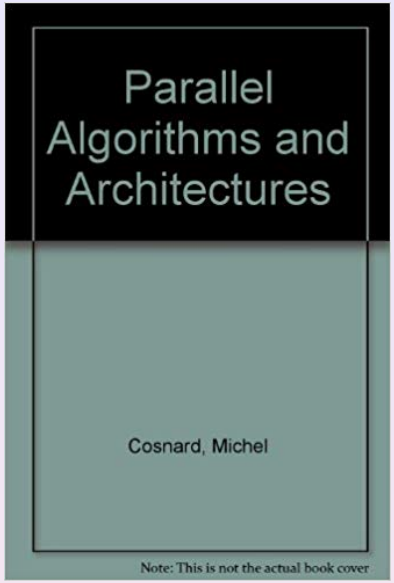
Algoritmos Genéticos Paralelos



Los puntos importantes relacionados con la migración son dos:

1. ¿A quién importar en una población?
2. ¿A quién reemplazar en una población?

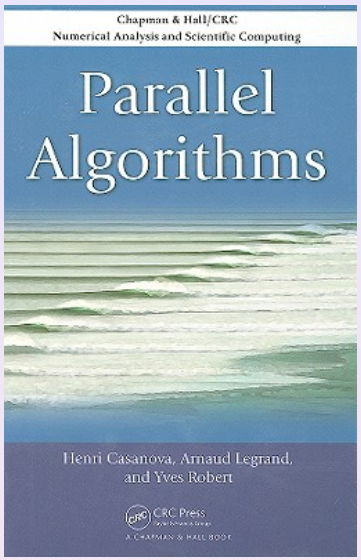
Algoritmos Genéticos Paralelos



Existen varios criterios para llevar a cabo estas 2 operaciones:

- Importar al azar y reemplazar al azar
- Importar al azar y reemplazar al peor individuo en el **deme**
- Importar el mejor y reemplazar al azar
- Importar el mejor y reemplazar el peor

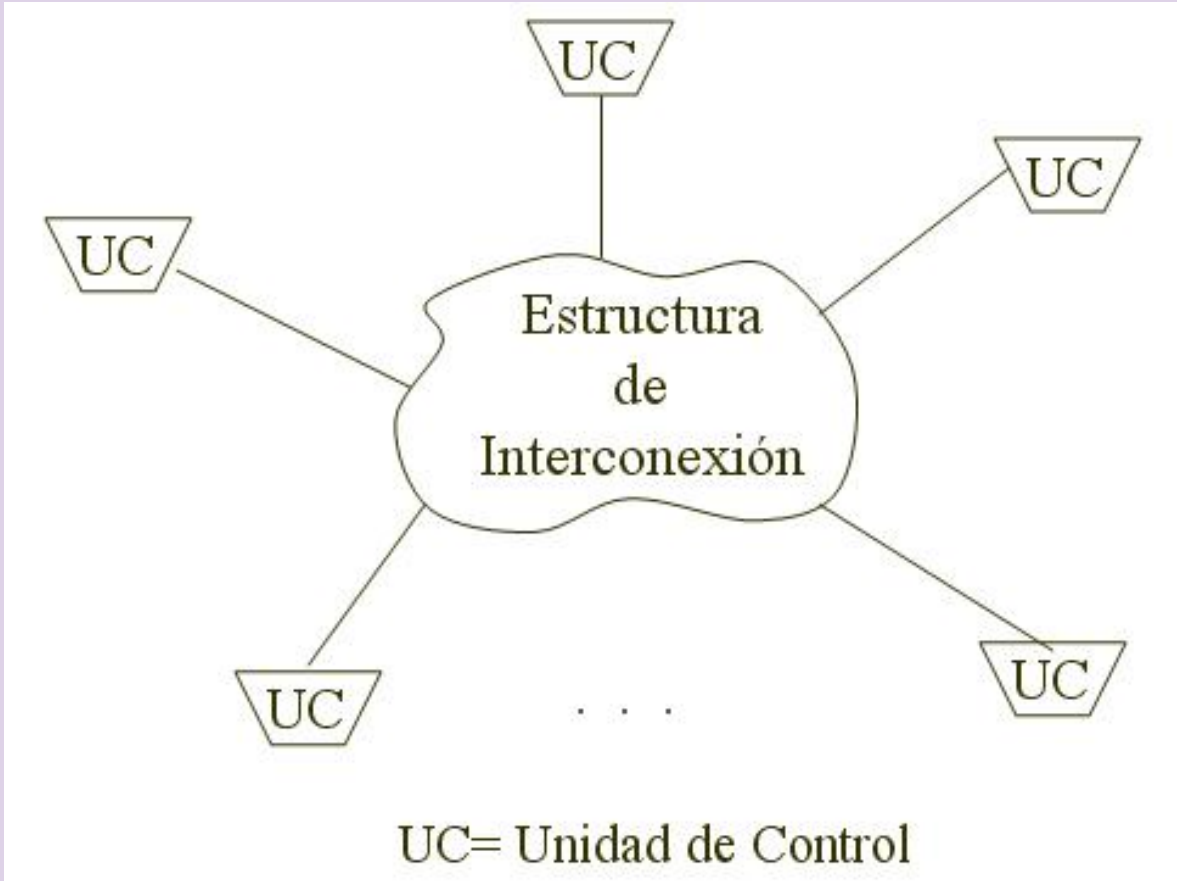
Algoritmos Genéticos Paralelos



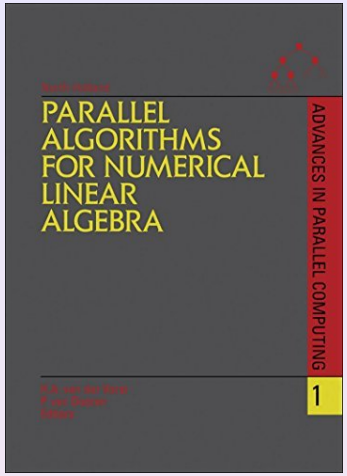
Se sabe que la política de reemplazo no es muy importante (no parece tener un efecto significativo en el desempeño de un algoritmo genético paralelo).

Sin embargo, el criterio de importación sí es importante (importar al mejor parece funcionar bien).

Por otro lado, tenemos las arquitecturas MIMD:



Algoritmos Genéticos Paralelos

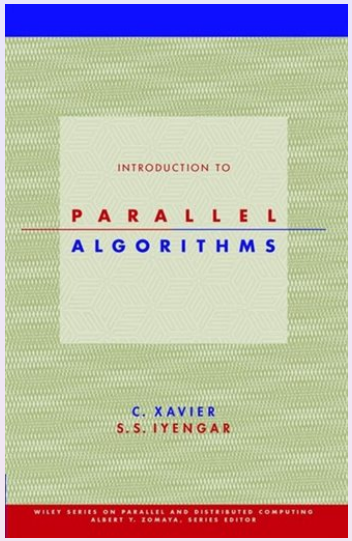


Las arquitecturas MIMD pueden ser de 2 tipos:

1. **Descentralizadas**: tienen poca o ninguna memoria global.
2. **Centralizadas**: cuentan con una memoria global compartida.

Las arquitecturas MIMD suelen asociarse con los algoritmos genéticos de grano grueso.

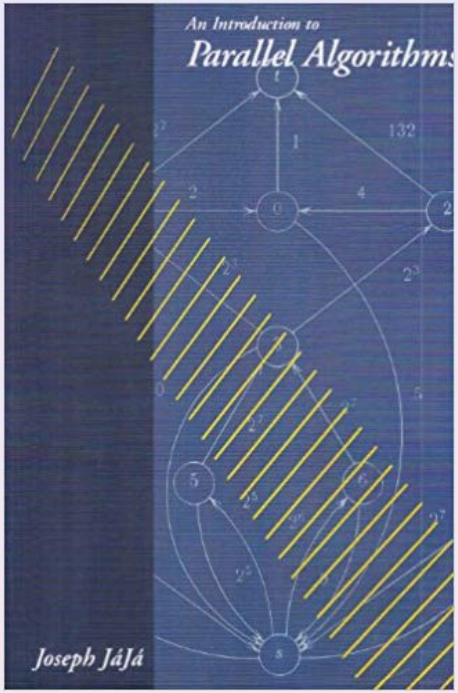
Algoritmos Genéticos Paralelos



En la arquitectura MIMD suele tenerse un número pequeño de **demes** (normalmente, menor a 40), pero el tamaño de cada uno de ellos suele ser grande.

Es posible usar la misma representación para cada **deme**, o mezclar diferentes representaciones.

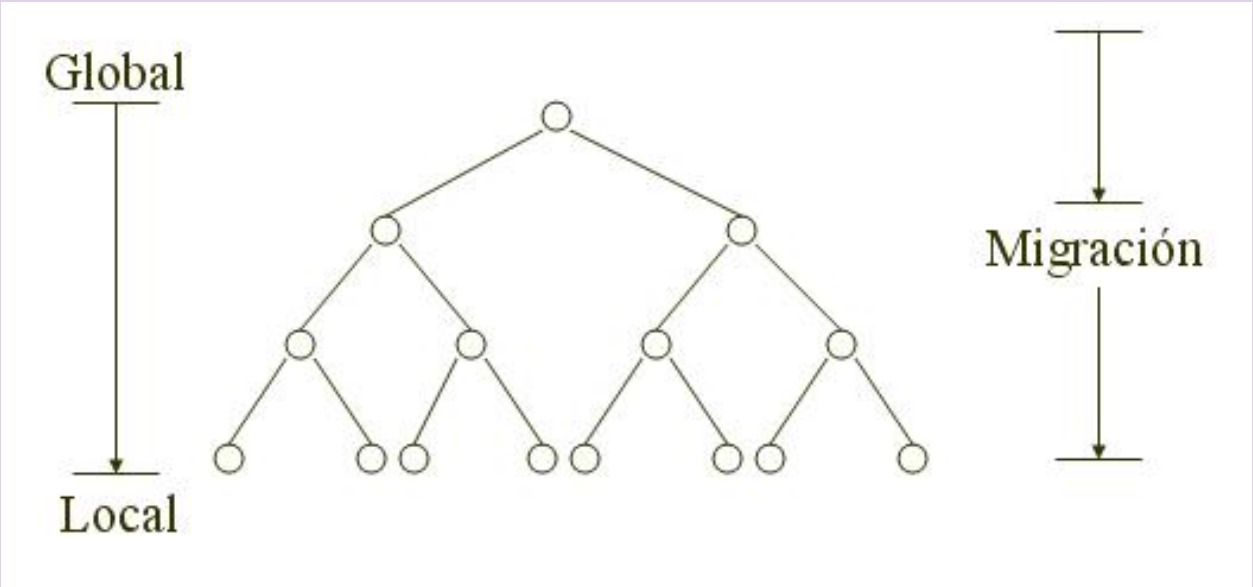
Algoritmos Genéticos Paralelos



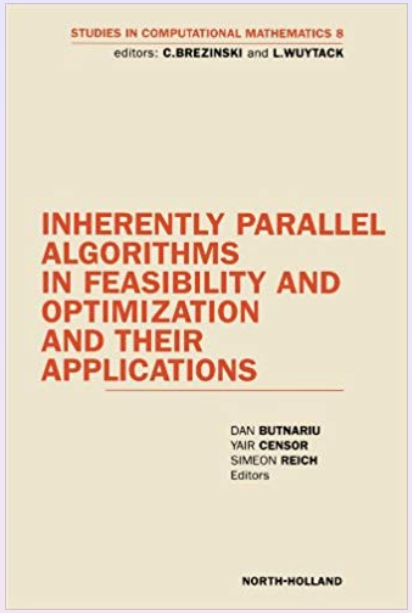
Los **demes** suelen diferenciarse debido al particionamiento del espacio de búsqueda.

Las políticas de migración, en este caso, están dictadas por el propósito de los **demes**.

Una topología que suele usarse con las arquitecturas MIMD es la de árbol, como se ilustra en esta figura:

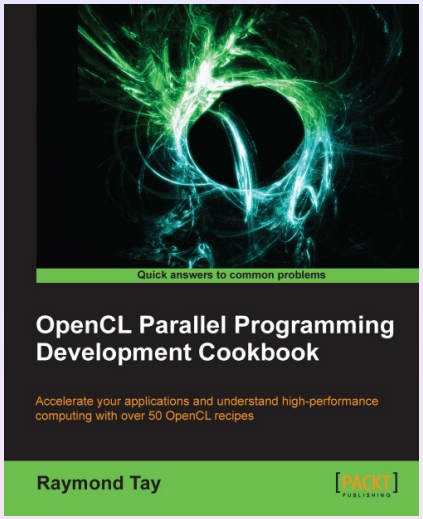


Algoritmos Genéticos Paralelos



La migración en este caso introduce 2 nuevos parámetros:

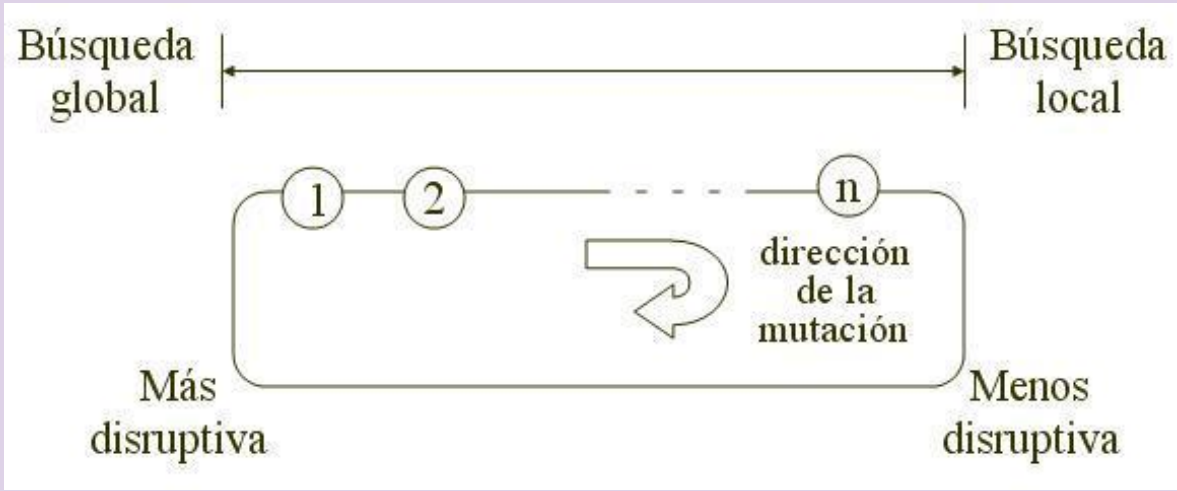
1. ¿Con qué frecuencia exportar? (siempre se exporta el mejor). Si se hace con mucha frecuencia, se produce disrupción. Si se hace con poca frecuencia, habrá poca recombinación y puede producirse convergencia prematura en ciertos **demes**.



2. ¿A qué **deme** exportar? Normalmente se usa una de las 2 siguientes opciones:
   - Exportar el mejor individuo hacia el peor **deme**.
   - Exportar hacia el **deme** donde se tenga una mejor correspondencia (*matching*) con respecto al individuo elitista, medida usando la distancia de *Hamming* (en el genotipo).

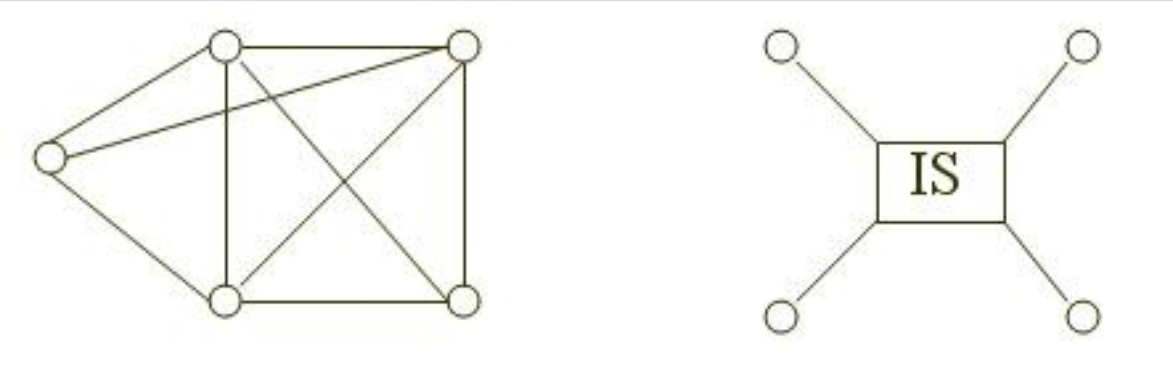
Algoritmos Genéticos Paralelos

Otra posibilidad para una arquitectura MIMD es usar una topología de anillo:



En este tipo de topología, la búsqueda puede hacerse más local cambiando la precisión de la representación.

Otra posibilidad es usar una topología de grafo con k interconexiones, o una topología de estrella:



En esta topología, se usan típicamente menos de 100 procesadores.

Algoritmos Genéticos Paralelos

Algunas opciones interesantes de la topología de grafo con k interconexiones (o de estrella) son las siguientes:

1. Mientras que la arquitectura sea MIMD, pueden usarse la misma estructura y las mismas opciones para migración que con la arquitectura SIMD.
2) **Pizarrones**: Usando datos globales, cada **deme** decide por sí mismo cuándo cambiar su "dirección" de búsqueda.

Algoritmos Genéticos Paralelos

Un punto interesante relacionado con los algoritmos genéticos paralelos es el de las medidas de desempeño.

El tiempo de cómputo consumido por un algoritmo genético paralelo se puede definir de varias formas.

Tiene sentido usar el tiempo del reloj, pues que éste considera los costos involucrados en la paralelización.

Bajo ciertas condiciones, es posible usar también el número de generaciones o el número de evaluaciones. Esto es factible si estas medidas reflejan el tiempo real de ejecución de una manera apropriada.

Algoritmos Genéticos Paralelos

Comparar la ejecución de un algoritmo genético paralelo con uno secuencial tiene también sus complicaciones. Como Alba [2002] indica, esta comparación sólo tiene sentido cuando ambos algoritmos alcanzan una solución con la misma precisión.

Definir **aceleración** (*speedup*) para un algoritmo genético paralelo es mucho más difícil que hacerlo para un algoritmo paralelo determinista.

A principios de siglo, hubieron diversos investigadores que aseguraban obtener aceleraciones super-lineales al implementar algoritmos genéticos paralelos.

E. Alba, "**Parallel Evolutionary Algorithms can Achieve Super-Linear Performance**", *Information Processing Letters*, Vol. 82, No. 1, pp. 7–13, 2002.

Algoritmos Genéticos Paralelos

Pronto se descubrió a que esto se debía, principalmente, a que las poblaciones usadas por estos algoritmos genéticos paralelos eran suficientemente pequeñas como para caber en la memoria caché de los procesadores, lo cual causaba estas aceleraciones super-lineales.

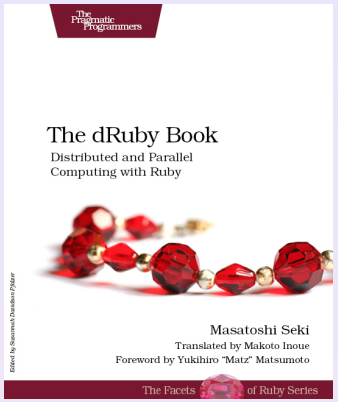
Alba [2002] define diferentes tipos de **aceleración** en los algoritmos genéticos paralelos:

- **Aceleración Fuerte:** El tiempo de ejecución de un algoritmo paralelo se compara contra el tiempo de ejecución del *mejor algoritmo secuencial conocido*. Esto se denomina *aceleración absoluta* [Barr and Hickman, 1993]. Como, en general, es difícil encontrar el mejor algoritmo secuencial, esta medida rara vez se usa en la práctica [Alba, 2002].

R.S. Barr and B.L. Hickman, "**Reporting computational experiments with parallel algorithms: Issues, measures, and experts' opinion**", *ORSA Journal on Computing*, Vol. 5, No. 1, pp. 2–18, 1993.

- **Aceleración Débil:** La aceleración de un algoritmo paralelo se compara con respecto a su *mejor tiempo de ejecución secuencial*. Esto da pie a dos versiones donde es posible precisar la noción de *su mejor ejecución secuencial*:
  1. **Una sola máquina/panmixia:** El algoritmo genético paralelo se compara contra una versión canónica, panmítica de sí mismo, ejecutándose en una computadora secuencial.
  2. **Ortodoxa:** El algoritmo genético paralelo ejecutándose en *m* máquinas se compara contra el mismo algoritmo genético paralelo ejecutado en una sola máquina. Barr y Hickman [1993] llamaron a esto *aceleración relativa*.

Algoritmos Genéticos Paralelos



Existen otras medidas de desempeño que se han utilizado con los algoritmos genétcos paralelos:

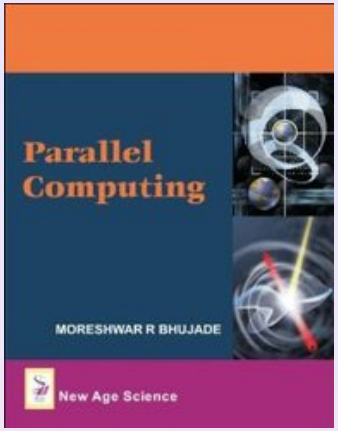
- **Velocidad de convergencia**: Tiempo (generaciones) en alcanzar el óptimo.
- **Precisión de la respuesta obtenida**: ¿Qué tan buena es la solución con respecto a la obtenida con otras técnicas?
- **Diversidad**: El grado en el cual los organismos (de una sola o de varias poblaciones) permanecen diferentes.
- **Velocidad de propagación de los esquemas**: ¿qué tan bien distribuidos están los esquemas “aptos”? O sea, ¿qué tan útil resulta la migración?

Existen expresiones estándar para medir la diversidad de un AG (serial o paralelo). Consideremos por ejemplo la siguiente:

$$\delta = \frac{1}{l} \times \sum_{i=1}^{l} \left(1 - 2 \times \left| \frac{\sum_{j=1}^{m} \sum_{k=1}^{n} bit(i,k,j)}{m \times n} \right| \right)$$

l = Longitud cromosómica
m = número de **demes**
n = tamaño de cada **deme**
bit $(\cdot)$ = Valor del i-ésimo bit en el k-ésimo miembro del j-ésimo **deme**
δ = diversidad

Algoritmos Genéticos Paralelos



En la fórmula del acetato anterior, $\delta \in [0, 1]$ representa la diversidad de una población (o conjunto de poblaciones).

Si las cadenas consisten de puros ceros, $\delta = 0$

Si las cadenas consisten de puros unos, $\delta = 0$

Si las cadenas son del tipo 101010...10, $\delta = 0$

Algoritmos Genéticos Paralelos

¿A qué se refiere la velocidad de propagación de esquemas?

Se refiere no sólo al porcentaje de **demes** en los que un cierto esquema está presente, sino también al porcentaje en el cual dicho esquema está presente en un **deme** vecino.

Idealmente, deberíamos conocer de antemano cuáles son los buenos esquemas.

Esto, sin embargo, es imposible en la práctica.

Una alternativa viable es:

- Escoger varios esquemas de antemano.
- Hacer que la propagación de esquemas sea la fracción máxima de **demes** en la cual aparece un esquema.

Veamos un ejemplo del cálculo de la propagación de esquemas (SP):

Esquemas seleccionados	% de **demes** en los que aparecen
*1*10*	3/9
*110**	4/9
*10*0*	5/9

En este caso: SP = 5/9

Algoritmos Genéticos Paralelos

En años recientes, el uso de *Graphics Processing Units* (GPUs) para acelerar el procesamiento de algoritmos evolutivos muy costosos, se ha vuelto relativamente popular.

Las GPUs se han usado mucho, por ejemplo, en programación genética.

W.B. Langdon, "**Graphics processing units and genetic programming: an overview**", *Soft Computing*, Vol. 15, No. 8, pp. 1657–1669, August 2011.

Algoritmos Genéticos Paralelos

También se han usado *Field Programmable Gate Arrays* (FPGAs) para acelerar la ejecución de algoritmos evolutivos computacionalmente costosos.

Pei-Yin Chen, Ren-Der Chen, Yu-Pin Chang, Leang-San Shieh and Heidar A. Malki, "**Hardware implementation for a genetic algorithm**", *IEEE Transactions on Instrumentation and Measurement*, Vol. 57, No. 4, pp. 699–705, April 2008.

Aplicaciones Exitosas de los Algoritmos Genéticos

Un equipo de **Unilever Research** usó algoritmos genéticos combinados con redes neuronales para diseñar nuevos péptidos bactericidas para usarse en limpiadores anti-bacterianos y preservativos de alimentos.

Las redes neuronales se utilizaron para predecir la actividad bactericida en los péptidos, y posteriormente se combinaron con algoritmos genéticos para optimizar péptidos virtuales.

El resultado fue la generación de más de 400 bactericidas virtuales potencialmente activos, de los cuales 5 fueron sintetizados.

Aplicaciones Exitosas de los Algoritmos Genéticos

La empresa de software escocesa **Quadstone**, usó algoritmos genéticos para resolver un problema de optimización de estrategias de producción de *British Petrol*.

El objetivo era maximizar el retorno financiero de un grupo de campos petrolíferos y de gas interdependientes.

El problema es bastante complejo debido a los muchos compromisos posibles entre beneficios y penalizaciones.

Aplicaciones Exitosas de los Algoritmos Genéticos

En este problema, aún una mejora relativamente pequeña puede traer enormes ahorros a la larga, ya que su impacto es acumulativo.

El uso de algoritmos genéticos en este problema produjo retornos netos substancialmente mejores que los producidos previamente por cualquier planeador humano o por cualquier otra técnica de optimización.

Aplicaciones Exitosas de los Algoritmos Genéticos

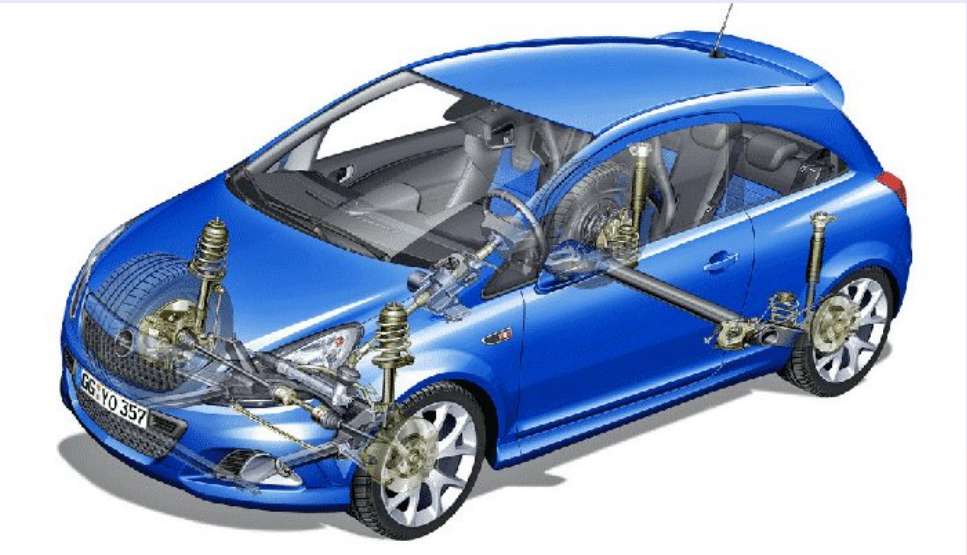
La empresa holandesa **Cap Gemini** y la empresa británica **KiQ Ltd** han desarrollado de forma conjunta un sistema llamado **Omega**, el cual usa algoritmos genéticos para resolver problemas de mercadotecnia, crédito y aplicaciones financieras relacionadas.

Omega usa como punto de partida un portafolio de comportamiento previo de un cliente, ya partir de él genera un modelo matemático que puede usarse posteriormente para predecir comportamientos de clientes que se encuentren fuera de los portafolios conocidos.

Aplicaciones Exitosas de los Algoritmos Genéticos

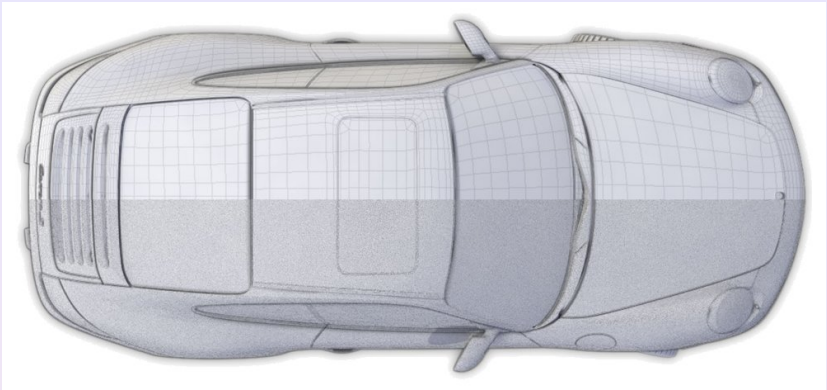
Un banco holandés comparó **Omega** contra su sistema de asignación de crédito tradicional (basado en sistemas expertos), encontrando que Omega era substancialmente superior tanto en cantidad (ofrecía más préstamos) como en calidad (se reducía el riesgo en los créditos) de los préstamos evaluados.

Aplicaciones Exitosas de los Algoritmos Genéticos



Investigadores del *Kanpur Genetic Algorithms Laboratory* (**KanGAL**), en la India, han utilizado algoritmos genéticos para diseñar un sistema de suspensión para un automóvil que es más cómodo que el previamente utilizado por una empresa automotriz reconocida mundialmente.

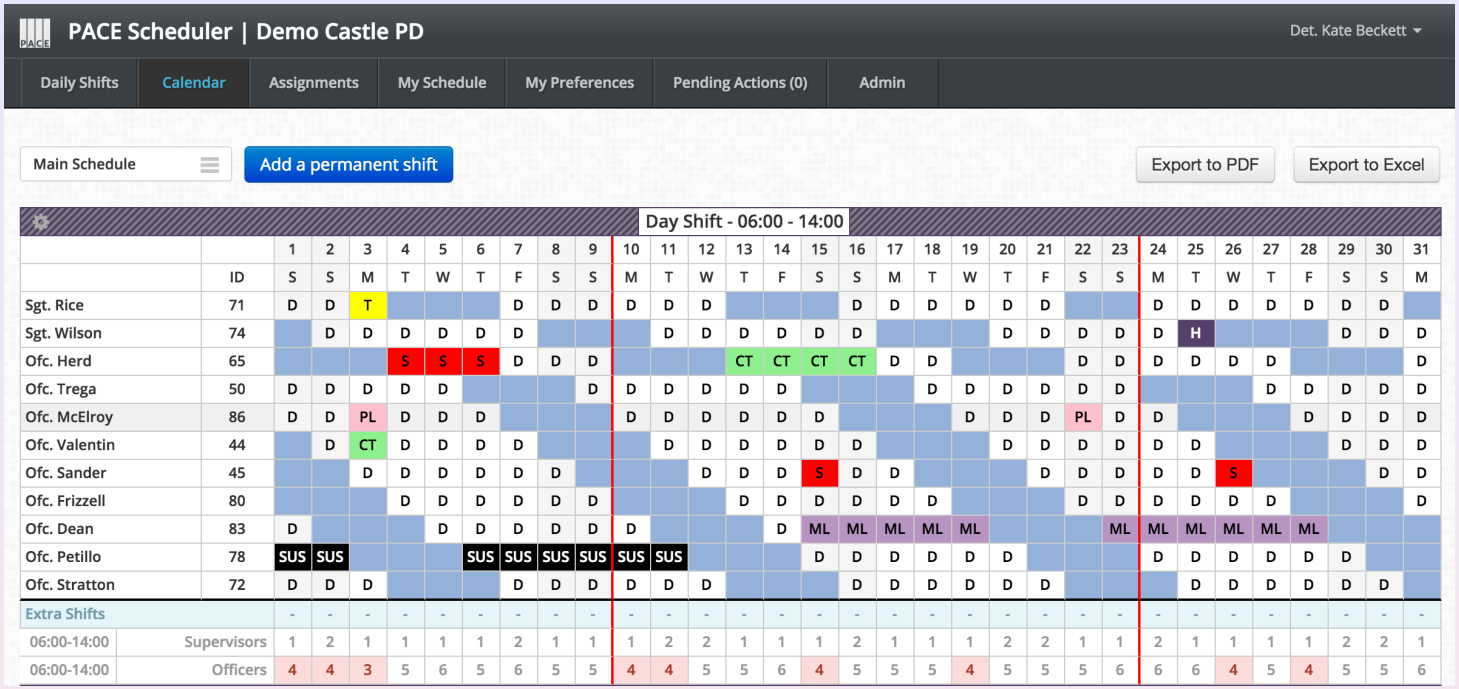
Aplicaciones Exitosas de los Algoritmos Genéticos



Utilizando un modelo tridimensional de un automóvil, optimizaron el diseño de los amortiguadores y los resortes de rigidez del vehículo.

Las simulaciones efectuadas mostraron que el sistema generado por el algoritmo genético hace que los pasajeros sufran menor aceleración vertical, de manera que se disfruta de un mayor confort que con los sistemas utilizados previamente por el fabricante de automóviles en cuestión.

Aplicaciones Exitosas de los Algoritmos Genéticos



Un grupo de investigadores del **Jožef Stefan Institute**, en Eslovenia, desarrollaron un sistema de programación de horarios basado en técnicas evolutivas.

El sistema ha reducido sustancialmente los costos de energía en la planta de prensado de una fábrica de automóviles.

Aplicaciones Exitosas de los Algoritmos Genéticos

El sistema utiliza una heurística “avariciosa” (*greedy*) para diseñar horarios iniciales que luego son utilizados como punto de partida por una técnica evolutiva que minimiza el consumo de energía durante las horas pico.

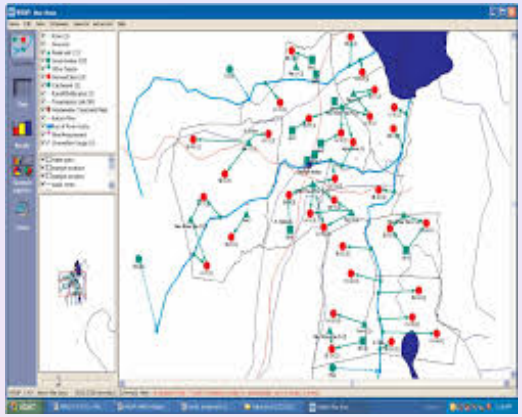
Aplicaciones Exitosas de los Algoritmos Genéticos

Optimización de una Red Hidráulica

Descripción del Problema:

- Extender y reforzar la infraestructura de suministro de agua para la región de York, en Ontario, Canadá.
- La población de la región se duplicará en el período de 1996 a 2031.

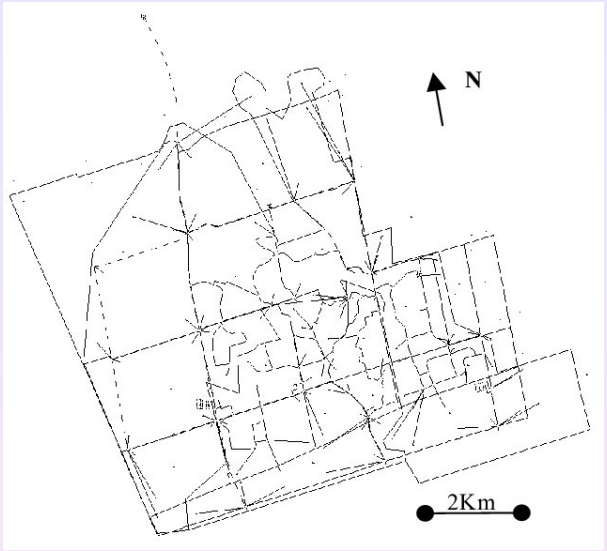
Aplicaciones Exitosas de los Algoritmos Genéticos



Complejidad del Problema: Para la nueva red se requieren 300 nuevos tipos de tubos, cada uno de los cuales puede adoptar uno de 14 diámetros comerciales disponibles (de 300 mm a 2100 mm).

Combinando las demás variantes del problema (p.ej. ubicación de las estaciones de bombeo, etc.) se estimó que el tamaño del espacio de búsqueda era de aproximadamente 10^{357}.

Aplicaciones Exitosas de los Algoritmos Genéticos



Análisis Previo:

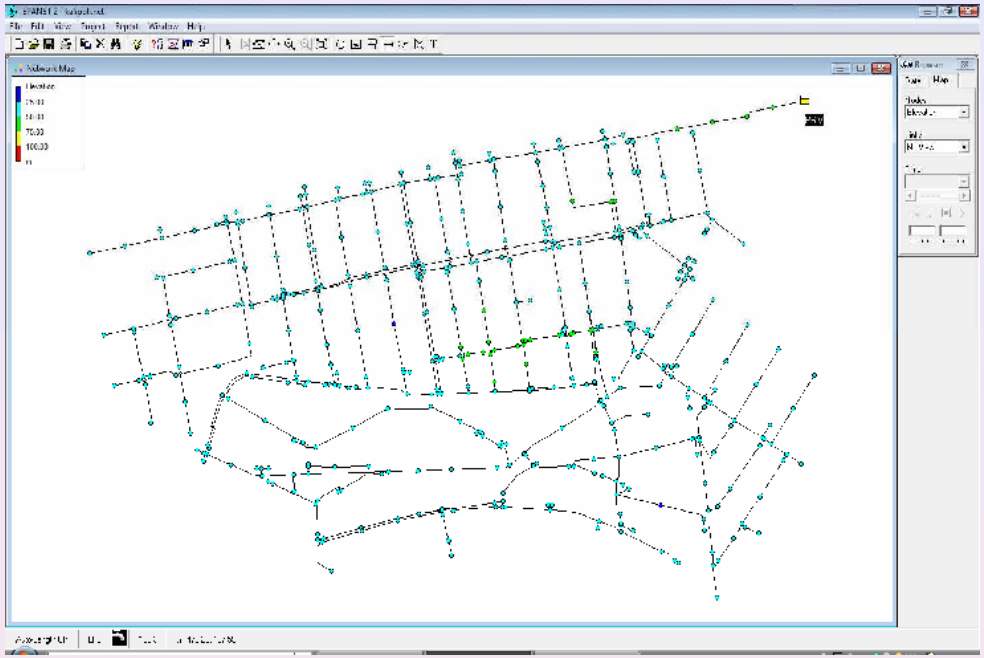
- Estudios de campo extensivos de la red y de las estaciones de bombeo.
- Se construyó un modelo de todos los cauces principales de la región, usando **StruMap**, que es un sistema de información geográfica que tiene un módulo integrado para resolver redes hidráulicas.

Aplicaciones Exitosas de los Algoritmos Genéticos

Análisis realizado por humanos:

- Sólo disponible hasta el año 2011, se extrapoló para el 2031.
- Costo estimado: $156 millones de dólares

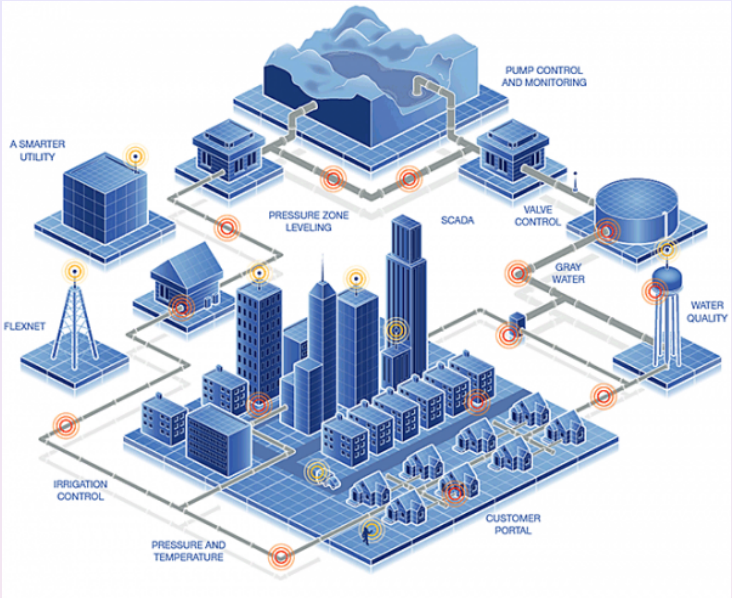
Aplicaciones Exitosas de los Algoritmos Genéticos



Uso de Computación Evolutiva:

- **GAnet**: una biblioteca de clases para desarrollar algoritmos genéticos.
- Incluye rutinas para simular redes hidráulicas.
- Maneja restricciones duras y blandas.

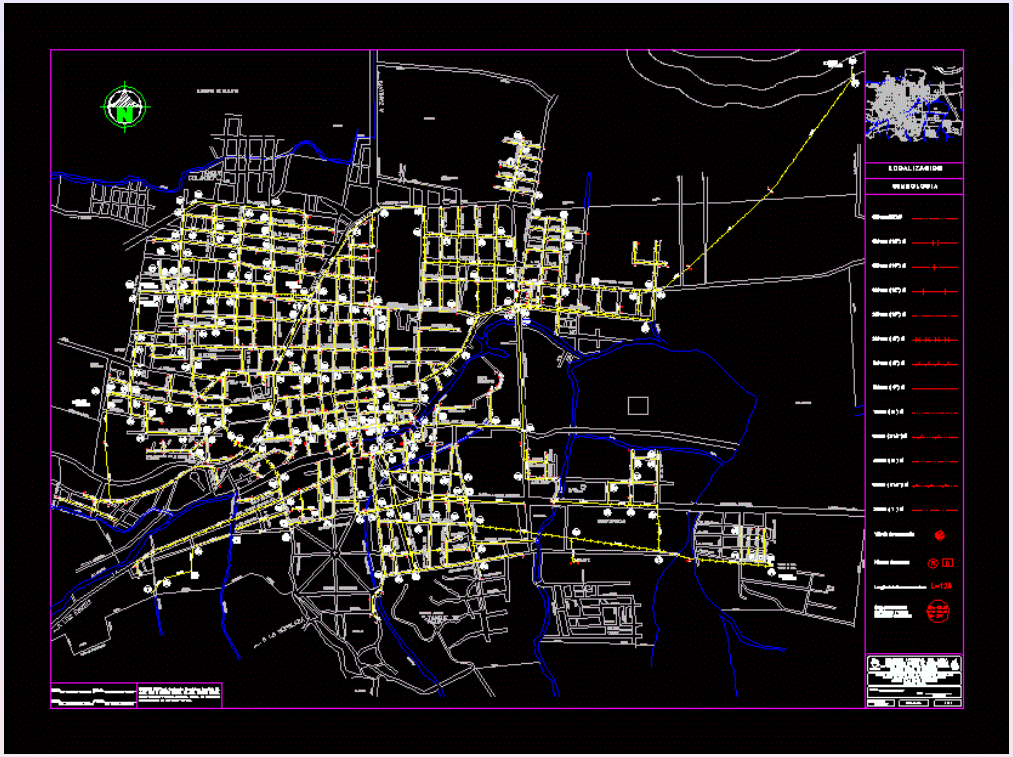
Aplicaciones Exitosas de los Algoritmos Genéticos



Resultados de GAnet:

- Agregar 85 tuberías principales a las 750 ya existentes.
- Se propusieron 6 nuevas estaciones de bombeo y se sugirió expandir 3 de las ya existentes, totalizando 42 nuevas bombas.

Aplicaciones Exitosas de los Algoritmos Genéticos



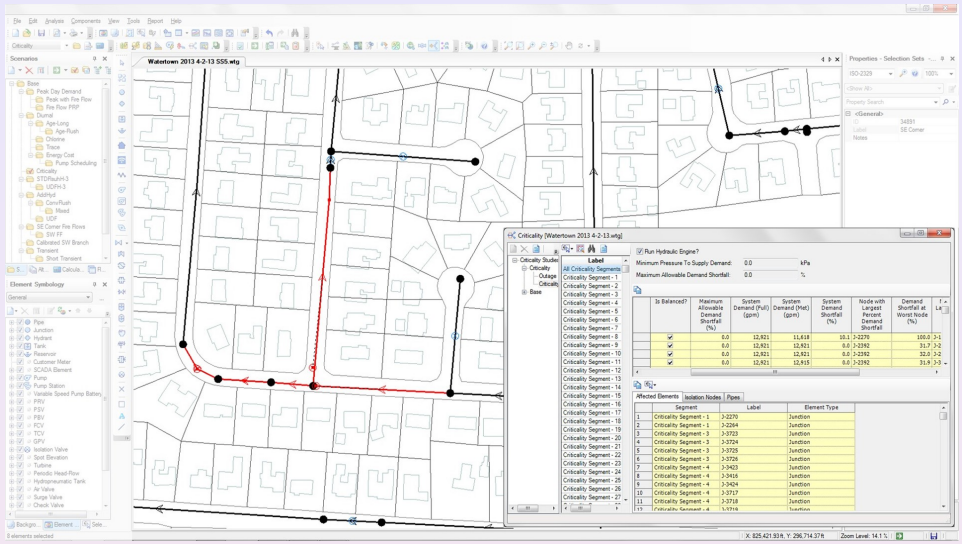
- Se propuso sacar de circulación a 3 estaciones de bombeo.
- Se propusieron 7 nuevos tanques elevados y se sugirió sacar de circulación a 2 de los existentes.

Aplicaciones Exitosas de los Algoritmos Genéticos

Resultados de GAnet:

- Costo: $102 millones de dólares.
- Solución 35% más económica que la propuesta por diseñadores humanos.
- Ahorro estimado: $54 millones de dólares.

Aplicaciones Exitosas de los Algoritmos Genéticos



GAnet es uno de los 3 productos principales de software de la empresa *Optimal Solutions*.
Los otros 2 son:

1) **GAser**: *Genetic Algorithms Sewer Analysis*, que se usa para optimizar redes de alcantarillado.
2) **GAcal**: Herramienta para calibrar modelos hidráulicos.

Aplicaciones Exitosas de los Algoritmos Genéticos

La empresa **Optimal Solutions** fue fundada en 1996: http://www.ewan.co.uk/os.html

Esta empresa surgió a partir de la investigación del Dr. Dragan Savic y sus colaboradores en la Universidad de Exeter (en Inglaterra).

Aplicaciones Exitosas de los Algoritmos Genéticos

Torsten Reil, un investigador de Oxford (en Inglaterra), utilizó un algoritmo genético para realizar animaciones de una figura humana que aprende por sí sola a caminar.

Aplicaciones Exitosas de los Algoritmos Genéticos

Reil desarrolló un modelo corporal simple (figuras sólidas con caras poligonales y uniones que hacen las veces de articulaciones) y posteriormente lo dotó de músculos virtuales y una red neuronal para controlarlos.

Aplicaciones Exitosas de los Algoritmos Genéticos

De tal forma, había que controlar los movimientos de los músculos a través de la red neuronal.

Esto involucró la optimización de 700 parámetros distintos. Hubo que definir, también, reglas especiales sobre cómo caminar correctamente (controlando la posición del centro de masa), a fin de evitar que la evolución “hiciera trampa”.

Aplicaciones Exitosas de los Algoritmos Genéticos

Aplicaciones Exitosas de los Algoritmos Genéticos

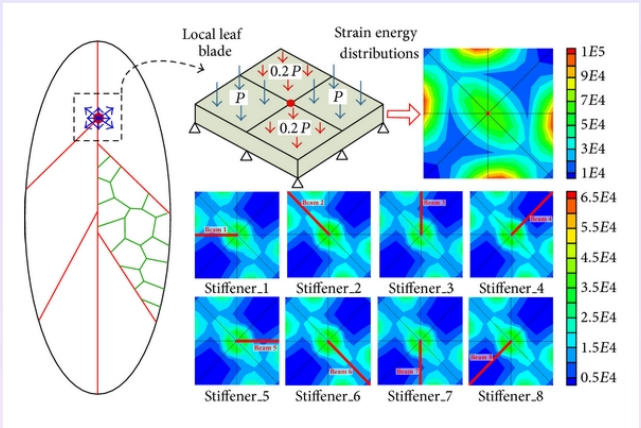
Reil decidió crear la empresa **Natural Motion** para comercializar el software creado, el cual permite crear animaciones realistas de movimientos humanos en 3D usando algoritmos genéticos.

Aplicaciones Exitosas de los Algoritmos Genéticos

La empresa británica **Redlands** produce losas prefabricadas de concreto.

El diseño de la topología de dichas losas ha sido optimizado por matemáticos usando técnicas tradicionales.

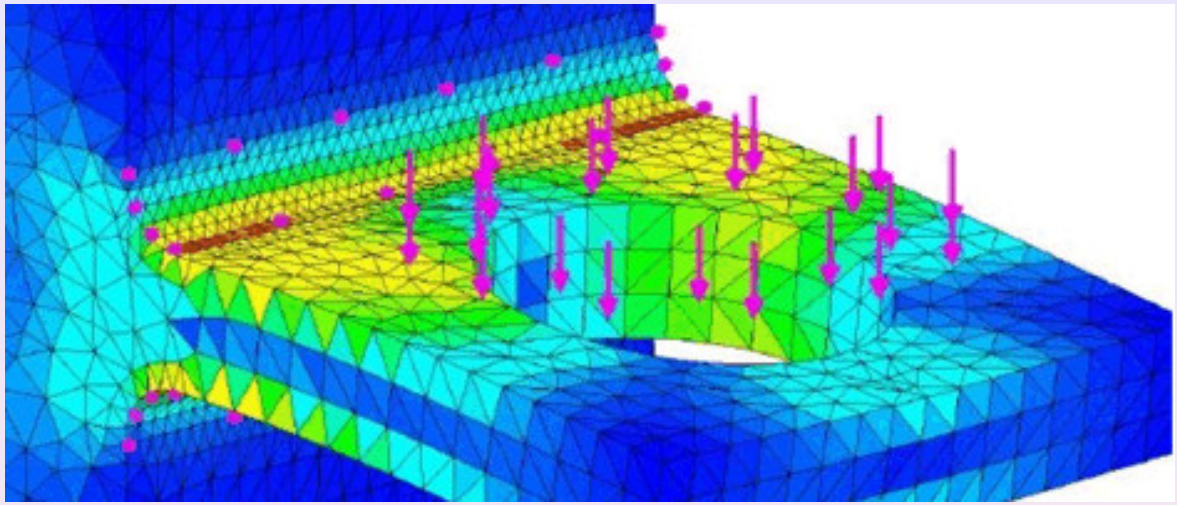
Aplicaciones Exitosas de los Algoritmos Genéticos



Complejidad del Problema:

- La losa se representa como una placa sujeta a cargas puntuales.
- Alta dimensionalidad: unas 400 variables.
- Una sola evaluación de la función de aptitud tomaba alrededor de 10 minutos en una estación de trabajo *Sun* con 6 procesadores.

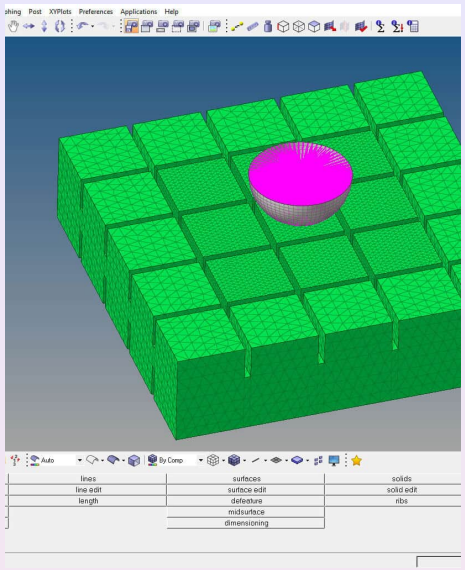
Aplicaciones Exitosas de los Algoritmos Genéticos



Análisis del Problema:

- Uso de ANSYS para el análisis por medio del elemento finito.
- Enfasis en la eficiencia del método a desarrollarse.
- ¿Podrá diseñarse un algoritmo genético para este problema?

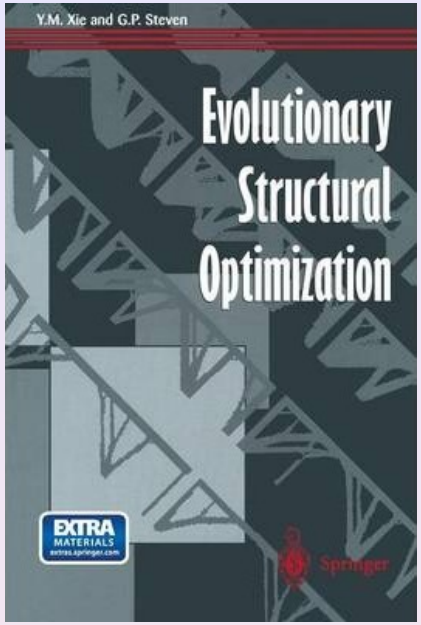
Aplicaciones Exitosas de los Algoritmos Genéticos



Uso de Computación Evolutiva:

- El algoritmo genético diseñado para resolver este problema fue implementado en FORTRAN.
- Esquema distribuido para evaluar la función de aptitud.
- Representación con cambio de granularidad.

Aplicaciones Exitosas de los Algoritmos Genéticos



Resultados Obtenidos:

- La solución del algoritmo genético resultó entre 3 y 5% mejor que la mejor encontrada con técnicas tradicionales.
- Los ahorros fueron de aproximadamente 1.3 millones de libras esterlinas por año.
- La inversión: unas 50 mil libras esterlinas en total.