Un estudio de la potencia computacional y robustez de los algoritmos evolutivos paralelos

Carlos Cotta, Enrique Alba, J.M. Troya

{ccottap,eat,troya}@lcc.uma.es
Dpto. de Lenguajes y Ciencias de la Computación
Universidad de Málaga
Campus de Teatinos (2.2.A.6)
29071 - Málaga, España

Resumen

El amplio éxito de los algoritmos evolutivos en múltiples disciplinas ha dado lugar a numerosos trabajos sobre aplicaciones, desarrollos teóricos y estudios empíricos. Sin embargo pensamos que es interesante disponer de trabajos donde se parta de una descripción formal fundamentada sobre su potencia computacional así como una posterior derivación de algoritmos útiles en la práctica. En este estudio todo ello resulta en una formalización de la potencia computacional (como máquinas de Turing), una particularización del modelo genérico de sistema adaptativo para dar lugar a la clase de los algoritmos genéticos secuenciales y paralelos y en una presentación de su robustez y análisis de los parámetros más influyentes sobre problemas complejos. En particular se concluye que los algoritmos paralelos son mejores que los secuenciales, que es más productivo migrar un individuo aleatorio, que los resultados mejoran cuantos más procesos-subpoblación se utilicen y que un algoritmo celular en cuadrícula debería tener forma alargada y no cuadrada.

Palabras Clave: Algoritmos Evolutivos, Algoritmos Genéticos Paralelos, Sistema Adaptativo, Eficacia.

1. Introducción

Los algoritmos evolutivos son un nuevo tipo de heurísticos cuyos principios de funcionamiento han sido inspirados por los procesos naturales de selección de las especies y supervivencia de los mejores. Existen numerosas aplicaciones con éxito en los campos de optimización combinatoria y funcional, I.A., comunicaciones, ... (Alba y Cotta, 1997), (Cantú-Paz, 1997), (Goldberg, 1989), (Bäck, Fogel y Michalewicz, 1997).

Este tipo de algoritmos permiten mantener un conjunto de soluciones plausibles en paralelo sobre las que se realizan modificaciones que se espera conduzcan a la solución óptima para el problema planteado. Las operaciones son típicamente estocásticas y se aplican de acuerdo a una probabilidad de éxito asociada.

Sobre el conjunto de operaciones se define un plan de reproducción que determina el orden de trabajo de los operadores. De esta forma, a lo largo de múltiples iteraciones las distintas poblaciones de estructuras se reemplazan unas a otras en forma de generaciones sucesivas hasta concluir en una población que contendrá una solución (sub)óptima.

La motivación de este trabajo es proporcionar la secuencia de pasos completa teórico-práctica para algoritmos evolutivos (y genéticos en particular) formalizándolos, derivando algoritmos reales, presentando una implementación de ellos y evaluándolos sobre problemas no triviales.

Concretamente, pretendemos caracterizar tanto la clase de los algoritmos evolutivos como su potencia computacional. Posteriormente se derivan de esta clase genérica tres subclases que dan lugar a los conocidos como algoritmos genéticos (AA.GG.) en sus variantes secuencial (Goldberg, 1989), celular (Spiessens y Manderick, 1991) y distribuida (Tanese, 1989). Tradicionalmente se distingue entre AA.GG. de grano grueso y de grano fino según que la relación entre computación y comunicación sea alta o no, respectivamente. En este trabajo no se proporciona esta visión tradicional del paralelismo, más bien los AA.GG. distribuidos (grano grueso) se consideran como una superclase que engloba a la clase secuencial y a la celular tanto como a cualquier otro esquema de evolución básica, definiendo al mismo tiempo un sistema de búsqueda distribuida con migración de estructuras entre los algoritmos.

Finalmente exponemos en el trabajo la aplicación de diferentes instancias de estas clases genéricas de algoritmos enfrentándolos a problemas de optimización combinatoria y funcional. Asimismo, presentamos un estudio sobre la influencia que ciertos parámetros importantes tienen sobre la evolución.

2. La Potencia Computacional de los Algoritmos Evolutivos

Para estudiar la potencia computacional de los algoritmos evolutivos es necesario en primer lugar caracterizarlos. Extendiendo la definición de *sistema adaptativo* (Daida *et al.*,1995), definimos los siguientes conceptos que nos resultan útiles:

Definición 1 (sistema adaptativo). Un sistema adaptativo es una tupla $S=(A,\Omega,\tau,B,\Omega_B,\tau_B,I)$, donde:

- $A = \{A_1, A_2, ...\}$ es el conjunto de estructuras con las que trabaja S.
- $\Omega = \{\omega_1, \omega_2, ...\}$ es un conjunto de operadores para modificar estructuras de A.
- $\tau : I \rightarrow \Omega$ es un plan adaptativo para determinar qué operador aplicar a la entrada I en un instante determinado.
- $B = \{B_1, B_2, ...\}$ es el conjunto de artefactos manejados por S.
- $\Omega_B = \{\omega_{B1}, \omega_{B2}, ...\}$ es un conjunto de operadores para modificar artefactos en B.
- $\tau_B : I \rightarrow \Omega_B$ es un plan adaptativo para determinar qué operador aplicar a la entrada I en un instante determinado.
- I es el conjunto de posibles entradas al sistema desde un entorno E (problema que resolver).

Nótese que existe un paralelismo entre los tres primeros y los tres siguientes elementos de S. Básicamente, ambas tripletas corresponden a los dominios del genotipo y fenotipo respectivamente (i.e., los primeros son representaciones internas de los segundos).

El siguiente concepto que es necesario definir hace referencia a la *granularidad* del algoritmo. Éste intenta capturar su estructura interna, la cual se considera una colección de *gránulos*. Dichos gránulos (modos de funcionamiento) son las unidades estructurales que, combinadas de manera apropiada, dan lugar a un algoritmo. La definición anterior puede ser extendida para reflejar la granularidad de un sistema adaptativo.

Definición 2 (sistema adaptativo granulado). Un sistema adaptativo granulado es una tupla $\Xi = (\Gamma, \Pi, \Phi)$, donde:

- $\Gamma = \{\Gamma_1, \Gamma_2, ..., \Gamma_m\}$ es un conjunto de gránulos, cada uno consistente en un sistema adaptativo $(A^i, \Omega^i, \tau^i, B^i, \Omega^i_B, \tau^i_B, I^i)$.
- Π es la política de activación de gránulos.
- $\Phi : \Gamma \times \Gamma \rightarrow \mathcal{P}(B)$ es una función que controla la comunicación entre gránulos (qué artefactos son compartidos entre ellos).

Por tanto un sistema adaptativo granulado comprende varios gránulos, siendo cada uno un sistema adaptativo que usa sus propias estructuras y lleva a cabo su propio plan adaptativo. Las conexiones entre gránulos se especifican mediante Φ, mientras que el flujo de control (secuencial o paralelo) queda determinado por Π.

A continuación se demostrará que la plantilla de un sistema adaptativo granulado es lo suficientemente potente como para realizar un cómputo arbitrario. Dado que cualquier computación puede ser representada como una máquina de Turing (Papadimitriou, 1994) (asumiendo modelos computacionales discretos) se mostrará que un sistema adaptativo granulado puede simular una máquina de Turing.

Sea $M = (\Sigma, Q, s, \delta)$ una máquina de Turing (TM en lo sucesivo). Sea $Q^+ = Q \cup \{h\}$ y $\Sigma^+ = \Sigma \cup \{\mathtt{L}, \mathtt{R}\}$. La función de transición $\delta: Q \times \Sigma \rightarrow Q^+ \times \Sigma^+$ puede descomponerse en una familia de funciones $(\delta_1,\delta_2,...,\delta_n)$ donde $n=|Q|$, tales que $\delta(q,a) = \delta_q(a)$.

Sea $\rho : MT \rightarrow \Sigma_\rho^*$ una función de codificación definida sobre Σ_ρ (donde $\Sigma_\rho \supset \Sigma$). La codificación de la cinta se define como sigue:

Definición 3 (codificación de la cinta). Sea $u_1\underline{a}u_2 \in (\Sigma^* \times \Sigma \times \Sigma^*)$ el contenido de la cinta. Se codifica como:

$$\rho(u_1\underline{a}u_2) = a(u_1)^{\mathrm{T}} : u_2 \qquad (1)$$

i.e., el símbolo bajo el cabezal concatenado con la cadena izquierda traspuesta, un delimitador (':' $\notin \Sigma$) y la cadena derecha.

Definición 4 (operador ◊). El operador infijo ◊ se define como:

$\lozenge: (\Sigma \to (Q^{+} \times \Sigma^{+})) \times (\Sigma^{*} \times \Sigma \times \Sigma^{*}) \to (\Sigma^{*} \times \Sigma \times \Sigma^{*})$

con la siguiente semántica:

$$\delta_i \lozenge u_1 \underline{a} u_2 = \begin{cases} u_1 \underline{a'} u_2 & \text{si } \pi_2(\delta_i(a) = a'),\ a' \in \Sigma \\ u'_1 \underline{x} a u_2 & \text{si } \pi_2(\delta_i(a) = \mathtt{L}),\ u'_1 x = u_1 \\ u_1 a \underline{x} u'_2 & \text{si } \pi_2(\delta_i(a) = \mathtt{R}),\ x u'_2 = u_2 \end{cases} \quad \textbf{(2)}$$

Dicho operador puede interpretarse como la aplicación de una regla de transición a una cadena de símbolos, y puede componerse fácilmente con la función ρ, resultando:

$$\rho(\delta_i \lozenge \rho^{-1}(a(u_1)^T : u_2)) = \begin{cases} a'(u_1)^T : u_2 & \text{si } \pi_2(\delta_i(a) = a'), a' \in \Sigma \\ x(u'_1)^T : a u_2 & \text{si } \pi_2(\delta_i(a) = \mathtt{L}),\ u'_1 x = u_1 \\ x a (u_1)^T : u'_2 & \text{si } \pi_2(\delta_i(a) = \mathtt{R}),\ x u'_2 = u_2 \end{cases} \textbf{(3)}$$

Tras definir los elementos necesarios, una MT puede simularse como sigue:

Lema 5 (simulación de una MT). *Sea M una MT. Existe un sistema adaptativo granulado Ξ que simula M.*

***Prueba**:* Sea $M = (\Sigma, Q, s, \delta)$. Ξ se define como:

- $\Gamma = \{\Gamma_1, \Gamma_2, ..., \Gamma_{|Q|}\}$ con $\Gamma_1 = (A^i, \Omega^i, \tau^i, B^i, \Omega^i_B, \tau^i_B, I^i)$ donde:
  - $A^i = \{\rho(u) \mid u \in (\Sigma^{*} \times \Sigma \times \Sigma^{*})\}$
  - $\Omega^i = \{\omega_1^{\mathrm{i}}\}$ tal que $\omega_1^{\mathrm{i}}(A_j^i) = \rho(\delta_i \lozenge \rho^{-1}(A_j^i))$
  - $\tau^i : I^i \to \Omega^i$ tal que $\tau^i(u) = \omega_1^{\mathrm{i}}, \forall u \in I^i$
  - $B^i = (\Sigma^{*} \times \Sigma \times \Sigma^{*})$
  - $\Omega^i_B = \{\omega_{\mathrm{B1}}^{\mathrm{i}}\}$ tal que $\omega_{\mathrm{B1}}^{\mathrm{i}}(B_j^i) = \delta_i \lozenge B_j^i$
  - $\tau^i_B : I^i \to \Omega^i_B$ tal que $\tau^i_B(u) = \omega_{\mathrm{B1}}^{\mathrm{i}}, \forall u \in I^i$
  - $I^i \subset B^i$
- $\Pi \equiv +$, i.e., activación secuencial de gránulos. El sistema comienza en modo 0, y cambia de modo después de la aplicación de cada operador.
- $\Phi\ (\Gamma_i, \Gamma_j) \equiv B$, i.e., todos los artefactos se comparten entre gránulos.

Cada gránulo Γ_i corresponde a un estado de M. El espacio de artefactos B representa las posibles configuraciones de la cinta. Una configuración B_i se representa internamente mediante una estructura A_j. Esta estructura sufrirá modificaciones a través de la aplicación de un operador ω_1^{i}. Este operador está definido para llevar a cabo las transiciones especificadas en δ_i. Inicialmente, el artefacto B_0 = (#u#) constituye la entrada del sistema.

Sea $\sigma_{i,j} : \Sigma_\rho^{*} \to \Sigma_\rho^{*}$ una función que devuelve los símbolos comprendidos entre las posiciones i-ésima y j-ésima (ambas inclusive) de su argumento. Sea σ_i una abreviatura de $\sigma_{i,i}$. Un cambio de modo ocurre cuando, estando en modo i, $\pi_1\ (\delta_i(\sigma_1(A_j))) \neq i$. En tal situación, el mecanismo de comunicación está definido para pasar el artefacto B_j (que contiene la configuración de la cinta) al gránulo que se activa o darlo como salida si Π determina la terminación (cuando π_1 $(\delta_i(\sigma_1(A_j)))=\mathtt{h}$).

3. Derivación de los Algoritmos Genéticos Secuenciales y Paralelos

En esta sección pretendemos derivar de las definiciones genéricas sobre sistemas adaptativos las definiciones que caracterizan a los algoritmos genéticos. En particular derivaremos los algoritmos genéticos secuenciales más tipicos tanto de estado estacionario como generacionales, los algoritmos celulares (masivamente paralelos) y los algoritmos distribuidos.

Para conseguir estos objetivos debemos ir instanciando los elementos que componen la tupla que define a un sistema adaptativo. Asimismo al final de cada subsección presentaremos las decisiones sobre operadores y parámetros que se hacen en este trabajo con la intención de hacer comprensibles los experimentos.

3.1. Caracterización de un A.G. Secuencial

Un algoritmo genético secuencial es un sistema adaptativo granulado que consta de un solo gránulo (véase la definición 2) de forma que, si la tupla genérica la notamos como $\Xi_{seq} = (\Gamma_{seq}, \Pi_{seq}, \Phi_{seq})$, cada componente se define como:

- $\Gamma_{seq} = \{\Gamma_1\}$
- Π_{seq} y Φ_{seq} -indiferente: sólo hay un gránulo-.

Con esta instanciación de valores para la tupla genérica basta para definir cualquier A.G. secuencial. Para generar las variantes de estado estacionario y generacional necesitamos también especificar el contenido del gránulo.

Definición 6 (algoritmo genético secuencial). El gránulo constituyente de un A.G. secuencial se define como $\Gamma = (A, \Omega, \tau, B, \Omega_B, \tau_B, I)$, donde:

- $A = \{A_1, A_2, ...\}$ es el conjunto de poblaciones posibles. De forma que $A_i = \langle \boldsymbol{a}_{i1}, \boldsymbol{a}_{i2}, ..., \boldsymbol{a}_{i\mu} \rangle$ donde $\boldsymbol{a}_{ij} \in G$. En los algoritmos genéticos tradicionales, $G = \{0,1\}^l$ o bien $G = \mathcal{R}^l$.
- $\Omega = \{\omega_{seq}\}$. Existe un único operador denominado ciclo reproductivo secuencial. Este operador se define por la concatenación de cuatro operadores internos:

  $\Omega_S: G^{\mu} \times \mathcal{R}^{\mu} \to G^{\mu'}$ $\qquad$ $\Omega_X: G^{\mu'} \to G^{\mu''}$

  $\Omega_M: G \to G$ $\qquad$ $\Omega_R: G^{\mu'+\mu''} \to G^{\mu}$

  $\omega_{seq}(A_i, \mathcal{R}_i) = \Omega_R(A_i \cup \langle \Omega_M(\boldsymbol{a}_{ij}) \mid \boldsymbol{a}_{ij} \in \Omega_X\ (\Omega_S(A_i, \mathcal{R}_i)) \rangle)$
- $\tau : I \to \Omega$ tal que $\tau(I) = \omega_{seq}(A_i, I)$.

- $B = \{B_1, B_2, ...\}$, tal que, B_i=<$\boldsymbol{b}_{i1},..\boldsymbol{b}_{ig}$>, donde $\boldsymbol{b}_{ij} \in P$ (conjunto de fenotipos). Existe una función de representación $\rho: P \rightarrow G$ tal que $B_i=\rho^{-1} \circ A_i$=<$\rho^{-1}(\boldsymbol{a}_{ij})$ | A_i=<$\boldsymbol{a}_{i1}, ..., \boldsymbol{a}_{i\mu}$>>. En AA.GG. con representación binaria es típico que
$$\mathbf{b}_{ji} = \rho^{-1}(\mathbf{a}_{ji}) = u_j + \frac{v_j - u_j}{2^{lx} - 1} \cdot \left(\sum_{r=1}^{lx} \mathbf{a}_{jir} \cdot 2^{r-1} \right)$$
y en caso de punto flotante $\boldsymbol{b}_{ij}=\boldsymbol{a}_{ij}$.
- $\Omega_B=\{\omega_{Bseq}\}$ donde $\omega_{Bseq}(B,I)=\rho^{-1} \circ (\omega_{seq}(\rho \circ B,I))$.
- $\tau_B : I \rightarrow \Omega_B$ se define como $\tau_B(I)=\omega_{Bseq}(B_i,I)$.
- $I = \mathcal{R}^{\mu}$, contiene los valores de adecuación de las estructuras en la población.

El funcionamiento del A.G. consiste pues en la repetida aplicación del operador ciclo reproductivo sobre una población de estructuras codificadas (A_i). Son los detalles de los operadores internos selección (Ω_s) y reemplazo (Ω_R) los que diferencian a los AA.GG. generacionales (μ‘=μ) de los AA.GG. de estado estacionario (μ‘=1 ó 2).

3.2. Caracterización de un A.G. Paralelo

Un algoritmo genético paralelo es un sistema adaptativo granulado de μ gránulos en el que cada componente se define como sigue:

Definición 7 (A.G. paralelo). Un A.G. paralelo es una tupla $\Xi_{par} = (\Gamma_{par}, \Pi_{par}, \Phi_{par})$, en la que:
- $\Gamma_{par} = \{\Gamma_1, ..., \Gamma_\mu\}$
- $\Pi_{par} \equiv \Gamma_1 \| \Gamma_2 \| ... \| \Gamma_\mu$, es decir, la política de activación es paralela, usualmente sincronizada entre los gránulos.
- $\Phi_{par}(\Gamma_i, \Gamma_j) = \begin{cases} \varnothing & i \notin \nu(j) \\ \psi_{ij}(b_i) & i \in \nu(j) \end{cases}$

  Esta expresión caracteriza la comunicación entre dos gránulos. En ella $b_i \in B^i$ representa el conjunto de estructuras contenidas en el gránulo Γ_i. La función ψ_{ij} selecciona de entre éstas a las estructuras distinguidas que se comparten entre los gránulos i y j. Usualmente $\psi_{ij}=\psi$, es decir, el mecanismo de comunicación es homogéneo. La función $\nu(i): \mathcal{N} \rightarrow \mathcal{P}(\mathcal{N})$ se denomina función de vecindad y determina qué gránulos comparten estructuras.

Tradicionalmente se han considerado dos modelos de A.G. paralelo: A.G. celular y A.G. distribuido. Ambos modelos quedan caracterizados por la anterior expresión. La diferencia entre ambos estriba en que, en los primeros, cada gránulo contiene un vecindario b_i, siendo μ^i igual a $|\nu(i)|$, y la función ψ_{ij} selecciona siempre el mismo elemento distinguido de este vecindario ($\psi_{ij} \neq \psi_{ik}(b_i)$ | $k \neq j; k,j \in \nu(i)$). En los segundos, cada gránulo contiene una subpoblación independiente (isla) de tamaño arbitrario, consistiendo usualmente la función ψ_{ij} en la selección de la mejor estructura de cada isla o de una estructura aleatoria de ésta. Adicionalmente, los AA.GG. celulares se sincronizan al final de cada ciclo reproductivo, mientras que los AA.GG. distribuidos lo hacen con mucha menor frecuencia.

Esta definición genérica nos permite derivar nuevos modelos paralelos en los que varios algoritmos celulares evolucionan en paralelo con migración esporádica de individuos entre ellos (por ejemplo en anillo) -`dcGA`- o bien a la inversa -`cdGA`-. Además por supuesto de poder derivarse las tradicionales islas de AA.GG. secuenciales -`dseqGA`-.

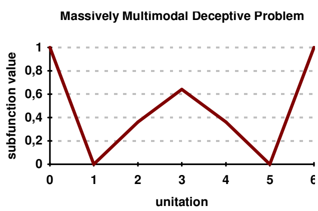
4. Resultados Experimentales y Análisis

En esta sección presentamos múltiples resultados empíricos destinados a demostrar la potencia y robustez de estos algoritmos.

Para ello vamos a estudiar la forma en la que los principales parámetros de cada tipo de algoritmo influyen en su comportamiento. En particular esto significa que haremos múltiples consideraciones sobre eficiencia y número de procesos en la ejecución paralela de estos algoritmos (secc. 4.1), tomaremos los resultados sobre un A.G. secuencial y comprobaremos cómo afectan la tasa de migración y la selección del individuo migrado en un A.G. distribuido (4.2) y finalmente cómo afecta la forma de la cuadrícula al comportamiento de un A.G. celular (4.3).

Nuestro objetivo es utilizar instancias complejas de problemas usuales en la literatura asociada (Fig. 1) aunque no tratamos de optimizar ningún parámetro de funcionamiento, sino que pretendemos presentar una comparativa para resaltar los comportamientos típicos y cómo cada uno de ellos muestra sus propias tendencias claras en cualquiera de los problemas.

6-BIT BIPOLAR DECEPTION	
unitation	**subfunction value**
0	1.000000
1	0.000000
2	0.360384
3	0.640576
4	0.360384
5	0.000000
6	1.000000



(a) **(b)**

Figura 1. *Aspecto tabulado* **(a)** *y gráfico* **(b)** *del problema de decepción bipolar de 6 bits.*

$$R(x_i\big|_{i=1..n}) = 10 \cdot n + \sum_{i=1}^{n} x_i^2 - 10 \cdot \cos(2 \cdot \pi x_i)$$

$$x_i \in [-5.12, 5.12]$$

Figura 1 (bis). *Formulación matemática de la función de Rastrigin generalizada.*

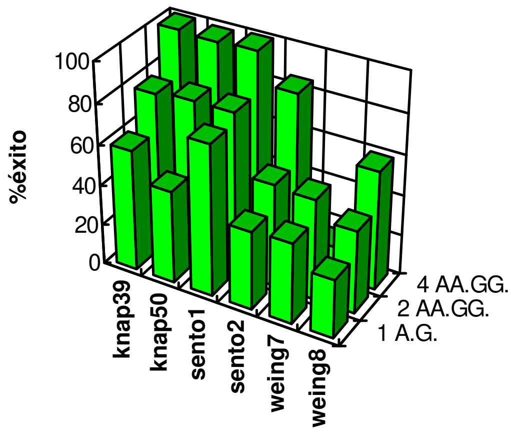
Para ello utilizaremos una instancia de 15 subproblemas (*Massively Multimodal Deceptive Problem*, MMDP15) para el caso de la función MMDP, otra de 20 variables para el caso de la función de Rastrigin (RAS20, aunque codificamos cada variable con 8 bits lo que facilita en cierta medida su solución) y varias instancias del problema de la mochila multidimensional (01-MKP en lo sucesivo; véase Khuri *et al.*, 1993).

4.1. Efecto del Número de Subpoblaciones

En principio hemos estudiado el efecto que el número de subpoblaciones independientes que se mantienen en un A.G. distribuido tiene sobre la calidad del resultado. Para ello se ha escogido el problema 01-MKP. Los experimentos realizados han considerado diferentes instancias tomadas de la literatura (OR-Library, Beasley, 1990) e instancias generadas aleatoriamente. En la figura 2 se presenta el porcentaje de éxito (ejecuciones que encuentran solución frente a totales) de un A.G. generacional distribuido con k=1,2,4 poblaciones de 100/k individuos cada una. Las migraciones (mejor individuo) se realizan cada 20 generaciones entre cualesquiera dos subpoblaciones.

Como puede apreciarse, el hecho de que la población global participe en varias pequeñas subpoblaciones da lugar a una mejora en la calidad final de la solución, debido al mantenimiento de una mayor diversidad en cada una de éstas. En estos resultados las mejoras al pasar de un algoritmo secuencial a otro con dos subpoblaciones son notablemente superiores que en el paso de usar 2 a 4 subpoblaciones. En definitiva, y según el número de procesadores disponibles, valores de 8 o 16 son adecuados.

Para obtener estos resultados hemos usado una red totalmente conectada de procesos secuenciales que ejecutan un A.G. generacional con intercambios de la mejor solución en cada paso de migración. Puesto que no estamos seguros de algunos de estos valores, en la siguiente sección dilucidamos si es mejor enviar a la mejor estructura o una aleatoria así como intentamos determinar un paso de migración adecuado pero usando una topología más básica y menos costosa en comunicaciones como es el anillo unidireccional de procesos. Por otro lado los procesos generacionales suelen ser lentos respecto a los de estado estacionario y por ello usamos estos últimos en las siguientes pruebas.



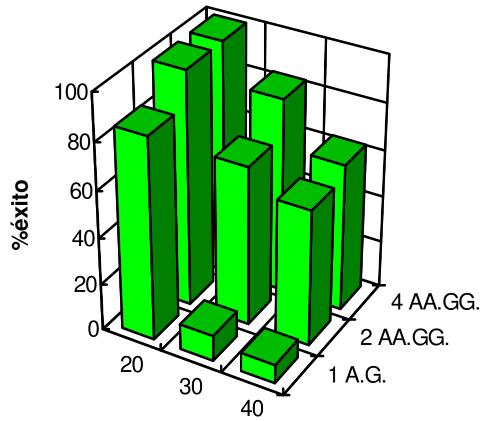


Figura 2. *Porcentaje de éxito en la resolución de 01-MKP en función del número de AA.GG. distribuidos para instancias de la OR-Library (arriba) e instancias generadas aleatoriamente (abajo).*

4.2. Análisis de la Política de Migración

En esta sección pretendemos describir el comportamiento de un A.G. distribuido que utiliza bien subpoblaciones secuenciales de estado estacionario o bien celulares. Aunque la mezcla en el mismo algoritmo distribuido está teórica y prácticamente permitida, aquí no presentamos resultados usando esta capacidad adicional.

Los dos factores principales que guían la búsqueda en este caso son el intervalo de migración y el criterio de selección de los individuos que migran. Resolvemos migrar un único individuo y utilizar una topología de anillo entre poblaciones para facilitar un estudio básico.

Los problemas usados son RAS20 (población de 8x22 individuos) y MMDP15 (8x32). En las figuras 3 y 4 presentamos la media de 20 ejecuciones independientes para mostrar el comportamiento en términos del porcentaje de éxito y número medio de evaluaciones (máximo $4 \cdot 10^5$) para conseguir alcanzar una solución al problema, tanto usando subpoblaciones de estado estacionario (`dseqGA`) como celulares (`dcGA`). El operador de cruce utiliza dos puntos (DPX-P_c=1.0) y genera un sólo hijo (el que tiene más porción del mejor padre) y mutación (P_m=0.01). Para el caso celular usamos siempre una cuadrícula de altura 2 individuos (como demostraremos, al ser estrecha, proporciona mejores resultados).

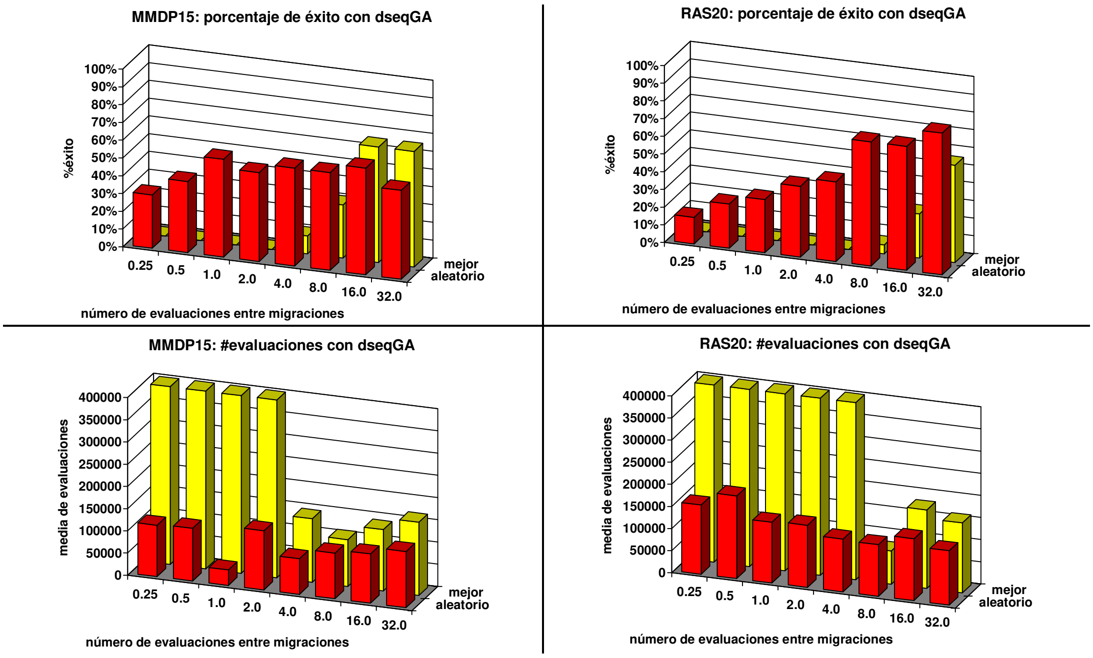


Figura 3. *Porcentaje de éxitos (arriba) y evaluaciones necesarias (abajo) de* `dseqGA` *para distintos intervalos de migración y políticas de selección de migrantes. Problema MMDP15 (izqda.) y RAS20 (dcha.).*

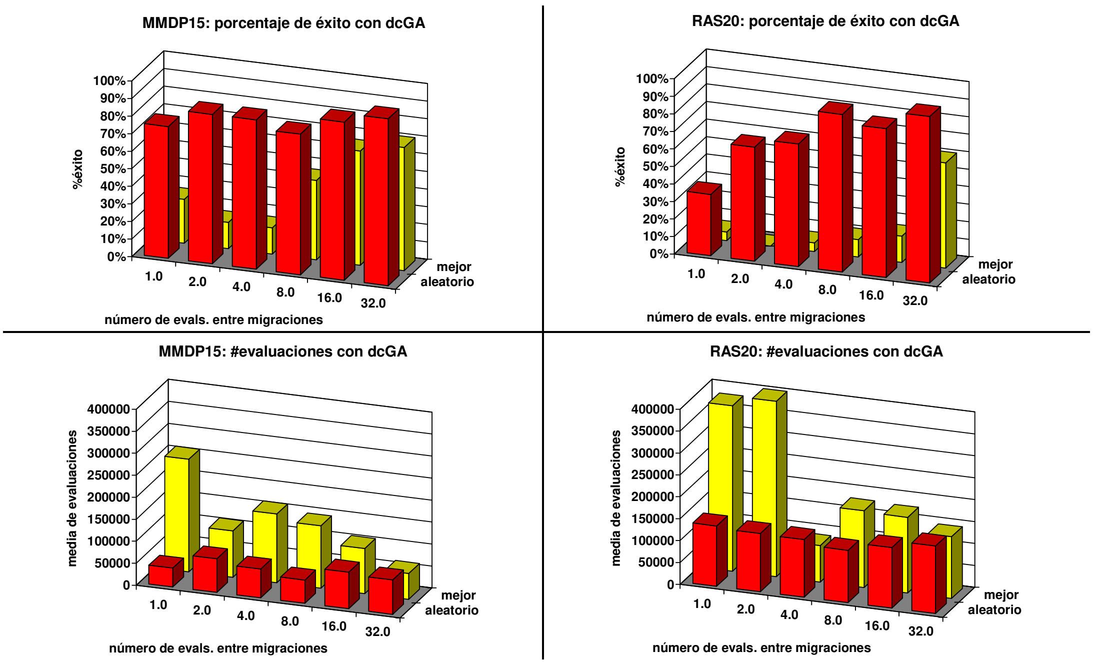


Figura 4. *Porcentaje de éxitos (arriba) y evaluaciones necesarias (abajo) de* `dcGA` *para distintos intervalos de migración y políticas de selección de migrantes. Problema MMDP15 (izqda.) y RAS20 (dcha.).*

Podemos extraer varias conclusiones de estos resultados. En primer lugar es mejor hacer migrar un individuo seleccionado aleatoriamente que hacer migrar una copia del mejor. Esto es así para cualquier valor del intervalo de migración y se explica por la mayor diversidad que introduce en las subpoblaciones permitiendo además una búsqueda global ligeramente guiada y común.

En segundo lugar respecto al intervalo de migración aparece como clara conclusión que son más interesantes valores altos. Aquí hemos expresado este parámetro como un múltiplo del tamaño de la población global. Valores de 16 o 32 son muy adecuados y permiten búsquedas independientes. El porcentaje de éxito es mayor y además la eficiencia tanto en número de evaluaciones totales como en tiempo real necesario (escaso coste de comunicación) es también superior para estos intervalos de migración.

Finalmente en relación al comportamiento de `dseqGA` y `dcGA` podemos comprobar cómo el segundo (`dcGA`) es en general mejor en eficacia y también eficiencia, al presentar un mayor porcentaje de éxito y un menor número de evaluaciones. Ambas características de este mismo algoritmo auguran su buen comportamiento en otros problemas.

4.3. Análisis sobre la Forma de la Cuadrícula

Uno de los principales factores de diseño de un algoritmo genético celular es decidir la topología de la forma bidimensional en la que residirán las estrucuturas, así como la elección de la función de vecindad entre ellas. En este aspecto el resultado de (Sarma y DeJong, 1996) nos permite usar una cuadrícula bidimensional como topología sin pérdida de generalidad, ya que definiendo una u otra vecindad podemos obtener la presión selectiva que deseemos simulando así cualquier otra topología o vecindario de un algoritmo distinto.

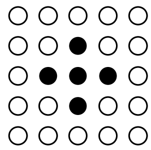
La clave para un estudio unificado en el campo de los AA.GG. celulares proviene de una definición adecuada del radio de la vecindad y del radio de la topología (Fig. 5). De su relación se deduce el comportamiento del algoritmo. Definimos ambos radios como la dispersión entre n^* patrones dentro de una elipse con centro ($\bar{x}$, $\bar{y}$). El valor n^* es el número de individuos involucrados (número de vecinos o bien número de individuos totales, según se trate de un vecindario o una topología). Para nuestro estudio pretendemos demostrar la influencia de la forma de la cuadrícula elegida en el resultado y en el proceso de búsqueda.

$$rad = \sqrt{\frac{\sum (x_i - \bar{x})^2 + \sum (y_i - \bar{y})^2}{n^*}}$$

$$\bar{x} = \frac{\sum_{i=1}^{n^*} x_i}{n^*} \qquad \bar{y} = \frac{\sum_{i=1}^{n^*} y_i}{n^*} \qquad\qquad ratio_{cGA} = \frac{rad_{vecindad}}{rad_{topología}}$$

(a) **(b)**

Figura 5. *Definición de radio* (**a**) *para la topología y vecindad y* (**b**) *su relación (ratio).*



$$rad_{lineal5} = \sqrt{\frac{2+2}{3+2}} = 0.8944$$

(a) **(b)**

Figura 6. *Nuestra vecindad NEWS (también conocida como lineal5)* (**a**) *y su radio* (**b**).

Por tanto podemos definir clases de equivalencia de distintas topologías y vecindades sobre ellas atendiendo a que todos los elementos de la misma clase compartan el mismo valor del *ratio* entre ambos radios. El caso secuencial tiene un *ratio* de 1.0. En el algoritmo que presentamos usamos siempre la vecindad NEWS cuyo radio es de valor 0.8944 (figuras 6a y 6b).

En la figura 7 presentamos dos conjuntos de resultados atendiendo a la solución de los problemas MMDP15 y RAS20 que lleva a cabo el A.G. celular usando los mismos parámetros excepto la forma de la cuadrícula. A medida que la cuadrícula es más estrecha se obtienen mejores resultados. Esto es lógico si pensamos que las mejores cadenas no hacen desaparecer a otras peores demasiado rápido, debido a su lenta propagación a través del eje más largo. Las peores cadenas se estancan en su vecindario próximo hasta desaparecer.

Un efecto negativo de este comportamiento es que necesitamos un número mayor de evaluaciones para alcanzar la solución a medida que la cuadrícula es más alargada. Sin embargo los beneficios de esta disposición son superiores a esta desventaja que no es muy pronunciada, ya que si ejecutamos varias islas en paralelo de algoritmos celulares obtenemos beneficios tanto en un menor tiempo real como en un menor número de evaluaciones necesarias. La ventaja respecto a mayor diversidad y mantenimiento paralelo de soluciones distintas asiladas por la distancia del `cGA` aumentan la eficacia (deseables).

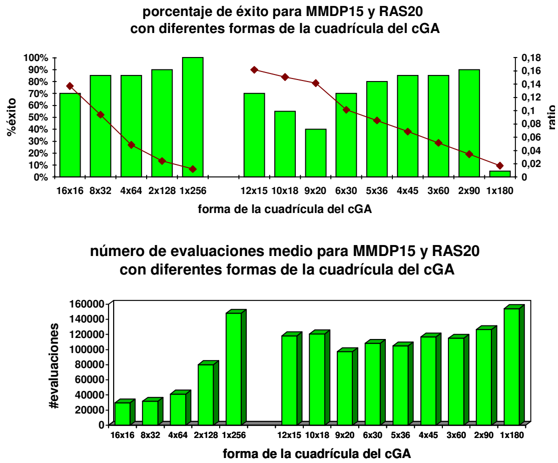


Figura 7. *Porcentaje de éxito (arriba) y esfuerzo computacional (abajo) para resolver MMDP15 y RAS20 con diferentes cuadrículas del cGA.*

Por tanto concluimos que los valores de altura deseables son pequeños (*ratii* en [0.02 .. 0.05]). Las topologías degeneradas (altura 1) funcionan de forma dependiente de la aplicación (bien para MMDP15 y mal para RAS20) lo que impide recomendarlas genéricamente.

5. Conclusiones y Trabajo Futuro

En este trabajo hemos desarrollado una nueva propuesta teórica que ha permitido definir un sistema adaptativo como máquina de Turing y por tanto capaz de realizar cualquier cómputo finito. Hemos presentado una definición unificada del paralelismo que nos permite derivar tanto islas de algoritmos celulares como células de algoritmos distribuidos, además por supuesto de cualquier A.G. secuencial.

Finalmente hemos analizado distintos algoritmos secuenciales y paralelos llegando a varias conclusiones importantes sobre parámetros vitales. En particular el intervalo de migración definido como múltiplo del tamaño de la población global permite mejores resultados para valores altos como 16 o 32.

Por otro lado es interesante usar un número no trivial de procesos isla, normalmente por encima de 4, ya que el porcentaje de éxito crece (aunque no de forma continuada) a medida que usamos más procesos. En cuanto a los algoritmos celulares nos interesan en forma de cuadrícula alargada para proveer un eje longitudinal de búsqueda que mejore la probabilidad de encontrar una solución.

La nueva propuesta de usar islas de algoritmos celulares aparece como ventajosa en términos tanto de eficiencia física como de eficacia respecto a las tradicionales islas secuenciales, incluso aunque sean de estado estacionario, cuyos resultados a su vez suelen ser mejores que las que utilizan evoluciones generacionales.

Como trabajo futuro podemos pensar en refinar estos resultados y arropar la robustez demostrada con aplicaciones a nuevos problemas. Asimismo sería interesante extender el modelo teórico hacia un modelo estadístico.

Referencias

(Alba y Cotta, 1997). Alba E., Cotta C., "Evolución de Estructuras de Datos Complejas", *Informática y Automática*, Vol. 30 Núm. 3, Madrid, 1997

(Bäck, Fogel, Michalewicz, 1997). Bäck T., Fogel D., Michalewicz Z. (eds.) *Handbook of Evolutionary Computation*, Oxford Univ. Press, 1997

(Beasley, 1990). Beasley J.E., "OR-Library: Distributing Test Problems by Electronic Mail", *Journal of the Operational Research Society*, 41(11), 1990, pp. 1069-1072

(Cantú-Paz, 1997). Cantú-Paz E., "A Summary of Research on Parallel Genetic Algorithms". *IlliGAL R. 95007*. Julio 1995. Revisión: *IlliGAL R. 97003*, Mayo 1997

(Daida *et al.*, 1995). Daida J.M., Ross S.J., Hannan B.C., "Biological Symbiosis as a Metaphor for Computational Hybridization", *Procs. of the Sixth ICGA*, Eshelman L.J. (ed.), Morgan Kaufmann, 1995, pp. 328-335

(Goldberg, 1989). Goldberg D.E.. *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, 1989

(Khuri *et al.*, 1993). Khuri S., Bäck T. y Heitkötter J., "The Zero/One Multiple Knapsack Problem and Genetic Algorithms", *Procs. of the ACM Sympos. of Applied Computation*, ACM Press, 1993

(Papadimitriou, 1994). Papadimitriou C.H., *Computational Complexity*, Addison-Wesley, 1994

(Sarma y DeJong, 1996). Sarma J., DeJong K.A., "An Analysis of the Effects of Neighborhood Size and Shape on Local Selection Algorithms", Voigt H.M.,Ebeling W.,Rechenberg I., Schwefel H.-P. (eds.), *Parallel Problem Solving from Nature IV*, Springer-Verlag, 1996, pp 236-244

(Spiessens y Manderick, 1991). Spiessens P., Manderick B., "A Massively Parallel Genetic Algorithm"; *Procs. of the 4th ICGA*, Belew R.K. and Booker L.B. (eds.), Morgan Kaufmann, 1991, pp. 279-286

(Tanese, 1989). Tanese R., "Distributed Genetic Algorithms"; *Procs. of the 3rd ICGA*, Schaffer J.D. (ed.), Morgan Kaufmann, 1989, pp. 434-439