Clases de Equivalencia en Algoritmos de Estimación de Distribuciones

Carlos Echegoyen, Alexander Mendiburu, Roberto Santana y Jose A. Lozano

Resumen—**Entender la relación que surge entre un algoritmo de búsqueda y el espacio de problemas es una cuestión fundamental en el campo de la optimización. En este trabajo nos centramos en la elaboración de taxonomías de problemas para algoritmos de estimación de distribuciones (EDAs). Mediante la utilización del modelo de población infinita y asumiendo selección basada en el ranqueo de las soluciones, agrupamos las funciones inyectivas según el comportamiento del EDA. Para llevar a cabo esta clasificación, se define una relación de equivalencia entre funciones que permite particionar el espacio de funciones en clases de equivalencia para las cuales el algoritmo tiene un comportamiento similar. Considerar diferentes modelos probabilísticos en el EDA genera diferentes particiones del conjunto de posibles problemas. Como consecuencia natural de las definiciones, todas las funciones objetivo están en la misma clase de equivalencia cuando el algoritmo no impone restricciones sobre el modelo probabilístico. Con el fin de crear una primera taxonomía de problemas, nos centramos en la partición que se produce cuando se considera un modelo probabilístico que asume independencia entre las variables. Para ello, primero fijamos las condiciones suficientes para decidir si dos funciones son equivalentes y segundo, obtenemos los operadores para describir y contar los miembros de una clase. En general, el presente trabajo sienta las bases para continuar el estudio del comportamiento de los EDAs y su relación con los problemas de optimización.**

Palabras clave— **Algoritmos de estimación de distribuciones, modelos probabilísticos, factorizaciones, selección basada en ranqueo, población infinita, clases de equivalencia, taxonomía de funciones.**

I. Introducción

Los algoritmos de estimación de distribuciones (EDAs) [1–3] constituyen un paradigma de optimización basado en poblaciones que ha adquirido especial relevancia en la última década. Como muestra de esta popularidad podemos destacar el desarrollo de nuevos y cada vez mas sofisticados EDAs [3, 4], la variedad de campos de aplicación [5–7] o los trabajos que estudian cuestiones fundamentales con el fin de entender su funcionamiento [8–11].

La principal característica de los EDAs es el uso de modelos probabilísticos para guiar la búsqueda hacia las áreas más prometedoras del espacio de búsqueda. Haciendo uso de los mejores individuos de la población, los modelos probabilísticos empleados permiten estimar distribuciones de probabilidad sobre el espacio de búsqueda en cada generación del algoritmo. De este modo, cada posible solución tiene una probabilidad asignada que varía durante el proceso de optimización. Estos valores de probabilidad son la fuente principal para determinar que soluciones serán devueltas por el algoritmo tras finalizar la búsqueda. Dado un problema, el comportamiento ideal del algoritmo sería conseguir incrementar cada vez más la probabilidad del óptimo a través del proceso iterativo. No obstante, cuando un EDA es aplicado para buscar la mejor solución a un problema dado, podemos esperar diferentes situaciones [10] que pueden diferir considerablemente del caso ideal. En este sentido, identificar los diferentes comportamientos que un algoritmo puede exhibir, y cómo estos comportamientos se relacionan con las características de los problemas de optimización, es una cuestión fundamental para entender los mecanismos subyacentes que gobiernan a este tipo de algoritmos.

Intelligent Systems Group, Universidad del País Vasco, Paseo Manuel de Lardizábal 1, 20018, San Sebastián - Donostia, Spain. E-mail:{carlos.echegoyen, alexander.mendiburu, roberto.santana, ja.lozano}@ehu.es

Comenzamos por considerar la siguiente cuestión: ¿Es posible agrupar los problemas de optimización según el comportamiento del EDA?. Con esta pregunta cuestionamos la existencia de grupos de problemas para los cuales el algoritmo se comporta igual y además, si los hay, queremos saber cómo son esos grupos. Con el fin de abordar esta cuestión, asumimos tres simplificaciones principalmente: 1) consideramos EDAs con población infinita, 2) la selección está basada en el ranqueo de las soluciones y 3) el algoritmo es aplicado en el espacio de las funciones inyectivas. Bajo estas asunciones, definimos una relación de equivalencia entre funciones basada en la secuencia de distribuciones de probabilidad que genera el EDA durante la búsqueda. Esta relación de equivalencia particiona el espacio de funciones en clases de equivalencia. Así, podemos establecer que el algoritmo tendrá un comportamiento similar para las funciones que pertenezcan a la misma clase.

Con el fin de llevar a cabo una primera exploración de las clases de equivalencia que un algoritmo genera en relación al modelo probabilístico que implementa, nos centraremos en un EDA que asume independencia entre las variables del problema. Para este tipo de algoritmo, primero fijamos las condiciones para identificar funciones equivalentes. A continuación, desarrollamos los operadores que nos permiten describir las funciones de una misma clase y así contar el número de elementos por clase. Además, mostramos que todas las funciones están en la misma clase de equivalencia cuando el algoritmo no asume ninguna restricción en el modelo probabilístico.

Este trabajo constituye un primer estudio de la relación entre funciones objetivo y EDAs desde el punto de vista de las clases de equivalencia que su comportamiento genera. Sin embargo, tras crear la agrupación de funciones respecto a un EDA, surgen nuevas cuestiones: ¿cómo podemos describir e identificar las clases que contienen problemas fáciles y difíciles? [12], ¿cómo el modelo probabilístico utilizado cambia la partición que se hace del espacio de funciones?, ¿cómo estudiar la convergencia del algoritmo para los problemas de una cierta clase? [13, 14], ¿cuáles son los descriptores del problema que permiten identificar la clase a la que éste pertenece?,

¿cómo se deben extrapolar estos resultados a los algoritmos que utilizan poblaciones finitas?, etc. La manera en la cual se pueden abordar algunas de estas cuestiones se discute a lo largo del artículo.

El resto del trabajo se organiza como sigue. En la Sección II se introducen los algoritmos de estimación de distribuciones. La Sección III explica el mecanismo de los EDAs con población infinita dentro del marco de este trabajo. La sección IV presenta y discute la definición formal de equivalencia. En la Sección V se establece la condición suficiente para decidir la equivalencia entre dos funciones para un EDA simple. En la Sección VI se desarrollan los operadores para alcanzar las funciones de una misma clase. La Sección VII discute las posibles líneas futuras del trabajo y finalmente en la Sección VIII se ofrecen las conclusiones recogidas tras el estudio.

II. Algoritmos de estimación de distribuciones

En este trabajo, se considera el siguiente problema de optimización:

$$\mathbf{x}^* = \arg \max_{\mathbf{x}} \mathrm{f}(\mathbf{x}), \tag{1}$$

donde $S = \{0,1\}^n$ representa el espacio de búsqueda, $\mathbf{x} = (x_1, \ldots, x_n) \in S$ es una solución y $f : S \to \mathbb{R}$ es la función objetivo. Por motivos de simplicidad, tratamos con variables binarias aunque los resultados fundamentales que se presentan no dependen de la cardinalidad de las variables. La cardinalidad del espacio de búsqueda es $|S| = 2^n$. A lo largo del trabajo, asumimos que $f(\mathbf{x})$ es inyectiva.

En EDAs se utilizan distribuciones de probabilidad con el fin de alcanzar la solución óptima $\mathbf{x}^*$ y resolver el Problema 1. Por lo tanto, necesitamos las siguientes definiciones. Sea $\mathbf{X} = (X_1, \ldots, X_n)$ un vector de n variables aleatorias binarias. Se define $p(\mathbf{X} = \mathbf{x})$, o simplemente $p(\mathbf{x})$, como la distribución de probabilidad conjunta sobre las variables $\mathbf{X}$ las cuales toman valores en el espacio de búsqueda S. El esquema general del EDA se resume en el Algoritmo 1

Algoritmo 1: **EDA**

1 $D_{t=0} \leftarrow$ Generar N individuos aleatoriamente
2 **do** $\{$
3 $\quad D_t \leftarrow$ Evaluar los individuos
4 $\quad D_t^s \leftarrow$ Seleccionar $M < N$ individuos de D_t según el esquema de selección
5 $\quad p_t(\mathbf{x}) = p(\mathbf{x}|D_t^s) \leftarrow$ Estimar la distribución de probabilidad conjunta
6 $\quad D_{t+1} \leftarrow$ Muestrear M individuos de $p_t(\mathbf{x})$ y crear la nueva población
7 $\quad t \leftarrow t + 1$
8 $\}$ **until** Criterio parada

De forma alternativa, el procedimiento de un EDA también puede explicarse en términos de las distribuciones de probabilidad involucradas en la búsqueda [15]

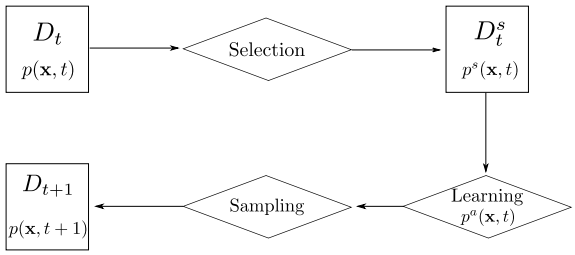

Fig. 1.Distribuciones de probabilidad relacionadas con los componentes del EDA. D_t, D_t^s, D_{t+1}: población e individuos seleccionados en la generación t, y población en la generación $t+1$. $p(\mathbf{x}, t)$, $p^s(\mathbf{x}, t)$, $p^a(\mathbf{x}, t)$: Distribuciones de probabilidad conjuntas para la población, el conjunto selección y el modelo probabilístico.

(ver Fig. 1). En primer lugar, D_t denota la población que el EDA maneja en la generación t y $p(\mathbf{x}, t)$ es la distribución subyacente a esta muestra. En segundo lugar, $p^s(\mathbf{x}, t)$ representa la distribución de probabilidad correspondiente al conjunto de individuos seleccionados D_t^s. Finalmente, $p^a(\mathbf{x}, t)$ es la distribución obtenida por el modelo probabilístico al aproximar $p^s(\mathbf{x}, t)$. Una vez que tenemos $p^a(\mathbf{x}, t)$, la población de la siguiente generación D_{t+1} se genera mediante un muestreo de esta distribución.

A. Esquemas de selección

Según algunos autores [16], los esquemas de selección pueden clasificarse en dos grupos: selección proporcionada y selección ordinal. Los procedimientos que implementan selección proporcionada seleccionan los individuos según el valor de función específico de cada uno. Sin embargo, los procedimientos de selección ordinal no seleccionan los individuos según su valor de función sino basándose en su ranqueo dentro de la población. Por lo tanto, este tipo de selección sólo tiene en cuenta información cualitativa dada por la función i.e. sólo tiene en cuenta el hecho de que $f(\mathbf{x}) > f(\mathbf{y})$ en lugar de considerar los valores de función específicos. En este trabajo, asumimos este tipo de selección en el algoritmo. Entre las implementaciones más comunes podemos encontrar la selección por truncación, torneo, ranking lineal o ranking exponencial. Estos procedimientos son considerados en una amplia variedad de algoritmos evolutivos tanto para su aplicación práctica como en estudios teóricos [13, 17, 18].

B. Modelos probabilísticos

Para tratar con distribuciones de probabilidad, el uso de modelos probabilísticos es esencial [19]. Un modelo probabilístico es capaz de expresar propiedades cualitativas y cuantitativas de una distribución de probabilidad. Así, este tipo de modelos permite codificar la distribución conjunta y manejarla en tiempos de cómputo razonables.

En EDAs, lo más común es utilizar modelos probabilísticos que factorizan la distribución conjunta por medio de funciones de probabilidad condicionadas. Esto es debido principalmente a motivos de eficiencia en el aprendizaje, el almacenamiento y especialmente en el

muestreo. En general, la factorización juega un papel fundamental en EDAs. La elección de una factorización apropiada para aproximar la distribución de probabilidad de los individuos seleccionados es un punto clave para el éxito de la búsqueda. De hecho, los diferentes EDAs suelen clasificarse según el tipo de modelo probabilístico que utilizan.

Para ilustrar los conceptos que nos permiten realizar una taxonomía de problemas para EDAs vamos a utilizar el algoritmo mas sencillo, es decir, aquel que supone que las variables del problema son independientes. Así, consideramos que la distribución $p^a(\mathbf{x}, t)$ se factoriza mediante distribuciones marginales univariadas de las siguiente manera:

$$p^a(\mathbf{x}) = \prod_{i=1}^{n} p^s(x_i), \tag{2}$$

donde $p^s(x_i)$ la distribución de probabilidad marginal de la variable i-ésima tras la selección.

III. EDAs con población infinita

A pesar de que los EDAs son básicamente un proceso iterativo de selección, aprendizaje y muestreo, el número de diferentes implementaciones que puede encontrarse en la literatura es considerablemente elevado. Además, este tipo de algoritmos puede ser aplicado en problemas muy diversos. Por lo tanto, ante una variedad tan amplia de situaciones y comportamientos, alcanzar un entendimiento general del funcionamiento de los EDAs es una tarea difícil y delicada.

Una alternativa para el estudio del comportamiento de los EDAs es el uso del modelo de población infinita (e.g. [13, 14, 20]) donde el algoritmo se reduce a su esencia y se elimina la aleatoriedad. En este modelo se asume que las distribuciones empíricas que se inducen de las soluciones en D_t y D_t^s (Fig. 1) convergerán, respectivamente, a las distribuciones de probabilidad subyacentes $p(\mathbf{x}, t)$ y $p^s(\mathbf{x}, t)$ conforme el tamaño de la muestra tiende a infinito. Por lo tanto, $p(\mathbf{x}, t)$ y $p^s(\mathbf{x}, t)$ pueden verse como la población y el conjunto de individuos seleccionados en la iteración t en EDAs con población infinita [14]. Según estos argumentos, como la muestra obtenida de $p^a(\mathbf{x}, t)$ tiende a infinito, podemos asumir que $p(\mathbf{x}, t+1) = p^a(\mathbf{x}, t)$.

Algoritmo 2: **EDA con población infinita,** $\mathcal{A}$

1 $\mathbf{p}_{t=0}$ es la población inicial
2 **do** {
3 Computar el esquema de selección ϕ sobre la población, $\mathbf{p}_t^s = \phi(\mathbf{p}_t)$
4 Computar $\mathbf{p}_t^a = \mathcal{M}(\mathbf{p}_t^s, \sigma)$ para aproximar $\mathbf{p}_t^s$.
5 $\mathbf{p}_{t+1} \leftarrow \mathbf{p}_t^a$
6 $t \leftarrow t + 1$
7 } **until** Convergencia

En el Algoritmo 2, presentamos el modelo de EDA que consideraremos a partir de ahora. Este algoritmo se denota como $\mathcal{A}$. Las poblaciones se representan como vectores de probabilidad $\mathbf{p}_t = (p_1, p_2, \ldots, p_{|S|})$ donde cada p_i es la probabilidad de una solución $\mathbf{x} \in S$. De la misma manera, la población seleccionada $\mathbf{p}^s$ y la aproximación $\mathbf{p}^a$ pueden representarse mediante estos vectores. Utilizamos el vector $\mathbf{p}$ para representar las poblaciones porque nos permitirá ordenar las probabilidades sin restricciones.

El espacio de todos los posibles vectores de probabilidad que puede tomar el algoritmo lo definimos como:

$$\Omega_{|S|} = \{(p_1, p_2, \ldots, p_{|S|}) | \sum_{i=1}^{|S|} p_i = 1, p_i \geq 0\}.$$

Para las poblaciones iniciales $\mathbf{p}_0 = (p_1, \ldots, p_{|S|})$ se establece que $p_i > 0$ para todo $i \in \{1, \ldots, 2^n\}$. Dada una población inicial, un algoritmo $\mathcal{A}$ aplicado a una función $f(\mathbf{x})$ produce secuencias deterministas de vectores de probabilidad o poblaciones:

$$\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \ldots$$

En el presente trabajo, utilizamos esta secuencia de poblaciones para describir el comportamiento del algoritmo.

A. Operador de selección basado en ranqueo

Como ya se dijo, en este trabajo sólo consideramos esquemas de selección basados en el ranqueo de las soluciones dentro de la población. Al trabajar con población infinita, en cada generación del algoritmo se considera que todo el espacio de búsqueda está representado en la población. La función $f(\mathbf{x})$ puede verse como una permutación σ de las soluciones en S. Así, la primera solución de σ, denotada como $\mathbf{x}_1$, tiene el mayor valor de función, $\mathbf{x}_2$ tiene el segundo mejor valor y así hasta la última solución $\mathbf{x}_{2^n}$ la cual tiene asignado el valor de función más bajo:

$$f(\mathbf{x}_1) > f(\mathbf{x}_2) > \ldots > f(\mathbf{x}_{2^n}).$$

Para nuestros propósitos, también nos es útil definir la función biyectiva $\sigma : S \rightarrow \{1, \ldots, 2^n\}$. Tenemos que $\sigma(\mathbf{x}) = \tau$ nos devuelve el ranking τ de la solución $\mathbf{x} \in S$. Hemos hecho cierto abuso de notación ya que σ representa tanto una permutación de las soluciones en S como la función $\sigma(\mathbf{x})$ que acabamos de definir. No obstante, ambos elementos serán utilizados sin ambigüedad. Una permutación σ dada representa a todas las funciones $f(\mathbf{x})$ que proporcionan el mismo ordenamiento de los elementos de S. Al considerar esta relación entre funciones y permutaciones, utilizaremos σ para representar funciones en el resto del artículo. En total, tenemos $2^n!$ funciones σ y definimos el conjunto de todas ellas como Π.

El la Tabla I, mostramos un ejemplo de σ para $n = 3$ donde las soluciones se han ordenado según una función $f(\mathbf{x})$. En la primera columna están los valores originales

$f(\mathbf{x})$	σ (x_1, x_2, x_3)	$\sigma(\mathbf{x})$	Población $\mathbf{p}$
100	(1,1,1)	1	$p(\sigma^{-1}(1)) = p_1$
50	(0,1,0)	2	$p(\sigma^{-1}(2)) = p_2$
45	(0,0,1)	3	$p(\sigma^{-1}(3)) = p_3$
20	(1,0,0)	4	$p(\sigma^{-1}(4)) = p_4$
10	(0,1,1)	5	$p(\sigma^{-1}(5)) = p_5$
3	(1,0,1)	6	$p(\sigma^{-1}(6)) = p_6$
1	(1,1,0)	7	$p(\sigma^{-1}(7)) = p_7$
0.5	(0,0,0)	8	$p(\sigma^{-1}(8)) = p_8$

TABLA I
EJEMPLO DE FUNCIÓN σ Y POBLACIÓN $\mathbf{p}$ PARA 3 VARIABLES.

de la función, en la segunda columna está la correspondiente permutación de soluciones σ y en la tercera columna están las posiciones del ranking dadas por $\sigma(\mathbf{x})$.

En este trabajo, asumimos que cualquier vector de probabilidad $\mathbf{p}$ que maneje el algoritmo $\mathcal{A}$ está ordenado según la función σ. Así, tenemos que p_1 es siempre la probabilidad de la solución óptima $\sigma^{-1}(1)$, p_2 es la probabilidad de la solución $\sigma^{-1}(2)$ y así sucesivamente hasta la última probabilidad p_{2^n} correspondiente a la peor solución. En general, tenemos que $p_i = p(\sigma^{-1}(i))$. La ordenación de la población se ilustra en la última columna de la Tabla I.

En el algoritmo $\mathcal{A}$, la selección se define formalmente como $\phi : \Omega_{|S|} \rightarrow \Omega_{|S|}$. La selección, para generar la nueva distribución, únicamente considera las posiciones en el ranking $\sigma(\mathbf{x})$ y no las configuraciones específicas $\mathbf{x}$. Por lo tanto, si obtenemos dos poblaciones iguales $\mathbf{p}_1$ y $\mathbf{p}_2$ cuando el algoritmo $\mathcal{A}$ es aplicado a dos funciones diferentes, tendremos que, tras el paso de selección ϕ, $\mathbf{p}_1^s = \mathbf{p}_2^s$ independientemente de σ. Podemos decir que la selección es un operador neutral que actúa igual ante cualquier función.

B. La aproximación $\mathcal{M}$

En el algoritmo $\mathcal{A}$, el paso de aproximación debe tener en cuenta la distribución de probabilidad $p^s(\mathbf{x})$ de los individuos seleccionados. Para ello, debemos unir el vector $\mathbf{p}^s$ con la permutación σ a través del par $(\mathbf{p}^s, \sigma)$. El par $(\mathbf{p}^s, \sigma)$ induce una distribución de probabilidad $p^s(\mathbf{x})$ tal que $p^s(\sigma^{-1}(i)) = p_i^s$. Por lo tanto, tenemos que $p_1^s = p^s(\sigma^{-1}(1))$ es la probabilidad del óptimo, $p_2^s = p^s(\sigma^{-1}(2))$ es la probabilidad del segundo mejor y así sucesivamente. La aproximación se define como una función $\mathcal{M} : \Omega_{|S|} \times \Pi \rightarrow \Omega_{|S|}$ que se aplica como $\mathbf{p}^a = \mathcal{M}(\mathbf{p}^s, \sigma)$ en el algoritmo $\mathcal{A}$. Por simplicidad, consideramos que la factorización que el algoritmo utiliza para aproximar la distribución de los individuos seleccionados está implícita en la función $\mathcal{M}$.

Como se comentó en la Sección II, asumimos que la distribución aproximada $p^a(\mathbf{x})$ se calcula mediante probabilidades marginales obtenidas de $p^s(\mathbf{x})$. Mas concretamente, nos centraremos en EDAs que introducen la factorización dada en la Ecuación 2. Por lo tanto, en este caso, la función $\mathcal{M}$ estará implementada según dicha Ecuación 2. En un EDA con poblaciones finitas, las probabilidades marginales $p(x_i)$ son estimadas del conjunto de individuos seleccionados e.g. mediante el cálculo de frecuencias relativas. No obstante, en el modelo de población infinita asumimos que las probabilidades se calculan de forma exacta como,

$$p^s(x_i) = \sum_{\mathbf{x} \setminus x_i} p^s(\mathbf{x}). \tag{3}$$

IV. EQUIVALENCIA ENTRE FUNCIONES

Un EDA $\mathcal{A}$ induce secuencias deterministas de vectores de probabilidad. La ejecución de una iteración del algoritmo puede expresarse mediante una función $\mathcal{G} : \Omega_{|S|} \times \Pi \rightarrow \Omega_{|S|}$ como $\mathbf{p}_{t+1} = \mathcal{G}(\mathbf{p}_t, \sigma)$. La función $\mathcal{G}$ es una composición de la selección ϕ y la función de factorización $\mathcal{M}$ tal que $\mathcal{G} = \mathcal{M} \circ \phi$. De esta forma, una secuencia de poblaciones generada por $\mathcal{A}$ puede expresarse como iteraciones de la función $\mathcal{G}$:

$$\mathbf{p}_0, \mathcal{G}(\mathbf{p}_0, \sigma), \mathcal{G}^2(\mathbf{p}_0, \sigma), \mathcal{G}^3(\mathbf{p}_0, \sigma), \ldots$$

donde $\mathcal{G}^t(\mathbf{p}_0, \sigma)$ es la población en la generación t.

Las propiedades de funciones similares a $\mathcal{G}^t$ han sido habitualmente analizadas mediante sistemas dinámicos. En [13, 21], se aportaron importantes resultados de convergencia para algunos EDAs utilizando esta aproximación. Nosotros, por ahora, no necesitamos considerar implementaciones concretas para ϕ o $\mathcal{M}$ y por lo tanto, $\mathcal{G}$ no tiene una formulación específica con la que estudiar la dinámica del algoritmo. Aunque la secuencia de poblaciones puede verse como un sistema dinámico, tomaremos una perspectiva más general para describir el comportamiento del EDA. Comenzamos por definir la equivalencia entre dos funciones.

Definición 1: Sean σ_1 y σ_2 dos funciones y sea $\mathcal{A}$ un EDA que implementa cualquier ϕ y $\mathcal{M}$ dados. Decimos que σ_1 y σ_2 son equivalentes para $\mathcal{A}$ si para cualquier población inicial $\mathbf{p}_0 \in \Omega_{|S|}$, $\mathcal{G}^t(\mathbf{p}_0, \sigma_1) = \mathcal{G}^t(\mathbf{p}_0, \sigma_2)$ para todo $t = 1, 2, 3, \ldots$.

Si dos funciones σ_1 y σ_2 son equivalentes para $\mathcal{A}$, las correspondientes secuencias de poblaciones que genera el algoritmo son iguales desde cualquier población inicial. La intuición que hay detrás de la Definición 1 es la siguiente. Decimos que dos funciones son equivalentes para un EDA cuando el comportamiento del algoritmo es el mismo desde el punto de vista de las posiciones que las soluciones ocupan en el ranking. Considérese que tenemos dos permutaciones diferentes σ_1 y σ_2 de las soluciones en S. Si aplicamos el algoritmo $\mathcal{A}$ en ambas funciones obtendremos dos secuencias de poblaciones. Si estas funciones son equivalentes para dicho algoritmo, éste asignará los mismos valores de probabilidad en las mismas posiciones $\sigma(\mathbf{x})$ del ranking en las dos ejecuciones. Lo importante es que esta asignación es independiente de las configuraciones específicas $\mathbf{x}$ que haya en cada posición del ranking. Como el algoritmo $\mathcal{A}$ ordena las poblaciones según σ, podemos decir que el algoritmo genera la misma secuencia de poblaciones para σ_1 y σ_2.

La Definición 1 proporciona una relación de equivalencia porque es una relación reflexiva, simétrica y transitiva. Dada esta relación de equivalencia, por cada σ, tenemos su clase de equivalencia denotada como $[\sigma]$. Esta clase está formada por el conjunto de las funciones que están relacionadas con σ por satisfacer la condición de la Definición 1. Mediante esta relación de equivalencia podemos particionar el espacio de funciones en clases de equivalencia para un algoritmo $\mathcal{A}$.

Teniendo en cuenta las anteriores definiciones de algoritmo, función y equivalencia, se puede deducir el siguiente resultado.

Teorema 1: Sea $\mathcal{A}$ un EDA que implementa $\mathcal{M}$ como $p^a(\mathbf{x}, t) = p^s(\mathbf{x}, t)$ para todo $t = 1, 2, 3 \ldots$. Todas las funciones σ pertenecen a la misma clase de equivalencia para $\mathcal{A}$.

La demostración del teorema es directa. Dado cualquier par de funciones, el algoritmo $\mathcal{A}$ es aplicado a ellas partiendo del mismo punto inicial $\mathbf{p}_0$. Debido a la neutralidad de ϕ, el algoritmo genera el mismo vector $\mathbf{p}_0^s$ tras la selección. A continuación, como $\mathcal{M}$ satisface que $p^a(\mathbf{x}) = p^s(\mathbf{x})$, tendremos la misma aproximación $\mathbf{p}_0^a$ en el caso de ambas funciones. Esta situación se repite en todas las generaciones. Por lo tanto, si se asume que $p(\mathbf{x}, t+1) = p^s(\mathbf{x}, t)$, el EDA exhibe exactamente el mismo comportamiento para todas las funciones. Podemos decir que todos los problemas son igualmente difíciles desde el punto de vista de este algoritmo.

A. Descriptores del comportamiento del EDA

Una vez que hemos dividido los problemas en clases de equivalencia, es interesante tratar de estudiar estas clases en relación al comportamiento del EDA dentro de ellas. Por un lado, el descriptor más básico de este comportamiento es la propia secuencia de poblaciones generada por el algoritmo. Por definición, sabemos que las secuencias que el algoritmo produce para las funciones de la misma clase son iguales. Si embargo, comparar las secuencias entre clases es una cuestión que merece especial atención. Por otro lado, podemos utilizar la idea de bases de atracción de las soluciones en S. Base de atracción es un término usado en sistemas dinámicos que adoptamos en este trabajo de manera simplificada. En términos generales, la base de atracción de un punto $\mathbf{x}$ en un sistema dinámico es el conjunto de puntos iniciales que convergen a $\mathbf{x}$. En nuestro contexto, teniendo en cuenta la función $\mathcal{G}$, decimos que la base de atracción de una solución $\mathbf{x} \in S$, es el conjunto $\mathcal{Z} \subseteq \Omega_{|S|}$ tal que $\forall \mathbf{p} \in \mathcal{Z}$, $lim_{t\to\infty}\mathcal{G}^t(\mathbf{p}, \sigma) = \mathbf{p}_\mathbf{x}$. Como trabajamos con vectores de probabilidad en lugar de con soluciones, utilizamos el vector $\mathbf{p}_\mathbf{x}$ para representar la población que ha convergido a la solución $\mathbf{x}$. Así, $\mathbf{p}_\mathbf{x}$ tiene un 1 en la posición que dicha solución ocupa en el ranking i.e. $p_{\sigma(\mathbf{x})} = 1$. Según esta definición, las bases de atracción producidas por $\mathcal{A}$ cuando se aplica a una función σ generan una partición de $\Omega_{|S|}$ dada por los conjuntos $\mathcal{Z}_1, \ldots, \mathcal{Z}_{|S|}$. Estos conjuntos contienen las poblaciones iniciales que convergen a cada solución del espacio de búsqueda.

Finalmente, de la Definición 1 se deduce que si dos funciones σ_1 y σ_2 son equivalentes, generan las mismas bases de atracción $\mathcal{Z}_1, \ldots, \mathcal{Z}_{|S|}$. Por lo tanto, todas las funciones pertenecientes a la misma clase tienen las mismas bases de atracción. Sin embargo, dos clases diferentes pueden tener las mismas bases de atracción pero siempre tendrán secuencias de poblaciones diferentes.

V. Condición de equivalencia

Este sección profundiza en la equivalencia entre funciones cuando el algoritmo utiliza probabilidades marginales univariadas para factorizar $p^a(\mathbf{x}, t)$ (Ecuación 2). Diferentes algoritmos como population-based incremental learning (PBIL), compact genetic algorithm (cGA) o univariate margnial distribution algorithm (UMDA) introducen este tipo de factorización. A continuación, se presenta la condición que deben cumplir dos funciones para ser equivalentes para un EDA que introduzca este tipo de modelo probabilístico.

La cuestión fundamental es la forma en la cual la función σ se relaciona con la estructura de la factorización utilizada para computar $\mathcal{M}$. Más concretamente, lo que tenemos en cuenta es cómo las posiciones $\sigma(\mathbf{x})$ de las soluciones en el ranking se organizan en las diferentes distribuciones marginales que forman la factorización. El cálculo de cada marginal $p^s(x_i)$ se puede relacionar con el ranqueo de las soluciones involucradas en el sumatorio. Así, utilizando la función inversa $\sigma^{-1}(\tau) = \mathbf{x}$, podemos reescribir la Ecuación 3 como,

$$p^s(x_i) = \sum_{\mathbf{x}\setminus x_i} p^s(\mathbf{x}) = \sum_{\tau \in Q_i^{x_i}} p^s(\sigma^{-1}(\tau)), \qquad (4)$$

donde $Q_i^{x_i} = \{\tau : \sigma^{-1}(\tau) = (y_1, \ldots, y_n) \wedge y_i = x_i\}$ es el conjunto de las posiciones que ocupan en el ranking las soluciones $\mathbf{x} \in S$ cuyos valores de probabilidad han sido utilizados para calcular la probabilidad marginal. La cardinalidad de cualquier conjunto $Q_i^{x_i}$ es siempre 2^{n-1}. Así, por cada probabilidad marginal $p(x_i)$, tenemos el conjunto Q_i^0 asociado a $p(X_i = 0)$ y el conjunto Q_i^1 asociado $p(X_i = 1)$. De esta manera, asociamos el conjunto $O_i = \{Q_i^0, Q_i^1\}$ a la distribución marginal $p(x_i)$. Estos conjuntos cumplen que para todo i, $Q_i^0 \cup Q_i^1 = \{1, \ldots, 2^n\}$ y $Q_i^0 \cap Q_i^1 = \emptyset$. Finalmente, definimos el conjunto $G_\sigma = \{O_1, O_2, \ldots, O_n\}$ que recoge toda la información necesaria para relacionar la función con la factorización. Los valores contenidos en los conjuntos $Q_i^{x_i}$ dependen de la función a la cual se aplica el algoritmo. La estructura de conjuntos que expresa la relación entre el modelo probabilístico y la función σ se ilustra en la Fig. 2. Para entender mejor esta relación, utilizamos el ejemplo de función que se muestra en la Tabla I. En este caso, mediante la Ecuación 4, el valor de probabilidad $p(X_1 = 0)$ se calcula a través de las soluciones cuyas posiciones en el ranking forman el conjunto $Q_1^0 = \{2, 3, 5, 8\}$. A su vez, $p(X_1 = 1)$ está asociado al conjunto de posiciones $Q_1^1 = \{1, 4, 6, 7\}$. Por lo tanto, tenemos el conjunto $O_1 = \{\{2, 3, 5, 8\}, \{1, 4, 6, 7\}\}$ asociado a la marginal $p(x_1)$. Siguiendo el ejemplo, tenemos que la marginal $p(x_2)$ tiene asociado el conjun-

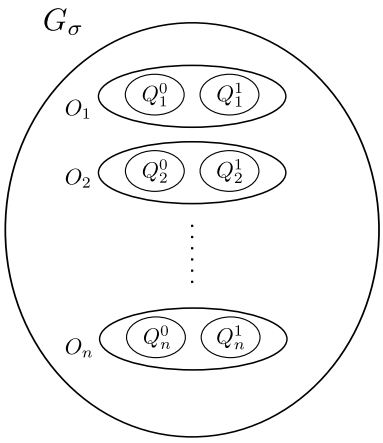

Fig. 2.Representación basada en conjuntos de la relación entre la función y el modelo probabilístico

to $O_2 = \{\{3,4,6,8\},\{1,2,5,7\}\}$ y $p(x_3)$ el conjunto $O_3 = \{\{2,4,7,8\},\{1,3,5,6\}\}$. Finalmente, construimos el correspondiente conjunto $G_\sigma = \{O_1, O_2, O_3\}$ asociado a la factorización y la función σ.

Mediante esta representación basada en conjuntos, podemos dar la siguiente condición de equivalencia que nos permitirá particionar el espacio de funciones.

Teorema 2: Sea $\mathcal{A}$ un EDA que implementa $\mathcal{M}$ como $p^a(\mathbf{x}) = \prod_{i=1}^n p^s(x_i)$. Dos funciones σ_1 y σ_2 son equivalentes para $\mathcal{A}$ si $G_{\sigma_1} = G_{\sigma_2}$.

Esta condición es suficiente pero no necesaria debido principalmente a que no hemos profundizado en las propiedades de la selección ϕ. La hipótesis es que la condición se podría transformar en suficiente y necesaria si imponemos ciertas restricciones básicas sobre ϕ. Sin embargo, tratar esta cuestión está fuera del alcance del presente trabajo. En cualquier caso, la condición de equivalencia impuesta por el Teorema 2 es válida para cualquier implementación de ϕ.

Debido a las limitaciones de espacio, en el presente artículo no presentamos la demostración rigurosa del Teorema 2. La intuición que hay detrás de dicho teorema es la siguiente. Dadas las dos funciones σ_1 y σ_2, el algoritmo asigna las mismas probabilidades a las mismas posiciones del ranking en la población inicial $\mathbf{p}_0$. La selección ϕ actúa según el ranqueo de las soluciones sin tener en cuenta la configuración $\mathbf{x}$ de dichas soluciones. Las probabilidades sufren las mismas modificaciones en el paso de selección y la función ϕ devuelve el mismo vector $\mathbf{p}_0^s$ para ambas funciones. En este punto, si los conjuntos G_{σ_1} y G_{σ_2} son iguales entonces las función de aproximación $\mathcal{M}$ produce el mismo vector $\mathbf{p}_0^a$ en los dos casos. La situación descrita anteriormente se repite a lo largo de todas las generaciones por lo que el algoritmo genera la misma secuencia de poblaciones al ser aplicado tanto a σ_1 y como a σ_2.

VI. Descripción de las funciones en la misma clase

Según el Teorema 2, si se cumple que $G_{\sigma_1} = G_{\sigma_2}$ entonces sabemos que el algoritmo genera la misma secuencia de poblaciones para las correspondientes funciones σ_1 y σ_2 y por lo tanto son equivalentes. De la igualdad de estos conjuntos también podemos derivar las operaciones que, dada una función, nos permiten obtener otras funciones pertenecientes a su clase de equivalencia. Así, con el fin de describir las funciones de la clase $[\sigma]$, debemos obtener todos los posibles $G_{\sigma'}$ tal que $G_\sigma = G_{\sigma'}$. Entonces, podemos afirmar que las correspondientes funciones σ' están en la clase $[\sigma]$.

Dado $G_\sigma = \{O_1, \ldots, O_n\} =$
$= \{\{Q_1^0, Q_1^1\}, \ldots, \{Q_n^0, Q_n^1\}\}$, las dos siguientes operaciones no modifican el conjunto G_σ y generan nuevas funciones equivalentes.

1. Dados $O_i = \{Q_i^0, Q_i^1\}$ y $O_j = \{Q_j^0, Q_j^1\}$, se pueden intercambiar los conjuntos Q entre ambos conjuntos O para obtener $O_i = \{Q_j^0, Q_j^1\}$, $O_j = \{Q_i^0, Q_i^1\}$. Los conjuntos Q_i^0 y Q_i^1 deben moverse juntos, de otra manera la probabilidad marginal no tendría sentido y demás obtendríamos conjuntos O diferentes a los anteriores. Esta operación puede ser interpretada como un intercambio de las posiciones del ranking que se han utilizado para calcular las marginales según la Ecuación 4. El efecto que esta operación tiene en una permutación σ es el de intercambiar los valores asignados a X_i por los de X_j en cada una de las posiciones del ranking. Esto produce una nueva función equivalente a σ. En general, tenemos $n!$ permutaciones de los conjuntos Q entre los n conjuntos O_i, $i \in \{1, \ldots, n\}$.
2. La segunda operación que mantiene el conjunto G_σ intacto es la siguiente. Las posiciones τ del ranking que contienen los conjuntos $Q_i^0, Q_i^1 \in O_i$ se pueden intercambiar (todas a la vez). El impacto que tiene este movimiento en σ es el de cambiar ceros por unos, y viceversa, en la correspondiente variable X_i. Por lo tanto, esto significa que la negación de una variable produce otra función equivalente. En total, tenemos 2^n posibles negaciones de las variables si se incluye la propia función σ sin modificar.

Ranqueo	σ (x_1, x_2, x_3)	σ' $(\neg x_1, x_2, x_3)$	σ'' $(\neg x_1, x_3, x_2)$
1	(1,1,1)	(0,1,1)	(0,1,1)
2	(0,1,0)	(1,1,0)	(1,0,1)
3	(0,0,1)	(1,0,1)	(1,1,0)
4	(1,0,0)	(0,0,0)	(0,0,0)
5	(0,1,1)	(1,1,1)	(1,1,1)
6	(1,0,1)	(0,0,1)	(0,1,0)
7	(1,1,0)	(0,1,0)	(0,0,1)
8	(0,0,0)	(1,0,0)	(1,0,0)

TABLA II
FUNCIONES EQUIVALENTES σ, σ' Y σ''.

En la Tabla II, damos un ejemplo de funciones equivalentes para $n = 3$. La función $\sigma' \in [\sigma]$ se ha obtenido negando los valores x_1 en σ. La función σ'', la cual está en la misma clase de equivalencia, se ha producido intercambiando los valores x_2 y x_3 en cada posición del ranking en σ'. Estas operaciones son indicadas

en las columnas correspondientes y puede comprobarse que $G_\sigma = G_{\sigma'} = G_{\sigma''}$.

De manera informal, podemos de decir que el intercambio y negación de variables en σ produce funciones equivalentes. Además, podemos afirmar que no hay más operaciones que puedan producir funciones equivalentes según el Teorema 2.

Finalmente, debido a que el Teorema 2 sólo da una condición suficiente, concluimos que el número mínimo de funciones por clase es $n!2^n$ y por lo tanto el número máximo de clases es

$$\frac{(2^n-1)!}{n!}.$$

En otras palabras, este es el número de comportamientos diferentes que el algoritmo puede exhibir en términos de secuencias de poblaciones. Además, como la factorización que hemos considerado es la más sencilla posible, no puede haber un EDA $\mathcal{A}$ que genere particiones más pequeñas del espacio de funciones.

VII. Trabajo futuro

Existe una gran variedad de cuestiones que deberían ser tratados en trabajos futuros. Éstas son las más destacadas:

- Deberíamos explorar en profundidad las clases dadas por el EDA que asume independencias entre las variables. Esto nos ayudaría a entender cómo son los diferentes comportamientos que se dan en cada clase. Para hacer esto, una primera aproximación sería realizar simulaciones numéricas del algoritmo en problemas con un número reducido de variables. Así, inicializando el algoritmo desde un conjunto suficientemente grande de poblaciones, podemos obtener las bases de atracción. Estas bases de atracción pueden ser interpretadas como una medida de dificultad del problema. Por ejemplo, si únicamente tenemos en cuenta la solución óptima, podríamos decir que un problema σ es más fácil que un problema σ' si la base de atracción del óptimo en σ es mayor que en σ'.
- En este trabajo, sólo consideramos el ranking dado por la función. Sin embargo, en la práctica, sólo disponemos de una información muy reducida sobre el problema. Creemos que es importante estudiar la relación entre los descriptores del problema y las clases a las cuales éstos pertenecen. Por ejemplo, descriptores como el número de óptimos locales, el grado de engaño o la descomposición en sumas de la función han sido relacionados con la dificultad de los problemas en algoritmos evolutivos [22]. Si conseguimos relacionar estos elementos con las clases de equivalencia, podríamos obtener una fuente de información para arrojar luz sobre la siguiente cuestión: ¿Cuáles son las características del problema que determinan el comportamiento del EDA?.
- La forma en la que el algoritmo converge depende fuertemente de la implementación de la selección. Con el fin de aportar resultados de convergencia, primero deberíamos estudiar la convergencia del algoritmo cuando se asume que $p(\mathbf{x}, t+1) = p^s(\mathbf{x}, t)$ i.e. cuando no se considera ninguna restricción en el modelo probabilístico. Para hacer esto, debemos profundizar en el estudio de la selección ϕ y sus propiedades.
- Para relacionar secuencias de diferentes clases, debemos estudiar propiedades de las secuencias de poblaciones dadas por la iteración de la función $\mathcal{G}$. Como ya discutimos previamente, podemos tener dos clases de equivalencia con las mismas bases de atracción pero sabemos, por definición, que dos clases diferentes siempre tienen secuencias de poblaciones diferentes. Por lo tanto, estas secuencias pueden ser usadas para distinguir comportamientos con precisión. Por ejemplo, consideremos todas las funciones que sólo tienen un óptimo local (el óptimo global). Sabemos por [13] que este óptimo es el único punto fijo atractivo para un EDA que asume independencia entre las variables. Según esto, podríamos pensar que el algoritmo tiene el mismo comportamiento para todas estas funciones. Sin embargo, en este caso simple, se puede calcular que el número de funciones con un óptimo local es mayor que el número de funciones que caben en una clase de equivalencia. Por lo tanto, aunque las bases de atracción para todas las funciones con un óptimo local sean iguales, se pueden distinguir diferentes comportamientos del algoritmo según las secuencias de poblaciones. Creemos que este tipo de consecuencias pueden ser útiles para extrapolar los resultados obtenidos con poblaciones infinitas a EDAs con poblaciones finitas.
- Finalmente, diferentes elementos del presente trabajo pueden ser generalizados. Por ejemplo, podemos desarrollar definiciones de equivalencia más generales que involucren diferentes algoritmos o grupos de modelos probabilísticos, también podemos extender los conjuntos G_σ para representar otras factorizaciones o podemos extender los resultados a funciones no inyectivas.

VIII. Conclusiones

En este trabajo, con el fin de arrojar luz sobre la relación que existe entre los EDAs y los problemas de optimización, hemos presentado diferentes definiciones y conceptos que nos permiten crear taxonomías de funciones para este tipo de algoritmos. En primer lugar hemos introducido los EDAs y su modelo de población infinita. A continuación, discutimos la existencia de clases de equivalencia en las cuales se agrupan las funciones según el comportamiento del algoritmo. En este sentido, el comportamiento del algoritmo se describe por las secuencias de poblaciones que éste genera. La definición de equivalencia que presentamos implica que el algoritmo genera las mismas secuencias de poblaciones para todas las funciones que pertenecen a la misma clase. Por lo tanto, sabemos que estas funciones también tienen asociadas las mismas bases de atracción.

Con el fin de realizar un estudio más profundo de la influencia que la factorización tiene en la partición del espacio de funciones, hemos considerado un EDA cuyo modelo probabilístico asume independencia entre las variables. Para este tipo de algoritmo y utilizando una representación de la factorización basada en conjuntos, damos la condición suficiente que nos permite decidir si dos funciones son equivalentes. A partir de esta condi-

ción, se deducen los operadores que nos permiten describir las funciones de una misma clase y contar el número de elementos que ésta contiene. Además, mostramos que todas las funciones son equivalentes cuando el algoritmo no factoriza la distribución conjunta. En este caso, el algoritmo tiene un comportamiento similar para todas las funciones.

Existen muchas conexiones entre las clases de equivalencia que presentamos aquí y otros trabajos teóricos relacionados con EDAs [13,14,21]. Los contenidos del presente artículo constituyen las bases de una línea de investigación que puede aportar resultados relacionados con la convergencia del algoritmo, la distinción entre funciones fáciles y difíciles, predicción sobre el éxito de una búsqueda, etc. Además, creemos que existen importantes conexiones entre las clases de equivalencia y ciertos descriptores del problema como el número de óptimos locales o la descomposición en sumas. En resumen, aunque hay diversas cuestiones que deben ser tratadas en profundidad, hemos sentado unas bases que permiten afrontar el estudio de la relación que surge entre los EDAs y el espacio de problemas de optimización.

AGRADECIMIENTOS

Este trabajo ha sido parcialmente financiado por los programas Saiotek y Research Groups 2007-2012 (IT-242-07) (Gobierno Vasco), el proyecto TIN2010-14931 (Ministerio Español de Ciencia e Innovación) y la red COMBIOMED en biomedicina computacional (Carlos III Health Institute). Carlos Echegoyen disfruta de una beca de la Universidad del País Vasco (UPV-EHU).

REFERENCIAS

[1] H. Mühlenbein and G. Paaß, "From recombination of genes to the estimation of distributions I. Binary parameters," in *Parallel Problem Solving from Nature - PPSN IV*, Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg, and Hans-Paul Schwefel, Eds., Berlin, 1996, vol. 1141 of *Lectures Notes in Computer Science*, pp. 178–187, Springer Verlag.

[2] P. Larrañaga and J. A. Lozano, Eds., *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, Kluwer Academic Publishers, Boston/Dordrecht/London, 2002.

[3] M. Pelikan, *Hierarchical Bayesian Optimization Algorithm. Toward a New Generation of Evolutionary Algorithms*, Studies in Fuzziness and Soft Computing. Springer, 2005.

[4] Mark W. Hauschild, Martin Pelikan, Kumara Sastry, and David E. Goldberg, "Using previous models to bias structural learning in the hierarchical BOA," in *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, New York, NY, USA, 2008, GECCO '08, pp. 415–422, ACM.

[5] Rubén Armañanzas, Iñaki Inza, Roberto Santana, Yvan Saeys, José L. Flores, José A. Lozano, Yves Van de Peer, Rosa Blanco, Víctor Robles, Concha Bielza, and Pedro Larrañaga, "A review of estimation of distribution algorithms in bioinformatics," *BioData Mining*, vol. 1, no. 6, pp. 1–12, 2008.

[6] Roberto Santana, P. Larrañaga, and J. A. Lozano, "Protein folding in simplified models with estimation of distribution algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 4, pp. 418–438, 2008.

[7] Alexander Brownlee, Martin Pelikan, John McCall, and Andrei Petrovski, "An application of a multivariate estimation of distribution algorithm to cancer chemotherapy," in *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2008*. 2008, pp. 1033–1040, ACM Press.

[8] J. L. Shapiro, "Drift and scaling in estimation of distribution algorithms," *Evolutionary Computation*, vol. 13, no. 1, pp. 99–123, 2005.

[9] Mark Hauschild, Martin Pelikan, Kumara Sastry, and Claudio Lima, "Analyzing Probabilistic Models in Hierarchical BOA," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 6, pp. 1199–1217, December 2009.

[10] C. Echegoyen, A. Mendiburu, R. Santana, and J.A. Lozano, "Towards understanding EDAs based on Bayesian networks through a quantitative analysis," *IEEE Transactions on Evolutionary Computation*, 2011, In press.

[11] Carlos Echegoyen, Qingfu Zhang, Alexander Mendiburu, Roberto Santana, and Jose A. Lozano, "On the limits of effectiveness in estimation of distribution algorithms," in *Evolutionary Computation (CEC), 2011 IEEE Congress on*, june 2011, pp. 1573 – 1580.

[12] T. Chen, K. Tang, G. Chen, and X. Yao, "Analysis of computational time of simple estimation of distribution algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 1, pp. 1–22, 2010.

[13] Qinfu Zhang, "On stability of fixed points of limit models of univariate marginal distribution algorithm and factorized distribution algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 1, pp. 80–93, 2004.

[14] Qinfu Zhang and Heinz Mühlenbein, "On the convergence of a class of estimation of distribution algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 2, pp. 127–136, 2004.

[15] Roberto Santana, P. Larrañaga, and J. A. Lozano, "Research topics on discrete estimation of distribution algorithms," *Memetic Computing*, vol. 1, no. 1, pp. 35–54, 2009.

[16] Kwang Y. Lee and Mohamed A. El-Sharkawi, Eds., *Modern Heuristic Optimization Techniques: Theory and Applications to Power Systems*, John Wiley & Sons, 2008.

[17] T. Blickle and L. Thiele, "A comparison of selection schemes used in evolutionary algorithms," *Evolutionary Computation*, vol. 4, no. 4, pp. 361–394, 1996.

[18] Adam Prügel-Bennett, "Finite population effects for ranking and tournament selection," *Complex Systems*, vol. 12, no. 2, pp. 183–205, 2000.

[19] E. Castillo, J. M. Gutierrez, and A. S. Hadi, *Expert Systems and Probabilistic Network Models*, Springer-Verlag, 1997.

[20] H. Mühlenbein, T. Mahnig, and A. Ochoa, "Schemata, distributions and graphical models in evolutionary optimization," *Journal of Heuristics*, vol. 5, no. 2, pp. 213–247, 1999.

[21] C. González, J. A. Lozano, and P. Larrañaga, "Analyzing the PBIL algorithm by means of discrete dynamical systems," *Complex Systems*, vol. 12, no. 4, pp. 465–479, 2001.

[22] L. Kallel, B. Naudts, and R. Reeves, "Properties of fitness functions and search landscapes," in *Theoretical Aspects of Evolutionary Computing*, L. Kallel, B. Naudts, and A. Rogers, Eds., pp. 177–208. Springer Verlag, 2000.