PBIL: un mismo nombre para distintos algoritmos. Un caso de estudio sobre un problema de optimización multi-objetivo.

M. Zangari[1], R. Santana[2], A. Mendiburu[2], A. T. R. Pozo[1]
{mzsouza, aurora}@inf.ufpr.br
{roberto.santana,alexander.mendiburu}@ehu.eus

[1]DInf - Federal University of Parana, CP: 19081, CEP 19031-970, Curitiba, Brazil
[2]Intelligent System Group, University of the Basque Country, San Sebastian, Spain

Resumen La optimización basada en el uso de técnicas que tratan de obtener un modelado probabilístico del espacio de búsqueda es una de las líneas de investigación que más ha avanzado en los últimos años en el ámbito de los Algoritmos Evolutivos. El algoritmo de aprendizaje incremental basado en poblaciones (PBIL) en una de las primeras propuestas introducidas en este campo, y ha sido ampliamente utilizado para resolver diferentes problemas de optimización. En este artículo queremos alertar de que las diferentes aplicaciones de PBIL publicadas en las literatura corresponden en realidad a dos implementaciones diferentes, en función de cómo ha sido definida la fase de aprendizaje del modelo. Estudiamos, analítica y empíricamente, el impacto que el método de aprendizaje utilizado tiene sobre el comportamiento del algoritmo. Como resultado de nuestra investigación, mostramos un caso de uso sobre un problema multi-objetivo, donde la elección de la variante PBIL puede producir resultados cualitativamente diferentes a lo largo del proceso de búsqueda. Utilizando diferentes versiones de PBIL como modelo base del algoritmo MOEA/D, evaluamos su efecto en la eficiencia de la búsqueda.

Palabras clave: PBIL, optimización multi-objetivo, MOEA/D

1. Introducción

El algoritmo de aprendizaje incremental basado en poblaciones (PBIL) [2] es uno de los algoritmos evolutivos basados en modelos más simple. Presumiblemente, es también uno de los primeros Algoritmos de Estimación de Distribuciones (AEDs) [11]. En una colección de artículos [2,3,4], Baluja y col. describen cómo se podría gestionar de manera eficiente el intercambio de información entre soluciones (implícito en los operadores de cruce), mediante la manipulación explícita de vectores de probabilidad que describan los estadísticos de la población. A grosso modo, PBIL funciona actualizando un vector que describe los estadísticos univariados de las mejores soluciones. La actualización de este sencillo modelo está controlada por un parámetro que establece el ratio de aprendizaje. El modelo es posteriormente utilizado para generar nuevas soluciones.

La sencillez de PBIL, junto con su similitud en comportamiento a los Algoritmos Genéticos (AGs) con operadores de cruce uniforme y en un punto, han contribuido a su popularidad. Se han propuesto diferentes aplicaciones y análisis teóricos del algoritmo [1,5,7,9,12,14], y también se han presentado extensiones al ámbito de los problemas definidos sobre el espacio continuo [15]. Un hecho curioso, que investigamos en este artículo, es que no existe una definición única del algoritmo PBIL en todos estos trabajos. Más concretamente, las diferencias entre las variantes de PBIL, aunque puedan parecer sutiles, resultan tener una influencia importante en el comportamiento del algoritmo durante el proceso de búsqueda.

Nuestro objetivo es analizar las diferentes variantes del algoritmo PBIL desde un punto de vista teórico y empírico. Mostramos cómo afectan al comportamiento del algoritmo los diferentes mecanismos utilizados en la fase de aprendizaje. Concretamente, mostramos un caso de uso basado en un problema multi-objetivo, donde la gran variabilidad de una de las variantes de PBIL presentadas puede ser utilizada como una manera de controlar la diversidad durante la búsqueda. En los experimentos, utilizaremos como modelo base del algoritmo MOEA/D (multiobjective evolutionary algorithm based on decomposition) las diferentes versiones de PBIL identificadas.

2. PBIL

Algoritmo 1: **PBIL (Descripción original) [2]**

```
1  P ← inicializar vector de probabilidad (cada posición = 0,5)
2  loop #Generaciones
3     I ← loop # MUESTRAS (Generar muestras)
4        muestra_I ← generar un vector de muestra en base a las probabilidades P
5        evaluar_I ← evaluar (muestra_I)
6     #(Buscar la mejor muestra)
7        max ← buscar el vector que se corresponde con la mejor evaluación
8     #(Actualizar el Vector de Probabilidades)
9     I ← loop # LONGITUD
10       P_I ← P_I * (1,0 − LR) + max_I * (LR)
11    #(Mutar el Vector de Probabilidades)
12    I ← loop # LONGITUD
13    if (random(0, 1] < MUT_PROBABILIDAD)
14          P_I ← P_I * (1,0 − MUT_CAMBIO) + random(0,0 o 1,0) *
            (MUT_CAMBIO)
```

El Algoritmo 1 muestra el pseudo-código de PBIL tal y como fue presentado en [2]. P representa un vector de probabilidades univariado asociado a la configuración de cada variable. De cara a analizar el algoritmo, empleamos la

notación original. Una característica relevante es que la actualización del vector univariado (Paso 8 del Algoritmo 1) se realiza teniendo en cuenta la mejor solución encontrada en cada generación. LR se refiere al ratio de aprendizaje. Nos referiremos a esta variante como *PBIL original*.

También en [2], se indica que, cuando se utilizan poblaciones de un tamaño considerable, actualizar el vector teniendo en cuenta únicamente la mejor solución conllevará el riesgo de ignorar el esfuerzo invertido y la información obtenida en la exploración llevada a cabo por el algoritmo. La solución directa sería actualizar el vector en base a las mejores ($M << N$) soluciones. Una propuesta se basaba en otorgar el mismo peso a cada una de las soluciones. Dicha propuesta fue presentada en [4]. Las diferencias principales entre el algoritmo PBIL presentado en [4] y la propuesta original se describen en el Algoritmo 2. En esta variante, las soluciones se ordenan primeramente de acuerdo a su valor de evalución (Paso 2) y, lo más importante, el modelo probabilístico aprendido es sensible a dicho orden (Pasos 4 y 5). Esta dependencia en el orden es debida a que los valores del vector de probabilidades se modifican de manera iterativa dentro del bucle que recorre las soluciones seleccionadas. Por lo tanto, la última solución tendrá un impacto mayor en la configuración del vector de probabilidades. A esta variante la denominaremos PBIL sensible al orden, o PBIL-OS.

Algoritmo 2: **PBIL (variante sensible al orden) [4]**

```
1  #(Ordenar vectores)
2     vector_soluciones = ordenar_vectores_de_mejor_a_peor_según_la_evaluación
3  #(Actualizar el Vector de Probabilidades hacia las mejores soluciones)
4     for j := 1 to M do
5        for i := 1 to LONGITUD do
               P[i] := P[i] * (1,0 − LR) + vector_soluciones[j][i] * (LR)
```

Finalmente, otra interpretación del algoritmo PBIL asume que, antes de actualizar el vector de probabilidades, se calcula un vector r con las probabilidades univariadas de las M soluciones seleccionadas. Posteriormente, dicho vector auxiliar r es utilizado para actualizar el vector de probabilidades. Nos referiremos a esta versión como incremental univariate distribution algorithm (iUMDA) PBIL, o *iUMDA*-PBIL, porque la estrategia utilizada es la misma que la propuesta para iUMDA en [10].

Todas estas variantes de PBIL han sido utilizadas y descritas en distintos trabajos de investigación. La versión original ha sido utilizada por ejemplo en [2,3,5]. Menciones a la versión sensible al orden pueden encontrarse en [4,14,18]. Finalmente, la versión iUMDA está presente en [1,7,9,12]. Sin embargo, ninguno de los trabajos mencionados ha estudiado el impacto que supone escoger una u otra variante del algoritmo PBIL.

Algoritmo 3: **PBIL (variante iUMDA) [10]**

```
1  #(Calcular las probabilidades actuales)
2     for i := 1 to LONGITUD do r[i] := (sum_{j=1}^{M} vector_soluciones[j][i]) / M
3  #(Actualizar Vector de Probabilidades)
4     for i := 1 to LONGITUD do
          P[i] := P[i] * (1,0 - LR) + r[i] * LR
```

3. Comportamiento esperado del algoritmo

Sea $\mathbf{X} = (X_1, \ldots, X_n)$ un vector de variables aleatorias discretas. Utilizamos $\mathbf{x} = (x_1, \ldots, x_n)$ para indicar una asignación a las variables. Representaremos una población como un conjunto de vectores $x^1, \ldots, x^N$ donde N es el tamaño de la población. De manera similar, x_i^l representa la asignación a la i-ésima variable de la solución l en la población. p denota una distribución, y $p(x_i)$ la probabilidad marginal $X_i = x_i$. Nos centraremos en problemas binarios.

Usando esta notación, la ecuación de actualización de la variante sensible al orden se puede expresar como $p(x_i) = (1-\alpha)p(x_i) + \alpha x_i$ donde α es el ratio de aprendizaje. Para *iUMDA*-PBIL, el resultado de la regla de actualización será igual independientemente del orden de las soluciones, $p(x_i) = b(1-\alpha) + \frac{\sum_{l=1}^{N} x_i^l}{N}$.

Por otro lado, no es difícil derivar la expresión parametrizada de $p(x_i)$ para una población de N soluciones en la variante PBIL-OS. Concretamente, resulta $p(x_i) = b(1-\alpha)^N + \alpha \sum_{l=1}^{N} x_i^l (1-\alpha)^{N-l}$.

Como las probabilidades univariadas de PBIL-OS dependen del orden de las soluciones, existirá un valor diferente para cada una de las $\binom{N}{k}$ formas en las que puede ser ordenado un vector con k unos. Este hecho proporciona una gran variabilidad al resultado producido por PBIL-OS. Por lo tanto, nos centraremos en determinar el valor esperado de $p(x_i)$ cuando todos los posibles órdenes son tenidos en cuenta.

$$\bar{p}(x_i) = \frac{\binom{N}{k} b(1-\alpha)^N + \alpha \sum_{j=1}^{\binom{N}{k}} \sum_{l=1}^{N} x_i^l (1-\alpha)^{N-l}}{\binom{N}{k}} \tag{1}$$

$$= \frac{\binom{N}{k} b(1-\alpha)^N + \frac{k\binom{N}{k}}{N} \alpha \left(\sum_{l=0}^{N-1} (1-\alpha)^{N-l-1} \right)}{\binom{N}{k}} \tag{2}$$

$$= b(1-\alpha)^N + \frac{k}{N} \alpha \left(\frac{1-(1-\alpha)^N}{1-(1-\alpha)} \right) \tag{3}$$

$$= b(1-\alpha)^N + \frac{k}{N} - \frac{k}{N}(1-\alpha)^N \tag{4}$$

$$= \frac{k}{N} + (1-\alpha)^N \left(b - \frac{k}{N}\right) \tag{5}$$

Para derivar este valor hemos tomado en consideración que el número de valores no cero en cada una de las N posiciones es $\frac{k\binom{N}{k}}{N}$, así como la fórmula

de la suma exponencial $\sum_{l=0}^{N-1} c^l = \frac{1-c^N}{1-c}$. Podemos observar en la Ecuación 5 que el valor esperado puede ser muy cercano a $\frac{k}{N}$, particularmente para valores grandes de N. Si la probabilidad inicial b es $\frac{k}{N}$, entonces el valor esperado $\bar{p}(x_i)$ es exactamente $\frac{k}{N}$.

4. Contexto de aplicación: MOEA/D-PBIL

El análisis presentado en la sección previa muestra la alta variabilidad del vector de probabilidad generado por la variante de PBIL sensible al orden. A continuación, procederemos a estudiar cómo afecta este hecho al comportamiento de un AE multi-objetivo basado en modelos, particularmente, el denominado multiobjective evolutionary algorithm based on decomposition (MOEA/D) [16].

4.1. Descripción de MOEA/D

Un problema multi-objetivo (MOP) puede ser definido como:

$$\textit{minimizar (o maximizar) } F(x) = (f_1(x), ..., f_m(x)); \quad \textit{sujeto a } x \in \Omega \qquad (6)$$

donde $x = (x_1, x_2, x_n)^T$ es la variable de decisión de tamaño n, Ω es el *espacio de decisión*, $F : \Omega \to R^m$ consta de m funciones objetivo y R^m es el *espacio de objectivos*.

De cara a definir el conjunto de soluciones óptimas se utiliza la definicion de solución **Pareto óptima**[1]**:** Sean $x, y \in \Omega$, se dice que x ***domina a*** y si y sólo si $f_i(x) \leq f_i(y)$ para todo $i \in \{1, .., m\}$ y $f_i(x) < f_i(y)$ para al menos un i. Una solución $x^* \in \Omega$ se denomina **Pareto óptima** si no existe ningún $x \in \Omega$ que *domina a* x^*. El conjunto de soluciones Pareto óptimas se denomina ***Pareto set (PS)*** y las soluciones mapeadas en el *espacio de objectivos* se denomina ***Pareto front (PF)***, es decir, $PF = \{F(x)|x \in PS\}$.

MOEA/D [16] descompone un MOP en un número de sub-problemas escalares y los optimiza simultáneamente. Cada sub-problema $i \in \{1, ..., N\}$ está asociado a un vector de pesos $\lambda^i \in \{\lambda^1, ..., \lambda^N\}$. Se han utilizado diversas **técnicas de descomposición** en MOEA/D [6]. La aproximación de Tchebycheff es una de las más populares, y por lo tanto es la utilizada en este artículo. En dicha aproximación, un sub-problema de optimización escalar se define como:

$$\textit{minimizar} \;\; g^{te}(x|\lambda, z^*) = \max_{1 \leq l \leq m} \{\lambda_l |f_l(x) - z_l^*|\}; \;\; \textit{sujecto a } x \in \Omega \qquad (7)$$

donde $\lambda = (\lambda_1, ..., \lambda_m)^T$ es el vector de pesos de este sub-problema y satisface $\sum_{l=1}^{m} \lambda_l = 1$ para todo $l \in \{1, ..., m\}$, y cada $\lambda_l \geq 0$. $z^* = (z_1^*, ..., z_m^*)^T$ es el punto de referencia, es decir, $z_l^* = \max\{f_l(x)|x \in \Omega\}$ para cada $l = 1, ..., m$. Para cada solución Pareto óptima x^* existe un vector de pesos λ tal que x^* es la solución optima de (7) y cada solución óptima de (7) es Pareto óptima de (6).

[1] La presente definición de dominación es para minimización. Para el caso de maximización, las inecuaciones deberían ser invertidas.

MOEA/D define una relación de vecindario entre los distintos sub-problemas, y esta relación juega un papel vital en el intercambio de información entre ellos. El vecindario i para cada sub-problema se define de acuerdo a la distancia euclidea de su vector de pesos λ^i. El conjunto de vectores de pesos debe ser tal que las soluciones óptimas de los diferentes sub-problemas se hallen uniformemente distribuidas a lo largo de ***PF***. En este artículo, fijamos los vectores de pesos $\{\lambda^1, ..., \lambda^N\}$ siguiendo la propuesta descrita en [16] . La relación entre vecinos se utiliza para seleccionar las soluciones padre y sustituir las soluciones antiguas. El Algoritmo 4 muestra el pseudo-código de MOEA/D utilizado en este trabajo.

Algoritmo 4: **Estructura general de MOEA/D, adaptada de [13]**

1 Fijar el tamaño del vecindario de cruce T_m y el tamaño del vecindario de reemplazo T_r. Generar la población inicial $x^1, ..., x^N$

2 Mientras no se cumpla una condición de parada

3 Para cada sub-problema $i \in 1, ..., N$ en cada generación
Establecer el conjunto de padres P^i de acuerdo a $T_m(i)$

4 $P^i \leftarrow$ SELECCIÓN_CRUCE($T_m(i)$,Población)
Completar la reproducción de P^i para generar una nueva solución x^i_{new}.

5 $x^i_{new} \leftarrow$ VARIACIÓN(x^i). Calcular $F(x^i_{new})$
Decidir que sub-problemas deben ser actualizados. Reemplazar las soluciones actuales de dichos sub-problemas por x^i_{new} si x^i_{new} tiene un valor de función de agregación mejor

6 Población $\leftarrow$ ACTUALIZAR_POBLACIÓN(T_r, x^i_{new}, Población)

4.2. MOEA/D y PBIL

En este artículo integramos las distintas variantes de PBIL dentro del algoritmo MOEA/D. Por lo tanto, en el paso 5 (VARIACIÓN(x^i)) del Algoritmo 4, se aprende un modelo a partir de las soluciones de los sub-problemas vecinos P^i, y se muestrea una nueva solución para obtener x^i_{new}. Por lo tanto, para cada sub-problema i, el algoritmo mantiene un vector de probabilidades univariadas $Pr^i = (Pr^i_1, ..., Pr^i_n)^T$. Para la variante PBIL-OS, utilizamos tres estrategias de ordenación diferentes para el conjunto de padres seleccionado P^i. La estrategia convencional, denominada PBIL-OS-c, consiste en ordenarlos en base a la distancia euclídea respecto a λ^i (primero el más cercano). En la segunda estrategia, las soluciones de P^i son evaluadas utilizando la función agregada escalar basada en λ^i (es decir, para cada solución $y \in P^i$ se calcula el valor escalar $g(y|\lambda^i)$), y posteriormente se ordenan de forma descendente, $g(y^{mejor}|\lambda^i) \leq g(y^{peor}|\lambda^i)$. Denominamos a esta variante PBIL-OS-d. La tercera estrategia se basa en PBIL-OS-d, pero situando las soluciones en orden inverso (de peor a mejor). La llamaremos PBIL-OS-a. La última estrategia considerada en los experimentos es la de iUMDA-PBIL (no depende del orden).

5. Experimentos

Los experimentos han sido diseñados con dos objetivos. Primero, determinar si las diferentes variantes de PBIL afectan al comportamiento del algoritmo MOEA/D. Segundo, en caso de que existan diferencias, determinar si PBIL-OS tiene un efecto negativo o positivo sobre la calidad del frente de Pareto obtenido. Todas las variantes de PBIL pueden ser instanciadas desde MOEAD/D y han sido implementadas en C++.

5.1. Problema utilizado: mUBQP

El problema de optimización cuadrática binaria multi-objetivo sin restricciones (mUBQP) es uno de los problemas más difíciles en el ámbito de la optimización combinatoria multi-objetivo [8]. Ha despertado mucho interés, puesto que permite representar una amplia variedad de problemas combinatorios [17]. En mUBQP se define una colección de n elementos, y cada par de elementos se asocia con $m \geq 2$ valores de beneficio que pueden ser positivos, negativos, o cero. Formalmente, para cada objetivo k se define una matriz simétrica racional $Q^k = (q^k_{ij})$ de tamaño $n \times n$. El objetivo es encontrar un vector binario $X = \{x_1, ..., x_i, ..., x_n\}$, $x_i \in \{0, 1\}$ que maximice el valor de las m funciones objectivo [8]:

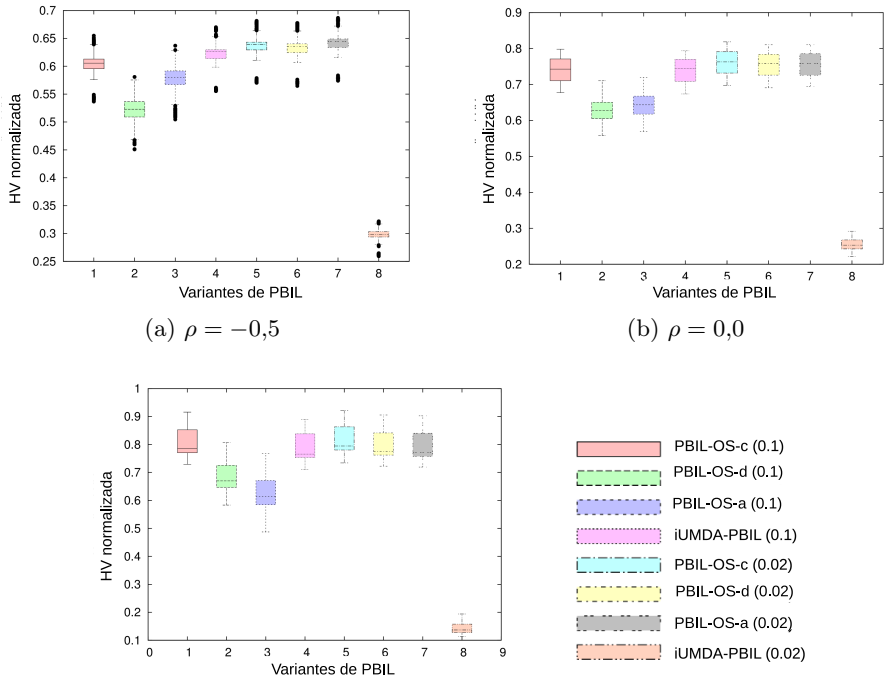
$$max\ F(X) = \sum_{i=1}^{n}\sum_{j=1}^{n} q^k_{ij} x_i x_j;\ k \in \{1, ..., m\};\quad t.q.\ x_i \in \{0, 1\};\ i \in \{1, ..., n\} \quad (8)$$

donde $F(x) = (f_1(X), ..., f_m(X))$ es un vector de funciones objetivo.

Hemos utilizado el generador de instancias mUBQP propuesto en [8]. Se han generado 30 instancias con un número fijo de variables, $n = 1000$, y $m = 2$ funciones objetivo. Se han utilizado distintos niveles de correlación entre ambos objetivos $\rho = \{-0{,}5, 0{,}0, 0{,}5\}$ (de más débil a más fuerte). Para cada valor de correlación se han generado 10 instancias diferentes.

5.2. Asignación de parámetros

Los parámetros de MOEA/D son los siguientes: $N = 201$ sub-problemas, el tamaño del vecindario de cruce $T_m = 30$, el tamaño del vecindario de reemplazo $T_r = 201$, y el máximo número de reemplazamientos $n_r = 2$. Para las variantes de PBIL, analizamos dos ratios de aprendizaje $\alpha = \{0{,}02, 0{,}1\}$. El primer valor es ampliamente utilizado en la literatura para la variante PBIL-OS, y el segundo para iUMDA-PBIL. Se han completado 30 ejecuciones para cada instancia. La condición de parada se basa 500 generaciones. utilizamos las medidas de rendimiento: el *Hipervolumen* (HV) y la cardinalidad (es decir, el número de soluciones presentes en PF). Además, se ha utilizado el test estadístico de *Kruskall-Wallis* para ordenar los resultados en base a HV.



(a) $\rho = -0{,}5$ (b) $\rho = 0{,}0$

(c) $\rho = 0{,}5$

Figura 1: Box plot de todos los valores HV obtenidos para los tres niveles diferentes de correlación $\rho = \{-0{,}5, 0{,}0, 0{,}5\}$. Cada columna es una combinación de una variante PBIL y $\alpha = \{0{,}1, 0{,}02\}$. El orden, de derecha a izquierda, es: PBIL-OS-c(0.1), PBIL-OS-d(0.1), PBIL-OS-a(0.1), iUMDA(0.1), PBIL-OS-c(0.02), PBIL-OS-d(0.02), PBIL-OS-a(0.02), iUMDA(0.02).

Cuadro 1: Ranking de *Kruskal-Wallis* para la medida HV

	PBIL-variant (α)							
	0,1				0,02			
ρ	PBIL-OS-c	PBIL-OS-d	PBIL-OS-a	iUMDA	PBIL-OS-c	PBIL-OS-d	PBIL-OS-a	iUMDA
-0,5	5	7	6	4	2	2,5	**1,5**	8
0,0	4	6,5	6,5	4	**2**	3	**2**	8
0,5	2,5	6	7	4	**1,5**	3,5	3,5	8

Cuadro 2: Media (desviación estandar) del tamaño de los frentes de Pareto

	PBIL-variant (α)							
	0,1				0,02			
ρ	PBIL-OS-c	PBIL-OS-d	PBIL-OS-a	iUMDA	PBIL-OS-c	PBIL-OS-d	PBIL-OS-a	iUMDA
-0,5	454,22	793,31	453,18	266,81	441,50	511,82	507,48	64,79
	(23,12)	(39,14)	(25,85)	(6,21)	(69,20)	(98,56)	(74,45)	(12,01)
0,0	344,86	411,10	385,57	197,30	337,51	368,20	351,13	36,07
	(40,09)	(21,30)	(32,14)	(8,29)	(37,05)	(50,20)	(22,04)	(4,81)
0,5	205,70	222,05	279,76	124,49	213,07	221,28	234,10	17,07
	(10,45)	(23,01)	(9,4)	(5,19)	(11,27)	(20,7)	(16,12)	(3,08)

5.3. Resultados numéricos

La Figura 1 presenta los valores de *HV* obtenidos para cada una de las variantes de PBIL y los dos ratios de aprendizaje α. El Cuadro 1 muestra el ranking en base a *Kruskal-Wallis* (nivel de significación del 5 %) de todas las ejecuciones y acorde al valor *HV*. Si el ranking final de dos o más variantes es el mismo, ello indica que no existen diferencias significativas entre ellas. Las variantes mejor clasificadas se muestran en negrita. Finalmente, el Cuadro 2 presenta los valores medios (y desviación estandar) de la medida *PF*.

Si nos fijamos en *HV*, la Figura 1 y el Cuadro 1 muestran que el comportamiento de PBIL-OS-(c,d,a) es significativamente diferente a iUMDA-PBIL. Además, PBIL-OS-c y PBIL-OS-a con $\alpha = 0{,}02$ son los que han conseguido los mejores resultados. En cuanto al tamaño de los frentes de Pareto finales, las tres variantes de PBIL-OS producen unos *PFs* más poblados que iUMDA-PBIL.

6. Conclusiones y trabajo futuro

Según nuestro conocimiento, este es el primer trabajo que estudia el comportamiento de las diferentes variantes de PBIL. Siendo éste uno de los AEDs más utilizados, creemos que se trata de una cuestión relevante, sobre todo si, como muestran los resultados, la variante utilizada influye de manera notable en el comportamiento del algoritmo. Además, también hemos estudiado el problema de manera analítica, derivando el valor esperado del vector de probabilidad para PBIL-OS. Hemos seleccionado un contexto de aplicación donde la diversidad producida por la variabilidad inherente a la variante de PBIL-OS tiene un efecto directo en la búsqueda. En dicho contexto, hemos mostrado que las variantes de PBIL son diferentes en esencia, hasta el punto de que diferentes valores del ratio de aprendizaje pueden provocar valores de hipervolumen estadísticamente diferentes. En la misma línea, los resultados han mostrado también que el tamaño medio del frente de Pareto es consistentemente mayor para PBIL-OS que para iUMDA-PBIL.

Como líneas de trabajo futuro: (i) Analizar el efecto de los diferentes mecanismos de aprendizaje sobre AEDs basados en modelos probabilísticos de mayor complejidad; (ii) estudiar el comportamiento de las variantes de PBIL sobre otros casos de estudio, tales como problemas del dominio continuo.

Agradecimientos

Este trabajo ha recibido el soporte del CNPq (Beca de productividad Nos. 305986/2012-0 y Programa Ciencia Sin Fronteras Nos. : 400125/2014-5) financiados por el CAPES (Gobierno Brasil), y de los programas IT-609-13 (Gobierno Vasco) y TIN2013-41272P (Ministerio de Ciencia e Innovacion).

Referencias

1. S. V. Ann. *Evolutionary multi-objective optimization using neural-based estimation of distribution algorithms*. PhD thesis, Department of Electrical & Computer Engineering. National University of Singapore, 2012.

2. S. Baluja. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical Report CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, PA, 1994.
3. S. Baluja. Genetic algorithms and explicit search statistics. *Advances in Neural Information Processing Systems*, 9:319–325, 1997.
4. S. Baluja and R. Caruana. Removing the genetics from the standard genetic algorithm. Technical report, Carnegie-Mellon University, Pittsburgh, 1995.
5. J. M. Chaves-González, M. A. Vega-Rodríguez, D. Domínguez-González, J. A. Gómez-Pulido, and J. M. Sánchez-Pérez. SS vs PBIL to solve a real-world frequency assignment problem in GSM networks. In *Applications of Evolutionary Computing*, pages 21–30. Springer, 2008.
6. I. Giagkiozis, R. C. Purshouse, and P. J. Fleming. Generalized decomposition. In R. C. Purshouse, P. J. Fleming, C. M. Fonseca, S. Greco, and J. Shaw, editors, *Evolutionary Multi-Criterion Optimization - 7th International Conference, EMO 2013, Sheffield, UK, March 19-22, 2013. Proceedings*, volume 7811 of *Lecture Notes in Computer Science*, pages 428–442. Springer, 2013.
7. C. González, J. A. Lozano, and P. Larrañaga. Analyzing the population based incremental learning algorithm by means of discrete dynamical systems. *Complex Systems*, 12(4):465–479, 2001.
8. A. Liefooghe, S. Verel, and J.-K. Hao. A hybrid Metaheuristic for Multiobjective Unconstrained Binary Quadratic Programming. *Appl. Soft Comp.*, 16:10–19, 2014.
9. A. Mendiburu, J. Miguel-Alonso, J. A. Lozano, M. Ostra, and C. Ubide. Parallel EDAs to create multivariate calibration models for quantitative chemical applications. *Journal of Parallel Distributed Computation*, 66(8):1002–1013, 2006.
10. H. Mühlenbein. The equation for response to selection and its use for prediction. *Evolutionary Computation*, 5(3):303–346, 1997.
11. H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions I. Binary parameters. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN IV*, volume 1141 of *Lectures Notes in Computer Science*, pages 178–187, Berlin, 1996. Springer.
12. R. Rastegar and M. R. Meybodi. A note on the population based incremental learning with infinite population size. In *Congress on Evolutionary Computation*, pages 198–205, 2005.
13. Z. Wang, Q. Zhang, G. Maoguo, and Z. Aimin. A Replacement Strategy for Balalancing Convergence and Diversity in MOEA/D. In *Proceedings of the Congress on Evolutionay Computation (CEC)*, pages 2132–2139. IEEE, 2014.
14. S. Yang and X. Yao. Experimental study on population-based incremental learning algorithms for dynamic optimization problems. *Soft Comp.*, 9(11):815–834, 2005.
15. B. Yuan and M. Gallagher. Playing in continuous spaces: Some analysis and extension of population-based incremental learning. In R. Sarker, R. Reynolds, H. Abbass, K. C. Tan, B. McKay, D. Essam, and T. Gedeon, editors, *Proceedings of the 2003 Congress on Evol. Computation CEC-2003*, pages 443–450. IEEE, 2003.
16. Q. Zhang and H. Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. on Evolutionary Computation*, 11(6):712–731, 2007.
17. Y. Zhou, J. Wang, and J. Yin. A Directional-biased Tabu Search Algorithm for Multi-objective Unconstrained Binary Quadratic Programming Problem. In *Proceedings of the 6th International Conference on Advanced Computational Intelligence (ICACI)*, pages 281–286. IEEE, 2013.
18. M. Zlochin, M. Birattari, N. Meuleau, and M. Dorigo. Model-based search for combinatorial optimization: A critical survey. *Annals of Operations Research*, 131(1-4):373–395, 2004.