El problema del ordenamiento óptimo de *buckets* usando metaheurísticas

Juan A. Aledo[1], José A. Gámez[2], and Alejandro Rosete[3]

[1] Departamento de Matemáticas, Universidad de Castilla-La Mancha, Albacete 02071, Spain, juanangel.aledo@uclm.es,
[2] Departamento de Sistemas Informáticos, Universidad de Castilla-La Mancha, Albacete 02071, Spain, jose.gamez@uclm.es,
[3] Instituto Superior Politécnico José Antonio Echeverría (Cujae), Marianao 19390, Havana, Cuba, rosete@ceis.cujae.edu.cu,

Resumen El problema del *ordenamiento óptimo de buckets* consiste en obtener el ranking completo (donde se admiten empates entre ítems) de consenso a partir de una base de datos de precedencias. En este trabajo abordamos ese problema mediante el uso de metaheurísticas, específicamente de estrategias evolutivas $(1 + \lambda)$. En particular, nuestro principal objetivo es analizar una serie de operadores de mutación (7) para este problema, algunos directamente tomados del caso en que la solución debe ser una permutación, y otros propuestos en este trabajo para abordar la nueva situación (empates). Además, se estudian distintas inicializaciones y estrategias de aplicación de las mutaciones. Las metaheurísticas resultantes son comparadas con heurísticas voraces *ad-hoc* propuestas en la literatura para este problema. De la experimentación sobre un conjunto de 15 bases de datos, se obtiene, en promedio, una ventaja de la mejor metaheurística respecto a la mejor heurística de alrededor de un 30 %.

Keywords: Bucket order, Kemeny, metaheurísticas, agregación de rankings, ranking de consenso

1. Introducción

El *problema de Kemeny (PK)* consiste en buscar la permutación con mayor consenso (menos desacuerdos) respecto a una base de datos de permutaciones [4,8]. El *problema del ordenamiento óptimo de buckets (POB)* [13] es una generalización del PK cuya solución es un ranking completo, en el que se admiten ítems empatados, que pasamos a describir/formalizar a continuación. Ambos problemas son NP-Completos. Tiene aplicación, por ejemplo, en consolidar búsquedas en internet [7], votaciones y estudios astronómicos[11].

Dado un conjunto de elementos $M = \{1, ..., n\}$, un *ordenamiento de buckets* B es una secuencia ordenada de k subconjuntos (buckets) $M_1, M_2, ..., M_k$ de M, $1 \leq k \leq n$, que constituyen una partición de M (esto es, son disjuntos dos a dos y la unión de todos ellos es M). Por comodidad, nos referiremos a los ordenamientos de buckets como *rankings*. Como notación usaremos una barra

© Ediciones Universidad de Salamanca

vertical | para separar los buckets y una coma para separar los elementos dentro de cada bucket. Por ejemplo, para $n = 4$ el ranking $1|3|2,4$ expresa que el 1 precede a los demás, que el 3 precede al 2 y al 4, y que para el 2 y el 4 que están contenidos en un mismo bucket, no hay preferencia entre ellos, están empatados.

Dados dos elementos $u, v \in M$ se considera que u antecede a v en el ranking B y se denota como $u \prec_B v$ si $u \in M_i$, $v \in M_j$ con $i < j$. Los elementos que pertenecen al mismo bucket en B se consideran no ordenados (o empatados). Si $u, v \in M$ pertenecen a un mismo bucket escribiremos $u \sim_B v$. En [13] se representa un ranking B mediante una matriz $n \times n$ donde $B(u, v) = 1$ si $u \prec_B v$, $B(u, v) = 0{,}5$ si $u \sim_B v$ y $B(u, v) = 0$ si $v \prec_B u$. En particular, nótese que $B(u, v) + B(v, u) = 1$ para cualesquiera $u, v \in M$ y que los elementos de la diagonal principal son todos igual a 0.5.

Sea C una matriz $n \times n$ tal que $C(u, v) \in [0, 1]$ $\forall u, v \in M$, $C(u, v)+C(v, u) = 1$ si $u \neq v$ y $C(u, u) = 0{,}5$. En nuestro contexto C se interpretará como una matriz de precedencias (posiblemente construida a partir de una base de datos que expresan preferencias), donde $C(u, v)$ se interpreta como la probabilidad de que u anteceda a v. Dadas dos matrices de precedencias C_1, C_2 de orden n, consideraremos la siguiente distancia matricial

$$D(C_1, C_2) = \sum_{u \neq v} |C_1(u, v) - C_2(u, v)|. \tag{1}$$

Así, dada una matriz de precedencias C el objetivo del POB consiste en encontrar el ranking B que minimiza la distancia respecto a la matriz C. Nótese que si el ranking B fuera una permutación se cumpliría que $B(u, v) \in \{0, 1\}$ para cualesquiera $u, v \in M$, $u \neq v$. Como además $B(u, v) + B(v, u) = 1$, siendo B una permutación se tiene que

$$D(C, B) = \sum_{u \neq v} |C(u, v) - B(u, v)| = 2 \sum_{u \prec_B v} C(v, u). \tag{2}$$

En particular, si C representa también una permutación en (2), entonces $D(C, B)$ es el doble de la distancia de Kendall entre las permutaciones C y B [9]. Así, si C es la matriz que expresa las precedencias de una base de datos de permutaciones (véase, por ejemplo, [3]) y pedimos que la solución sea también una permutación, POB y PK coinciden. En este sentido, el POB puede considerarse como una generalización del PK. Esta generalización de Kemeny al POB es diferente a la que representa el *problema de agregación de rankings* (RAP) [2,7], ya que en RAP se permiten empates (buckets) en los rankings de entrada, pero la solución debe ser una permutación. Sin embargo, no es difícil observar la conveniencia de permitir, soluciones que contengan empates (POB). Para ilustrar este comentario, consideremos la base de datos de permutaciones $\{1|2|3|4, 2|1|3|4, 1|2|4|3, 2|1|4|3\}$. Es obvio que todas las permutaciones coinciden en que $i \prec j$ para $i \in \{1, 2\}$ y $j \in \{3, 4\}$. Sin embargo, no hay consenso respecto a $1 \prec 2$, $2 \prec 1$, $3 \prec 4$ y $4 \prec 3$. Para esta base de datos, las mejores soluciones del PK serían las 4 permutaciones de la base de datos. Sin embargo, al menos conceptualmente, el ranking $1, 2|3, 4$ expresaría más adecuadamente las precedencias existentes en la base de datos.

© Ediciones Universidad de Salamanca

Para el POB se han desarrollado varios algoritmos heurísticos *ad-hoc*, siendo el algoritmo BPA [13] uno de los más reconocidos [10]. BPA recibe un conjunto de ítems I y una matriz de precedencias C y funciona de forma similar al algoritmo *quicksort*. Dado un conjunto de ítems se elige aleatoriamente uno de ellos, p, como pivote. En función de un umbral β el resto de ítems se dividen en tres conjuntos: $L = \{l_i \in I \mid C(l_i, p) > \beta\}$; $R = \{r_i \in I \mid C(p, r_i) > \beta\}$; y el resto $\{c_1, \ldots, c_c\}$ se sitúan en un bucket con p. El ranking a devolver se genera entonces llamando recursivamente a BPA: $BPA(L, C)|c_1, \ldots, c_c, p|BPA(R, C)$. Otro algoritmo heurístico que puede usarse para resolver el POB es el de Borda [6], ya que si bien fue diseñado para el problema de Kemeny, internamente rompe los empates arbitrariamente para generar una permutación, por tanto, si los empates obtenidos no se rompen, se genera un ranking con buckets.

En el presente artículo se propone el uso de metaheurísticas para abordar el POB. La sección 2 describe la propuesta. En particular, definimos 7 operadores (mutaciones) para trabajar con rankings que generalizan a algunos conocidos operadores para permutaciones. La sección 3 presenta un estudio experimental sobre 15 bases de datos tomadas de PrefLib[11] considerando 12 configuraciones de estrategias evolutivas $(1 + \lambda)$, combinando distintos tipos de inicialización y de estrategias de aplicación de las mutaciones.

2. Aproximación metaheurística al Problema de Ordenación de Buckets

Al igual que en BPA [13] asumimos que la entrada a nuestro problema es una matriz de precedencias C (véase la sección anterior). En nuestro caso, dicha matriz se obtiene a partir de una base de datos de permutaciones[4] y $C(u, v)$ refleja la proporción de permutaciones en las cuales $u \prec v$. Una solución potencial al problema o *bucket ranking* se representa como una lista ordenada de buckets, siendo cada bucket un conjunto no vacío de elementos. La función de adecuación o fitness, dado un ranking B, vendrá dada por:

$$f(B, C) = D(m(B), C) = \sum_{u \neq v} |m(B)(u, v) - C(u, v)|, \tag{3}$$

es decir, la aplicación de la distancia matricial definida en (1) entre C y la decodificación del ranking B en su matriz de preferencias $m(B)$. Evidentemente nuestro objetivo es encontrar el ranking B que **minimice** esta función. Nótese que en ocasiones se abusa de la notación, usándose B para referirnos tanto al ranking como a la matriz de precedencia $(m(B))$ que se obtiene a partir de él.

Además de estos dos elementos comunes a toda metaheurística (representación de soluciones y evaluación), describimos ahora la generación de soluciones iniciales y los operadores de mutación, por ser estos componentes clave en las *estrategias evolutivas*[12], la metaheurística seleccionada (ver Figura 1). Aunque

[4] También puede obtenerse a partir de rankings incompletos y/o con buckets [3]

© Ediciones Universidad de Salamanca

en su planteamiento original las estrategias evolutivas solo se aplicaban a problemas discretos, aquí se ha obviado este matiz pues el esquema general de la Figura 1 se ajusta a nuestra propuesta.

Figura 1. Estrategia Evolutiva $(1 + \lambda)$ (adaptada de [12])

Inicializar una población con 1 individuo (padre);
Evaluar el individuo;
Repetir
    Generar λ hijos a partir del padre;
    Evaluar los λ hijos;
    Remplazar el padre actual con el mejor entre los λ hijos y el padre;
Hasta que se cumple la condición de parada
Salida: la mejor solución encontrada.

Métodos de inicialización.- Hemos seleccionado cuatro métodos de construcción de soluciones iniciales, desde muy informados a aleatorios.

- BPA: Algoritmo heurístico de propósito específico para POB [13].
- Borda: El algoritmo heurístico de propósito específico para el problema de Kemeny [6], pero sin deshacer los empates que resultan de la suma de las precedencias para los ítems. Los ítems *empatados* van al mismo bucket.
- B1: Todos los elementos se ubican en único bucket.
- A: Se genera un ranking con bucket aleatoriamente. Primero se genera una permutación aleatoria. Después, se recorre la permutación de mayor a menor precedencia y cada ítem se sitúa en el mismo bucket que el ítem anterior o en un bucket diferente con probabilidades 0.5/0.5.

Operadores de mutación.- Existen muchos operadores de mutación definidos para trabajar con permutaciones [12] los cuales tienen diferentes comportamientos según el problema [5]. Como una permutación puede verse como un ranking con buckets de un elemento, comenzamos por adaptar los tres operadores de mutación más comúnmente usados en el espacio de permutaciones:

- Inserción de bucket: Mueve un bucket de una posición aleatoria a otra. Ejemplo: $1, 2|3|4, 5, 6|\mathbf{7}, \mathbf{8} \longrightarrow \mathbf{7}, \mathbf{8}|1, 2|3|4, 5, 6$
- Intercambio de buckets: Toma dos buckets aleatoriamente y los intercambia. Ejemplo: $\mathbf{1}, \mathbf{2}|3|4, 5, 6|\mathbf{7}, \mathbf{8} \longrightarrow \mathbf{7}, \mathbf{8}|3|4, 5, 6|\mathbf{1}, \mathbf{2}$
- Inversión de buckets: Selecciona dos posiciones aleatorias e invierte el orden de los buckets en esa sublista. Ejemplo: $\mathbf{1}, \mathbf{2}|\mathbf{3}|\mathbf{4}, \mathbf{5}, \mathbf{6}|7, 8 \longrightarrow \mathbf{4}, \mathbf{5}, \mathbf{6}|\mathbf{3}|\mathbf{1}, \mathbf{2}|7, 8$

Estos operadores no cambian ni el número de buckets, ni su tamaño o composición, sólo alteran su orden en el ranking. Por tanto, para mayor diversidad en el proceso de búsqueda, consideramos además otros 4 nuevos operadores:

© Ediciones Universidad de Salamanca

- Unión de buckets: Toma dos buckets consecutivos aleatorios y los fusiona. Ejemplo: $\mathbf{1}, \mathbf{2}|\mathbf{3}|4, 5, 6|7, 8 \longrightarrow \mathbf{1}, \mathbf{2}, \mathbf{3}|4, 5, 6|7, 8$
- División de un bucket: Selecciona un bucket aleatoriamente (de más de dos elementos) y lo divide aleatoriamente en dos nuevos buckets. Ejemplo: $1, 2|3|\mathbf{4}, \mathbf{5}, \mathbf{6}|7, 8 \longrightarrow 1, 2|3|\mathbf{4}, \mathbf{5}|\mathbf{6}|7, 8$
- Inserción de elemento: Toma un elemento (aleatorio) de un bucket (aleatorio) y lo inserta (aleatoriamente) en cualquier posición distinta del ranking, pudiendo agregarse a un bucket existente o convertirse en uno nuevo unitario. Nunca se inserta en el bucket de partida. Si el bucket de donde salió el elemento quedara vacío, se elimina. Ejemplo: $1, 2|3|\mathbf{4}, 5, 6|7, 8 \longrightarrow 1, 2, \mathbf{4}|3|5, 6|7, 8$.
- Intercambio de elementos: Selecciona aleatoriamente dos elementos de buckets distintos y los intercambia. Ejemplo: $1, 2|\mathbf{3}|\mathbf{4}, 5, 6|7, 8 \longrightarrow 1, 2|\mathbf{4}|\mathbf{3}, 5, 6|7, 8$.

Cabe señalar que siempre existe una secuencia de mutaciones de entre las anteriores que permite llegar desde un ranking arbitrario a cualquier otro.

3. Estudio experimental

Los experimentos se basan en las matrices de precedencia de 15 bases de datos (rankings completos y sin empates) tomadas de PrefLib[5] [11]. Estas bases de datos (del tipo *Election Data*) se describen en las primeras cinco columnas de la Tabla 1, mostrándose su identificador en PrefLib (BD); el número de elementos a ordenar (n); la proporción de celdas de C que están a distancia menor de 0.05 de 0 o 1 ($C_{\{0,1\}}$); la proporción que están a distancia menor de 0.05 de 0.5 ($C_{\{0,5\}}$); y la proporción que están a distancia mayor de 0.05 de 0, 0.5 o 1 ($\mathcal{D}$).

Creemos que estas medidas ilustran la complejidad de cada problema tanto en términos del espacio de búsqueda (n) como la proporción de precedencias que tienden a estar más definidas (hacia un empate o una ventaja) o más inciertas. Todos los experimentos descritos se realizaron en una computadora con un procesador Intel i7-6700 a 3,40 GHz, 8 núcleos y 16 Gb de memoria RAM, con sistema operativo Windows 8. Los algoritmos se implementaron en Prolog.

3.1. Comparación de métodos de inicialización

El primer experimento se orientó a comparar los métodos de inicialización o construcción de soluciones iniciales descritos en la sección 2. Como BPA y A son estocásticos, se realizaron 20 repeticiones para cada uno de ellos. La tabla 1 (columnas 6-9) muestra los resultados obtenidos (promedio para BPA y A). Los resultados corroboran lo que podría esperarse, así, analizando los resultados de la tabla 1 se observa que BPA obtiene la mejor solución en 10 problemas, Borda en 4 y, sorprendentemente, $B1$ en 1 ocasión. Es lógico que las heurísticas específicas para la búsqueda de rankings de consenso obtengan soluciones mucho mejores, y también lo es comprobar (a partir de la media) que el algoritmo diseñado *ad-hoc* para tratar con buckets (BPA) es mucho mejor que la adaptación de Borda.

[5] En PrefLib (http://www.preflib.org/) las matrices se codifican en ficheros pwg

© Ediciones Universidad de Salamanca

Tabla 1. Bases de Datos y comparación entre heurísticas de construcción

BD	n	$C_{\{0,1\}}$	$C_{\{0,5\}}$	$\mathcal{D}$	Borda	BPA	$B1$	A
14.1	10	0	0,22	0,78	30,94	17,14	**14,08**	38,08
15.48	10	0,23	0	0,77	21,67	**15,85**	26,33	40,73
6.11	20	0,62	0	0,38	19,11	**14,64**	170,89	190,96
6.12	20	0,07	0	0,93	9	**5,67**	180,22	188,16
15.74	20	0,41	0	0,59	56,67	**53,7**	137,33	188,87
6.46	30	0,76	0	0,24	30,29	**22,36**	406,14	431,55
15.67	30	0,34	0,25	0,41	159	**148,6**	287	414,95
15.46	40	0,36	0,22	0,43	261	**249**	533	764,85
15.65	40	0,22	0,24	0,54	358,5	**343,6**	454,5	776,9
15.55	52	0,37	0,15	0,48	**423**	478,35	944	1327,55
15.66	52	0,4	0,16	0,44	**398,5**	453,05	961,5	1348,1
15.54	60	0,34	0,24	0,42	632,5	**554,05**	1170,5	1747,9
15.16	70	0,28	0,2	0,52	**931**	935,95	1547	2450,85
15.69	81	0,34	0,18	0,48	**1109**	1288,85	2227	3218,65
15.42	100	1	0	0	2081	**1898,6**	2956	4945,6

Como también podría esperarse, los métodos $B1$ y A son mucho peores, si bien es interesante ver como $B1$ que representa la solución de máxima incertidumbre, es sistemática y significativamente mejor que el método que construye los rankings de buckets de forma aleatoria.

En base a estos resultados, se realizaron pruebas estadísticas usando la herramienta Keel [1]. La prueba de Friedman asigna el siguiente ranking promedio a los algoritmos: BPA(1.33), Borda(1.8), $B1$(2.87) y A(4), obteniéndose un p-valor de 0 que implica que al menos hay algún algoritmo diferente al resto. El análisis post-hoc revela que no hay diferencia estadísticamente significativa ($\alpha = 0{,}05$) entre BPA y Borda, pero si entre ellos y $B1$ y A. También se aprecia diferencia a favor de $B1$ con respecto a A. Vale la pena observar que BPA supera a Borda en los 7 problemas más pequeños, mientras que los problemas con más de 40 nodos BPA gana 5 veces y Borda gana 4. Eso podría sugerir que el aumento del tamaño del problema afecta más a BPA que a Borda. Para verificarlo, se analizó el coeficiente de correlación de Pearson entre las columnas de la tabla 1 que describen a los conjuntos de datos (n, $C_{\{0,1\}}$, etc.) y las que representan los resultados obtenidos por BPA y Borda. Una correlación positiva (resp. negativa) significa que según crece la métrica empeora (resp. mejora) el comportamiento del algoritmo. Si es cercano a 0 es que no existe casi influencia. Del estudio realizado podemos deducir que Borda tiende a funcionar relativamente mejor según crecen n (correlación de -0.57) y $C_{\{0,1\}}$ (-0.39), pero empeora cuanto mayor sea la proporción de celdas dudosas $\mathcal{D}$ (0,51). Por su parte, BPA tiende a funcionar mejor según crece $C_{\{0,1\}}$ (-0.34) pero empeora según crecen $C_{\{0,5\}}$ (0.34) y $\mathcal{D}$ (0.21). Para finalizar, se debe indicar que Borda es el algoritmo de inicialización que más se ve afectado por los parámetros que describen las bases de datos y BPA el menos sensible, pues el promedio de los valores absolutos para los coe-

© Ediciones Universidad de Salamanca

ficientes de correlación es (0.40) para Borda, seguido por A (0.30), $B1$ (0.29) y BPA (0.24).

3.2. Comparación de las variantes metaheurísticas

El segundo experimento se orientó al estudio del comportamiento de los métodos anteriores de inicialización combinado con diferentes formas de evolucionarlos usando las mutaciones definidas en la sección 2. Se ha elegido $\lambda = 7$ para la estrategia evolutiva (ver Figura 1 por ser este el número de mutaciones definidas para estudiar diferentes estrategias de aplicación de ellas. En todos los casos, se comienza generando una población inicial con una solución (usando alguno de los métodos de inicialización descritos en la sección 2). Se experimentaron 3 estrategias de aplicación de las mutaciones: *Aleatoria* (*Ale*), escoger el tipo de mutación a aplicar y sus parámetros de forma aleatoria; *Todas* (*Tod*), los siete operadores de mutación son aplicados, generando un hijo cada uno; y *Única* (*Una*), generando los siete hijos mediante el mismo operador de mutación, y cambiándolo cíclicamente para cada generación. Las estrategias *Ale* y *Tod* hacen que la vecindad sea potencialmente mayor, mientras que la estrategia *Una* estaría más cerca de una búsqueda con vecindad variable.

Combinando los 4 métodos de inicialización (Borda, BPA, $B1$ y A) con las 3 estrategias de aplicación de las mutaciones (*Ale*, *Tod* y *Una*) se obtienen 12 configuraciones: *Borda-Ale*, *Borda-Una*, *Borda-Tod*, BPA-*Ale*, BPA-*Una*, BPA-*Tod*, *B*1-*Ale*, *B*1-*Una*, *B*1-*Tod*, *A-Ale*, *A-Una*, *A-Tod*. Para cada base de datos y configuración se realizaron 20 repeticiones y se permitieron hasta $10n^2$ evaluaciones de la función objetivo, para aumentar la exploración según aumenta el espacio de búsqueda. Los resultados se muestran en la Tabla 2. Se muestra el promedio de las mejores soluciones en cada repetición y, entre paréntesis, el % que representa este valor respecto al valor obtenido por la respectiva heurística de construcción (Tabla 1). Si en una casilla los tres algoritmos (*Ale*, *Tod*, *Una* con igual inicialización) obtuvieron el mismo resultado, se muestra una sola vez.

Basaremos nuestras conclusiones en el análisis estadístico realizado. La prueba de Friedman devolvió el siguiente orden de los algoritmos con sus rankings promedios: Borda-*Una* (3.2), Borda-*Ale* (4.33), Borda-*Tod* (5.07), *A-Una* (5.37), *A-Tod* (5.47), *A-Ale* (5.57), BPA-*Tod* (7.57), *B*1-*Tod* (7.57), BPA-*Una* (7.83), *B*1-*Una* (8.3), BPA-*Ale* (8.47), *B*1-*Ale* (9.37). El p-valor obtenido fue menor de 0.000003 lo cual demuestra que al menos existe un algoritmo significativamente diferente en el grupo. Vistos los resultados del experimento anterior, resulta curioso que en los primeros puestos estén los algoritmos basados en Borda y A con bastante separación respecto a los basado en BPA y $B1$. De hecho, Borda-*Una* obtiene el mejor promedio en 13 de las bases de datos, seguido por Borda-*Ale* que lo obtiene en 7, y Borda-*Tod*, *A-Ale* y *A-Tod* que lo obtienen en 6. Nótese que en varias de las bases de datos de menor dimensión hay empates. Usando Borda-*Una* como referencia (o control) se realizó el análisis post-hoc usando todos los tests presentes en Keel a tal efecto, detectándose que Borda-*Una* es significativamente superior a las 6 configuraciones basadas en $B1$ y BPA según

© Ediciones Universidad de Salamanca

Tabla 2. Comparación de las configuraciones de las estrategias evolutivas $(1 + \lambda)$

BD	*Borda-Ale* *Borda-Una* *Borda-Tod*	BPA-*Ale* BPA-*Una* BPA-*Tod*	*B*1-*Ale* *B*1-*Una* *B*1-*Tod*	*A-Ale* *A-Una* *A-Tod*
14.1	**13,09** (42,31 %)	**13,09** (76,36 %)	**13,09** (92,94 %)	**13,09** (34,38 %)
15.48	**13** (60 %)	**13** (82,02 %)	**13** (49,37 %)	**13** (31,91 %)
6.11	**14,22** (74,42 %)	14,23 (97,19 %)	14,43 (8,44 %)	**14,22** (7,45 %)
	14,22 (74,42 %)	14,24 (97,27 %)	**14,22** (8,32 %)	**14,22** (7,45 %)
	14,22 (74,42 %)	14,23 (97,19 %)	**14,22** (8,32 %)	**14,22** (7,45 %)
6.12	**5,67** (62,96 %)	**5,67** (100 %)	**5,67** (3,14 %)	**5,67** (3,01 %)
15.74	39,72 (70,09 %)	40,05 (74,58 %)	40,08 (29,19 %)	39,83 (21,09 %)
	39,47 (69,65 %)	40,02 (74,52 %)	40,33 (29,37 %)	39,77 (21,06 %)
	39,78 (70,21 %)	39,92 (74,33 %)	40,03 (29,15 %)	39,78 (21,06 %)
6.46	**19,29** (63,68 %)	19,3 (86,33 %)	**19,29** (4,75 %)	**19,29** (4,47 %)
	19,29 (63,68 %)	19,3 (86,33 %)	**19,29** (4,75 %)	19,3 (4,47 %)
	19,29 (63,68 %)	19,31 (86,39 %)	**19,29** (4,75 %)	**19,29** (4,47 %)
15.67	105 (66,04 %)	107,8 (72,54 %)	109,1 (38,01 %)	107,45 (25,89 %)
	104,75 (65,88 %)	108,6 (73,08 %)	108,65 (37,86 %)	107,35 (25,87 %)
	106,2 (66,79 %)	107,85 (72,58 %)	107,35 (37,4 %)	107,75 (25,97 %)
15.46	**170** (65,13 %)	171,3 (68,8 %)	172,4 (32,35 %)	**170** (22,23 %)
	170 (65,13 %)	171,35 (68,82 %)	172,4 (32,35 %)	170,9 (22,34 %)
	170 (65,13 %)	170,9 (68,63 %)	172,4 (32,35 %)	170,45 (22,29 %)
15.65	248,4 (69,29 %)	255,4 (74,33 %)	257,05 (56,56 %)	250,1 (32,19 %)
	245,35 (68,44 %)	251,1 (73,08 %)	251,95 (55,43 %)	249,15 (32,07 %)
	252,6 (70,46 %)	249,35 (72,57 %)	245,4 (53,99 %)	251,4 (32,36 %)
15.55	297,35 (70,3 %)	300,35 (62,79 %)	319,3 (33,82 %)	298,2 (22,46 %)
	292,65 (69,18 %)	299,15 (62,54 %)	297,9 (31,56 %)	295,15 (22,23 %)
	296,5 (70,09 %)	299,9 (62,69 %)	306,3 (32,45 %)	296,35 (22,32 %)
15.66	293,7 (73,7 %)	301,35 (66,52 %)	305,7 (31,79 %)	298,3 (22,13 %)
	292,15 (73,31 %)	301,3 (66,5 %)	297,6 (30,95 %)	295,7 (21,93 %)
	294,25 (73,84 %)	303,5 (66,99 %)	297,95 (30,99 %)	297,1 (22,04 %)
15.54	413,25 (65,34 %)	414,8 (74,87 %)	416,25 (35,56 %)	411,2 (23,53 %)
	411,15 (65 %)	418,25 (75,49 %)	416,4 (35,57 %)	411,2 (23,53 %)
	412,5 (65,22 %)	416,45 (75,16 %)	412,45 (35,24 %)	**411** (23,51 %)
15.16	**673** (72,29 %)	693,2 (74,06 %)	701,7 (45,36 %)	687,25 (28,04 %)
	690,6 (74,18 %)	700,3 (74,82 %)	688,6 (44,51 %)	686,45 (28,01 %)
	679,8 (73,02 %)	694,4 (74,19 %)	711,1 (45,97 %)	695 (28,36 %)
15.69	814,6 (73,45 %)	834,6 (64,76 %)	833,6 (37,43 %)	821 (25,51 %)
	811,7 (73,19 %)	830,9 (64,47 %)	901,3 (40,47 %)	826,05 (25,66 %)
	816,25 (73,6 %)	835,75 (64,84 %)	848,85 (38,12 %)	830,05 (25,79 %)
15.42	1387,1 (66,66 %)	1358,65 (71,56 %)	1374,25 (46,49 %)	1372,65 (27,75 %)
	1355,75 (65,15 %)	1358,8 (71,57 %)	1436,5 (48,6 %)	1378,55 (27,87 %)
	1374,25 (66,04 %)	1358,45 (71,55 %)	1401,75 (47,42 %)	1362,55 (27,55 %)

© Ediciones Universidad de Salamanca

todos los postprocesamientos (p-valores menores de 0.02). En la comparación entre Borda-Una y los demás métodos basados en Borda (Borda-Tod y Borda-Ale) o en inicio aleatorio (A-Ale,A-Tod,A-Una) todos los p-valores son superiores a 0.1, por lo que no existen diferencias significativas.

En general se puede concluir que los métodos con inicio de Borda (y luego los de inicio aleatorio) son los mejores. Es interesante observar que los métodos basados en BPA, la heurística de inicialización más informada, a pesar de la ventaja inicial que les otorga la mejor inicialización, finalmente son superados por los inicializados con Borda y A. Si se observa los valores entre paréntesis en la tabla 2 se puede ver que estos mejoran menos la solución inicial, quedándose en un 76 % del resultado de BPA (mejoría de alrededor de un 24 %), si bien esto puede ser razonable debido a la mejor calidad del punto de inicio con respecto a las otras inicializaciones, el hecho de verse superadas finalmente por ellas nos hace pensar que las soluciones iniciales propuestas por BPA son *demasiado* buenas, estando en la zona de atracción de óptimos locales inferiores (según los operadores usados) a las zonas de atracción de las soluciones generadas por Borda. La mejoría lograda por Borda es de alrededor de un 34 % que combinado con que parten de soluciones bastante buenas y cercanas en evaluación a las de BPA resulten en una combinación mejor. Las mejorías obtenidas por los métodos basados en inicio aleatorio son de alrededor del 78 % lo cual les permiten no diferenciarse significativamente de los basados en Borda. Por último, los métodos basados en $B1$ logran mejorías de alrededor de un 64 % pero no llegan a acercarse significativamente a los mejores. La estrategias de aplicación de las mutaciones Una es la que mejores resultados obtiene por escaso margen, logrando que las soluciones obtenidas sean las que más mejoran respecto al inicio en las variantes Borda y A; y Tod en las BPA y $B1$. Sin embargo, los valores de mejoría promedio son muy cercanas: Una (49,73 %), Tod (49,71 %) y Ale (49,67 %). La ligera ventaja de Una es congruente con que la mejor combinación sea Borda-Una, pero el estrecho margen también explica porque hay combinaciones basadas en las tres estrategias tanto entre los mejores como entre las peores configuraciones.

Por último, se puede comparar a las heurísticas con las metaheurísticas de dos maneras: la mejor de las heurísticas (en negrita en la Tabla 1) contra la mejor de las metaheurísticas (en negrita en la Tabla 2) en cada problema o la mejor heurística globalmente (BPA) contra mejor metaheurística globalmente (Borda-Una). Las metaheurísticas son 29.6 % y un 34.9 % mejores en promedio, respectivamente. Ambas mejorías aumentan con la dimensión del problema (coeficiente de correlación de Pearson entre n y las mejorías de 0.59 y 0.55, respectivamente) y con $C_{\{0,5\}}$ (0,51 y 0.59), pero decrece con $\mathcal{D}$ (-0.44 y -0.29).

4. Conclusiones

En este trabajo se ha propuesto una solución basada en metaheurísticas para el POB. Se han definido 7 operadores de mutación para este problema y se han probado 4 formas de inicialización y 3 estrategias para aplicar las mutaciones. Los resultados muestran que las variantes basadas en el algoritmo de Borda (sin

© Ediciones Universidad de Salamanca

romper los empates) con la estretegia de aplicar la misma mutación para generar todos los hijos de una generación obtiene los mejores resultados, aunque otras variantes obtienen resultados cercanos. Un resultado interesante es que la mejor heurística de construcción (BPA) no fue la que mejores resultados obtuvo como método de inicialización. La solución presentada basada en estrategias evolutivas $(1 + \lambda)$ logra mejorar en alrededor del 30 % a las mejores soluciones obtenidas por las heurísticas ad-hoc en los problemas.

Agradecimientos

Este artículo ha sido parcialmente financiado por la Junta de Comunidades de Castilla-La Mancha, la Universidad de Castilla-La Mancha y el Fondo Europeo de Desarrollo Regional mediante las ayudas PEII-2014-049-P y CCI nº 2014ES16RFOP010.

Referencias

1. Alcalá-Fdez,J., Fernandez,A., Luengo,J., Derrac,J., García,S., Sánchez,L., Herrera.,F.: KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework, Journal of Multiple-Valued Logic and Soft Computing, 2011, 17:2-3, 255–287
2. Aledo, J.A., Gámez, J.A., Molina,D.: Using metaheuristic algorithms for parameter estimation in generalized Mallows models, Applied Soft Computing , 2016, 38, 308 – 320
3. Aledo, J.A., Gámez, J.A., Molina, D.: Using extension sets to aggregate partial rankings in a flexible setting, (submitted), 2016.
4. Ali, A., Meila, M.: Experiments with Kemeny ranking: What works when? (2012) Mathematical Social Sciences, 64 (1), pp. 28-40;
5. Ceberio, J., Irurozki, E., Mendiburu, A., Lozano, J. A.: A review of distances for the Mallows and Generalized Mallows estimation of distribution algorithms, Computational Optimization and Applications, 2015, 62 (2), 545–564
6. Borda, J.: Memoire sur les elections au scrutin, Histoire de l'Academie Royal des Sciences; 1781, 657–665
7. Dwork, C., Kumar, R., Naor, M., Sivakumar, D.: Rank aggregation methods for the web (2001), Proceedings of the 10th International Conference on World Wide Web, WWW'01, Hong Kong, pp. 613-622;
8. Kemeny, J.L., Snell, J.G.: Mathematical Models in the Social Sciences, Blaisdell, New York, 1962
9. Kendall, M.G.: A new measure of rank correlation, Biometrika, 30, pp. 81-93, 1938.
10. Kenkre,S., Khan,A., Pandit,V.: On discovering bucket orders from preference data, Proc. of the SIAM International Conference on Data Mining, pp. 872-883, 2011
11. Mattei, N., Walsh, T.: PrefLib: A Library of Preference Data http://preflib.org, Proceedings of the 3rd International Conference on Algorithmic Decision Theory (ADT 2013), LNAI, pp. 259–270, Springer, 2013.
12. El-Ghazali Talbi: Metaheuristics: From Design to Implementation, Wiley, 2009
13. Ukkonen, A., Puolamki, K., Gionis, A., MannilaH.: A randomized approximation algorithm for computing bucket orders, Information Processing Letters, 2009, 109(7), 356 – 359

© Ediciones Universidad de Salamanca