Evaluación parcial en el problema de agregación de rankings

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 Este trabajo presenta un estudio teórico y experimental sobre la aplicación de la evaluación parcial en el *problema de agregación de rankings (RAP)*. La evaluación parcial solo calcula la parte de la función objetivo afectada por la aplicación de ciertos operadores. En particular, se estudian los operadores habituales en búsqueda local de inserción, intercambio e inversión. El estudio teórico muestra que el costo cuadrático de la evaluación de la función objetivo puede reducirse a lineal aplicando la evaluación parcial tras la inserción o el intercambio. El estudio experimental valida las expresiones teóricas y muestra que para los operadores de inserción e intercambio se puede dividir el tiempo de evaluación por 10 (30) para problemas con más de 50 (100) elementos. En el caso de la inversión, la ganancia es mucho menor.

Keywords: metaheurísticas, operadores de vecindad, RAP, evaluación parcial

1. Introducción

Las *funciones sustitutas (surrogates)* son versiones alternativas a la función objetivo original pero con menor costo computacional [6]. En este trabajo, nuestro foco está en las funciones sustitutas que son equivalentes a la original y no en sus versiones aproximadas [6,9,13]. Sin aproximar la función objetivo, uno de los caminos para la reducción del coste computacional es la concepción de estructuras de datos eficientes para cada problema [15,21]. Otra posibilidad es detener la evaluación de la función objetivo (*early stopping*) cuando la solución no sea relevante para la búsqueda, lo cual ha sido aplicado p.e. en el trazado de grafos [17] y la planificación de tareas [4].

En [4] se comenta que la evaluación de la función objetivo en el problema de planificación de máquinas puede hacerse solo considerando los cambios introducidos por los operadores. Esto también había sido empleado eficientemente en [11] para problemas de programación binaria y en su generalización al caso bipartito [12], estudiando el efecto del cambio de valor de una variable binaria. A esto le llamaron *evaluación incremental*.

Actas de la XVII Conferencia de la Asociación Española para la Inteligencia Artificial, pp. 221-230

En [19] se presentó un algoritmo heurístico para el *Linear Ordering Problem (LOP)* que emplea una estructura de datos basada en grafos para representar las permutaciones y evaluar de manera eficiente la vecindad que genera el operador de inserción en este problema. Más recientemente, en [18] se emplean los algoritmos anteriores dentro de un contexto de uso de metaheurísticas. En [8] se desarrolla un método eficiente para el problema LOP basado en analizar las operaciones de inserción, descartando las no prometedoras. Otro trabajo interesante en este contexto es [7], donde se estudian varias mutaciones[4] sobre permutaciones y establecen semejanzas entre la búsqueda local y los EDA. También estudian cómo afectan ciertos operadores a la función objetivo de diversos problemas definidos sobre permutaciones.

Este trabajo se enfoca en el problema RAP (*Rank aggregation Problem* o *Problema de Agregación de Rankings*) [2,3,10] cuya función objetivo tiene una complejidad cuadrática en el número de items a rankear, $O(n^2)$ Nuestro objetivo es definir una forma de evaluar esta función objetivo que sea computacionalmente más eficiente. El principio que se empleará, al que llamaremos **evaluación parcial**, se asemeja al de la evaluación incremental [11,12] y consiste en no evaluar una solución desde cero, sino hacerlo considerando la información que aporta la solución previa y la semántica del operador empleado. Así, se sustituye la evaluación completa de la función objetivo por una evaluación parcial que evalúa las diferencias entre las soluciones. Para esto, hay que determinar la afectación que produce en la evaluación de la función el empleo de ciertos operadores, como se había hecho en [7] para otros problemas. Aquí nos centraremos en la evaluación parcial después de aplicar mutaciones a soluciones potenciales para el RAP.

El artículo está organizado de la manera siguiente. La Sección 2 analiza la complejidad computacional cuadrática de la función objetivo del problema RAP. La Sección 3 muestra cómo puede reducirse ese costo aplicando evaluación parcial y se obtienen expresiones que demuestran que la complejidad computacional de la evaluación parcial basada en los operadores de inserción y de intercambio se reduce en el caso promedio de cuadrática a lineal. La sección 4 presenta un estudio experimental que corrobora las expresiones teóricas. Finalizamos en la Sección 5 presentando nuestras conclusiones.

2. Costo computacional de la evaluación en RAP

Un ranking expresa un orden de preferencias entre un conjunto de elementos $[[n]] = \{1, ..., n\}$. Un ranking completo y sin empates es una permutación sobre $[[n]]$. Denotamos por $\mathcal{S}_n$ al conjunto de permutaciones de n elementos.

Dado un conjunto de rankings $\Pi = \{\pi_1, \ldots, \pi_N\}$, se puede calcular una matriz Q de modo que Q_{ij} refleje las precedencias entre cada par de elementos i y j en los rankings de Π. Las formas de calcular Q_{ij} van desde contar la cantidad de veces que i antecede a j en el conjunto de rankings (cuando estos son permutaciones) hasta formas más complejas que consideran los casos de

[4] En este trabajo se usarán los términos mutación y operador local o de vecindad indistintamente

empates y ausencias [1,10]. En lo que sigue, asumimos como entrada la matriz Q, independientemente de cómo ésta fue calculada. La función a optimizar es

$$f(P) = \sum_{i \prec_P j} Q_{ji}, \qquad P \in \mathcal{S}_n \tag{1}$$

donde $i \prec_P j$ se cumple si i antecede a j en la permutación P. Por ejemplo, cuando los rankings π_k son permutaciones y Q_{ij} representa las veces que i antecede a j en las permutaciones de Π, la función f mide el número de desacuerdos de P con respecto a Π (distancia de Kendall) y la permutación P que minimiza f es la solución al problema de Kemeny [14].

Llamaremos $R_P = \{(i,j) : i \prec_P j,\, i,j \in [[n]]\}$ al conjunto de relaciones de precedencia presentes en P. Así, podemos reescribir la función f como

$$f(P) = \sum_{(i,j) \in R_P} Q_{ji}, \tag{2}$$

Para una permutación de n elementos, $|R_P| = \sum_{i=0}^{n-1} i = \frac{n(n-1)}{2}$ lo cual implica un crecimiento cuadrático de la evaluación de la función objetivo, siendo necesario evaluar y sumar una cantidad de casos $E(n) = \frac{n(n-1)}{2}$. Esto no pasa en otros problemas que trabajan con permutaciones, como p.e. el TSP, que se puede evaluar en tiempo lineal porque la función objetivo solo tiene en cuenta las relaciones de adyacencia en la permutación, no las precedencias. Dadas dos permutaciones cualesquiera P_1 y P_2, el conjunto $R_{P_1} \cap R_{P_2}$ solo sería vacío cuando se trate de una permutación y su inversa. En general

$$f(P_1) = \sum_{(i,j) \in R_{P_1} \cap R_{P_2}} Q_{ji} + \sum_{(i,j) \in R_{P_1} - R_{P_2}} Q_{ji} \tag{3}$$

La primera suma es común para los dos miembros del par, de donde

$$f(P_2) = f(P_1) - \sum_{(i,j) \in R_{P_1} - R_{P_2}} Q_{ji} + \sum_{(i,j) \in R_{P_2} - R_{P_1}} Q_{ji} \tag{4}$$

Si $|R_{P_1} \cap R_{P_2}| \gg |R_{P_1} - R_{P_2}| + |R_{P_2} - R_{P_1}|$ y si se tuviera el valor $f(P_1)$ ya calculado, entonces usando (4) podría realizarse una evaluación de $f(P_2)$ con un menor costo computacional que si se usara directamente (2). El costo computacional de emplear (4) teniendo ya el valor de $f(P_1)$ depende de la cantidad de precedencias que diferencien a ambas permutaciones. Estas precedencias a evaluar se corresponden con accesos a la matriz Q y sumas, tanto para determinar la cantidad que se sumará como la que se restará empleando (4). De (4) puede entenderse que el costo de evaluar P_2 en función de P_1 será más eficiente computacionalmente mientras más elementos tengan en común ambas permutaciones.

En la próxima sección se derivan fórmulas que expresan la cantidad esperada de elementos comunes entre dos permutaciones y el valor esperado de la cantidad de precedencias a evaluar. En este trabajo se ignora la posibilidad de emplear eficientemente que $Q_{ij} + Q_{ji} = 1$ para cualesquiera $i,j \in [[n]]$, $i \neq j$, como

pasa en el problema de Kemeny. En este caso las dos sumas de (4) tendrían una estrecha relación ya que si Q_{ij} aparece en una, entonces $Q_{ji} = 1 - Q_{ij}$ aparecerá en la otra. Es decir, $(i,j) \in R_{P1} - R_{P2}$ si y solo si $(j,i) \in R_{P_2} - R_{P_1}$. Por tanto, podría reescribirse (4) como

$$f(P_2) = f(P_1) + \sum_{(i,j)\in R_{P_2}-R_{P_1}} Q_{ji} - Q_{ij} = f(P_1) + \sum_{(i,j)\in R_{P_2}-R_{P_1}} 2Q_{ji} - 1 \quad (5)$$

Esto implica que todas las reducciones aplicando evaluación parcial que se estudian en este trabajo podrían hacerse en la mitad del tiempo, ya que usando (5) habría la mitad de accesos y la mitad de sumas. En lo que sigue se obviará esta posibilidad para analizar el caso más general del problema RAP.

3. Afectación a las precedencias por las mutaciones y costo computacional de la evaluación parcial

Para calcular cuántas precedencias son comunes entre dos permutaciones, habría que calcular la distancia de Kendall entre ambas permutaciones [2,3]. Esto implica una complejidad cuadrática $O(n^2)$, aunque en algunos casos concretos, como es la comparación de dos rankings, se han logrado algoritmos mejores que no llegan a ser lineales [5]. Sin embargo, en tiempo lineal se pueden obtener los elementos comunes al inicio y al final de las permutaciones de n elementos.

Supongamos que una permutación P_2 se obtuvo de una permutación P_1 aplicando una mutación y que usamos dos listas L_1 y L_2 para representar ambas permutaciones. Esta listas se pueden expresar como la concatenación de una lista inicial común (I) , con una lista intermedia diferente en cada permutación (S_1 y S_2), más una lista final común (F). Es decir, $L_1 = I \odot S_1 \odot F$ y $L_2 = I \odot S_2 \odot F$, donde el símbolo $\odot$ expresa la concatenación de dos listas. Nótese que las listas S_1 y S_2 son dos permutaciones sobre el mismo conjunto de m elementos. En estos casos, las únicas relaciones de precedencia que podrían ser distintas entre P_1 y P_2 son las que hay entre los elementos presentes en S_1 y S_2.

Podemos determinar el tamaño esperado $\overline{m_n}$ de la lista S_1 (y S_2) para una permutación de n elementos. Como la mutación afecta a S_1 para convertirla en S_2, a S_1 le llamaremos *lista afectada* y por tanto $\overline{m_n}$ es el tamaño esperado de la lista afectada para una permutación de n elementos. Nos centramos en tres operadores de mutación muy usados en las metaheurísticas cuando se trabaja con permutaciones:

- El operador de inserción: toma un elemento de una posición aleatoria y lo coloca en otra posición aleatoria.
- El operador de intercambio: toma dos elementos de posiciones aleatorias diferentes y los intercambia.
- El operador de inversión: escoge una sublista aleatoria y la invierte [20].

Independientemente de cómo operan, el tamaño esperado de la lista afectada para los tres operadores es el mismo, porque ellos comienzan seleccionando aleatoriamente dos puntos i y j, $i < j$, entre los cuales se producirá la afectación.

De esta manera, la lista I tendría $i-1$ elementos, la lista F tendría $n-j$ elementos, mientras S_1 tendría $j-i+1$ elementos. Por ejemplo, para $n=8$, $L_1=1,2,3,4,5,6,7,8$, $i=3$ y $j=7$, quedaría $I=1,2$, $S_1=3,4,5,6,7$, $F=8$.

Como i y j se seleccionan aleatoriamente, i sigue una distribución uniforme entre 1 y $n-1$, mientras que j sigue una distribución uniforme entre $i+1$ y n. Todas estas combinaciones de valores son equiprobables y se puede determinar el tamaño esperado $\overline{m_n}$ de la lista afectada S_1 sumando todos los tamaños posibles y dividiéndolos entre la cantidad total de listas posibles. La mayor lista afectada ocurre cuando $i=1$ y $j=n$ con tamaño n. Igualmente, se puede notar que habría 2 listas de tamaño $n-1$ cuando (i,j) valen $(1,n-1)$ y $(2,n)$, 3 listas de tamaño $n-2$ cuando (i,j) valen $(1,n-2)$, $(2,n-1)$ y $(3,n)$, y así sucesivamente hasta $n-1$ listas de tamaño 2. En general, para una permutación de tamaño n habrá $K(n,m)=n-m+1$ listas de tamaño m, $2\leq m\leq n$. A partir de eso se puede determinar:

$$\begin{aligned}\overline{m_n} &= \frac{\sum_{m=2}^{n} mK(n,m)}{\sum_{m=2}^{n} K(n,m)} = \frac{[\sum_{m=1}^{n} m(n-m+1)]-n}{[\sum_{m=1}^{n}(n-m+1)]-n} \\ &= \frac{[(n+1)\sum_{m=1}^{n} m]-[\sum_{m=1}^{n} m^2]-n}{[(n+1)\sum_{m=1}^{n} 1]-[\sum_{m=1}^{n} m]-n} \\ &= \frac{[\frac{(n+1)^2 n}{2}]-[\frac{n(n+1)(2n+1)}{6}]-n}{[n(n+1)]-[\frac{n(n+1)}{2}]-n} = \frac{n(n-1)(n+4)}{3n(n-1)} = \frac{n+4}{3}\end{aligned}$$

Aunque el tamaño de la lista afectada sea el mismo para los operadores de inversión, intercambio y de inserción, la cantidad de precedencias afectadas (y por tanto la cantidad de casos a evaluar aplicando evaluación parcial) no es el mismo, ya que su semántica tiene distintos efectos en las precedencias dentro de la lista afectada. La inversión p.e. altera el orden relativo de todos los elementos de la sublista afectada y, por tanto, se afectan todas las precedencias, mientras que el operador de inserción solo afecta las relaciones de precedencia con respecto al elemento movido. A continuación se analiza cada caso.

Inversión.- Es el operador que afecta a más precedencias. De hecho, para una lista afectada de tamaño m, se afectan todas las precedencias entre sus elementos, $m(m-1)/2$, ya que todos cambian su orden relativo. Por ejemplo, si la lista afectada (de tamaño $m=5$) por una inversión es $S_1=3,4,5,6,7$; la lista tras la inversión sería $S_2=7,6,5,4,3$ y habría que buscar los $\frac{5\cdot 4}{2}=10$ valores Q_{ji} sumados en P_1 y que dejarán de estar en P_2: Q_{43}, Q_{53}, Q_{63}, Q_{73}, Q_{54}, Q_{64}, Q_{74}, Q_{65}, Q_{75} y Q_{76}. Igualmente, habría que buscar los nuevos valores a sumar para evaluar P_2, asociados a precedencias que no había en P_1: Q_{34}, Q_{35}, Q_{36}, Q_{37}, Q_{45}, Q_{46}, Q_{47}, Q_{56}, Q_{57} y Q_{67}. Esta afectación de 10 precedencias de S_1 provoca el doble de accesos a la matriz Q y de sumas, porque hay que calcular ambas sumas mostradas en (4). A esta combinación de accesos y sumas le llamamos *casos a evaluar*, y está directamente vinculado al costo computacional de la evaluación. En general, para una lista afectada de tamaño m la cantidad de casos a evaluar tras una inversión será, según hemos razonado, $2\frac{m(m-1)}{2}=m(m-1)$.

Actas de la XVII Conferencia de la Asociación Española para la Inteligencia Artificial, pp. 221-230

Análogamente al cálculo del valor esperado para $\overline{m_n}$, se puede estimar la cantidad de casos a evaluar para un tamaño $\overline{m_n}$ luego de una inversión $E_r(n)$ como

$$E_r(n) = \overline{m_n}(\overline{m_n} - 1) = \left(\frac{n+4}{3}\right)\left(\frac{n+4}{3} + 1\right) = \frac{n^2+5n+4}{9} \qquad (6)$$

Aunque (6) es cuadrática al igual que (2), el coeficiente del término cuadrático es menor ($\frac{1}{9}$ vs $\frac{1}{2}$), lo cual implica ya una ventaja.

Intercambio.- En la mutación de intercambio se mueven solo los elementos extremos de la lista afectada, por lo que ésta tendría la forma $S1 = A \odot W \odot B$ donde A y B son los elementos intercambiados. Los $m-2$ elementos de la lista W no cambian su orden relativo, por lo que aquí solo se afectan las $n-2$ relaciones entre A y los miembros de W, las $n-2$ relaciones entre B y los miembros de W, y la predecencia entre A y B. Esto permite ver que la cantidad de precedencias afectadas es $2(m-2)+1 = 2m-3$. Como se dijo antes, por cada precedencia afectada hay que evaluar dos casos, por lo que la cantidad esperada de casos a evaluar para un tamaño $\overline{m_n}$ luego de un intercambio $E_x(n)$ es

$$E_x(n) = 2(2\overline{m_n} - 3) = 4\left(\frac{n+4}{3}\right) - 6 = \frac{4n-2}{3} \qquad (7)$$

Resultando en un coste computacional de orden *lineal*.

Inserción.- En este caso la lista afectada puede tener dos formas: $S1 = A \odot W$ o $S1 = W \odot A$, donde A es el elemento que se movió y W es el resto de la lista. Las precedencias que cambian son las $m-1$ precedencias de A respecto a los $m-1$ elementos de W. Al haber dos casos por cada precedencia afectada, la cantidad esperada de casos a evaluar para un tamaño $\overline{m_n}$ luego de una inserción $E_i(n)$ es

$$E_i(n) = 2(\overline{m_n} - 1) = 2\left(\frac{n+4}{3}\right) - 2 = \frac{2n+2}{3} \qquad (8)$$

De nuevo resulta un coste computacional de orden *lineal*, pero mejorando el coeficiente con respecto a intercambio.

4. Estudio experimental

El objetivo de los experimentos realizados fue comprobar las expresiones teóricas obtenidas y evaluar el efecto real tanto en la disminución de la cantidad esperada de casos a evaluar como en el tiempo de ejecución. Se escogieron 30 bases de datos de PrefLib[16] con cantidades de elementos entre 10 y 242. Se escogieron bases de datos de permutaciones (ranking completos sin ausencias) y se construyeron las matrices de precedencia Q a partir de los ficheros pwg disponible en PrefLib. Concretamente, se usaron las siguientes bases de datos, todas de tipo Election Data: ED-00006-Skate Data(2, 11, 12, 46), ED-00011-Web Search(1), ED-00014-Sushi Data(1) y ED-00015-Clean Web Search(1, 2, 3,

4, 7, 14, 12, 16, 20, 23, 24, 41, 46,48 , 54, 55,65, 66, 69, 42, 40, 32, 67, 74). La intención de la selección fue tomar una muestra con valores de n distribuidos en el intervalo $[10, 250]$, dentro de los disponibles en PrefLib. El grupo escogido incluye la siguiente cantidad de bases de datos para los 18 valores distintos de n: 10(2), 20(3), 30(2), 40(2), 52(2), 60(1), 70(2), 81(1), 91(1), 100(2), 110(1), 122(1), 131(1), 142(1), 153(1), 163(1), 240(3), 242(3). Otros detalles de estas bases de datos están disponibles en PrefLib.org.

Para cada conjunto de datos se realizaron 100 repeticiones del siguiente método: se generó una permutación aleatoria, se le aplicó una mutación de cada tipo (inserción, intercambio e inversión) y para cada una de ellas se registró la cantidad de casos a evaluar y la proporción que representa el tiempo de demora de la evaluación completa respecto a la evaluación parcial. Luego se promediaron los resultados de las 100 repeticiones. Como la ejecución no depende en sí de los valores concretos en la matriz Q, se promediaron los valores que correspondían al mismo valor de n y así se obtuvieron los siguientes estadísticos: casos a evaluar después de inserción E'_i, de intercambio E'_x y de inversión E'_r; proporción de tiempo de evaluación completa respecto a la evaluación parcial después de una inserción (T'/T'_i), de intercambio (T'/T'_x) y de inversión (T'/T'_r). Cada una de estas series de datos incluye 18 valores asociados a cada n. Para los análisis que siguen también se calculó la cantidad de casos a evaluar usando evaluación completa E y evaluación parcial luego de inserción E_i, intercambio E_x e inversión E_r que predicen las expresiones obtenidas en la Sección 3.

Todos los experimentos se realizaron en una computadora con un procesador Intel i7-6700 a 3.40 GHz con 8 núcleos y 16 Gb de memoria RAM. El sistema operativo es Windows 8 y los algoritmos se implementaron en Prolog.

El primer experimento se centró en la cantidad de casos a evaluar para cada valor de n. Para analizar la relación entre los valores medidos y los teóricos, se calculó el coeficiente de correlación de Pearson: 0.994 entre E_i y E'_i, 0.998 entre E_x y E'_x y 0.989 entre E_r y E'_r. El promedio de la diferencia (error) entre cada par de series fue del 5 % de los valores absolutos. La prueba de Wilcoxon no detectó diferencias significativas entre los pares de series ($\alpha = 0.05$). Se buscaron las tendencias de cada una de las series experimentales y se obtuvieron las siguientes ecuaciones:

$$\begin{array}{ll} E'_i(n) = 0.662n - 0.474 & (R^2 = 0.989) \\ E'_x(n) = 1.34n - 1.8 & (R^2 = 0.996) \\ E'_r(n) = 0.128n^2 + 1.74n - 127.73 & (R^2 = 0.978) \end{array}$$

Las ecuaciones son muy parecidas a las teóricas, sobre todo en los coeficientes de mayor grado de los polinomios, corroborándose de este modo las tendencias lineales predichas para E_i y E_x así como la cuadrática para E_r. En resumen, se observa una gran semejanza entre los valores teóricos y los experimentales.

El segundo experimento se centró en estimar las proporción de tiempo que consume la evaluación completa respecto a la evaluación parcial. Para ello, se analizaron las series obtenidas (T'/T'_i,T'/T'_x y T'/T'_r). Se muestra a proporción de tiempo de un tipo de evaluación respecto a la otra porque esta medida no de-

pende de las características del entorno de ejecución, como pasaría con cualquier medición absoluta de tiempo. La figura 1 muestra estas proporciones.

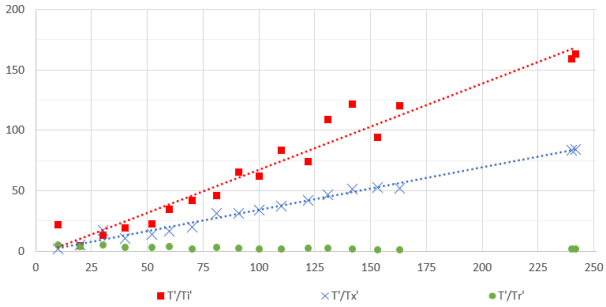

Figura 1. Proporción de tiempo de evaluación parcial respecto a la evaluación completa

Para la mutación de inversión se obtuvo una línea de tendencia:

$$T'/T'_r = -0.013n + 4.079\,(R^2 = 0.559)$$

Esta tendencia experimental marcada por la constante 4.079 es muy cercana a la que podría preverse de la relación entre los coeficientes de los términos de mayor grado en (2) y (6), ya que $\frac{9}{2} = 4.5$. De hecho, el promedio de la serie es de 4.062 y la mediana es de 4.229. Esto implica que la evaluación completa es alrededor de 4 veces más lenta que la evaluación parcial después de una inversión. Es sobretodo interesante analizar las tendencias para la inserción y el intercambio. Sus líneas de tendencia fueron:

$$T'/T'_i = 0.732n - 6.677 \qquad (R^2 = 0.96)$$
$$T'/T'_x = 0.352n - 1.267 \qquad (R^2 = 0.985)$$

Para ambas se observa una tendencia claramente lineal que muestra que mientras crece n también crece la cantidad de veces que la evaluación completa es más lenta que la parcial. Dado el coste cuadrático de la evaluación completa, esta esta relación confirma la tendencia lineal del tiempo en la evaluación parcial. A manera de ejemplos, para $n = 50$ la evaluación completa consume 23 veces más tiempo que la parcial luego de una inserción y 14 veces si es un intercambio. Para $n = 100$ es 62 veces más lenta que la evaluación parcial luego de la inserción y 34 veces si es un intercambio. Las proporciones del total de casos (E) respecto a los casos a evaluar usando evaluación parcial (E'_i,E'_x y E'_r) siguen una relación parecida que no se muestra por razones de espacio.

Adicionalmente, considerando todos los valores experimentales de las proporciones de tiempo (T'/T'_i, T'/T'_x, T'/T'_r) y de casos a evaluar (E/E'_i, E/E'_x, E/E'_r) se obtuvieron dos series de datos que resumen todas las proporciones de tiempos (T/T'_p) y casos (E/E'_p) de la evaluación completa respecto a la parcial. El valor de correlación entre ambas series fue de 0.88. Más aún, se pudo encontrar una tendencia $T/T'_p = 0.8717(E/E'_p) + 5.934 (R^2 = 0.78)$ que expresa la relación lineal entre casos y tiempo. Esto confirma que los pasos adicionales necesarios en la evaluación parcial (p.e. análisis del inicio y final común) no cambian la tendencia teórica que se veía en términos de casos a evaluar. El efecto final es una reducción notable en el tiempo de evaluación de la función objetivo.

5. Conclusiones

Este trabajo ha demostrado, de forma teórica y experimental, que en el problema RAP se pueden lograr reducciones notables en el tiempo de evaluación de las soluciones candidatas si se aplica evaluación parcial, explotando el hecho de que una buena parte de la función no tiene que ser reevaluada. Para los operadores de inserción y de intercambio se demostró que la reducción supone pasar de orden cuadrático a lineal, obteniéndose evaluaciones 10 (30) veces más rápidas para valores de n igual a 50 (100). En el caso de la inversón, el coste se mantiene cuadrático pero se divide por dos la constante, obteniéndose evaluaciones 4 veces más rápidas. Estos resultados pueden ser útiles para resolver el problema RAP con metaheurísticas que empleen mutaciones como pueden ser la Búsqueda Tabú, el Recocido Simulado, la Ascensión de Colinas, las Estrategias Evolutivas, etc. Se pretende en el futuro, extender este tipo de análisis para otros operadores de mutación (y cruce), tanto en el problema RAP como en otros problemas relacionados que son muy costosos computacionalmente.

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. Ailon, N., Charikar, M., Newman, A.: Aggregating inconsistent information: ranking and clustering, Journal of the ACM, 55 (5), 23-27, 2008.
2. Aledo, J.A., Gámez, J.A., Molina, D.: Using extension sets to aggregate partial rankings in a flexible setting, submitted, 2016.
3. Ali, A., Meila, M.: Experiments with Kemeny ranking: What works when? Mathematical Social Sciences, 64 (1), pp. 28-40, 2012

4. Avalos-Rosales, O., Angel-Bello, F., Alvarez, A.: Efficient metaheuristic algorithm and re-formulations for the unrelated parallel machine scheduling problem with sequence and machine-dependent setup times, The International Journal of Advanced Manufacturing Technology, 76(9), 1705–1718, 2014.
5. Bansal, M. S. , Fernández-Baca, D.: Computing distances between partial rankings, Information Processing Letters, 109 (4), 238 - 241, 2009.
6. Brownlee, A. E.I., Woodward, J. R., Swan, J.: Metaheuristic design pattern: surrogate fitness functions, Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation,1261–1264,ACM,New York, USA, 2015.
7. 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, 62(2), 545–564, 2015.
8. Ceberio, J., Mendiburu, A., Lozano, J.A.: The linear ordering problem revisited, European Journal of Operational Research, 241 (3), 686–696, 2015.
9. Chow, Joseph Y. J. and Regan, Amelia C.: A surrogate-based multiobjective metaheuristic and network degradation simulation model for robust toll pricing, Optimization and Engineering, 15(1), 137–165, 2013.
10. Dwork, C., Kumar, R., Naor, M., Sivakumar, D.: Rank aggregation methods for the web, Proceedings of the 10th International Conference on World Wide Web, WWW'01, Hong Kong, pp. 613-622, 2001.
11. Glover, F., Jin-Kao H.: Efficient evaluations for solving large 0-1 unconstrained quadratic optimisation problems, Int. J. Metaheuristics, 1(1), 3-10, 2010.
12. Glover, F., Tao, Y., Punnen, A.P., Kochenberger,G.A.: Integrating tabu search and VLSN search to develop enhanced algorithms: A case study using bipartite boolean quadratic programs,European Journal of Operational Research,697-707,2015.
13. Jin, Y.: A comprehensive survey of fitness approximation in evolutionary computation, Soft Computing, 9(1), 3–12, 2003.
14. Kemeny, J.L., Snell, J.G.: Mathematical models in the social sciences, Blaisdell, New York, 1962
15. Klusáček, D., Rudová, H.: A metaheuristic for optimizing the performance and the fairness in job scheduling systems, Artificial Intelligence Applications in Information and Communication Technologies, 3–29, 2015.
16. 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), Lecture Notes in Artificial Intelligence, pp. 259–270, Springer, Bruxelles, Belgium, November 12-14, 2013.
17. Rosete, A., Ochoa, A., Sebag,M.: Efficient-discarding fitness functions, Proc. of Late-Breaking Papers presented at Genetic and Evolutionary Computation Conference, GECCO-99, Orlando, U.S.A., 1999.
18. Sakuraba, C. S., Ronconi, D. P., Birgin, E. G., Yagiura, M.: Metaheuristics for large-scale instances of the linear ordering problem, Expert Systems with Applications, 42(9), 4432 - 4442, 2015.
19. Sakuraba, C. S., Yagiura, M.: Efficient local search algorithms for the linear ordering problem, International Transactions in Operational Research, 17(6), 711–737, 2010.
20. Talbi, E. G.: Metaheuristics: From design to implementation, Wiley, 2009
21. Thomas, J., Chaudhari, N. S.: A new metaheuristic genetic-based placement algorithm for 2D strip packing, Journal of Industrial Engineering International, 10(1), 1–16, 2014.