USO DE UN ALGORITMO STACKELBERG-EVOLUTIVO PARA RESOLVER EL PROBLEMA DE FIJACIÓN DE CUOTAS EN UNA RED DE TRANSPORTE

Pamela Jocelyn Palomo Martínez
Co autor: José Fernando Camacho Vallejo
UANL-FCFM
Universidad Autónoma de Nuevo León
Facultad de Ciencias Físico Matemáticas
San Nicolás de los Garza, Nuevo León, México

Resumen:
El problema de fijación de cuotas en una red de transporte de bienes múltiples TOP (por sus siglas en inglés, Toll Optimization Problem) se modela como una problema binivel en el cual el líder busca determinar un conjunto de tarifas que se asignan a determinados arcos de dicha red y el seguidor debe elegir por cuáles arcos transportar los bienes, sabiendo que debe pagar las cuotas establecidas por el líder. El siguiente trabajo presenta un algoritmo Stackelberg-Evolutivo para resolver el TOP que estudia la interacción entre el líder y el seguidor como un juego de Stackelberg; asímismo, explota los principios de la computación evolutiva, en la cual se realizan cambios aleatorios entre los individuos de una población de soluciones factibles del problema, seleccionando individuos que contribuyen en mayor medida a mejorar la utilidad de los agentes del juego de Stackelberg. Dicha selección permite que en cada iteración las mejores soluciones tengan mayor probabilidad de sobrevivir, permitiendo que las utilidades del líder y del seguidor vayan mejorando su calidad.

Palabras claves:
optimización de cuotas, algoritmos evolutivos, programación binivel

Introducción

El problema de fijación de cuotas óptimas en una red de transporte involucra a dos agentes: el líder, quien es el dueño de la red y cuyo objetivo consiste en maximizar la ganancia que obtendrá al cobrar tarifas a los usuarios de la red por utilizar un subconjunto de los arcos de ella; y, por otro lado, el seguidor, papel tomado por los usuarios que buscarán minimizar el costo de transportar bienes a través de la red. Se supone que entre cada par de nodos entre los cuales los usuarios desean transportar bienes, llamados par origen-destino, existe un camino formado por arcos libres de cuota; además, el líder no podrá establecer cuotas negativas a los arcos. Así, el problema reside en que el líder debe seleccionar las tarifas de modo que asegure que los usuarios de la red utilizarán los arcos a los que se les asignó cuota y no el camino libre.

La modelación original del TOP aparece por primera vez en 1998 [1], desde entonces han surgido numerosos estudios acerca de la complejidad del problema, se han hecho reformulaciones del mismo y se han estudiado métodos de solución exactos y heurísticos. Por el lado de la complejidad del TOP, se ha demostrado que el TOP es un problema *NP-hard* [1]. Posteriormente, se llegó a la conclusión de que el TOP es un problema fuertemente *NP-hard* [2][3]. Con respecto a las reformulaciones del TOP, se han encontrado nuevas desigualdades válidas [4]. Por otro lado, se realizó una remodelación de la red basándose en caminos más cortos [5]. Además, se ha reformulado el TOP como un programa matemático con restricciones de equilibrio MPEC (por sus siglas en inglés, Mathematical Program with Equilibrium Constraints) al considerar el flujo dinámico del tráfico en la red [6]. Por otro lado, el TOP se puede reformular como un problema entero mixto MIP (por sus siglas en inglés, Mixed Integer Program) si se linealizan las condiciones de holgura complementaria; de este modo, el problema puede ser resuelto por métodos tradicionales para los MIP.

En el campo de los métodos exactos, se ha analizado al TOP como un problema combinatorio al considerar que maximizar la utilidad del líder es compatible con una solución determinada del seguidor, la cual será uno de los caminos que puede elegir; logrando llegar al óptimo de instancias medianas y aproximándose al óptimo de instancias grandes [7]. Bajo el mismo enfoque, se desarrolló un generador de caminos eficiente, logrando obtener cotas superiores para la función objetivo del líder [8]. Por otro lado, se implementó un algoritmo de función de penalización, un algoritmo quasi-Newton, un algoritmo basado en aproximación de gradiente y algoritmo de búsqueda directa haciendo uso del método símplex de Nelder-Mead [9].

Por el lado de los métodos heurísticos, se desarrolla y se aplica a redes que transportan un único bien un heurístico primal-dual basándose en penalizaciones de la función objetivo del nivel superior [10]. Posteriormente, extienden el algoritmo a redes de múltiples bienes y se aplica a instancias pequeñas [11]. Se desarrolló también un heurístico que busca la convergencia a óptimos globales [12 y un algoritmo genético buscando también óptimos globales [13]; además de un algoritmo de búsqueda directa que logró convergencia a resultados similares que el genético, pero en menor tiempo [14]. Por último, se desarrolla un algoritmo de búsqueda tabú que resultó eficiente en instancias grandes [15].

Modelo matemático

Sea N un conjunto de nodos, A un conjunto de arcos y sea G(N,A) un grafo conformado por los conjuntos N y A, que representará a la red de transporte de bienes múltiples. Consideremos $A_1 \subset A$ como el conjunto de los arcos a los que el líder les asignará una tarifa y a $A_2 \subset A$ como el conjunto de los arcos libres de cuota; además, cada arco es libre o se le asigna tarifa y no existen arcos que sean libres y a la vez de cuota, es decir $A_1 \cup A_2 = A$ y $A_1 \cap A_2 = \emptyset$. Cada arco $a \in A$ tiene un costo fijo asociado por utilizarlo c_a; asímismo, t_a será el costo que el líder asignará a cada arco $a \in A_1$ y denotaremos como x_a^k al flujo de transporte en el arco a asociado con el bien $k \in K$, donde K es el conjunto de bienes que se desean transportar. Por otra parte, al considerar a n^k como la demanda entre el nodo origen o(k) y el nodo destino $d(k)$, podemos asociar a cada nodo $i \in N$ con el bien $k \in K$ y con n^k de la siguiente manera:

$$b_i^k = \begin{cases} n^k, & \text{si } i = o(k) \\ -n^k, & \text{si } i = d(k) \\ 0, & \text{en otro caso} \end{cases}$$

Por último, consideremos como i^+ al conjunto de arcos de A que tienen al nodo i como cabeza e i^- será el conjunto de arcos que tienen al nodo i como cola.

La modelación matemática del TOP será la siguiente:

$$\max_{t,x} \sum_{k \in K} \sum_{a \in A_1} t_a x_a^k \qquad (1)$$

$$\text{sujeto a:} \quad 0 \le t_a \le t_a^{max}, \quad \forall a \in A_1 \tag{2}$$

$$\forall k \in K:$$

$$x_a^k \in Arg\min \sum_{a \in A_1} (c_a + t_a)x_a + \sum_{a \in A_2} c_a x_a \tag{3}$$

$$\text{sujeto a:} \quad -\sum_{a \in i^-} x_a + \sum_{a \in i^+} x_a = b_i^k, \quad \forall i \in N \tag{4}$$

$$x_a^k \ge 0, \quad \forall a \in A \tag{5}$$

La expresión (1) indica la función objetivo del líder, la cual es la ganancia que obtiene al cobrar a los usuarios las tarifas que fije. La expresión (2) muestra que las tarifas no pueden ser negativas y que, además, no pueden exceder a un límite máximo t_a^{max}. Por otro lado, (3) indica la función objetivo del seguidor, la cual es su costo de transporte. La expresión (4) es una ecuación de conservación de flujo en la red. Por último, (5) indica que no existen flujos negativos.

Algoritmo Stackelberg-Evolutivo

Al resolver un modelo binivel, el objetivo es optimizar la función objetivo del líder, pero esta no solo depende de sus variables de decisión, sino también de las variables de decisión del seguidor. Es posible evaluar a una población de soluciones factibles si analizamos la interacción entre el líder y el seguidor como un juego de Stackelberg. Consideremos a cada una de los individuos de una población como un conjunto de estrategias del líder, entonces el seguidor tendrá su función de reacción que consiste en elegir las variables de decisión que minimicen su función objetivo dada la solución del nivel superior. El líder será capaz de anticipar las reacciones del seguidor para cada una de sus estrategias, entonces es posible que elija la solución que maximice su función. El objetivo entonces, será buscar un equilibrio de Stackelberg entre el líder y el seguidor. En la **Figura 1** se muestra el pseudocódigo del algoritmo Stackelberg-Evolutivo.

Primero es necesario determinar un número de generaciones y la cantidad de elementos que existirán en la población inicial *T*. Por otro lado, es necesario determinar el valor de t_a^{max}, si es que este no está predeterminado. Una forma sencilla de encontrar esta cota es la siguiente: sea k una matriz de dimensiones $n \times 2$, donde *n* es el número de bienes que se desean transportar en la red y sea *k(i,1)* el nodo de origen de transporte del bien i y el nodo destino será *k(i,2),* con *i=1,...,n.* Se procede a buscar d_1 la distancia del camino más corto

```
Proceso Algoritmo Stackelberg-Evolutivo
    Leer Generaciones;
    Generar la población inicial T;
    Evaluación de Stackelberg de T;
    Para i←1 Hasta Generaciones Hacer
        Realizar torneos;
        Seleccionar a la población de padres T';
        Cruza de T' generando los hijos T";
        Mutación de los hijos generando T"';
        Evaluación de Stackelberg de T"';
        Selección de T de la generación i+1;
    FinPara
FinProceso
```

Figura 1. Pseudocódigo del Algoritmo Stackelberg-Evolutivo.

entre *k(i,1)* y *k(i,2)* y d_2 la distancia del camino más largo libre de cuota entre *k(i,1)* y *k(i,2)*. Entonces, una cota *t(i)* estará dada por (6).

$$t(i) = d_2 - d_1 \tag{6}$$

Finalmente, la cota t_a^{max} estará dada por la ecuación (7).

$$t_a^{max} = \max_i \{t(i)\} \tag{7}$$

Después de ello se procede a realizar los pasos siguientes:

Generar la población inicial T: En el caso del TOP, las soluciones del líder son tarifas que debe fijar al conjunto A_1 de arcos, por ello la representación elegida para los individuos de la población inicial *T* será un vector de $|A_1|$ componentes continuas entre 0 y t_a^{max}; de este modo, cada componente representará la cuota asignada a cada arco $a \in A_1$ y los límites 0 y t_a^{max} asegurarán la factibilidad de la solución. Una vez establecida la forma de los individuos de *T* se generan aleatoriamente tantos

vectores de cuotas como se haya decidido. Se puede notar que la cota t_a^{max} encontrada no es una cota para los arcos, sino para cada uno de los caminos entre los pares origen-destino. En consecuencia, se eligió una distribución de probabilidad triangular que favorezca la aparición de cantidades más cercanas al límite aproximado para cada arco que a t_a^{max} para generar a la población inicial. La distribución triangular tiene la moda que se muestra en la ecuación (8) con $i=1,\ldots,n$.

$$moda = \min_i promedio(i) \tag{8}$$

en donde:

$$promedio(i) = \frac{t(i)}{\text{número de arcos de } camino_1} \tag{9}$$

En donde $camino_1$ es el camino más corto con cuota entre $k\,(i,1)$ y $k\,(i,2)$.

Evaluación de Stackelberg de T: Sea t_j un vector de cuotas perteneciente a la población T, con $j=1,\ldots,|T|$. Para realizar la evaluación de Stackelberg del individuo t_j debe resolverse el problema de programación que consiste en (3) - (5), es decir, el problema del seguidor. Al ser conocidos los costos fijos c_a y las tarifas asignadas por el líder, el problema del seguidor se convierte en un problema de programación lineal cuyas variables de decisión son los flujos x_a^k. Posteriormente, al conocer los valores de x_a^k, se calcula el valor de la función objetivo del líder dada por (1) a la cual denotaremos por $lider(j)$. La evaluación de Stackelberg debe realizarse para cada uno de los individuos $t_j \in T$.

Realizar torneos: Al inicio de los torneos, cada uno de los individuos de T cuenta con cero victorias; es decir, $victorias_j=0$ para $j=1,\ldots,|T|$. Para cada individuo $t_j \in T$ se elije para competir con él a otro individuo $t_k \in T$, donde k es un entero aleatorio procedente de una distribución uniforme tal que $1 \leq k \leq |T|$ y, además, $j \neq k$. A continuación, se comparan los valores de $lider(j)$ y $lider(k)$, si $lider\ (j) \geq lider\ (k)$, entonces t_j obtendrá una victoria, es decir, $victorias(j)=victorias(j)+1$; en *el caso contrario, si* $lider(j)<lider(k)$, *tendremos que* $victorias(k)=victorias(k)+1$. El proceso se repite por el número de torneos deseado, no es conveniente realizar una gran cantidad de ellos, ya que entre más repeticiones haya del proceso, cada individuo tenderá a competir la misma cantidad de veces debido a la ley de los grandes números; consecuentemente, los individuos que hayan tenido los mayores valores en la función objetivo serán los que ganen más torneos convirtiendo a la selección en un proceso determinista. En el algoritmo diseñado para el presente artículo se realizaron cinco torneos.

Seleccionar a la población de padres T': Para realizar la selección de los individuos de T que se convertirán en padres T', se procede a ordenar en una lista a los individuos de T en orden descendiente según el número de victorias que hayan obtenido. Los individuos que se encuentren en las primeras $\frac{|T|}{2}$ posiciones serán los individuos que formarán parte de la población T'.

Cruza de T' generando los hijos T'': Para cada individuo $t'_j \in T'$ con $j = 1,\ldots,|T'|$ se selecciona de manera aleatoria a otro individuo t'_k donde k es un entero procedente de una distribución uniforme tal que $1 \leq k \leq |T'|$ y, además, $j \neq k$. Adicionalmente, se elige un entero aleatorio l procedente de una distribución uniforme tal que $1 \leq l \leq |A_1|$ al cual llamaremos punta de cruza. Los individuos t'_j y t'_k se reproducirán para crear a un hijo $t''_j \in T''$ donde T'' será la población de hijos de T'. Las componentes $1,\ldots, l$ del hijo t''_j serán las componentes $1,\ldots, l$ de t'_j, del mismo modo, las componentes $l+1,\ldots, |A_1|$ del vector t''_j serán las mismas que las del vector t'_k.

Mutación de los hijos generando T''': En el algoritmo desarrollado para el presente artículo, una mutación consistirá en una alteración de una componente de un individuo de T'' y para ello se utilizó una probabilidad de mutación de 0.1. Para llevar a cabo el proceso de la mutación, para cada individuo $t''_j \in T''$ se selecciona un número aleatorio l procedente de una distribución uniforme continua tal que $0 \leq l \leq 1$. Si $l > 0.1$, el individuo t''_j pasará a formar parte de la población T''' sin sufrir modificaciones y se denotará como t'''_j; por el contrario si $l \leq 0.1$, el individuo t''_j sufrirá una mutación, la cual se realiza generando un entero aleatorio m procedente de una distribución uniforme tal que $1 \leq m \leq |A_1|$; de modo que la componente m de t''_j será la que mutará.

El primer paso para la mutación de la componente m de t''_j es elegir aleatoriamente si se sumará o restará una cantidad a dicha componente. Posteriormente, se elige una cantidad aleatoria procedente de una distribución uniforme continua que será sumada o restada a la componente según se haya decidido; dicha cantidad debe estar entre 0 y un tamaño máximo de mutación. Para este artículo, el tamaño máximo de la mutación, denotado por $tam\_mutacion$, fue seleccionado de modo que una mutación permita explorar una nueva región del espacio factible del problema y está dado por (10).

$$tam\_mutacion = 0.1 \times t\_max \tag{10}$$

Al individuo resultante de la mutación lo denotaremos por t'''_j y pasará a formar parte de la población T'''.

Evaluación de Stackelberg de *T'''*: La evaluación de Stackelberg de la población T''' se realiza del mismo modo que se realizó la evaluación de Stackelberg de la población T, el valor de la función objetivo del líder dada en (1) se llamará $lider'''(k)$ con $k = 1,\ldots,T'''$.

Selección de T de la generación i+1: Para seleccionar a la población inicial T de la generación $i + 1$ se ordena en una lista a los individuos de TUT''' en orden descendiente según el valor de lider(j) y lider^''' (k). Los individuos que conformarán la nueva población inicial T serán los primeros |T| individuos en la lista, donde |T| es el número de individuos en la población inicial que se eligió al comenzar el algoritmo.

Resultados numéricos

El algoritmo Stackelberg-Evolutivo se codificó y se probó en lenguaje M en el entorno de experimentación del *software* Matlab 7.5 en una computadora Acer Aspire 5250 con un procesador AMD E-300 APU con Radeon HD Graphics 1.30 Ghz y con una memoria RAM de 2 Gb con el sistema operativo Windows 7 Starter. Además, se probó en instancias pequeñas con 7 nodos, 12 arcos y 2 bienes, y en instancias medianas con 25 nodos, 40 arcos y 3 bienes tomadas de [9].

Los parámetros que se variaron para la experimentación, fueron el número de individuos de la población inicial I y el número de generaciones G. Para las instancias pequeñas, los tamaños de población inicial utilizados fueron I=50,100 y el número de generaciones G=25,50. Asímismo, para las instancias medianas, los parámetros utilizados fueron I=150,200,250 y G=50,75. Para cada tamaño de instancia se experimentó con cinco ejemplos distintos, entre los cuales varía el costo fijo c_a de los arcos. **La Tabla 1** corresponde a los valores promedio encontrados de la función objetivo del líder con el algoritmo Stackelberg-Evolutivo. De la columna 2 a la 5 se hallan los resultados para las instancias chicas; y de la columna 6 a la 9, los de las instancias medianas. El primer número de cada casilla corresponde al promedio de la función objetivo del líder encontrado en la experimentación y el número entre paréntesis corresponde al promedio del tiempo de ejecución en segundos.

El desempeño del algoritmo Stackelberg-Evolutivo con I individuos en la población inicial y con G generaciones (G,I) se comparó contra el algoritmo de función de penalización P, el algoritmo quasi-Newton QN, el algoritmo basado en aproximación de gradiente G y el algoritmo de búsqueda directa haciendo uso del método símplex de Nelder-Mead NM de [9]. Para realizar estas comparaciones, se realizaron pruebas estadísticas en las que se encontró que no existen diferencias significativas en el promedio de las funciones objetivo del líder entre las distintas combinaciones de los parámetros G e I, tanto para las instancias pequeñas como para las medianas; por otro lado, se encontró que para las instancias pequeñas, el tiempo de ejecución del algoritmo es menor con los parámetros (25,50) y (25,100). Del mismo modo, el tiempo de ejecución para las instancias medianas es significativamente menor o igual con los parámetros (50,150) y (50,200). Por tanto, se eligieron los resultados obtenidos con los

Tabla 1: Promedios de la función objetivo del líder

	Instancias pequeñas				Instancias medianas			
Ejemplo	**(25,50)**	**(25,100)**	**(50,50)**	**(50,100)**	**(50,150)**	**(50,200)**	**(75,150)**	**(75,200)**
1	157.23 (71.18)	161.09 (133.04)	161.70 (143.77)	162.18 (276.49)	1856.93 (423.71)	1886.87 (622.13)	1849.42 (639.92)	1903.01 (829.07)
2	259.82 (66.25)	266.42 (131.19)	266.88 (133.60)	272.91 (279.12)	2695.91 (429.26)	2695.41 (634.47)	2702.10 (596.87)	2724.35 (796.36)
3	189.77 (75.40)	196.42 (143.65)	193.75 (147.06)	198.84 (282.98)	3860.28 (423.30)	3869.79 (522.68)	3792.46 (641.70)	3922.36 (784.60)
4	154.45 (82.05)	155.73 (192.87)	155.19 (147.89)	156.31 (373.39)	2333.86 (382.15)	2344.00 (544.74)	2330.57 (582.56)	2358.11 (755.59)
5	130.92 (85.18)	134.67 (124.74)	123.14 (164.30)	134.94 (259.19)	6908.20 (360.18)	6743.55 (453.09)	6827.44 (550.02)	6815.35 (716.13)

parámetros anteriormente mencionados para realizar las comparaciones.

El desempeño del algoritmo Stackelberg-Evolutivo (de aquí en adelante: SE) con I individuos en la población inicial y con G generaciones, (G,I). Se hicieron comparaciones contra los algoritmos de penalización P, quasi-Newton QN, basado en aproximación de gradiente G y de búsqueda directa haciendo uso del método símplex de Nelder-Mead NM descritos en [9]. Para realizar estas comparaciones, se realizaron pruebas estadísticas en las que se encontró que no existen diferencias significativas en el promedio de las funciones objetivo del líder entre las distintas combinaciones de los parámetros G e I, tanto para las instancias pequeñas como para las medianas; por otro lado, se encontró que para las instancias pequeñas, el tiempo de ejecución del algoritmo es menor con los parámetros (25,50) y (25,100). De manera análoga, para las instancias medianas consideramos los parámetros (50,150) y (50,200) para realizar las comparaciones.

La **Tabla 2** resume las comparaciones entre los distintos algoritmos. Las columnas 2 y 3 corresponden a las instancias pequeñas y muestran los *gaps* en porcentaje de los resultados obtenidos con el algoritmo SE con los parámetros (25,50) y (25,100) reportados en la **Tabla 1** con respecto al mejor resultado conocido de entre los algoritmos P, QN, G y NM. El primer número de cada celda indica el *gap* de la función objetivo del líder calculado de la siguiente forma:

$$\text{GAP} = \frac{\alpha(G,I)}{\text{Mejor valor conocido}} \times 100 \qquad (11)$$

en donde $\alpha(G,I)$= Mejor valor conocido - Valor obtenido con el $SE(G,I)$, con G=25 e I=50, 100. Por otra parte, el número que se encuentra entre paréntesis indica el *gap* del tiempo de ejecución calculado con (11), con G=25 e I=50, 100. Por ello, un gap negativo para los resultados de la función objetivo, indicará que el algoritmo SE obtuvo un mejor resultado, lo mismo indicará un gap positivo para los tiempos de ejecución. Análogamente, para las instancias medianas, los hacen las columnas 5 y 6, calculando los *gaps* con (11) y con G=50 e I=150, 200. La columnas 4 y 7 indican el mejor valor conocido de la función valor objetivo del líder para las instancias chicas y medianas, respectivamente, que es el primer número de la celda acompañado por el algoritmo que alcanzó dicho valor. Además, el segundo número indica el mejor valor encontrado para los tiempos de ejecución. Al realizar un análisis estadístico de los resultados obtenidos, se encontró que para las instancias chicas no existen diferencias significativas entre los promedios de las funciones objetivo del líder encontradas por el algoritmo SE con G=25 e I=50, 100 con respecto al mejor valor conocido encontrado por los algoritmos P, QN, G o NM. Sin embargo, se encontró que los tiempos de ejecución del algoritmo propuesto son significativamente mayores, lo cual concuerda con el hecho de que los algoritmos evolutivos son poco prácticos para ser utilizados en problemas sencillos. Por otra parte, en las instancias medianas, se encontró también que no hay diferencias significativas entre los valores de la función objetivo del líder para los distintos algoritmos; sin embargo, los tiempos de ejecución del algoritmo SE son significativamente menores o iguales y, además, en algunos de los ejemplos logró acercarse al

Tabla 2: Comparaciones entre los algoritmos

	Instancias pequeñas			Instancias medianas		
Ejemplo	**(25,50)**	**(25,100)**	**Mejor valor conocido**	**(50,150)**	**(50,200)**	**Mejor valor conocido**
1	3.527 (-5576.7)	1.154 (-10509.8)	162.975 (G) 1.254 (G)	-5.270 (50.385)	-6.968 (27.151)	1763.962 (QN) 854 (NM)
2	5.514 (-4118.0)	3.115 (-8252.28)	274.986 (G) 1.571 (QN)	2.284 (-4.698)	2.302 (-54.749)	2758.923 (QN) 410 (QN)
3	-72.624 (-7507.1)	-78.672 (-14392.5)	109.931 (G) 0.991 (QN)	7.891 (13.436)	7.664 (-6.888)	4190.971 (QN) 489 (G)
4	1.615 (-4586.2)	0.801 (-10916.3)	156.987 (P,G) 1.751 (G)	1.434 (20.052)	1.006 (-13.962)	2367.820 (QN) 478 (QN)
5	4.405 (-9284.3)	1.666 (-13642.3)	136.953 (QN) 0.908 (QN)	-13.381 (64.757)	-10.678 (55.666)	6092.925 (QN) 1022 (QN)

óptimo global del problema, no así los algoritmos contra los que se comparó.

Conclusiones

El algoritmo propuesto en este artículo mostró tener un buen desempeño al mostrar valores para la función objetivo del líder similares a los encontrados en la literatura pero en tiempos de ejecución menores o iguales en instancias medianas. En instancias pequeñas obtuvo resultados similares para la función objetivo pero en tiempos mucho mayores; sin embargo, este resultado se esperaba debido a que la computación evolutiva ha sido diseñada para la solución de problemas complejos. Debido a lo anterior, se espera que el algoritmo SE llegue a mostrar buenos resultados en la solución de instancias grandes del TOP. Por otro lado, es posible que, al realizar un estudio más detallado de la afinación de los parámetros del algoritmo y de las distribuciones de probabilidad utilizadas, se puedan encontrar resultados mejores que los reportados para las instancias medianas. Además, cabe agregar que al realizar una apropiada codificación de las soluciones del nivel superior, es posible extender el uso de este algoritmo a otros problemas que puedan formularse como un problema binivel, como lo son la solución de problemas ambientales, de logística humanitaria, de diseño de redes, de transporte, de localización, etcétera.

Referencias

[1] Labbé, M., Marcotte, P., Savard, G. "A bilevel model of taxation and its application to optimal highway pricing". *Management Science*. Vol 44, pp. 595-607. 1998

[2] Roch, S., Savard, G., Marcotte, P. "An approximation algorithm for Stackelberg network pricing". *Networks*. Vol. 46, pp. 57-67. 2005.

[3] Grigoriev, G., van Hoesel, S., van der Kraaij, A., Uetz, M., Bouhtou, M. "Pricing bridges to cross a river". *Naval Research Logistics*. Vol. 54, pp. 411-420. 2007.

[4] Dewez, S., Labbé, M., Marcotte, P., Savard, G. "New formulations and valid inequalities for a bilevel pricing problem". *Operations Research Letters*. Vol. 36, pp.141-149. 2008.

[5] Bouhtou, M., van Hoesel, S., van der Kraaij, A., Lutton, J.L. "Tariff optimization in networks". *INFORMS Journal on Computing*. Vol. 19, pp.458-469. 2007.

[6] Joksimovic D, Bliemer MCJ, Boby PHL. "Optimal Toll Design Problem in Dynamic Traffic Networks – with Joint Route and Departure Time Choice". *Transportation Research Record: Journal of the Transportation Research Board*. Pp. 61-72. 2005.

[7] Dibi-Biha, M., Marcotte, P., Savard, G. "Path-based formulations of a bilevel toll setting problem". *Optimization with multivalued mappings theory: theory, applications and algorithms*. Pp. 29-50. 2006.

[8] Brotcorne, L., Cirinei, F., Marcotte, P., Savard, G. "An exact algorithm for the network princing problem". *Discrete Optimization*. Vol. 35, pp. 1-14. 2011.

[9] Kalashnikov, V., Camacho, F., Askin, R., Kalashnykova, N. "Comparison of algorithms for solving a bi-level toll setting problem". *International Journal of Innovative Computing, Information and Control*. Vol. 6, Nu.8. pp.1-14. 2010.

[10] Brotcorne, L., Labbé, M., Marcotte, P., Savard, G. "A bilevel model and solution algorithm for a freight tariff setting problem". *Transportation Sc*. Vol. 34 pp.289-302. 2000.

[11] Brotcorne, L., Labbé, M., Marcotte, P., Savard, G. "A bilevel model for toll optimization on a multi-commodity transportation network". *Transportation Science*. Nu. 35, pp.1-14. 2001.

[12] Lotito, P.A., Mancinelli, E.M., Quadrat, J.P., Wynter, L. "A global descent heuristic for bilevel network pricing". http://www-rocq.inria.fr/metalau/quadrat/bpalgocomplet.pdf 2004.

[13] Dimitriou, L., Tsekeris, T., Stathopoulos, A. "Genetic computation of a road network design and pricing Stackelberg games with multi-class users". *Applications of Evolutionary Computing*. Pp. 669-678. 2008.

[14] Dimitriou, L., Tsekeris, T. "Elastic multi-user stochastic equilibrium toll design with direct search meta-heuristics". *Proceedings of the 8th EU/MEeeting on Metaheuristics in the Service Industry*. Pp. 9-16. 2007.

[15] Brotcorne, L., Cirinei, F., Marcotte, P., Savard, G. "A tabu search algorithm for the network pricing problem". *Computers and operations research*. Vol. 39, pp.2603-2611. 2011.

Datos de los autores:

Pamela Jocelyn Palomo Martínez

Cursó sus estudios en Licenciatura en Matemáticas en la Facultad de Ciencias Físico Matemáticas de la Universidad Autónoma de Nuevo León. En la actualidad, se encuentra estudiando la Maestría en Ciencias en Ingeniería de Sistemas en la Facultad de Ingeniería Mecánica y Eléctrica de la Universidad Autónoma de Nuevo León. Las líneas de interés son la aplicación de la investigación de operaciones en la modelación y optimización de procesos industriales, así como la resolución de modelos de optimización con metaheurísticas.

Dirección del autor: Ciudad Universitaria, S/N. C.P. 66451, San Nicolás de los Garza, Nuevo León, México.
Email: pamela.jpm@hotmail.com

José Fernando Camacho Vallejo

El Dr. Camacho tiene Licenciatura en Matemáticas por la Facultad de Ciencias Físico-Matemáticas de la Universidad Autónoma de Nuevo León, Maestría en Ciencias en Ingeniería con especialidad en Ingeniería Industrial por Arizona State University y Doctorado en Ciencias de la Ingeniería con especialidad en Ingeniería Industrial otorgado por el ITESM campus Monterrey. Actualmente, se encuentra laborando como profesor-investigador exclusivo y de tiempo completo en CICFIM y como coordinador del Posgrado en Ciencias con Orientación en Matemáticas de la FCFM en la UANL. Las líneas de investigación de interés son resolución de problemas de investigación de operaciones, en particular sobre teoría y aplicaciones de programación binivel, diseño de métodos numéricos y técnicas heurísticas para resolver problemas de programación binivel.

Dirección del autor: Ciudad Universitaria, S/N, C.P. 66451, San Nicolás de los Garza, Nuevo León, México.

Email: jose.camachovl@uanl.edu.mx