Una heurística basada en memoria para el problema del diseño de recorridos en transporte público urbano

Antonio Mauttone [•]
María E. Urquhart [•]
Departamento de Investigación Operativa, Instituto de Computación,
Facultad de Ingeniería, Universidad de la República, Uruguay

22 de Marzo de 2007

Palabras clave: optimización de recorridos y frecuencias, memorias adaptativas, metaheurísticas.

Resumen

El problema del diseño de recorridos en transporte público urbano consiste en encontrar un conjunto de recorridos y frecuencias, que optimicen objetivos de usuarios y empresas, en base a la red de calles y a la demanda de viajes. Se presenta un algoritmo basado en la técnica de memorias adaptativas para su resolución aproximada; una memoria de largo plazo almacena componentes de soluciones para la construcción de nuevas alternativas. Algunos resultados numéricos muestran que la heurística propuesta converge rápidamente a una buena solución en comparación con una variante GRASP que produce mejores soluciones pero a un mayor costo computacional.

[•] Dirección: J. Herrera y Reissig 565, Piso 5. E-mail: {mauttone|urquhart}@fing.edu.uy.

1. Introducción

El problema del diseño de recorridos para transporte público urbano, conocido como TNDP (Transit Network Design Problem [2]) consiste en encontrar un conjunto de recorridos y valores de frecuencias, que optimizan simultáneamente los objetivos contrapuestos de los usuarios y los operadores, teniendo en cuenta la red de calles y la demanda entre diferentes puntos de una ciudad. Las restricciones generalmente refieren a niveles de servicio requeridos (tiempos de viaje y de espera, ocupación de los buses) y disponibilidad de recursos (buses). Las frecuencias son incluídas como variables de decisión, dado que tienen influencia directa en la estructura de costos tanto de los usuarios como de los operadores. El TNDP se presenta como uno de los problemas a resolver en el contexto de la planificación estratégica de un sistema de transporte público urbano [5]. Su resolución exacta presenta algunas fuentes de dificultad [2, 4], siendo las principales: alta complejidad combinatoria, utilización de un sub-modelo de asignación y naturaleza multi-objetivo del problema. Los métodos de enumeración completa de soluciones son impracticables por su elevado costo computacional; formulaciones de programación matemática existen para versiones simplificadas del problema [3, 15]. El TNDP se ha resuelto en forma aproximada, mediante heurísticas [2] y metaheurísticas, en particular utilizando Algoritmos Genéticos [14], Tabu Search [6] y GRASP [9].

En la Sección 3 se presenta el algoritmo basado en la técnica de memorias adaptativas (AMP, Adaptive Memory Programming [12]) que resuelve el TNDP en forma aproximada. El modelo de optimización utilizado (Sección 2) es de función objetivo única, construída mediante una suma ponderada de los objetivos de los usuarios y de los operadores. Los resultados numéricos (Sección 4) muestran que el algoritmo basado en AMP converge rápidamente a una mejor solución que la obtenida por el algoritmo GRASP-TNDP [9]; sin embargo, este último devuelve mejores soluciones si dispone de un mayor esfuerzo computacional tanto en tiempo como en cantidad de iteraciones. Como trabajo pendiente se detecta la necesidad de la definición de una estructura de vecindad que permita realizar una búsqueda local en todas las variables de decisión del problema (recorridos y frecuencias).

2. Notación y modelo de optimización

Se considera un grafo no dirigido $G = (N,E)$ que modela la red de calles; N es el conjunto de vértices ($|N| = n$) y E es el conjunto de aristas. El costo c_e de una arista $e = (i,j) \in E$ modela el tiempo de viaje en vehículo (a bordo de los buses) entre los vértices i y j. La matriz origen-destino $D = \{d_{ij},\ i,j \in [1..n]\}$ caracteriza la demanda, donde d_{ij} indica la cantidad de viajes requeridos desde el vértice i al vértice j, expresada en viajes (a ser realizados por una persona que ocupa un lugar en el bus) por unidad de tiempo en un horizonte temporal dado. Un recorrido es una secuencia de vértices adyacentes en G. Una solución S para el TNDP es un par (R,F), donde $R = \{r_1,...,r_r\}$ es un conjunto de recorridos y $F = \{f_1,...,f_r\}$ sus correspondientes frecuencias; cada f_k, $k \in [1..r]$ es un valor real que representa el inverso del tiempo entre dos buses consecutivos en el recorrido r_k.

Los objetivos de los usuarios y de los operadores son modelados con las funciones Z_1 y Z_2 respectivamente, cuya suma ponderada se busca minimizar, siendo

$$Z_1(S) = \sum_{i=1..n} \sum_{j=1..n} d_{ij} \left(tv_{ij} + te_{ij} + tt_{ij} \right), \qquad (1)$$

donde tv es el tiempo de viaje en vehículo (a bordo de los buses), te es el tiempo de espera y tt es el tiempo de penalización por transbordos. Estos tres componentes son calculados para

cada solución S aplicando un sub-modelo de asignación [5]. Los objetivos de los operadores son representados mediante la minimización del tamaño de la flota, expresado como

$$Z_2(S) = \sum_{r_k \in R} f_k t_k \ , \qquad (2)$$

donde $t_k = 2\sum_{e \in r_k} c_e$ es la duración total del recorrido r_k.

Dada una solución $S = (R,F)$, se define $D_0(S) \in [0,1]$ como la proporción del total de la demanda $D_{tot} = \sum_{i=1..n} \sum_{j=1..n} d_{ij}$, satisfecha por recorridos de R directamente (sin transbordos). Análogamente, $D_{01}(S)$ es la proporción de D_{tot} satisfecha directa o indirectamente (con un transbordo como máximo). D_{0min} y D_{01min} son constantes que definen las restricciones

$$D_0(S) \geq D_{0min} \ , \qquad (3)$$
$$D_{01}(S) \geq D_{01min} \ . \qquad (4)$$

Las frecuencias de una solución deben respetar la restricción (5) correspondiente a las frecuencias mínima y máxima y la restricción (6) correspondiente al factor de carga de los buses [10].

El modelo de optimización se define como

$$\min Z(S) = \alpha Z_1(S) + \beta Z_2(S)$$
s. a.
restricciones (3), (4), (5) y (6).

α y β (con $\alpha+\beta=1$) son coeficientes que modelan la importancia relativa de los diferentes objetivos, los valores de Z_1 y Z_2 se encuentran normalizados. La descripción completa del modelo puede encontrarse en [10].

3. Un algoritmo basado en memoria

Adaptive Memory Programming (AMP) se define en [7] (ligado a Tabu Search) como un conjunto de métodos que se basan en el uso de memoria para la resolución de problemas de optimización combinatoria de alta complejidad; existen memorias de corto y largo plazo, así como estrategias de intensificación y diversificación asociadas al manejo de las memorias. Más recientemente, en [12] se agrupan varias metodologías bajo el nombre de AMP, que tienen como característica común el uso de memoria. AMP ha probado ser una técnica efectiva en determinados problemas de optimización combinatoria [12], en particular en ruteo de vehículos [8, 11, 13], problema que comparte algunas características con el TNDP.

En este trabajo se presenta un algoritmo basado en AMP para la resolución aproximada del TNDP en base al modelo de optimización de la Sección 2. El algoritmo, denominado AMP-TNDP, utiliza una memoria de largo plazo donde almacena componentes (recorridos) que forman parte de soluciones previamente generadas. Iterativamente se repite un proceso de construcción de soluciones utilizando los recorridos almacenados en la memoria, y de actualización de la memoria a partir de las soluciones generadas. De esta forma, las nuevas soluciones se construyen a partir de componentes que forman parte de buenas soluciones y la memoria es actualizada cada vez que se supera la mejor solución encontrada hasta el momento. La Figura 1 muestra un seudocódigo del cuerpo principal del algoritmo.

```
procedure AMP-TNDP(out Smejor);
begin
 1   Generar I soluciones iniciales;
 2   Inicializar M con los recorridos de las soluciones generadas en 1;
 3   for i=1 to AMPiters do
 4      R = Generar nuevo conjunto de recorridos a partir de M;
 5      Completar construcción de R si elementos de M no fueron suficientes en  4;
 6      Calcular conjunto de frecuencias F para R;
 7      if Z(S=(R,F)) < Z(Smejor) then
 8         Smejor = S;
 9         Actualizar M con elementos de R;
10      end if;
11   end for;
12   return Smejor;
end AMP_TNDP;
```

Figura 1

La generación de soluciones iniciales (línea 1) se realiza ejecutando I iteraciones del algoritmo GRASP-TNDP [10]. La memoria M se inicializa (línea 2) con los recorridos de las I soluciones generadas. M tiene un tamaño fijo dado $Mtam$, y sus elementos son recorridos r que están ordenados en forma no decreciente según $f(r)$, el valor de la función objetivo evaluada en su correspondiente solución S, $f(r) = Z(S)$, $r \in R$ con $S = (R,F)$.

La generación de un nuevo conjunto de recorridos R (línea 4) se realiza seleccionando iterativamente elementos de M. La selección es aleatoria (según distribución de probabilidades utilizada en [11]) y cada vez que un recorrido r es seleccionado, se eliminan de M todos los recorridos que satisfacen la demanda (dada por la matriz origen-destino D) de pares de vértices satisfecha por r. Dado que este procedimiento es destructivo respecto a la memoria, se trabaja con una copia local a la iteración AMP (memoria local) de la memoria global. La construcción de un conjunto de recorridos finaliza cuando se cumplen las restricciones (3) y (4) del modelo de optimización. Si la memoria local se vacía antes de finalizar la construcción, se generan recorridos adicionales (línea 5) utilizando el procedimiento constructivo explicado en [10]; dicho procedimiento aplica una estrategia ávida aleatoria a partir de una solución parcial (eventualmente vacía), y consiste en la generación de recorridos en base a caminos más cortos entre pares de vértices en el grafo G y en la inserción de pares de vértices en recorridos existentes.

El conjunto de frecuencias F se inicializa con las mínimas frecuencias factibles, y se inicia un procedimiento de búsqueda local (línea 6), donde la vecindad de la solución $S = (R,F)$ se define como

$$V_S = \{S' = (R,F') \text{ con } F' = \{f'_1,...,f'_r\} \in \Theta^r / \exists\, j \in [1..r] \text{ que cumple } f_j \text{ y } f'_j \text{ son consecutivas en } \Theta \text{ y } f_i = f'_i \;\forall\, i \in [1..r],\, i \neq j\}\ ,$$

siendo $\Theta = \{\theta_1,...,\theta_f\}$ un conjunto de valores de frecuencias, ordenado en forma creciente. De esta forma se discretiza el dominio de las frecuencias y la búsqueda local trata de mejorar el valor objetivo de $Z(S)$ aumentando o disminuyendo las frecuencias de cada recorrido de R; se utiliza una estrategia *first-improving* y la búsqueda finaliza cuando no hay ningún vecino de S según V_S que mejora la solución. El resultado cumple con las restricciones (5) y (6) relativas a frecuencias del modelo de optimización. Observar que V_S se define con respecto a las frecuencias solamente.

Cuando una solución mejora el costo de la mejor solución obtenida hasta el momento, la memoria M se actualiza con sus recorridos (línea 9). El sub-modelo de asignación de Baaj y Mahmassani [1] es utilizado para calcular la componente Z_1 de la función objetivo Z y para verificar el cumplimiento de la restricción (6) del modelo de optimización. El esquema de

búsqueda local utilizado por el algoritmo AMP-TNDP es el mismo que el utilizado en GRASP-TNDP [10].

4. Resultados numéricos

Los algoritmos AMP-TNDP y GRASP-TNDP se probaron con el caso de Mandl (tomado de [2]), consistente en una red de 15 vértices y 21 aristas y una matriz origen-destino con el 76% de sus elementos no nulos. La implementación fue realizada en C++ y las pruebas fueron ejecutadas en un PC Pentium 4 de 1,6 GHz con 512 MB de memoria RAM. Se configuraron los parámetros de cantidad de soluciones iniciales I = 20 y tamaño de la memoria *Mtam* = 250.

La Tabla 1 presenta resultados obtenidos a partir de una replicación de cada algoritmo, para diferentes tiempos fijos de ejecución. Ambas implementaciones utilizan los mismos componentes (rutinas y estructuras de datos). La ejecución se finaliza una vez que se supera el tiempo máximo indicado (columna *Tiempo*), retornándose la mejor solución encontrada antes de dicho evento y la cantidad de iteraciones realizadas por cada algoritmo.

Se observa que AMP-TNDP se estanca en el valor objetivo encontrado, a partir de los 200 segundos de ejecución. El algoritmo GRASP-TNDP mejora Z acorde aumenta el tiempo máximo de ejecución, lo que permite un mayor número de iteraciones; sin embargo la convergencia a una solución de valor objetivo comparable a la producida por AMP-TNDP es lenta (los valores objetivo se igualan para 1000 segundos). Para 1500 segundos GRASP-TNDP supera el mejor valor objetivo encontrado por AMP-TNDP, que permanece estancado.

Tabla 1. Comparación de algoritmos.

Tiempo (segs.)	*AMP-TNDP*		*GRASP-TNDP*	
	Valor objetivo Z	*Iteraciones AMP*	*Valor objetivo Z*	*Iteraciones GRASP*
100	0,4874	46	0,4788	146
200	0,4778	92	0,4783	280
500	0,4778	235	0,4782	664
1000	0,4778	470	0,4778	1385
1500	0,4778	691	0,4774	2075

5. Conclusiones y trabajos futuros

Teniendo en cuenta que el tiempo aquí es utilizado simplemente a los efectos de la comparación y no interesa como medida absoluta, podemos decir que el algoritmo AMP-TNDP converge más rápido a una buena solución, en relación a GRASP-TNDP, que lo supera para un tiempo de 1500 segundos. Sin embargo, AMP-TNDP se estanca en el valor objetivo encontrado. La rápida convergencia de AMP-TNDP se atribuye a la memoria de largo plazo utilizada, que permite intensificar la búsqueda, encontrando tempranamente buenas soluciones en relación a GRASP-TNDP, que no utiliza ningún mecanismo de memoria. El estancamiento de AMP-TNDP se atribuye a la falta de un mecanismo efectivo de diversificación que permita alimentar la memoria con nuevos recorridos, acorde avanza la evolución del algoritmo. Aunque la construcción de una solución se completa utilizando un procedimiento aleatorio (línea 5 en Figura 1), que genera recorridos que eventualmente no se han generado previamente, es necesaria la definición de otra vecindad, diferente de V_S que permita modificar la estructura de los recorridos de una solución.

La definición de una vecindad con respecto a la estructura de los recorridos en el TNDP plantea dos dificultades: i) es altamente probable que una movida de búsqueda local produzca soluciones no factibles con respecto a las restricciones (3) y (4), ii) la invocación al sub-

modelo de asignación es computacionalmente costosa, lo que hace necesario la búsqueda de métodos eficientes de evaluación de las movidas. Posibles líneas de investigación para atacar estas dificultades son: i) incluir un mecanismo que permita visitar soluciones no factibles, utilizando una función objetivo con términos adicionales que penalicen el grado de violación de las restricciones, ii) utilizar una función objetivo alternativa *Z'*, representativa de la original *Z*, y de evaluación menos costosa en términos computacionales.

Otra posible dirección para desarrollos futuros, es la consideración de estructuras alternativas para la memoria *M* utilizada por el algoritmo AMP-TNDP. Puede pensarse que en lugar de almacenar recorridos completos en *M*, pueden almacenarse porciones de estos, dado que un recorrido puede ser "bueno" (y por lo tanto poseer una probabilidad alta de ser seleccionado para formar parte de una solución en construcción) porque solamente algun tramo de él satisface una cantidad importante de demanda. Una idea similar es explorada en [13] para ruteo de vehículos, con probados buenos resultados.

Agradecimientos: A la Comisión Sectorial de Investigación Científica (CSIC) de la Universidad de la República, por la financiación del proyecto I+D "El problema del transporte público colectivo urbano".

Referencias

[1] M. H. Baaj, H. S. Mahmassani, *TRUST: A LISP program for the analysis of transit route configurations,* Transportation Research Record 1283, 1990, páginas 125 - 135.

[2] M. H. Baaj, H. S. Mahmassani, *An AI-based approach for transit route system planning and design*, Journal of Advanced Transportation 25(2), 1991, páginas 187 - 210.

[3] R. Borndörfer, M. Grötschel, M. E. Pfetsch, *Models for line planning in public transport*, 9th International Conference on Computer Aided Scheduling of Public Transport, San Diego, Estados Unidos, 2004.

[4] P. Chakroborty, *Genetic Algorithms for optimal urban transit network design*, Computer Aided Civil and Infrastructure Engineering 18(3), 2003, páginas 184 - 200.

[5] G. Desaulniers, M. Hickman, *Public transit*, Reporte Técnico G-2003-77, GERAD, 2003.

[6] W. Fan, R. B. Machemehl, *A Tabu Search based heuristic method for the transit route network design problem*, 9th International Conference on Computer Aided Scheduling of Public Transport, San Diego, Estados Unidos, 2004.

[7] F. Glover, M. Laguna, *Tabu Search*, Kluwer Academic Publishers, 1997.

[8] B. L. Golden, G. Laporte, E. D. Taillard, *An adaptive memory heuristic for a class of vehicle routing problems with min-max objective*, Computers and Operations Research 24(5), 1997, páginas 445 - 452.

[9] A. Mauttone, M. E. Urquhart, *A multi-objective metaheuristic approach for the Transit Network Design Problem*, 10th International Conference on Computer Aided Scheduling of Public Transport, Leeds, Inglaterra, 2006.

[10] A. Mauttone, *Optimización de recorridos y frecuencias en sistemas de transporte público urbano colectivo*, Tesis de Maestría en Informática, Universidad de la República, Uruguay, 2005.

[11] A. Olivera, O. Viera, *Adaptive memory programming for the vehicle routing problem with multiple trips*, Computers and Operations Research, 2005, en prensa.

[12] E. D. Taillard, L. M. Gambardella, M. Gendrau, J. Y. Potvin, *Adaptive memory programming: a unified view of metaheuristics*, European Journal of Operational Research 135, 2001, páginas 1 - 16.

[13] C. D. Tarantilis, C. T. Kiranoudis, *BoneRoute: An Adaptive Memory-Based Method for Effective Fleet Management*, Annals of Operations Research 115, 2002, páginas 227 - 241.

[14] V. M. Tom, S. Mohan, *Transit route network design using frequency coded Genetic Algorithm*, Journal of Transportation Engineering 129(2), 2003, páginas 186 - 195.

[15] Q. K. Wan, H. K. Lo, *A mixed integer formulation for multiple-route transit network design*, Journal of Mathematical Modelling and Algorithms 2(4), 2003, páginas 299 - 308.