HEURÍSTICA DE BÚSQUEDA DE ENTORNO VARIABLE PARA EL PROBLEMA DE RUTEO DE TRANSPORTE PÚBLICO URBANO

Rafael Alvarez Arispe, Héctor Martínez Luz, Antonio Mauttone
Instituto de Computación, Facultad de Ingeniería, Universidad de la República
J. Herrera y Reissig 565, Piso 5
{ralvarez | hectorm | mauttone}@fing.edu.uy

RESUMEN

En este trabajo se considera un modelo de optimización combinatoria para el problema de ruteo del transporte público urbano. El problema consiste en encontrar un conjunto eficiente de recorridos de transporte público sobre una infraestructura dada, de forma de satisfacer una demanda dada, sujeto a un conjunto de restricciones. Proponemos un algoritmo de resolución basado en la metaheurística de Búsqueda de Entorno Variable (VNS). Se presenta un algoritmo de construcción de soluciones iniciales y un conjunto de estructuras de entorno, que se combinan con el uso de un criterio de aceptación de tipo Simulated Annealing y una función objetivo que incluye penalización para permitir visitar soluciones no factibles. El algoritmo se prueba utilizando la instancia de Mandl. Presentamos resultados numéricos que muestran que nuestro algoritmo es competitivo e incluso mejora soluciones existentes en la literatura.

PALABRAS CLAVE. Transporte público, Optimización de recorridos, Búsqueda de entorno variable. Area principal: Metaheurísticas, Logística y Transporte, Optimización Combinatoria.

ABSTRACT

In this paper, we consider a combinatorial optimization model for the urban transit routing problem. The objective is to develop an efficient set of bus routes on a given infrastructure in order to satisfy a given demand, subject to a set of constraints. We propose a solution algorithm based on the metaheuristic Variable Neighborhood Search (VNS). We present an algorithm for the construction of initial solutions and a set of neighborhood structures. We use a Simulated Annealing acceptance criterion and an objective function which includes penalties to allow infeasible solutions. The algorithm is tested using the instance of Mandl. Numerical results are presented, that show that our algorithm is competitive and even outperforms existing solutions in the literature.

KEYWORDS. Public transportation, Bus route optimization, Variable Neighborhood Search. Main area: Metaheuristics, Logistic and Transportation, Combinatorial Optimization.

1 Introducción

La planificación del transporte público urbano colectivo cobra cada vez más importancia, tanto en los países desarrollados como en los en vías de desarrollo (Mauttone *et al*, 2003). Dicha importancia surge debido al impacto del trasporte público en la calidad de vida de la población de una ciudad. Una proporción importante de los viajes en las ciudades medianas y grandes son efectuados utilizando transporte público colectivo. Por otro lado, las herramientas de apoyo a la toma de decisiones en muchos casos complementan el conocimiento y experiencia profesional con elementos cuantitativos.

La planificación de un sistema de transporte público urbano colectivo implica determinar un plan de recorridos, frecuencias, horarios, asignación de personal y flota, en lo posible óptimas. Este proceso se puede descomponer en etapas (Ceder y Wilson, 1986) de la siguiente manera:

1. Diseño de los recorridos: cantidad de líneas y el trazado de sus recorridos.
2. Determinación de frecuencias: de pasadas para cada línea, eventualmente variable en el tiempo.
3. Determinación de horarios: tablas de horarios de cada línea y sincronización de despachos entre aquellas que comparten puntos de transbordos.
4. Asignación de flota: en base a los vehículos disponibles para realizar los viajes.
5. Asignación de personal y recursos disponibles a los viajes programados por línea.

Las dos primeras etapas son generalmente ejecutadas por las entidades reguladoras, es decir, el estado, la municipalidad. Las tres últimas etapas son generalmente ejecutadas por los operadores de los servicios, las empresas de transporte.

En este trabajo se considera un modelo de optimización combinatoria para atacar la primera etapa de esta división. El problema de ruteo del transporte público urbano (UTRP, de sus siglas en inglés Urban Transit Routing Problem), consiste en construir un conjunto eficiente de recorridos de vehículos sobre una infraestructura dada (red vial, paradas). El modelo es concebido como herramienta de apoyo a la toma de decisiones de la entidad reguladora; su solución debe ser un conjunto de recorridos conveniente tanto para los usuarios como para las empresas de transporte.

El UTRP ha sido abordado por diferentes autores, ya sea por sí solo, o conjuntamente con la determinación de las frecuencias (Desaulniers y Hickman, 2007). Una característica de la literatura referente a este problema, es que no existe una formulación estándar ni un conjunto de instancias de prueba con resultados asociados contra los cuales comparar. Por otro lado, los métodos de resolución existentes son en la mayoría heurísticas (Chakroborty y Dwivedi, 2002; Fan y Machemehl, 2004; Mauttone y Urquhart, 2009), dado que la formulación y resolución del problema tienen como principales dificultades (Baaj y Mahmassani, 1991) la alta complejidad combinatoria y la necesidad de representar el comportamiento de los usuarios para evaluar las soluciones del modelo de optimización.

En este trabajo se toma el modelo propuesto por (Fan y Mumford, 2010), se propone un algoritmo de resolución basado en la metaheurística VNS (Hansen y Mladenovic, 2001) y se prueba utilizando la instancia de Mandl. Los resultados obtenidos se comparan con los publicados en (Fan y Mumford, 2010).

El documento continúa de la siguiente manera. En la sección 2 se define formalmente el problema, mientras que en la sección 3 se presenta el procedimiento implementado para resolverlo. En la

sección 4 se detalla el conjunto de pruebas así como los resultados obtenidos mientras que en la sección 5 se presentan las conclusiones y se delinean trabajos futuros.

2 Definición del problema

Tomamos la definición del problema dada por (Fan y Mumford, 2010). Como entrada al UTRP, la formulación asume que existe:

- Un grafo no dirigido G (V, A), donde $V = \{v_1, ..., v_n\}$ es el conjunto de vértices y $A = \{a_1, ..., a_m\}$ es el conjunto de aristas. En dicho grafo los vértices pueden representar intersecciones de calles, paradas del sistema de transporte público o centroides de zonas (Ortúzar y Willumsen, 1996). Las aristas representan una conexión entre vértices, que permite la definición de un recorrido sobre la misma.

- Una matriz origen-destino, también llamada matriz de demanda $D=\{d_{ij}\}$, que representa la demanda entre pares de vértices de G, sobre una base diaria o para un período específico del día; cada d_{ij} representa la cantidad de individuos que desean transportarse desde el vértice i hacia el j por unidad de tiempo (asumimos que D es simétrica).

- Una matriz de costos $C=\{c_{ij}\}$, donde c_{ij} representa el costo (distancia o tiempo) de transportar un vehículo de i a j. Una entrada en la matriz C de valor infinito (c_{ij} = inf) indica que los vértices i y j no están conectados directamente.

Un recorrido se define como una secuencia de vértices adyacentes en G. Una solución R consiste en un conjunto recorridos. Dados G y R, denominamos G^R al grafo inducido por R sobre G.

Como restricciones del problema:

- G^R debe ser conexo.

- Es necesario que ningún recorrido posea un ciclo.

- Hay exactamente r recorridos en la solución.

- El número de vértices en cada recorrido debe ser mayor a 1 y menor o igual que MAX.

Dada una solución R, la función objetivo que se busca minimizar se define como:

$$A\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} d_{ij}\,p_{ij} + B\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} d_{ij}\,t_{ij}\ ,$$

donde

- p_{ij} es el costo del camino de menor costo entre los vértices i y j sobre G^R,

- t_{ij} es la mínima cantidad de transbordos entre los vértices i y j utilizando un camino de costo p_{ij} en G^R,

- A y B son constantes utilizadas para definir el peso de los dos componentes de la función objetivo.

La figura 1 ilustra el cálculo de t_{ij}. Sea c el camino de menor costo que conecta i con j. Existen dos formas para transitar c según el conjunto de recorridos $R = \{r1, r2, r3\}$. La primera es realizar un transbordo sobre el vértice u (efectuando el cambio de recorrido $r3$ –

r2) y luego otro sobre el vértice *v* (*r2 – r1*). La segunda implica transitar sobre *r3* y realizar un transbordo en el vértice *v* (*r3 – r1*). La segunda forma es la que realiza la mínima cantidad de transbordos (uno) y por lo tanto $t_{ij} = 1$.

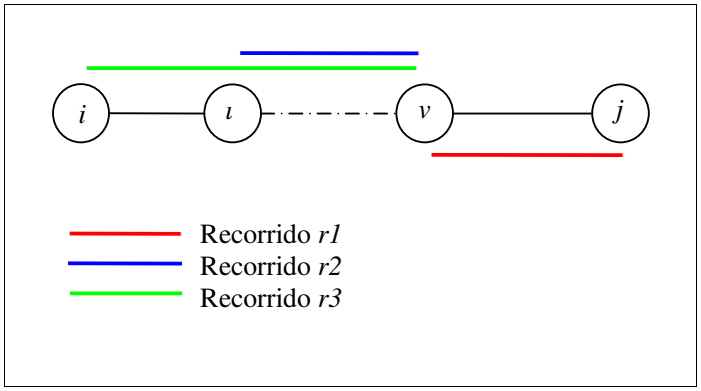

Figura 1 – Definición de recorridos y transbordos.

3 Resolución

Esta sección describe el procedimiento construido para atacar el UTRP, que se basa en la meta heurística VNS (Variable Neighbourhood Search) (Hansen y Mladenovic, 2001). La idea fundamental de VNS es el cambio sistemático de vecindad junto con una búsqueda local. Partiendo de una solución inicial, primero se inicia una fase llamada agitación en la cual se escoge de manera aleatoria una solución dentro de una vecindad, luego se intenta mejorar la solución encontrada mediante una búsqueda local, finalmente se debe decidir si se cambia la solución inicial por la nueva solución encontrada. En la figura 2 se muestran los pasos de VNS básico.

Inicialización:

Seleccionar el conjunto de vecindades N_k (k=1,...,k_{max}), que se utilizaran en la búsqueda; encontrar una solución inicial x; escoger una condición de parada.

Repetir hasta que se cumpla la condición de parada:

1. $k = 1$
2. Repetir hasta que $k=k_{max}$

   a) Agitación. Generar una solución aleatoria x' a partir de la k_ésima vecindad de x ($N_k(x)$).

   b) Búsqueda Local. Aplicar algún método de búsqueda local tomando x' como solución inicial, denotemos x'' al óptimo local obtenido.

   c) Decisión de Aceptación. Si x'' es mejor que la solución actual, o se cumple algún criterio de aceptación, moverse a la nueva solución ($x = x''$) y continuar la búsqueda con $k = 1$; de otra forma, $k = k+1$.

Figura 2 - VNS básico.

3.1 Solución Inicial

El algoritmo construido para generar la solución inicial, crea una solución de exactamente r recorridos, cada uno conteniendo hasta MAX vértices. Para ello utiliza la matriz origen-destino D así como la matriz de costos C.

El algoritmo trabaja en dos fases, distribución de demanda y construcción de recorridos.

Distribución de demanda:

En esta fase generamos una matriz, la cual llamamos de "flujo de demanda" $DD=\{dd_{ij}\}$, a partir de la matriz de demanda D. En la matriz DD solo los vértices que están conectados por aristas de G ($c_{ij} < \inf$) pueden tener valor de flujo mayor a cero, el resto tiene valor de flujo igual a cero. Para construir la matriz DD tenemos que distribuir la demanda de los pares de vértices (i, j) para los cuales el valor en la matriz de demanda es mayor a cero ($d_{ij} > 0$) pero el costo de viajar del vértice i al j (y viceversa) es infinito.

Dado un par de vértices (i, j) con valores $d_{ij} > 0$ y $c_{ij} = \inf$, construimos los posibles recorridos que tomarían los pasajeros para ir del vértice i al vértice j. Suponiendo que los pasajeros escogen los recorridos debido al costo en tiempo (independiente de la cantidad de transbordos), la idea es construir un conjunto de caminos de menor costo entre los vértices (i, j). Para esto utilizamos el algoritmo de Yen (Yen, 1971), que retorna los K caminos de menor costo sobre un grafo para un par de vértices.

Luego, la demanda del par no conectado (i, j) es dividida en forma proporcional al costo del camino, sobre las aristas en G de cada uno de esos K caminos. De esta manera se obtiene una matriz que representa para cada par de vértices (i, j) unidos por una arista en G, el "tráfico" que debe soportar en base a personas que potencialmente transitan esa arista (independiente de cualquier recorrido).

Construcción de recorridos:

Utilizando la matriz DD, generada en la fase de distribución de demanda, implementamos un algoritmo que se inspira en el algoritmo de ahorro de Clarke y Wright (versión secuencial), en donde la matriz de ahorro aquí corresponde a la matriz DD. Primero, se enumeran las aristas (i,j) de manera decreciente según su correspondiente valor dd_{ij}. Sea AA esta enumeración. Luego se comienzan a construir los r recorridos.

En el algoritmo de ahorro de Clarke y Wright (Clarke y Wright, 1964) no hay límite a la cantidad de recorridos a construir y por lo tanto al aplicarlo directamente, estamos potencialmente violando la restricción de la cantidad de recorridos. Para solucionar esto, cortamos el algoritmo al construir los r recorridos. Si al finalizar el algoritmo tenemos menos de r recorridos, entonces creamos tantos recorridos de una sola arista como sean necesarios utilizando la enumeración AA.

3.2 Agitación

Las vecindades son parte esencial de VNS. En nuestro procedimiento utilizamos una única operación para definir las distintas vecindades, basada en el procedimiento “make_small_change” (Fan y Mumford, 2010). Este procedimiento realiza un único cambio sobre un recorrido escogido de manera aleatoria de la solución actual; este cambio puede ser agregar un vértice al final del recorrido o eliminar el primer vértice del recorrido. Para definir las distintas vecindades extendimos esta operación de manera de agregar o eliminar k vértices. La eliminación de los k primeros vértices es trivial, sin embargo agregar k vértices merece una explicación. Para agregar k vértices implementamos un algoritmo recursivo que en cada paso, dado un vértice v, construye una

lista de candidatos con todos los vértices adyacentes a *v* en *G* que pueden ser agregados al recorrido sin generar un ciclo. Luego selecciona al azar para ser agregado, a un vértice de la lista de candidatos y se invoca nuevamente utilizando el vértice seleccionado de la lista. Concluye cuando se agregan los *k* vértices, o la lista de candidatos es vacía. Llamaremos al procedimiento implementado "make_k_changes". Utilizando la operación "make_k_changes" creamos dos tipos de estructuras de entornos o vecindades, Tipo 1 y Tipo R. Las vecindades Tipo 1 fuerzan la agregación o eliminación de *k* vértices en un recorrido (donde *k* depende del número de vecindad). En las vecindades Tipo R se permite agregar o eliminar entre 1 y *k* vértices, dependiendo esto de un factor aleatorio (con distribución uniforme). La idea de las vecindades Tipo R, es dar más oportunidades a los pequeños cambios y de esta manera explorar soluciones más cercanas (Hemmelmayr *et al.*, 2009).

3.3 Búsqueda Local

La búsqueda local encuentra la mejor solución que diste a un solo cambio de una solución dada. Simplemente se itera sobre todos los recorridos de la solución aplicando la operación “make_small_change_to_i” sobre cada recorrido. La operación "make_small_change_to_i" acciona de igual manera que "make_small_change" pero afecta solamente al recorrido *i*.

3.4 Decisión de Aceptación

La decisión de aceptación refiere a escoger entre la solución actual (mejor solución encontrada hasta el momento) y la solución que resulta de la búsqueda local. VNS básico escoge siempre la mejor solución. Nosotros experimentamos con otro esquema que se basa en la técnica Simulated Anneling (SA) (Kirkpatrick *et al.*, 1983), en base al trabajo de (Hemmelmayr *et al.*, 2009). Con este esquema, soluciones mejores siempre se aceptan, en cambio soluciones con mayor valor objetivo se aceptan con probabilidad $\exp((f(x)-f(x''))/T)$, donde f es la función objetivo. La aceptación de una solución con mayor valor objetivo depende de una temperatura T dada y de la diferencia entre el valor objetivo de las soluciones. Se utiliza el esquema de decrecimiento lineal de la temperatura *T*. Cada η iteraciones (etapa de SA) *T* disminuye $\eta T/\mu$, donde μ es la cantidad máxima de iteraciones. De esta manera $T=0$ en las últimas η iteraciones.

3.5 Cálculo de Función Objetivo

Nuestra versión de la función objetivo *f* utilizada en el algoritmo de resolución, se basa en la definida por (Fan y Mumford, 2010), a la que agregamos penalizaciones a la demanda satisfecha por los transbordos y a la demanda insatisfecha, quedando de la siguiente forma:

$$A\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} d_{ij}p_{ij} + B\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} d_{ij}t_{ij} + p_1 d_1 + p_2 d_2 + p_{3oMas} d_{3oMas} + p_{un} d_{un}$$

donde

- p_1, penalización extra por hacer un transbordo
- p_2, penalización extra por hacer dos transbordos
- p_{3oMas}, penalización extra por hacer tres o más transbordos
- p_{un}, penalización por no existir camino entre los vértices *i* y *j*
- d_1, porcentaje de la demanda satisfecha mediante un transbordo

- d_2, porcentaje de la demanda satisfecha mediante dos transbordos
- d_{3oMas}, porcentaje de la demanda satisfecha mediante tres o más transbordos
- d_{un}, porcentaje de la demanda no satisfecha

La diferencia de nuestra versión se encuentra en la penalización de la demanda insatisfecha, dado que las demás penalizaciones pueden realizarse utilizando los parámetros A y B (en el caso lineal). La penalización por demanda insatisfecha es necesaria dado que optamos por permitir movimientos por soluciones no factibles ($d_{un} > 0$). La restricción de conectividad de G^R es verificada a posteriori.

La dificultad del cálculo de la función objetivo se halla en calcular los valores p_{ij} y t_{ij}. Para calcular p_{ij} implementamos el algoritmo Floyd-Warshall (Floyd, 1962), que encuentra el camino de menor costo entre todo par de vértices de un grafo en orden N^3, donde N es la cantidad de vértices del grafo. Nuestra implementación de este algoritmo produce dos matrices, la de costos mínimos que contiene los p_{ij} y la matriz PP que contiene el vértice previo en el camino de menor costo. Para calcular t_{ij} creamos un algoritmo que dado el conjunto de recorridos y la matriz PP, obtiene la menor cantidad de transbordos. El algoritmo se basa en los mismos conceptos del algoritmo de Floyd-Warshall.

4 Experimentos

4.1 Instancias

Para probar nuestro enfoque al problema usamos la instancia de Mandl (obtenida de OR Library, http://people.brunel.ac.uk/~mastjjb/jeb/orlib/utraninfo.html), que consta de 15 vértices, 21 aristas y 172 valores de demanda diferentes de cero. Como valores de referencia para nuestro trabajo utilizamos los resultados obtenidos por diferentes autores presentados en (Fan y Mumford, 2010). Realizamos experimentos para valores de r iguales a 4, 6, 7 y 8 recorridos, donde cada recorrido puede tener un máximo de $MAX = 8$ vértices.

4.2 Parámetros

Los parámetros utilizados en las pruebas fueron ajustados en base a una experimentación previa. Los valores de los parámetros de la función objetivo utilizados son los siguientes: $A = 0.1$, $B = 0.5$, $p_1 = 100$, $p_2 = 200$, $p_{3oMas} = 300$, $p_{un} = 10000$. Los valores de p_1, p_2, p_{3oMas} pretenden controlar los transbordos. p_{un} es un valor relativamente grande que guía la búsqueda hacia soluciones sin demanda insatisfecha, e indirectamente dependiendo de la instancia, lograr que la solución sea conexa. Para la decisión de aceptación necesitamos ajustar la temperatura inicial, la cantidad de iteraciones por etapa y la cantidad máxima de iteraciones de SA. Los valores utilizados son los siguientes: la temperatura se inicializa en un 5% del valor objetivo de la solución inicial, se realizan 1000 iteraciones por etapa y como máximo 100000 iteraciones (100 etapas). El algoritmo termina cuando transcurre una etapa sin lograr una mejora o al fin de la etapa con temperatura igual a cero. Luego de ajustados los parámetros se realizaron 30 ejecuciones independientes del algoritmo, para cada valor de r.

4.3 Evaluación de soluciones

Para evaluar las soluciones, utilizamos las métricas att y d_m definidas en (Fan y Mumford, 2010). En este caso, para el cálculo de los caminos de menor costo se asume una penalización de 5

minutos por cada transbordo realizado. Por lo tanto el cálculo de p_{ij} se realiza de manera diferente que en el cálculo de la función objetivo. Para todos los caminos de menor costo p_{ij} encontrados para el par (i, j) se calculan los transbordos necesarios para recorrerlos. Así, t_{ij} es la mínima cantidad de transbordos necesarios para recorrer alguno de los caminos de menor costo para el par (i, j).

- $$att = \frac{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} d_{ij}\, p_{ij}}{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} d_{ij}}$$

- $$d_m = \frac{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} d_{ij}\, t_{ij}^{m}}{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} d_{ij}}, \quad donde\ t_{ij}^{m} = \left\{1 \Leftrightarrow t_{ij} = m, \quad 0\ en\ otro\ caso\right\}$$

4.4 Resultados

La tabla 1 muestra los valores de las soluciones obtenidas por nuestro algoritmo para el mejor caso y el caso promedio sobre las 30 ejecuciones independientes (columnas Mejor y Prom, para vecindades Tipo 1 y Tipo R). Las columnas FM Mejor y Prom corresponden a los mejores resultados y promedios utilizando Simulated Annealing por (Fan y Mumford, 2010), mientras que la columna CD corresponde a los resultados obtenidos por (Chakroborty y Dwivedi, 2002) utilizando un algoritmo genético. Como podemos observar, nuestro algoritmo mejora las soluciones para d_0 (porcentaje de demanda satisfecha sin necesidad de transbordos) y d_1 en los experimentos de 6, 7 y 8 recorridos. También mejora el *att* en los experimentos de 4 y 8 recorridos, siendo marginalmente superior a los mejores resultados en los casos de 6 y 7 recorridos. Nuestros resultados (tanto los mejores como los promedios) mejoran sensiblemente cuando se incrementa la cantidad de recorridos, no así los resultados de Fan y Mumford los cuales tienen un notable desempeño en el experimento con menor cantidad de recorridos en comparación con el resto. La diferencia entre nuestros mejores resultados y nuestros promedios tiende a decrecer al incrementar la cantidad de recorridos mientras que la de Fan y Mumford es similar en todos los experimentos.

La tabla 2 muestra los recorridos de las mejores soluciones obtenidas en los experimentos. Consideramos mejor solución aquella que logra el menor valor de *att*, que no necesariamente coincide con la de los mejores valores en d_0.

r	Param.	CD	FM mejor	FM prom.	Mejor Tipo 1	Prom Tipo 1	Mejor Tipo R	Prom Tipo R
4	d_0	86.86	**93.26**	92.48	88.31	81.75	84.78	78.65
	d_1	12.00	**6.74**	7.52	10.66	16.87	14.71	19.14
	d_2	1.14	**0.00**	**0.00**	1.03	1.34	0.51	2.09
	d_{un}	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	att	11.90	11.37	11.55	**10.84**	11.77	11.35	13.45
6	d_0	86.04	91.52	90.87	93.19	91.69	**94.41**	91.68
	d_1	13.96	8.48	8.74	6.49	7.90	**5.20**	7.82
	d_2	**0.00**	**0.00**	0.39	0.32	0.41	0.39	0.49
	d_{un}	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	att	**10.30**	10.48	10.65	10.37	10.54	10.36	10.58
7	d_0	89.15	93.32	92.47	96.79	94.91	**97.88**	94.20
	d_1	10.85	6.36	6.95	3.21	4.93	**2.12**	5.57
	d_2	**0.00**	0.32	0.58	**0.00**	0.16	**0.00**	0.23
	d_{un}	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	att	**10.15**	10.42	10.62	10.19	10.33	10.16	10.38
8	d_0	90.38	94.54	93.65	97.69	96.11	**98.14**	96.36
	d_1	9.62	5.46	5.88	2.31	3.84	**1.86**	3.61
	d_2	**0.00**	**0.00**	0.47	**0.00**	0.05	**0.00**	0.04
	d_{un}	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	att	10.46	10.36	10.58	10.14	10.25	**10.12**	10.24

Tabla 1 – Comparación de resultados, se resaltan en negrita los mejores resultados de cada experimento.

r	**Mejor solución Tipo 1**	**Mejor solución Tipo R**
4	4-3-5-7-9-12-10-11	4-3-5-7-9-12-10-11
	0-1-2-5-14-6-9-10	0-1-2-5-14-6-9-12
	12-13-9-6-14-7-5	8-14-6-9-10-11-3-5
	8-14-7-9-10-11-3-1	13-9-12-10-11-3-5-7
6	0-1-2-5-7-9-10-12	3-1-2-5-7-9-6-14
	3-4-1-2-5-7-9-13	4-1-2-5-7-14-6-9
	4-3-5-7-9-10-12-13	0-1-2-5-7-9-10-11
	8-14-6-9-10-11-3-5	12-10-9-6-14-7-5-3
	0-1-2-5-14-6-9-10	4-3-5-7-9-6-14-8
	2-1-3-11-10-9-7-14	9-13-12-10-11-3-5-7
7	4-3-5-7-9-13-12-10	0-1-3-11-10-9-6-14
	0-1-2-5-14-6-9-13	2-1-4-3-5-7-14-6
	0-1-4-3-5-7-14-6	1-2-5-14-6-9-13-12
	0-1-3-11-10-9-6-14	0-1-2-5-7-9-10-12
	8-14-6-9-10-11-3-5	0-1-4-3-11-10-12-13
	12-10-9-7-5-2-1-0	4-3-5-7-9-6-14-8
	8-14-5-3-11-10-12	8-14-5-2-1-3-11-10
8	8-14-6-9-10-11-3-5	4-3-5-7-9-13-12-10
	4-1-2-5-7-14-6	0-1-3-11-10-12-13-9
	0-1-4-3-5-14-6-9	8-14-6-9-10-11-3-4
	3-1-2-5-7-14-6-9	0-1-4-3-5-14-6-9
	0-1-3-11-10-12-13	8-14-5-2-1-4-3-11
	0-1-2-5-7-9-10-12	0-1-2-5-14-6-9-13
	4-3-5-7-9-10-12	0-1-2-5-7-9-10-12
	0-1-2-5-14-6-9-13	6-14-7-5-2-1-4-3

Tabla 2 – Recorridos de nuestras mejores soluciones.

5 Conclusiones y trabajos futuros

Presentamos un algoritmo que ataca el UTRP utilizando la meta heurística VNS. Construimos el algoritmo basándonos en la operación "make_small_change" y la utilización de un esquema de aceptación al estilo Simulated Annealing. El algoritmo mejoró las soluciones existentes en la literatura para la instancia de Mandl en casi todos los experimentos realizados.

Creamos un algoritmo para construir soluciones iniciales. El algoritmo construye una solución en dos fases. La primer fase se basa en el concepto que denominamos distribución de flujo de demanda, en la cual se crea un flujo de pasajeros ficticio basado en los posibles caminos que pueden utilizar los usuarios del sistema. La segunda fase utiliza el mecanismo de construcción de recorridos del algoritmo de ahorro de Clarke y Wright secuencial. Aquí notamos que no siempre es la mejor decisión ponderar las aristas solamente por el valor de demanda distribuida. Sería bueno tener en cuenta otros factores que involucran la topología de *G*, por ejemplo dar más chance a vértices difíciles de alcanzar (vértices con pocos adyacentes).

Si bien se lograron resultados competentes en relación a la literatura existente, observamos que la construcción de las estructuras de entornos en base a la operación "make_small_change" no resulta siempre efectiva como forma de escapar de los óptimos locales en el contexto de VNS (mediante el cambio de vecindad). El problema radica en que una vez que los recorridos empiezan a completarse (llegan al máximo de vértices posibles), la única opción que queda es eliminar vértices y esto nunca mejora las soluciones. Como se pudo apreciar en los resultados de los experimentos (sección 4.4), no encontramos diferencia significativa en la utilización de los dos tipos de estructuras de entorno.

Como trabajos a futuro es de nuestro interés probar el algoritmo con instancias más grandes (en cantidad de vértices de *G*) que el caso Mandl. Además, sería bueno complementar las estructuras de entorno con alguna otra operación que permita generar cambios en una solución sin perder aristas.

Referencias

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

Ceder, A. y Wilson, N., Bus Network Design, *Transportation Research B*, 20, 331-344, 1986.

Chakroborty, P. y Dwivedi, T., Optimal route network design for transit systems using genetic algorithms, *Engineering Optimization*, 34(1), 83-100, 2002.

Chakroborty, P., Genetic algorithms for optimal urban transit network design, *Computer Aided Civil Infrastructure Engineering*, 18, 184-200, 2003.

Clarke, G. y Wright, J., Scheduling of vehicles from a central depot to a number of delivery points, *Operations Research,* 12, 568-581, 1964.

Desaulniers, G. y Hickman, M., Public transit, en Barnhart, C. y Laporte, G. (eds.) *Handbooks in Operations Research and Management Science Volume 14*, North Holland, Amsterdam, 69-127, 2007.

Fan, W. y Machemehl, R. A Tabu search based heuristic method for the transit route network design problem. En *9th International Conference on Computer Aided Scheduling of Public*

Transport, SanDiego, UnitedStates, 2004.

Fan, L. y Mumford, C., A metaheuristic approach to the urban transit routing problem, *Journal of Heuristics*, 16(3), 353-372, 2010.

Floyd, R., Algorithm 97, Shortest Path, *Communications ACM*, 5-345, 1962.

Hansen, P., y Mladenovic, N., Variable neighborhood search: Principles and applications, *European Journal of Operational Research*, 130, 449-467, 2001.

Hemmelmayr, V., Doerner, K. y Hartl, R., Variable neighborhood search heuristic for periodic routing problems, *European Journal of Operational Research*, 195, 791-802, 2009.

Kirkpatrick, S., Gelatt, **C. y Vecchi, M.P.**, Optimization by Simulated Annealing, *Science*, 220 (4598), 671-680, 1983.

Mauttone, A., Cancela, H. y Urquhart, M., Diseño y Optimización de Rutas y Frecuencias en el Transporte Colectivo Urbano, Modelos Y Algoritmos, *Actas del XI Congreso Chileno de Ingeniería de Transporte*, 2003.

Mauttone, A. y Urquhart, M., A multi-objective metaheuristic approach for the Transit Network Design Problem, *Public Transport*, 1(4), 253-273, 2009.

Ortuzar, J., Willumsen, L., *Modelling transport*, Wiley, 1994.

Yen, J., Finding the K Shortest Loopless Paths in a Network, *Managment Science* 17(11), 712-716, 1971.