Universidad de la República
Facultad de Ingeniería

Tiempo mínimo de difusión en redes

Memoria de proyecto presentada a la Facultad de Ingeniería de la Universidad de la República por

Gabriela Gallo, Santiago Gutierrez

en cumplimiento de los requerimientos
para la obtención del título de
Ingeniero en Computación.

Tutores

Dr. Pablo Rodríguez-Bocca ..... Universidad de la República
Dr. Pablo Romero ..... Universidad de la República
Dr. Franco Robledo ..... Universidad de la República

Tribunal

Dra. Libertad Tansini ..... Universidad de la República
Dr. Santiago Iturriaga ..... Universidad de la República
Dr. Juan Kalemkerián ..... Universidad de la República

Montevideo
jueves 13 septiembre, 2018

Tiempo mínimo de difusión en redes, Gabriela Gallo, Santiago Gutierrez.

Contiene un total de 116 páginas.
Compilada el jueves 13 septiembre, 2018.
`https://www.fing.edu.uy/inco`

"Si he visto más lejos es porque estoy sentado sobre los hombros de gigantes"

Isaac Newton (1643-1727)

Agradecimientos

Expresamos nuestra inmensa gratitud a nuestros tutores, Franco Robledo, Pablo Rodriguez-Bocca y Pablo Romero, quienes estuvieron en continuo contacto y con una excelente disposición desde el comienzo de este trabajo hasta el final. Ellos fueron durante todo el proceso nuestros reales mentores, acompañándonos cálidamente a afrontar una tarea que, lejos de ser rutinaria en nuestro pasaje por la facultad así como en nuestro desarrollo profesional en la industria, fue del todo atípica y novedosa.

Agradecemos sinceramente al Dr. Amaro de Sousa, Universidad de Aveiro - Portugal por sus aportes en la construcción del artículo [8] que antecede a este documento y por su buena disposición. En el marco de este proyecto, tuvimos la oportunidad de intercambiar tanto con él como con nuestros tutores de una forma muy horizontal, aprendiendo muchísimo sobre el trabajo científico académico.

Finalmente, agradecemos a nuestras familias y amigos, que gracias a su apoyo y contención han sabido acompañarnos y alentarnos en todo nuestro proceso formativo.

A nuestras familias.

Resumen

El problema en estudio es el tiempo mínimo de difusión (MBT). A partir de un grafo conexo no dirigido y de un nodo origen que posee un mensaje, el objetivo es transmitir ese mensaje lo antes posible a todos los nodos restantes del grafo. La comunicación tiene lugar entre nodos vecinos de forma selectiva y cada reenvío toma un intervalo de tiempo.

Históricamente, el MBT encuentra aplicaciones en servicios telefónicos, sin embargo, sirve como problema inspirador para el diseño de esquemas actuales de reenvío en sistemas de comunicación modernos como redes de entrega de contenido y redes de pares.

El MBT pertenece a la clase de problemas $\mathcal{NP}$-Completos. Como consecuencia, la literatura ofrece heurísticas, algoritmos de aproximación y soluciones exactas de tiempo exponencial. Así mismo, se encuentran diferentes propuestas de ingeniería inversa, donde se construyen topologías de grafos para minimizar el tiempo total de difusión.

La contribución principal de este trabajo es desarrollar una heurística competitiva llamada *TreeBlock*. *TreeBlock* explota la optimalidad para el MBT en grafos de tipo árbol, y el hecho de que los grafos conexos arbitrarios acepten una descomposición en una estructura de árbol de bloques, donde los bloques de construcción son componentes biconexas. Además, como consecuencia de un trabajo conjunto con investigadores locales y externos, hemos medido el potencial de nuestra heurística comparando con una solución exacta desarrollada mediante una técnica de programación lineal entera. Estos resultados complementan el estudio sobre la eficacia de *TreeBlock* en relación a la correctitud de los resultados arrojados.

Una comparación justa entre *TreeBlock* y heurísticas anteriores resalta también la efectividad de la propuesta.

Palabras claves: *Tiempo mínimo de difusión, Complejidad computacional, Programación lineal entera, Heurística.*

Prefacio

El presente Proyecto de Grado ha dado como producto la siguiente publicación arbitrada:
- Amaro de Sousa, Gabriela Gallo, Santiago Gutierrez, Pablo Rodriguez-Bocca, Franco Robledo, Pablo Romero. Heuristics for the Minimum Broadcast Time [8]. Aprobado para su publicación en la revista Electronic Notes in Discrete Mathematics de Elsevier, como Special Issue de la conferencia ALIO/EURO 2018, realizada entre el 25 y el 27 de junio de 2018 en Bolonia, Italia.

El presente documento es una extensión del aprobado en conferencia EURO/ALIO, con un tratamiento bibliográfico más extensivo y un estudio experimental que incluye nuevas instancias de prueba.

Está planificado enviar a la revista International Transactions in Operational Research, de Wiley, un artículo con los componentes principales de este trabajo.

Asimismo, vale destacar que el problema de estudio permanece en investigación, y actualmente hay una Tesis de Maestría de PEDECIBA Informática en marcha del Ingeniero Mauricio D' Ambrosio, orientada por los mismos tutores.

Tabla de contenidos

Capítulo 1

Introducción

La identificación de un esquema de reenvío ideal es un problema desafiante en las telecomunicaciones. La literatura científica ofrece modelos de fluidos para sistemas de comunicación masivos [1], esquemas de reenvío de árbol o de uno a uno [24], la estrategia *toma y daca* (de Toma y dame acá, o Tit for tat en su expresión original en inglés) de teoría de juegos [20], entre muchos otros.

Qiu y Srikant [35] presentaron por primera vez un modelo de fluidos fundamental para una red completa de pares. En este modelo, los autores consideran una red completa con usuarios idénticos, donde el reenvío simultáneo lleva proporcionalmente más tiempo que el envío punto a punto. Los autores proclaman un resultado contrario a la intuición, donde la solución *uno a uno parecería ser una buena estrategia de reenvío.*

Si bien los autores estudian la capacidad del servicio en un sistema de intercambio de archivos entre pares, su formulación es lo suficientemente genérica. Ellos encuentran una fórmula concreta para el tiempo de espera promedio en un esquema de reenvío uno a uno cuando la población es una potencia de dos. En [27], se incluye una prueba formal de que el esquema de reenvío uno a uno logra el tiempo de espera mínimo, cuando la población tiene una potencia de dos.

Así mismo, trabajos recientes confirman que la optimalidad también es válida para grafos arbitrarios (no completos) [26].

El esquema de reenvío uno a uno es abierto, en el sentido de que se debe determinar la estrategia de selección vecina. Esto significa que este problema es equivalente a un problema histórico bien conocido, llamado *tiempo mínimo de difusión* (Minimum Broadcast Time), o MBT para abreviar [10].

El MBT encuentra aplicaciones originalmente en la red telefónica pública conmutada. Sin embargo, sirve como un problema de inspiración para el diseño de esquemas actuales de reemisión tolerante al retardo en sistemas de comunicación modernos como redes de entrega de contenido (CDN) y redes de pares (P2P) [6], o también redes radio cognitivas (CR) [21], por lo que se entiende como un problema antiguo pero altamente vigente.

También debe mencionarse que la complejidad del problema [11] demanda el desarrollo de algoritmos aproximados, como metaheurísticas, y algoritmos exactos de tiempo exponencial.

El objetivo de este trabajo es entonces contribuir al estado del arte ofreciendo un algoritmo competitivo que se acerque a la solución óptima del problema. Esto se aborda mediante una técnica heurística.

Este trabajo está organizado de la siguiente manera.

En el Capítulo 2 se brinda la terminología necesaria de *Teoría de Grafos* y nociones sobre *Complejidad Computacional*, que serán de utilidad a lo largo del documento.

El Capítulo 3 presenta una descripción del problema de estudio y trabajos relacionados.

En el Capítulo 4 se presenta una formulación de programación lineal entera, un aporte novedoso surgido en el marco del proyecto por la colaboración del Dr. Amaro de Sousa.

La contribución principal del trabajo, que corresponde a la construcción de la heurística $TreeBlock$ se describe completamente en el Capítulo 5.

Seguidamente, en el Capítulo 6 se lleva a cabo una comparación justa contra los resultados alcanzados por las heurísticas más competitivas encontradas en la literatura. Además, se realiza una comparación con los resultados obtenidos por la formulación exacta en términos de resultados y tiempos de ejecución siempre que fuera posible. En ese mismo capítulo, se estudia también la escalabilidad de la heurística propuesta en función de la cantidad de nodos y la densidad de aristas para diferentes instancias de grafos del tipo *pequeño mundo*.

Finalmente, el Capítulo 7 presenta conclusiones y líneas de trabajo futuro.

Capítulo 2

Conceptos previos

2.1. Teoría de grafos

La teoría de grafos es una importante disciplina tanto para la matemática como para la teoría de la computación.

Gracias a la teoría de grafos se pueden resolver diversos problemas. Por ejemplo: encontrar el camino óptimo a través de las calles de una ciudad, calcular el número de combinaciones diferentes de vuelo entre dos ciudades de una red aérea, optimizar los tiempos para concretar una serie de trabajos, colorear cualquier mapa con cuatro colores de tal forma que dos países vecinos nunca tengan el mismo color.

En cuanto a la teoría de la computación, se destacan otras aplicaciones como utilizar grafos predecibles para la creación de sistemas operativos, diseño de bases de datos o redes, entre otras.

El origen de la teoría de grafos se ubica en el siglo XVIII gracias a los resultados obtenidos por Leonhard Euler en su trabajo conocido como "El problema de los puentes de Königsberg". La idea era determinar un camino que recorriese todos los puentes sin visitar más de una vez por cada uno. Euler sustituyó cada una de las regiones por un punto y cada uno de los puentes por una línea resultando en una figura denominada grafo. Así demostró que el problema en cuestión es irresoluble y este es considerado el primer resultado de la disciplina.

A continuación, se enumeran diferentes definiciones básicas utilizadas a lo largo de este trabajo. La terminología fue adoptada del libro de Douglas B. West, "Introduction to Graph Theory" [34]

Definición 1. Se define formalmente un grafo como el par (V, E), y en general se denota $G = (V, E)$, con $V \neq \emptyset$ el conjunto de nodos (o vértices) y $E \subseteq V \times V$ el conjunto de aristas (o enlaces).

Definición 2. Un nodo w es adyacente a otro nodo v si el grafo contiene una arista (v, w) que los une. Además, los nodos w y v son vecinos.

Definición 3. Una arista e es adyacente a un nodo n, cuando uno de sus extremos es el nodo n.

Definición 4. Dado un grafo $G = (V, E)$ y un nodo $i \in V$, el grado de i es el número de aristas adyacentes a él.

Definición 5. Un subgrafo de un grafo G es un grafo G' cuyos conjuntos de vértices y aristas son subconjuntos de los de G.

Definición 6. El subgrafo inducido G' (V', A') es un subgrafo de $G(V, A)$ tal que $V' \subset V$ y contiene todas las aristas adyacentes al subconjunto de vértices de G.

Definición 7. Un bucle es una arista cuyos dos nodos extremos son el mismo.

Definición 8. Un grafo con multiarista es aquel que para un par de nodos adyacentes, existe más de una arista que los une.

Definición 9. Un grafo es simple si no contiene bucles ni multiaristas.

Definición 10. Un camino es un grafo simple, cuyos nodos pueden ser ordenados en una lista de tal forma que dos nodos son adyacentes si y solo si, son nodos consecutivos en la lista.

Definición 11. Un ciclo es un grafo con igual cantidad de nodos y de aristas, cuyos nodos pueden ser dispuestos alrededor de un círculo, tal que dos nodos son adyacentes si y solo si aparecen consecutivamente en el círculo.

Definición 12. Un grafo es acíclico si no existe ningún sugbrafo inducido que sea un ciclo.

Definición 13. Grafo conexo, un grafo es conexo si existe una camino entre todo par de nodos.

Definición 14. La distancia entre dos nodos u y v, $d(u, v)$, es el largo del camino más corto que los comunica.

Definición 15. Excentricidad de un nodo de un grafo conexo, es la distancia máxima entre el nodo y cualquier otro nodo.

Definición 16. Diámetro, es la excentricidad máxima de cualquier nodo del grafo.

Definición 17. Radio, es la excentricidad mínima de cualquier nodo del grafo.

Definición 18. Las componentes conexas de un grafo G son cada uno de los subgrafos conexos inducidos de G maximales.

Definición 19. Un nodo $v \in V(G)$ es llamado *punto de corte* o *punto de articulación* si $G - v$ tiene más componentes conexas que G.

Definición 20. Un bloque de un grafo G es un subgrafo maximal conexo de G que no tiene puntos de corte. Si G es conexo y no tiene puntos de corte, entonces G es un bloque

Definición 21. Árbol, es un grafo en el que dos nodos cualesquiera están conectados exactamente por un solo camino.

Definición 22. Árbol de bloques, es un árbol de componentes biconexas que están unidos por los puntos de corte.

2.2. Elementos de complejidad computacional

En este apartado se brindan nociones generales sobre Complejidad Computacional. El tema se trata extensivamente en el libro autorizado por Garey y Johnson [11].

La Teoría de la complejidad computacional es una disciplina de la teoría de la computación y matemática que estudia la clasificación de los problemas computacionales de acuerdo a su dificultad en base a los recursos necesarios para su resolución y la relación entre dichas clases de complejidad. En este sentido, estudia la eficiencia de los algoritmos estableciendo su efectividad generalmente de acuerdo al tiempo (número de pasos de ejecución de un algoritmo para resolver un problema) y al espacio requerido (cantidad de memoria utilizada para resolver un problema), ayudando a evaluar la viabilidad de la implementación práctica en tiempo y costo.

La idea de medir el tiempo y espacio como una función de la longitud de la entrada, se originó a principios de los 60's por Hartmanis y Stearns, y así, nació la teoría de la complejidad computacional. En los inicios, los investigadores trataban de entender las nuevas medidas de complejidad, y cómo se relacionaban unas con otras. En 1965, Edmonds definió un algoritmo para el cual un polinomio acotaba el tiempo de ejecución (de aquí nace el término tiempo polinómico), lo que conllevó a la formulación de uno de los conceptos más importantes de la teoría de la complejidad computacional.

Definición 23. Un problema se compone de una serie de parámetros de entrada y una salida o solución que cumple ciertas propiedades deseadas.

Definición 24. La serie de pasos definidos para la resolución de un problema se conoce como algoritmo.

Definición 25. Función de complejidad. Para comparar la complejidad de dos algoritmos resulta necesario definir una función, $f(n)$ donde n representa el tamaño de la entrada del problema. Esta función indica el número de operaciones que requiere la ejecución del algoritmo.

Definición 26. Orden de complejidad. La función $f(n)$ es de orden $O(g(n))$ si existe M, constante, tal que $f(n) \leq Mg(n)$.

Definición 27. Los principales órdenes de complejidad son los siguientes:

- constante: $O(1)$
- logarítmica: $O(logn)$
- lineal: $O(n)$
- casi lineal: $O(nlogn)$
- cuadrática: $O(n^2)$
- cúbica: $O(n^3)$
- exponencial: $O(a^n)$

Definición 28. Jerarquía de órdenes:
$O(1) \subset O(logn) \subset O(n) \subset O(nlogn) \subset (n^2) \subset O(n^3) \subset O(a^n)$

Definición 29. Reducción polinomial:
Un problema A es reducible a B si las soluciones de B existen y dan solución para A siempre que A tenga solución. Entonces A no puede ser más difícil de resolver que B, la reducción polinomial son aquellos que pueden ser reducidos y resueltos en dicho tiempo, su notación es $A \leq_{\mathcal{P}} B$

Definición 30. Clase $\mathcal{L}$: Es el conjunto de los problemas de decisión que pueden ser resueltos en espacio $log(n)$, donde n es el tamaño de la entrada, por una máquina de Turing determinista tal que la solución si existe es única.

Definición 31. Clase $\mathcal{NL}$: La clase de complejidad NL (espacio logarítmico no determinista) es el conjunto de los problemas de decisión que pueden ser resueltos en espacio $log(n)$, donde n es el tamaño de la entrada, por una máquina de Turing no determinista tal que la solución si existe es única.

Definición 32. Clase $\mathcal{P}$: Son todos aquellos problemas de decisión que pueden ser resueltos en una máquina determinista secuencial en un período de

tiempo polinómico en proporción a los datos de entrada.

Definición 33. Clase $\mathcal{NP}$: Es el conjunto de problemas que pueden ser resueltos en tiempo polinómico por una máquina de Turing no determinista. O lo que es lo mismo, son los problemas de decisión donde puede verificarse la correctitud de una solución en tiempo polinomial por una máquina secuencial determinista.

La importancia de esta clase de problemas de decisión es que contiene muchos problemas de búsqueda y de optimización para los que se desea saber si existe una cierta solución o si existe una mejor solución que las conocidas. Un ejemplo es el clásico problema del viajante.

Definición 34. Clase $\mathcal{NP}$-Completo: Los problemas de $\mathcal{NP}$-completos son problemas en $\mathcal{NP}$, tales que todo otro problema en $\mathcal{NP}$ se puede reducir polinomialmente a ellos.

Definición 35. Clase $\mathcal{NP}$-Difícil. Todo otro problema en $\mathcal{NP}$ se puede reducir polinomialmente a ellos. Es decir, cumplen la segunda parte de la definición de $\mathcal{NP}$, puesto que no necesariamente deben ser $\mathcal{NP}$

Capítulo 3

Tiempo Mínimo de Difusión

3.1. Formulación del Problema

Originalmente, el problema MBT se enmarca en el contexto de redes telefónicas y la posibilidad de inundar la red con un mensaje, pero aplica a toda red punto a punto donde se quiera difundir un mensaje. Una descripción natural del problema es entonces en términos de llamadas telefónicas (se puede identificar reenvío o rumores, dependiendo del contexto); suponiendo que un miembro origina un mensaje que debe ser comunicado a todos los otros miembros de la red. Esto debe ser logrado lo más rápido posible mediante una serie de llamadas realizadas a través de líneas de la red. En el MBT, hay tres reglas estrictas de transmisión:

i) Cada llamada telefónica requiere un intervalo de tiempo,

ii) un miembro puede participar en una sola llamada por vez, y

iii) un miembro solo puede llamar a un miembro vecino.

Entonces, el problema a estudiar es el siguiente: dado un grafo conexo G y un nodo origen v_0, encontrar cuál es el número mínimo de intervalos de tiempo necesarios para completar la transmisión a todos los nodos del grafo partiendo desde v_0 y siguiendo las reglas de transmisión anteriores. Aquí el grafo representa una red de comunicación, donde los nodos V son los miembros, y las aristas E son los posibles enlaces de comunicación entre ellos.

Definiendo un nodo origen v_0, el *tiempo mínimo de difusión (MBT)*, denotado $b(v_0, G)$ es el número mínimo de intervalos de tiempo necesarios para difundir un mensaje desde v_0 a todos los demás nodos de G. Muchos de los trabajos estudian el MBT de un grafo, $b(G)$, como el peor tiempo entre todos los miembros potenciales de ser el original en el grafo G, es decir, $b(G) = max\{b(v_0, G)|v_0 \in V\}$.

Entonces debería haber al menos una *estrategia de reenvío* que logre $b(v_0, G)$, y podría especificarse como una secuencia de llamadas paralelas que sigue las reglas de difusión e informa a todos los nodos en la red G en $b(v_0, G)$ intervalos de tiempo.

3.2. Antecedentes y complejidad

El problema MBT es presentado formalmente por Garey y Johnson en [11], y fue incluido en su lista de problemas $\mathcal{NP}$-Completos. Debido a la complejidad, es poco probable que se resuelva exactamente. Una prueba de este resultado para un grafo arbitrario se encuentra en [29].

Sin embargo, varios enfoques para hacer frente al problema MBT se han considerado en la literatura: encontrando el óptimo para familias específicas de grafos; construyéndose diferentes algoritmos de aproximación y hallando los limites superiores e inferiores del óptimo; centrándose en diseñar esquemas óptimos de transmisión principalmente enfocados en clases específicas de grafos.

Además, se han reportado varios estudios donde el objetivo pasa a ser la construcción de topologías para las cuales el tiempo de difusión se acerque o alcance el mínimo.

A continuación veremos los trabajos mas relevantes en todos estos aspectos.

3.2.1. Optimalidad en tipos de grafos particulares

El esquema de reenvío óptimo es conocido solo para familias de grafos específicos. En particular, se repasan los resultados disponibles en la literatura para los grafos de tipo grillas y los árboles.

Optimalidad en grillas de dos dimensiones

Los grafos de tipo grilla en dos dimensiones aceptan un esquema de reenvío óptimo [15].

A partir de una grilla $R \times C$, el tiempo de transmisión es precisamente el diámetro $D = R - 1 + C - 1 = R + C - 2$. El reenvío para este caso es intuitivo, y se resuelve comenzando desde una esquina y terminando en la opuesta.

Optimalidad en árboles

La optimalidad en los árboles se logra recursivamente mediante una heurística golosa propuesta en [29].

La idea es un proceso de abajo hacia arriba, reenviando el mensaje primero a aquellos miembros que poseen un retraso máximo.

3.2.2. Estudios por métodos de aproximación

Kortsarz y Peleg [17] idearon el primer algoritmo de aproximación para el problema de difusión, con una cota de $O(\sqrt{n})$.

En un avance [25], Ravi diseñó un algoritmo que proporciona una aproximación de $O\left(\frac{log^2 n}{loglogn}\right)$, siendo n la cantidad de nodos del grafo. Con el aporte de este algoritmo, claramente se encuentra un límite superior para el problema.

Los resultados hallados en [22], para grafos de tipo $bounded - treewidth$ tienen una aplicación interesante en el contexto de encontrar esquemas de transmisión óptimos. Combinando sus resultados para grafos

bounded − treewidth con las técnicas en [25], obtienen un algoritmo de aproximación $O\left(\frac{logn}{loglogn}\right)$ para el problema mínimo de tiempo de difusión.

En otra contribución, Elkin y Kortsarz [9] encuentran un algoritmo de aproximación en $O(logn)$ para los casos de grafos *chordal* y *k-outerplanar*.

En la publicación *Broadcasting in Harary-like Graphs* [5], se estudia el problema en el marco de grafos de tipo Harary $H_{k,n}$, que son grafos k-conexos minimales en n nodos. Se presenta un algoritmo de aproximación en $O(log\frac{k-2}{2})$. Asimismo, los autores dan un paso más, construyendo una nueva familia de grafos a partir de los grafos de Harary, a la que llaman *modified-Harary*, para la cual el algoritmo que proponen alcanza el óptimo.

Otra contribución desde la construcción de algoritmos de aproximación, es la realizada por Shan, Wan y Hu en [28], donde proponen dos diferentes algoritmos de aproximación que mejoran los resultados expuestos hasta el momento.

Adicionalmente, se encuentran en la literatura otras contribuciones que estudian problemas levemente diferentes al MBT, también desde un enfoque de algoritmos de aproximación. Es el caso de [7], donde los autores encuentran un algoritmo para el problema de mínimo tiempo de difusión, restringido a que la ruta de envío de mensaje a cada nodo, debe ser la más corta a partir del nodo origen.

3.2.3. Estudios por heurísticas

Los trabajos presentes en la literatura que abordan el problema MBT, o alguna modificación muy ligera del mismo, son varios. De los explorados, algunos además de la descripción de la heurística que presentan, cuentan con datos, resultados y caracterización del universo de instancias utilizadas para las pruebas.

Para comenzar, en [3] se proponen dos heurísticas para la resolución del problema. Si bien se trabaja con el versionado del problema donde son varios los nodos que originalmente poseen el mensaje, destacan que el caso particular de que sea uno solo no ofrece mejoras respecto a los resultados que encuentran. Para este caso el algoritmo que postulan lo llaman RH y posee un orden computacional de $O(R|V|^2|E|)$, donde R corresponde al número de rondas de difusión. Se ofrecen resultados para grafos de tipo: CCC_k (*k-dimensional cube-connected-cycles*), SE_k (*k-th shuffle-exchange graph*), $Butterfly_k$ (*k-th butterfly*) y $DeBruijn_k$ (*k-th de Bruijn graph*). Los autores resaltan que pueden alcanzar resultados cercanos al óptimo en bajo tiempo de procesamiento.

Más adelante, Harutyunyan y Shao en [13], presentan una heurística basada en árboles que llaman TBA. Esta heurística genera el óptimo para los grafos de tipo anillo, árbol y grilla cuando el nodo origen es un vértice de esquina. Es destacable que en su trabajo experimental, refieren a [3] para comparar sus resultados con la heurística RH allí propuesta. El orden computacional del algoritmo propuesto es mejor que el de la anterior publicación, estando en $O(R|E|)$.

En [14], Harutyunyan y Wang construyen una nueva heurística, también basada en árboles a la que llaman $NTBA$. Como en la publicación anterior,

deciden mostrar los resultados comparando tanto con [13] como con [3] para algunas de las familias de grafos. Con este aporte, obtienen una heurística más simple que TBA, y con un orden computacional menor, alcanzado por $O(|E|)$. Es necesario resaltar que en esta contribución se presentan casos experimentales en instancias de tamaño considerablemente mayores que los casos anteriores; alcanzando el orden de los dos mil nodos.

Otros aportes enfocados a la construcción de heurísticas que resuelven el problema están dados por [36] y [21]. En ambos casos, y a diferencia de los anteriormente mencionados, las contribuciones se realizan enfocadas a familias de grafos específicas. Por un lado, en *Broadcasting on cactus graphs* [36], se presenta un algoritmo de orden $O(n)$ para encontrar el tiempo de difusión en grafos de tipo cactus $k - restringidos$, es decir: un grafo donde no más de k ciclos pueden tener más de un vértice en común. Asimismo, muestran que resolver el MBT para todos los nodos (suponiendo que todos pueden ser posibles originarios del mensaje) es posible con su algoritmo, pero se elevaría el orden a $O(n^2)$

Por otra parte, en [21], se desarrollan dos heurísticas de orden polinomial, así como una fórmula de programación lineal entera, para la familia de redes Radio Cognitivas. En su estrategia, comparan los resultados arrojados por las heurísticas, frente a los óptimos calculados por la fórmula exacta; afirmando que la efectividad de las heurísticas es buena. Las dos versiones de algoritmos difieren en la estrategia de resolución del problema, mientras que una comienza desde los nodos más lejanos al origen, la otra inicia justamente por los más cercanos. En ambas heurísticas, intentan tomar tantos pares de nodos/canales transmisores como sea posible para cada intervalo de tiempo, a la vez que evitan las colisiones.

Existen más trabajos en la literatura que exploran este antiguo problema desde la óptica de las heurísticas. Otro ejemplo puede ser [16], que proponen un algoritmo genético como técnica heurística para encontrar una solución cercana a la óptima.

3.2.4. Topologías con Mínimo Tiempo de Difusión

Finalmente, se resalta que en la literatura científica se identifica el correspondiente problema de diseño del MBT: hallar las familias de grafos que poseen el mínimo tiempo de difusión $log_2(n)$.

En primer lugar, el MBT de un grafo completo $G = K_n$ es claramente $b(K_n) = \lceil log_2(n) \rceil$, sin embargo K_n no es minimal con esta propiedad. El numero mínimo de enlaces para alcanzar $b(K_n)$ es un problema de diseño inverso con un progreso valioso. De hecho, cuando $n = 2^m$ para algunos naturales m, el hipercubo Q_m tiene $m2^{m-1}$ enlaces y el tiempo de reenvío es también $b(Q_m) = m = log_2(n)$. Por lo tanto, el hipercubo es la estructura mas eficiente para el reenvío cuando la población es potencia de dos.

Algunas publicaciones donde se construyen grafos para minimizar el tiempo de difusión se pueden encontrar en [2], [33], [19].

Capítulo 4

Solución exacta

4.1. Formulación vía Programación Lineal Entera

En el marco de este proyecto, el equipo se pone en contacto con el investigador portugués Dr. Amaro de Sousa, quien elabora y comparte una solución exacta al problema en base a una propuesta de Programación Linea Entera (ILP).

A continuación, en esta sección, se introduce este importante aporte generado.

Se recuerda que en cada intervalo de tiempo, un nodo i que posee el mensaje puede reenviarlo a un único miembro del conjunto de sus vecinos $V(i)$. El tiempo de transmisión total es el número de intervalos de tiempo necesarios para que el mensaje alcance a todo el conjunto V. El objetivo es minimizar el tiempo de transmisión.

Sea $G = (V, E)$ un grafo conectado, donde $V = \{0, 1, \ldots, n-1\}$ es el conjunto de nodos. Siendo $V(i)$ el conjunto de vecinos del nodo i. Por simplicidad, se elige $v_0 = 0$ como el nodo origen.

Para poder encontrar una formulación ILP eficiente, se considera un límite superior T para el tiempo mínimo de completado. Se considera también el conjunto de variables binarias x_{ij}^t tales que, cuando es igual a uno, significa que el mensaje fue transmitido desde el nodo i al nodo j durante el espacio de tiempo t. Además, se define una variable entera z cuyo valor representa el tiempo de difusión (medido en intervalos de tiempo).

Entonces, el modelo ILP para MBT se expresa de la siguiente forma:

$$\begin{aligned} &\text{Minimizar } z \\ &\text{Sujeto a:} \end{aligned} \tag{4.1}$$

$$\sum_{j \in V(i)} x_{ij}^1 = \begin{cases} 1\,, i = 0 \\ 0\,, i \neq 0 \end{cases} \quad , i = 0, ..., n-1 \tag{4.2}$$

$$\sum_{j \in V(i)} \sum_{t=1}^{T} x_{ji}^{t} = 1 \qquad , i = 1, ..., n-1 \tag{4.3}$$

$$\sum_{j \in V(i)} x_{ij}^{t} \leq 1 \qquad \begin{matrix} , i = 0, ..., n-1 \\ , t = 2, ..., T \end{matrix} \tag{4.4}$$

$$\sum_{\tau=1}^{t-1} \sum_{k \in V(i) \setminus \{j\}} x_{ki}^{\tau} \geq x_{ij}^{t} \qquad \begin{matrix} , (i,j) \in A : i \neq 0 \\ , t = 2, ..., T \end{matrix} \tag{4.5}$$

$$\sum_{t=1}^{T} t \cdot x_{ij}^{t} \leq z \qquad , (i,j) \in A \tag{4.6}$$

$$x_{ij}^{t} \in \{0, 1\} \qquad \begin{matrix} , (i,j) \in A \\ , t = 1, ..., T \end{matrix} \tag{4.7}$$

$$z \in \mathbb{N} \tag{4.8}$$

La función objetivo 4.1 es el tiempo total de completado, es decir, cuando el mensaje alcanza a todos los nodos del grafo G.

A continuación se expresa el significado de cada una de las restricciones.

- La restricción 4.2 indica que el mensaje es transmitido solo una vez por el nodo de origen 0 durante el primer intervalo de tiempo $t = 1$.
- La restricción 4.3 garantiza que el mensaje es recibido por cada nodo exactamente una vez desde algún nodo vecino.
- La restricción 4.4 garantiza que para los espacios de tiempo $t = 2, ..., T$, un nodo puede transmitir el mensaje a un solo nodo vecino como máximo. Esta restricción para el caso de $t = 1$ es redundante a la definida en 4.2.
- La restricción 4.5 indica que el mensaje es transmitido desde un nodo origen i (excepto cuando se trata del nodo origen 0) a un nodo j en el espacio de tiempo t solo si el mensaje fue recibido por el nodo i en el espacio de tiempo anterior desde algún vecino de i que no fuese j.
- La restricción 4.6 garantiza que el tiempo de completado no es inferior al intervalo de tiempo de cualquier transmisión de mensajes.
- Finalmente, las restricciones 4.7 y 4.8 representan el dominio de variables.

4.1.1. Fortalecimiento del modelo

El tiempo de ejecución requerido para resolver el problema de optimización usando un *solver* estándar es significativamente más corto con la introducción de las siguientes mejoras.

En primer lugar, pudiéndose obtener fácilmente mediante un algoritmo de camino más corto estándar, se considera que la cantidad mínima de saltos de cualquier camino en G del nodo origen 0 al nodo i está dado por h_i.

Para un número determinado de saltos h, se considera el conjunto N_h compuesto por todos los nodos tales que $h_i = h$. Un nodo $i \in N_h$ no puede transmitir el mensaje en cualquiera de los intervalos de tiempo $t = 1, ..., h_i$.

Por lo tanto, para cada nodo $i \in N_h$, se puede asignar 0 a las variables x_{ij}^t, para $t = 1, ..., h$; resultando en la práctica, en una reducción del número de variables.

Entonces, un nodo $i \in N_h$ puede transmitir como mucho un mensaje en un intervalo de tiempo $t = h+1$; y, generalizando, a lo sumo π mensajes en un tiempo $t = h + 1 + \pi$.

Se pueden definir entonces las siguientes desigualdades válidas:

$$\sum_{i \in N_h} \sum_{j \in V(i)} x_{ij}^{h+1+\pi} \leq 1 + \pi \qquad \begin{matrix} , h = 1, ..., T-1 \\ , \pi = 0, ..., T-h+1 \end{matrix} \tag{4.9}$$

Las pruebas computacionales demostraron que la restricción 4.9 no mejora de modo significativo el modelo cuando $\pi > 0$; y por esta razón, los resultados fueron obtenidos agregando las inecuaciones 4.9 solo para casos de $\pi = 0$.

Finalmente, se puede reducir el número de variables aún más de la siguiente manera. Se considera d el grado del nodo origen 0. Sin pérdida de generalidad, se puede decir que si los nodos vecinos d son usados por el nodo origen 0 para transmitir el mensaje, entonces, esas transmisiones sucederán en los primeros d intervalos de tiempo. Por lo tanto es posible establecer en 0 todas las variables x_{0j}^t, para $(0, j) \in A$ y $t = d + 1, \ldots, T$.

4.2. Validación y utilidad

La capacidad de resolver el modelo depende en gran medida de un buen límite superior T del valor mínimo de tiempo de completado. Los buenos límites superiores reducen significativamente tanto el número de variables como las restricciones del modelo resultante.

Se validó el ILP anterior en primer lugar para instancias de redes del tipo grilla.

La suposición es que un límite superior válido para el tiempo de completado en una grilla con filas R y columnas C viene dado por $T = (R-1)+(C-1)$. Téngase en cuenta que si esto no es válido para una instancia problema en particular, el problema debe volverse inviable, lo que nunca ocurrió.

La disponibilidad de este método es realmente importante para, en el Capítulo 6, hacer un análisis del potencial del método heurístico que se propone en el Capítulo 5.

Capítulo 5

Solución heurística

5.1. Generalidades

En este capítulo se introduce una nueva heurística para el MBT. En particular, se busca resolver el problema MBT para cuando el nodo origen está dado previamente.

El algoritmo principal ha sido diseñado inspirados por el enrutamiento óptimo en árboles, la descomposición del grafo en bloques biconexos y un enrutamiento eficiente en el árbol definido por estos bloques.

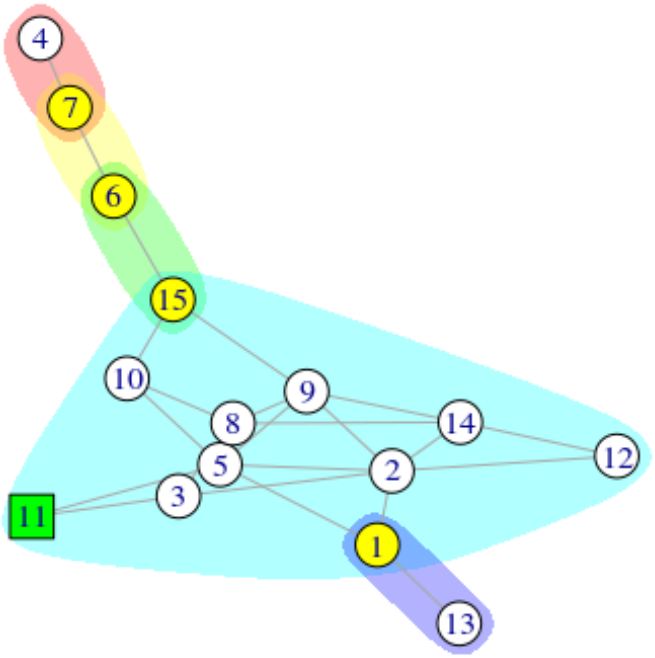
Se recuerda que el MBT acepta un esquema de reenvío óptimo en árboles. Aquí este enrutamiento es llamado $BroadcastTree$.

Para comenzar, se recuerdan algunas definiciones importantes mencionadas en el Capítulo 2. Un nodo $v \in V(G)$ es llamado *punto de corte* si $G - v$ tiene más componentes que G. Un bloque es un subgrafo maximal de G que no tiene puntos de corte. Claramente, si G es conexo y no tiene puntos de corte, entonces G es un bloque. Cada grafo conexo G acepta una descomposición árbol de bloques biconexos, donde cada bloque B_i es un subgrafo inducido sin puntos de corte, y cada punto de corte $v \in V(G)$ es incidente a los bloques que lo incluyen.

Formalmente, dado un grafo conexo arbitrario $G = (V, E)$, el árbol de bloques del grafo es $B(G) = (B \cup U, E')$, donde $U = \{v_1, \ldots, v_r\}$ es el conjunto de puntos de corte en G, $B = \{B_1, \ldots, B_s\}$ el conjunto de bloques de G, y $E' \subseteq B \times U$ consiste de las conexiones entre v_i y B_j siempre que el punto de corte pertenezca a B_j. Por su definición, el árbol de bloques es un árbol, donde algunos nodos son bloques. Observar que si G no tiene puntos de corte, entonces $B(G)$ tiene un solo nodo aislado, y también es un árbol. Adicionalmente, en un árbol no trivial, existen algunas *hojas-bloque*, a las cuales les incide un solo punto de corte [34].

Figura 5.1: Ejemplo de bloques biconexos para una instancia de grafo

Los nodos coloreados en amarillo corresponden a los puntos de corte. El nodo representado como un cuadrado y coloreado en verde corresponde al nodo origen. Cada bloque esta representado coloreando el área bajo el subgrafo que lo compone.



Las dos bloques de construcción principales de nuestro algoritmo son las funciones $BroadcastTree$ y $BroadcastBlock$. Esencialmente, $BroadcastTree$ es un algoritmo de enrutamiento *inter-bloques* encargado de reenviar el mensaje en la estructura del árbol de bloques. Mientras tanto, $BroadcastBlock$ es un algoritmo de enrutamiento *intra-bloque*, que sirve para el reenvío del mensaje pero en este caso dentro de cada bloque. Una estrategia de reenvío global es obtenida por la cooperación de las dos funciones.

5.2. Descripción de la heurística

5.2.1. Algoritmo $TreeBlock$

En primer lugar, se presenta el algoritmo $TreeBlock$, que corresponde a la estrategia de reenvío global. Este algoritmo se compone de básicamente tres partes:

i) Una primera parte de código, cuyo objetivo es generar el árbol de bloques biconexos para el grafo inicial G (entre las Líneas 1 y 12).

ii) La aplicación del algoritmo $BroadcastTree$ para encontrar el enrutamiento inter-bloques.

iii) La aplicación del algoritmo $BroadcastBlock$ para encontrar el enrutamiento intra-bloque en cada uno de los bloques determinados en [i)].

Algorithm 1 $F = TreeBlock(G, v_0)$

1: $U \leftarrow CutPoints(G)$
2: $U_0 \leftarrow \{v_0\} \cup U$
3: $d \leftarrow Distances(U_0)$
4: $T \leftarrow MST(U_0, d)$
5: $B \leftarrow Blocks(G)$
6: $L \leftarrow LeafBlocks(B)$
7: **for all** $B_i \in L$ **do**
8: $\quad z_i \leftarrow B_i \cap CutPoints(G)$
9: $\quad e_i \leftarrow max_{v \in B_i}\{d(w_i, v)\}$
10: $\quad w(z_i, B_i) \leftarrow e_i$
11: **end for**
12: $T \leftarrow T \cup \{(z_i, B_i)\}_{i=1,\ldots,|L|}$
13: $F \leftarrow BroadcastTree(T, v_0)$
14: **for all** $B_i \in B(v_0)$ **do**
15: $\quad F_i \leftarrow BroadcastBlock(B_i, F(B_i))$
16: $\quad F \leftarrow F \cup F_i$
17: **end for**
18: **return** F

A continuación se realiza una descripción paso a paso del algoritmo $TreeBlock$.

En primer lugar, en la Línea 1, se determina el conjunto de nodos U que pertenecen G, constituido a partir de todos los puntos de corte del grafo G. Luego, el nodo origen es agregado a este conjunto, generándose el conjunto de nodos U_0 (Línea 2).

Figura 5.2: Grafo completo y ponderado construido en la Línea 3 del algoritmo *TreeBlock*

Los nodos corresponden a los puntos de cortes encontrados (amarillos) y al nodo origen (verde), mientras que el peso de cada una de las aristas corresponde a la distancia en el grafo original entre cada nodo que conecta.

Figura 5.3: MST del grafo completo y ponderado (Figura 5.2)

El árbol de recubrimiento mínimo es construido mediante el algoritmo de Kruskal en la Línea 4 del algoritmo *TreeBlock*.

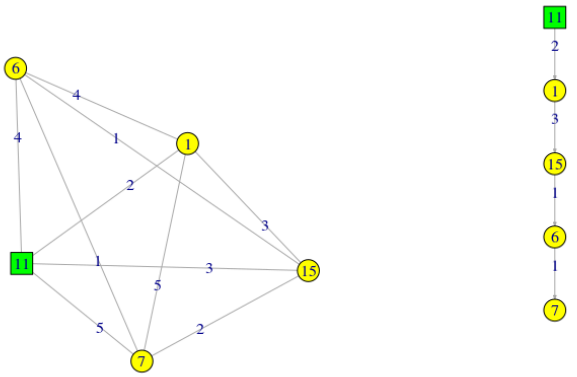
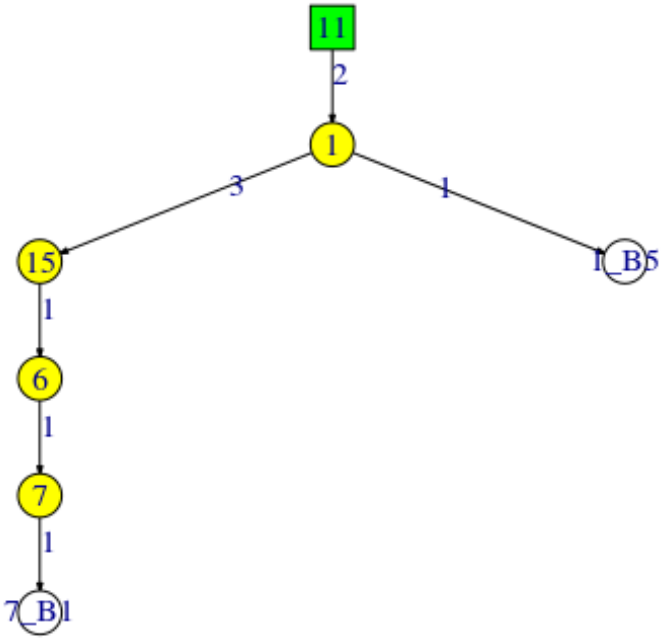


Figura 5.4: Árbol de bloques

Construido a partir del MST de la Figura 5.3, agregándose los bloques hojas (representados en blanco) encontrados en el grafo original, que se denominan en esta secuencia de ejemplos 1_B5 y 7_B1.



A partir de allí, se construye un grafo ponderado completo en la Línea 3, donde el peso de cada una de las aristas corresponde a la distancia entre cada par de nodos de U_0 en el grafo original. Luego, un árbol de recubrimiento mínimo T es construido usando el algoritmo Kruskal [18] [23] en la Línea 4. Este árbol va a servir de *esqueleto* para el esquema de reenvío inter-bloques. El peso entre los enlaces representa la *espera* entre los diferentes puntos de corte. Estos pasos se encuentran representados en las Figuras 5.2 y 5.3 para la instancia del grafo mostrado en la Figura 5.1.

Se observa además que hay una espera adicional con los bloque hojas, que debería ser sumada para alcanzar todos los nodos en el grafo. Con estos datos, en Líneas 5 y 6, se encuentran todos los bloques y todos los bloque hoja de G. Se recuerda que un bloque hoja tiene un solo punto de corte. En el bloque de Líneas 7-11, se halla la excentricidad e_i del punto de corte $z_i \in B_i$ para cada bloque hoja $B_i \in L$. Esta latencia adicional es considerada en Línea 12 adicionándola al árbol de bloques obtenido hasta el momento. El resultado entonces es que los nuevos nodos z_i son nodos hoja en T, y los enlaces correspondientes tienen peso e_i. En la Figura 5.4 se puede apreciar esta etapa del algoritmo para el ejemplo que se viene trabajando en anteriores figuras.

En la Línea 13, un esquema de reenvío inter-bloque F es realizado usando la función $BroadcastTree$ con el nodo origen v_0 para el árbol T. Se recuerda que esta función $BroadcastTree$ retorna el esquema óptimo de reenvío en el árbol ponderado. Una representación de este último paso es mostrada en las Figura 5.5.

Finalmente, el reenvío intra-bloques es realizado usando la función $BroadcastBlock$, que combina una estrategia recursiva de exploración de todo el grafo, y también de reenvío inter-bloques. La función $BroadcastBlock$ es invocada para el nodo origen v_0 y para todos los bloques en que v_0 se encuentre. Este conjunto de bloques se denomina $B(v_0)$.

5.2.2. Algoritmo $BroadcastTree$

En los siguientes párrafos se presentan los detalles sobre las funciones $BroadcastTree$ y $BroadcastBlock$.

Algorithm 2 $F = BroadcastTree(T, v_0)$

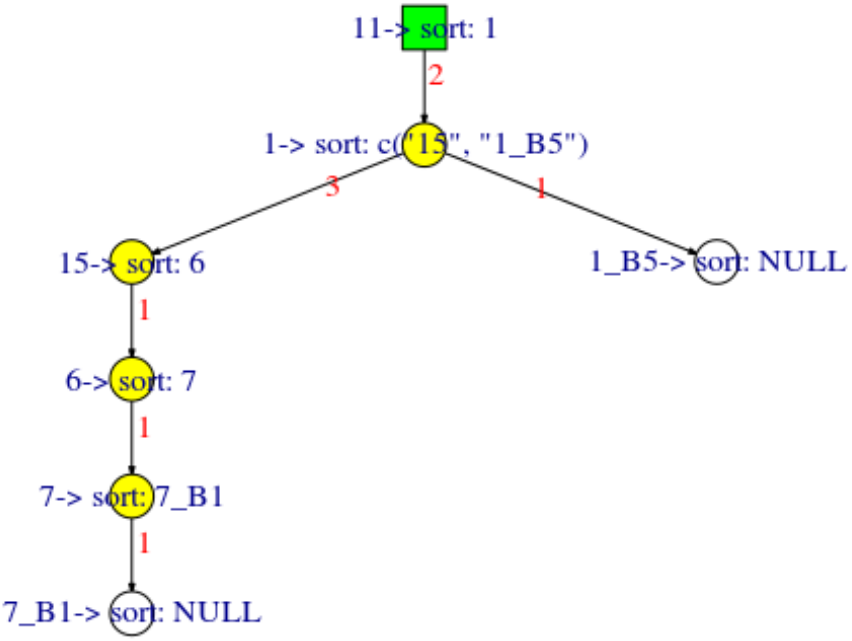
```
1:  H ← FindHeight(T, v_0)
2:  for all v ∈ T : height(v) = 0 do
3:      l(v) ← 0
4:  end for
5:  for h = 1 to H do
6:      T(h) = {v ∈ T : height(v) = h}
7:      for all v ∈ T(h) do
8:          (d_1, ..., d_r) ← SortChildren(v)
9:          l(v) ← max_{1≤i≤r}{d_i + i}
10:     end for
11: end for
12: F ← LargestLabel(v_0)
13: return F
```

El algoritmo $BroadcastTree$ comienza en la Línea 1, encontrando la altura H del árbol de bloques respecto al nodo origen. Se recuerda que el árbol de bloques fue construido tal que la raíz corresponde al nodo origen que posee el mensaje a difundir.

En el bloque de Líneas 2-4, todos los nodos que son hoja son etiquetados con $l(v) = 0$. Esta etiqueta representa la *prioridad* para enviar el mensaje hacia dicho nodo. En este caso, la etiqueta es por defecto y significa que el orden de envío en los nodos hojas es indiferente. Esto responde a que luego de la hoja ya no existe nadie más a quien enviar, por tanto, no importa que sea el último o primer receptor del mensaje. Los nodos restantes son etiquetados durante la iteración entre las Líneas 5-11. Para ello, se consideran iterativamente las diferentes alturas del árbol, representadas por el subconjunto de nodos $T(h) \subseteq T$ (ver Línea 6). Para cada una, se consideran los nodos $v \in T(h)$, con r hijos. Estos hijos son ordenados desde la etiqueta más grande d_1 a la más chica d_r (Línea 8); y la etiqueta de v es encontrada entonces en Línea 9. El cálculo de esta última línea, representa cuánto tiempo demora el nodo v en transmitir a todos sus hijos el mensaje, adicionándose el retraso que se va generando en cada altura del árbol. Es por eso que para la etiqueta de v no solo se considera la suma de la etiqueta de los hijos (que ya traen el retraso por la altura del árbol), sino que además se incrementa para cada hijo un intervalo de tiempo, necesario para efectivizar el reenvío a todos.

Figura 5.5: Resultado de aplicar el algoritmo *BroadcastTree*

Luego de aplicado el algoritmo *BroadcastTree*, queda completamente definido el orden de reenvío de mensaje para cada uno de los nodos en el árbol. En el ejemplo de esta figura, se puede apreciar además del identificador de cada nodo, el peso de las aristas (en rojo) que correspondía a la espera entre cada nodo, y finalmente, luego de la palabra *sort*, el orden en que cada nodo reenviará el mensaje a sus hijos. Se destaca que para los nodos hoja (1_B5 y 7_B1), el orden de reenvío esta indicado como $NULL$, representando que no tienen ningún hijo a quien transmitirles el mensaje. Luego, la mayoría de los nodos en el árbol tienen un solo hijo, por tanto, el orden de reenvío es trivial. El único caso que se presenta en el ejemplo, donde se decidió a cuál hijo reenviar primero, es para el nodo 1, reenviando primero al hijo 15 y luego al 1_B5.



Vale la pena remarcar como funciona el proceso de etiquetado: el mensaje es transmitido al nodo con la restricción de tiempo más rigurosa (esto es, la mayor etiqueta) primero.

Este esquema de reenvío al estilo goloso (Línea 12) alcanza optimalidad en árboles, y es devuelto en la Línea 13.

En la Figura 5.5 se puede observar cómo resulta el algoritmo para la instancia de grafo que se viene presentando en figuras anteriores.

5.2.3. Algoritmo $BroadcastBlock$

Finalmente se considera el algoritmo $BroadcastBlock$. Este algoritmo recibe un bloque B_i, el orden de reenvío esperado $F(B_i)$ y todos los k puntos de corte, a visitar, $v'_1, v'_2, \ldots, v'_k$. Para el bloque, el nodo v'_1 es el nodo origen desde donde el mensaje comienza a ser difundido.

Algorithm 3 $F_i = BroadcastBlock(B_i, v'_1, v'_2, \ldots, v'_k)$

```
1: T ← DynamicTree(B_i, v'_1)
2: for j = 1 to k do
3:    for all B_j ∈ B(v'_i) do
4:       F_j ← BroadcastBlock(B_j, F(B_j))
5:       T ← T ∪ {(v'_i, B_j)}
6:       w(v'_i, B_j) ← F_j
7:    end for
8: end for
9: F_i ← BroadcastTree(T, v'_1)
10: return F_i
```

En la Línea 1, el algoritmo $DynamicTree$ es aplicado al bloque B_i para encontrar el árbol de recubrimiento mínimo, con raíz en v'_1, que mejor se ajuste para optimizar el reenvío del mensaje en el bloque. Este paso es de vital importancia para tratar de encontrar un mejor árbol de recubrimiento que el retornado por el método Kruskal, que no se preocupa por minimizar los tiempos, sino solamente la cantidad de reenvíos que se realizan.

Una instancia de grafo que muestra claramente la necesidad del algoritmo auxiliar $DynamicTree$ es el grafo completo (se puede observar en la Figura 5.6), donde un árbol de recubrimiento mínimo es el mostrado en la Figura 5.7.

Algorithm 4 $TB_i = DynamicTree(B_i, v_0)$

```
1: for all v ∈ B_i do
2:    T ← T ∪ shortestPath(B_i, v_0, v)
3: end for
4: TB_i ← BroadcastTree(T, v_0)
5: for all v ∈ B_i do
6:    N'(v) = {n ∈ B_i : n ∈ N_{B_i}(v) and n ∉ N_{TB_i}(v)}
7:    t_v = receivedTime(v, TB_i)
8:    sendsDone = #(N_{TB_i}(v))
9:    for all n ∈ N'(v) do
10:      t_n = receivedTime(n, TB_i)
11:      t'_n = t_v + sendsDone + 1
12:      if t_n > t'_n then
13:         TB_i ← TB_i − E(to(n))
14:         TB_i ← TB_i + (v, n)
15:         diff = t_n − t'_n
16:         updateReceivedTime(TB_i, n, diff)
17:         sendsDone = sendsDone + 1
18:      end if
19:   end for
20: end for
21: return TB_i
```

Ciertamente, la cantidad de reenvíos es óptima, puesto que para transmitir a todos los nodos solo se requieren cinco reenvíos. Lo que no es óptimo en ese ejemplo es justamente el tiempo necesario para completar la difusión. Esto se debe a que de acuerdo al árbol generado, solo el nodo origen tiene la capacidad de reenviar el mensaje, por tanto le costará cinco intervalos de tiempo culminar con la difusión.

Suponiendo para simplificar, que el nodo origen reenvía los mensajes en el siguiente orden:[2,3,4,5,6]; es importante mostrar y resaltar que el nodo 2 posee el mensaje desde el tiempo $T = 1$, teniendo oportunidad de reenviarlo a cualquiera de sus nodos vecinos. En el grafo completo, los vecinos del nodo 2 son todos, pero de acuerdo al árbol de recubrimiento generado, esta oportunidad de reenvío no es aprovechada. Lo mismo sucede para el nodo 3 a partir del tiempo $T = 2$, y sucesivamente para el resto de los nodos del grafo.

Queda en evidencia entonces como la mayoría de los potenciales canales de comunicación no son utilizados. Se acuerda llamar a estos enlaces no utilizados *canales ociosos*, y corresponden entonces a todas las aristas que se encuentran en el grafo original, pero no en el MST generado.

Figura 5.6: Bloque grafo completo

Este grafo es un ejemplo de posible bloque sobre el cual se debe difundir el mensaje. Tiene la particularidad de contar con todos los posibles canales de comunicación, ya que cada nodo esta conectado con todos los restantes por una arista.

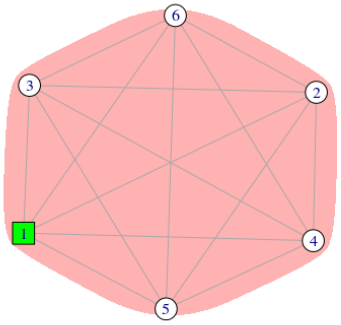
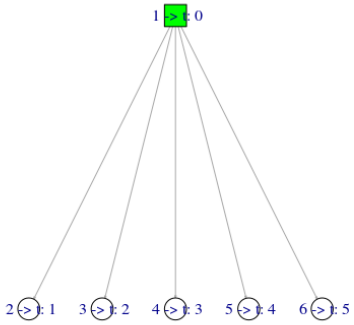


Figura 5.7: Paso 0 del algoritmo *DynamicTree*

Se genera un MST para el grafo mostrado en la Figura 5.6. Se destaca que si bien es mínimo en cantidad de enlaces, no lo es para resolver el problema MBT, debido a que bajo esa estructura, no es mínimo el tiempo requerido para la difusión del mensaje.



Entonces, *DynamicTree* es aplicado de la siguiente forma.

Primero, en el bloque de Líneas 1-3, un árbol de cubrimiento mínimo es construido a partir de un grafo vacío. La manera es tomar como nodo raíz a v'_1, y luego ir agregando uno a uno, los caminos más cortos desde éste a todos los otros nodos en el bloque. Se puede visualizar en la Figura 5.7 este paso del algoritmo.

Seguidamente, para poder encontrar el tiempo en el cual cada nodo recibirá el mensaje si el árbol generado fuese el definitivo, se utiliza en la Línea 4 el método *BoadcastTree*. Hasta este momento entonces se tiene una primera versión de cómo se reenviará el mensaje en el bloque, a través del árbol de cubrimiento generado, y en los tiempos recién encontrados. El árbol encontrado se denomina TB_i. En las siguientes etapas del algoritmo lo que se busca es ir mejorando el esquema de reenvío.

En el bloque de Líneas 5-20, para cada nodo v y en orden secuencial, el algoritmo explora la *capacidad ociosa* del mismo, es decir, todos los canales ociosos que potencialmente mejorarían el tiempo de difusión del mensaje en el bloque. Dicho de otro modo, lo que se hace es para cada nodo v, explorar a todos sus vecinos en el bloque B_i $(N_{B_i}(v))$, que no son vecinos en el árbol de recubrimiento mínimo TB_i $(N_{TB_i}(v))$. Se elige llamar a este conjunto de nodos $N'(v)$.

Esta exploración se realiza porque se sabe que luego de que v termine de enviar el mensaje según lo indicado por en el árbol de cubrimiento TB_i, se podría utilizar la ventaja de su capacidad ociosa en el bloque enviando a cualquiera de los nodos

de $N'(v)$ para así reducir los tiempos.

El procedimiento entonces consiste en, para cada nodo n en $N'(v)$, si el tiempo en el que el mensaje puede ser enviado a n desde v es menor que el tiempo en el que era enviado en el árbol TB_i, se altera TB_i de modo de impactar esta mejoría. Cabe destacar que el tiempo en el que posiblemente v pudiese enviar el mensaje a n, es un intervalo más que el tiempo que gastó en enviar a sus hijos en TB_i.

Se puede entender este paso del algoritmo visualizando la Figura 5.7 que representa el árbol TB_i. Si se considera por ejemplo el nodo $v = 2$, el conjunto $N'(v)$ esta constituido por los nodos 1, 3, 4, 5 y 6. El nodo 2 recibe el mensaje en $T = 1$, por tanto, podría en primer lugar reenviarlo a cualquiera de los nodos de $N'(v)$ en $T = 2$.

Al realizar la exploración, se encuentra que:

- para el nodo 1: el mensaje ya lo tenía en $T = 0$, por lo tanto no mejora el tiempo en el que se recibe reenviárselo en $T = 2$,
- para el nodo 3: el mensaje ya lo tenía en $T = 2$, por lo tanto no mejora el tiempo en el que se recibe reenviárselo en $T = 2$,
- para el nodo 4: el mensaje lo tenía en $T = 3$, por lo tanto, que el nodo 2 le reenvíe el mensaje en $T = 2$ mejora el tiempo en que lo recibe y se debe entonces alterar el árbol TB_i.

Si sucede que se debe modificar el árbol TB_i, porque se encontró un canal ocioso $v - n$ que mejora los tiempos, se realiza de la siguiente forma.

Primero, se borra el enlace por el cual el mensaje originalmente había sido enviado a n (Línea 13). Esto significa que el mensaje no continuará llegando desde el elegido en el árbol original. En el caso del ejemplo que se viene estudiando, el enlace de TB_i que se borra es el que va del nodo 1 al nodo 4.

Segundo, se agrega el enlace desde v hasta n (Línea 14), porque el nuevo canal usado es justamente a partir de v. En el ejemplo, el enlace a agregar sería el que va desde el nodo 2 hasta el nodo 4. Este paso puede verse en la Figura 5.8.

Cabe señalar que el sub-árbol de TB_i que cuelga de n se mantiene de la misma forma, con la única diferencia de que n cambió de padre. Esto significa que en principio, luego de que el mensaje alcance n, se seguirá reenviado como antes de la modificación. Igualmente, cuando el algoritmo avance, y se llegue a iterar sobre n, explorando su capacidad ociosa (o sea, cuando v sea justamente el nodo n), se tendrá eventualmente la oportunidad de cambiar.

Finalmente, el próximo paso consiste en actualizar los tiempos de arribo para el sub-árbol de TB_i que cuelga de n (Línea 16) ante el cambio previamente realizado. Esto es, impactar la diferencia de tiempo ganada por el cambio en cada nodo del sub-árbol, incluido la raíz n. El cambio en los tiempos se calcula de forma simple, restando a cada uno de los nodos del sub-árbol, la ganancia de haber recibido el mensaje en n antes. Si el mensaje se adelanta un intervalo de tiempo como en el ejemplo, sencillamente se resta 1 a los tiempos de cada uno de los nodos del sub-árbol. Esto esta fuertemente fundamentado en lo mencionado en el párrafo anterior: el orden de reenvío en el sub-árbol no se modifica.

Figura 5.8: *DynamicTree*: iteración 1

Comienza la exploración de los nodos del árbol y sus canales ociosos a partir del nodo 2. Se pasa por 1 y por 3 sin encontrar una posible mejora. Al llegar al nodo 4 se encuentra una mejoría en los tiempos, por lo tanto, se altera el árbol, tanto en la forma como en los tiempos de arribo del mensaje.

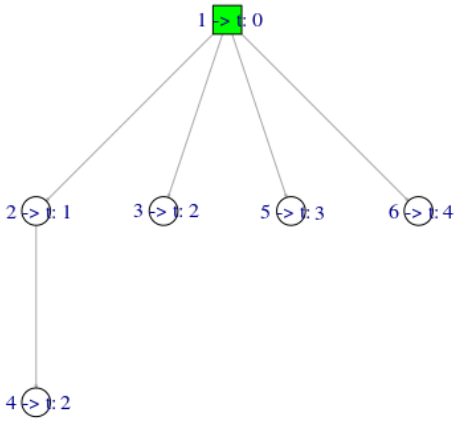
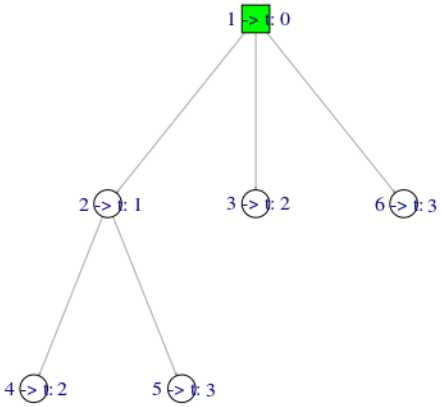


Figura 5.9: *DynamicTree*: iteración 2

Continua la exploración de los canales ociosos a partir del nodo 2. Para el nodo 5 se encuentra una mejoría en los tiempos, por lo tanto, se vuelve a alterar el árbol.



En las Figuras 5.8 y 5.9 se puede ver como estos pasos sucedieron para el ejemplo propuesto.

Como consecuencia de aplicar este algoritmo, puede que ahora el nodo v envíe el mensaje a uno o más de sus vecinos (Línea 17). En el ejemplo, el nodo 2 luego de este procesamiento pasa a enviar el mensaje a dos de sus vecinos, mientras que antes no reenviaba a ninguno.

Es muy importante recordar que todo cobra sentido, siempre y cuando efectivamente el tiempo en el que el mensaje llega a los nodos de $N'(v)$ sea menor con el cambio frente a lo que se obtenía con el árbol original.

Este procedimiento es realizado una vez y solo una vez por cada nodo en el bloque B_i.

Para el caso del ejemplo, luego de la iteración 2, si bien se itera sobre la totalidad de los nodos, ya no se encuentra mejoría en cuanto a los tiempos, y por lo tanto, no se vuelve a alterar el árbol y esa sección del código culmina.

Considerando nuevamente el algoritmo *BroadcastBlock*, se destacan dos importantes características: primero, que el algoritmo debe retornar el árbol de reenvío para los nodos del bloque; segundo, que dentro del bloque posiblemente existan otros puntos de corte (a menos que sea un bloque hoja), lo que significa la posibilidad de salir a explorar otros bloques con los cuales el bloque en cuestión se conecta.

Mientras sea más rentable, se querrá salir y explorar los otros bloques antes de continuar con el reenvío dentro del mismo. Pero hay casos en que esta estrategia no es la más adecuada. Es por eso que la exploración se realiza recursivamente; y antes de definir completamente cuál es el orden de reenvío, se realiza un procedimiento artificial, que se detalla a continuación.

Por cada punto de corte en el bloque, recursivamente es aplicado $BroadcastBlock$ para obtener los intervalos de tiempo requeridos para cubrir los bloques que están conectados por estos nodos. Esto significa, las intervalos de tiempo necesarios para completar el reenvío en dicha dirección. Para cada uno, luego de haber obtenido el resultado F_j, un nuevo nodo artificial es agregado al bloque desde el punto de corte y el enlace correspondiente tiene peso F_j (Líneas 2-7).

Finalmente, la función $BroadcastTree$ es aplicada en el árbol del bloque con la última modificación descrita, para definir un esquema de reenvío intra-bloque (Línea 9) y el resultado es retornado en Línea 10.

Se debe tener en cuenta que cuando el bloque no tienen ningún punto de corte, entonces este procedimiento artificial no es realizado, y simplemente la función $BroadcastTree$ es aplicada en el árbol encontrado por $DynamicTree$.

Una importante aclaración acerca de cómo el algoritmo $BroadcastTree$ es aplicado, refiere al tratamiento de los nodos artificiales que fueron previamente agregados. En la Línea 9 del algoritmo $BroadcastTree$, la etiqueta es calculada agregando el valor i, que corresponde a la posición relativa en la que cada nodo fue notificado. Para un nodo ordinario, este continua siendo el caso. Sin embargo, para los nuevos nodos artificiales, i no significa solo una unidad de tiempo (incremental por cada hijo). Estos casos refieren en cambio, al tiempo requerido para notificar a cada hijo del punto de corte en el bloque que esta siendo representado antes de volver al esquema de reenvíos en el bloque original. Por lo tanto el valor i, ya no corresponde a uno, sino al numero de hijos del punto de corte en el bloque representado por el nodo artificial.

Es así que con la combinación de los algoritmos presentados, se puede calcular tanto tiempo de difusión desde v_0, así como el árbol de reenvío para el grafo G original.

Capítulo 6

Análisis experimental

El rendimiento de la nueva heurística *TreeBlock* creada para el MBT es evaluado desde diferentes aspectos.

En primer lugar, como el problema se estudió previamente para topologías conocidas, se comparan los resultados alcanzados mediante la heurística con los arrojados por las otras soluciones propuestas en la literatura para los mismos grafos.

En segundo lugar, se explora el rendimiento de la heurística sobre un conjunto de grafos generados siguiendo el modelo pequeño mundo de Watts-Strogatz [32], que es un modelo aceptable para redes de comunicación reales [31]. Para este conjunto de datos, se busca comparar contra los resultados obtenidos por el método exacto, siempre que sea posible.

Finalmente, en las últimas dos subsecciones se busca entender cómo escala la heurística frente a cambios en la dimensión de las instancias de grafos, tanto a nivel de cantidad de nodos como de densidad de aristas, manteniendo la topología Watts-Strogatz.

En cuanto a la infrastructura utilizada para cada uno de los casos de prueba, se destacan los siguientes aspectos. Todas las pruebas de la solución heurística fueron ejecutadas en equipos domésticos: *Quad Core de 2.5GHz, con 16 Gb de RAM*. Las pruebas de la solución exacta se ejecutaron en CPLEX, un paquete comercial de código altamente optimizado, dentro de un servidor dedicado de alto rendimiento. Asimismo, se desconoce las características de los equipos de ejecución utilizados para las pruebas presentadas en la literatura abordada.

Debido a la heterogeneidad del *hardware* utilizado para cada uno de los casos, el foco de este estudio no está puesto en el tiempo de cómputo.

6.1. Topologías conocidas

Siguiendo los estudios previos, se presentan los resultados de la heurística para cinco topologías bien conocidas: *Lattice*, *De Bruijn*, *Hypercube*, *Harary* y *Chord Harary*.

Las Tablas 6.1 a 6.5 muestran para cada instancia probada, el número de nodos

(N), el resultado óptimo según la formulación ILP (ILP) y el resultado heurístico $TreeBlock$ (TB). Las otras columnas corresponden a datos específicos de cada una de las topologías.

La primera topología de red que se estudia es la que corresponde a los grafos de tipo grilla en dos dimensiones. En las Figuras 6.1 y 6.2 se pueden ver ejemplos de este tipo de grafo. En este caso, una instancia de problema se define por el número de filas R y el número de columnas C de la grilla, y por el nodo origen definido como (r, c), donde $1 \leq r \leq R$ corresponde al número de fila y $1 \leq c \leq C$ al número de columna.

Figura 6.1: Ejemplo de Lattice (grilla): dos dimensiones, $R = 5$ y $C = 4$

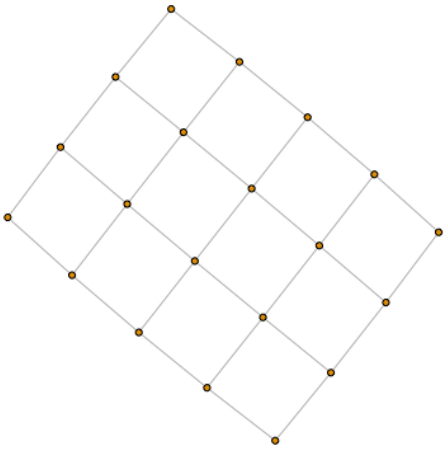
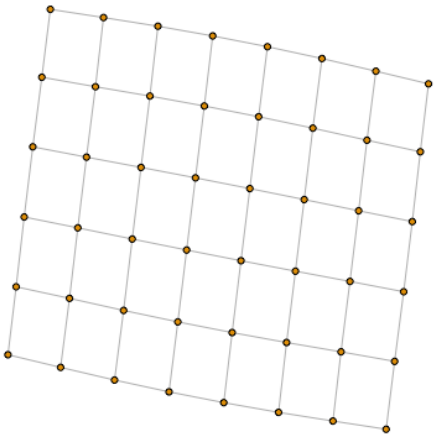


Figura 6.2: Ejemplo de Lattice (grilla): dos dimensiones, $R = 8$ y $C = 6$



Siguiendo [15], cuando el nodo origen es $(1, 1)$, el tiempo de difusión es precisamente el diámetro $D = R - 1 + C - 1 = R + C - 2$. La Tabla 6.1 resume los resultados de 16 instancias de grillas 2D aleatorias. La heurística $TreeBlock$ alcanza el resultado óptimo en todos los casos.

Tabla 6.1: Lattice

R	C	r	c	N	ILP	TB
7	10	1	1	70	15	15
7	10	2	3	70	12	12
7	10	4	1	70	13	13
7	10	4	5	70	9	9
9	15	1	1	135	22	22
9	15	3	4	135	17	17
9	15	5	1	135	19	19
9	15	5	8	135	13	13
11	20	1	1	220	29	29
11	20	3	5	220	23	23
11	20	6	1	220	25	25
11	20	6	10	220	16	16
13	30	1	1	390	41	41
13	30	4	8	390	31	31
13	30	7	1	390	36	36
13	30	7	15	390	22	22

En segundo lugar se estudia una topología de grafo introducida por Frank Harary en [12]. Estos grafos son los llamados *grafos de Harary*, y se denotan $H_{k,n}$. Los grafos de Harary son k-conexos en n nodos que tienen un grado de al menos k. La estructura de $H_{k,n}$ depende de las paridades de k y de n. En las Figuras 6.3 y 6.4 se pueden ver ejemplos de este tipo de grafo.

Figura 6.3: Ejemplo de Harary $H_{2,8}$

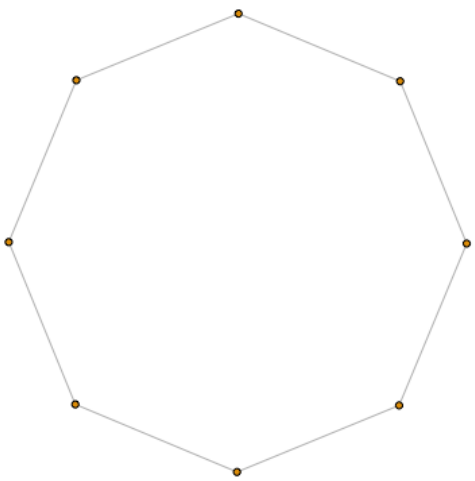
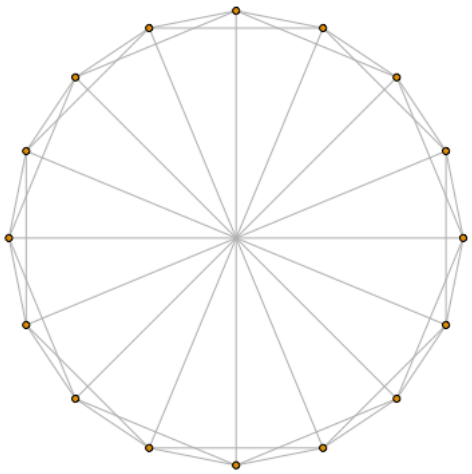


Figura 6.4: Ejemplo de Harary $H_{5,16}$



El algoritmo analizado en [5] asegura encontrar el resultado exacto del tiempo

de difusión para la mitad de los casos. Los resultados son en su mayoría óptimos en las instancias estudiadas con la heurística $TreeBlock$; en los otros casos, resultan ser muy cercanos, siendo un intervalo de tiempo la diferencia.

Tabla 6.2: Harary. Broadcast scheme S. [4]

N	k	n	ILP	TB
17	2	17	9	9
17	3	17	5	5
17	5	17	5	5
17	6	17	5	5
17	7	17	5	5
30	2	30	15	15
30	3	30	9	9
30	8	30	5	6
30	9	30	5	6
30	10	30	5	6
50	2	50	25	25
50	3	50	14	14
50	11	50	-	7
50	20	50	-	8
50	21	50	-	7
100	2	100	50	50

Otro caso de estudio son los grafos Chord-Harary. Estos son una variación del grafo de Harary, $CH_{k,n}$, introducidos en [4] intencionalmente, en función de obtener un tipo de grafos tal que se pueda alcanzar valores bajos de tiempo de difusión mediante un algoritmo también propuesto por los autores. Además, fue demostrado en esa publicación que el algoritmo que se propone alcanza el nivel óptimo para un subconjunto de estos grafos, donde k mantiene cierta relación con n. En las Figuras 6.5 y 6.6 se pueden ver ejemplos de este tipo de grafo.

Figura 6.5: Ejemplo de Chord-Harary $CH_{5,16}$

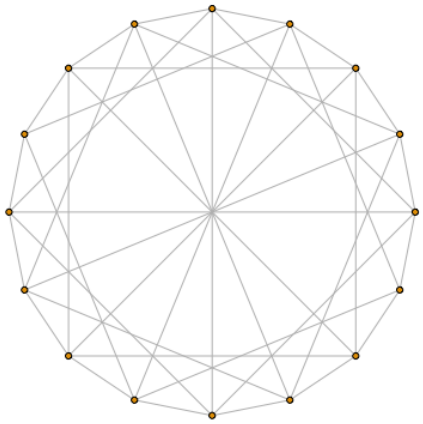
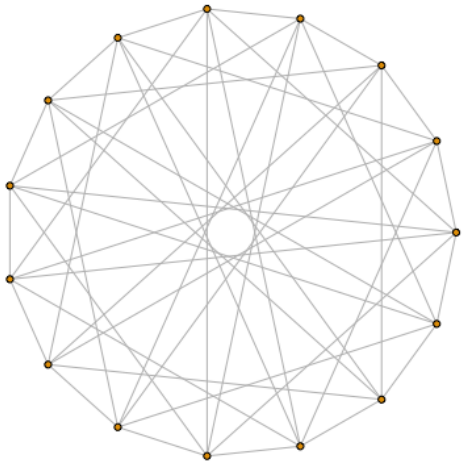


Figura 6.6: Ejemplo de Chord-Harary $CH_{7,15}$



La Tabla 6.3 muestra que los resultados alcanzados por la heurística *TreeBlock* siguen siendo competitivos, e incluso en algunos casos alcanza el resultado exacto. Las instancias elegidas no tuvieron en cuenta la relación entre los parámetros para los cuales el algoritmo de [4] logra la optimalidad precisamente para observar cómo se comporta en general.

Tabla 6.3: Chord-Harary. Broadcast Scheme B. [4]

N	k	n	ILP	TB
10	3	10	4	4
10	5	10	4	4
30	3	30	9	9
30	5	30	5	6
30	7	30	5	6
50	5	50	6	7
50	7	50	-	7
50	9	50	-	7
50	11	50	-	6
100	7	100	7	8
100	11	100	-	8
100	13	100	-	8
300	7	300	-	15
300	11	300	-	10
300	17	300	-	10

Figura 6.7: Ejemplo de HyperCube H_3

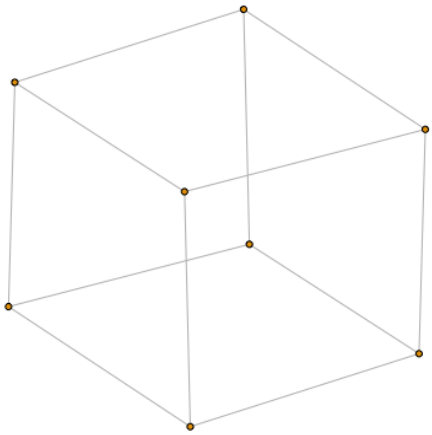


Figura 6.8: Ejemplo de HyperCube H_4

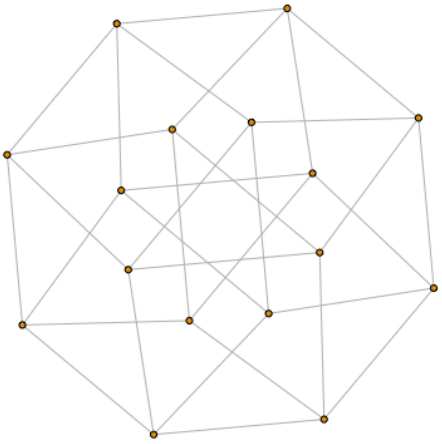


Figura 6.9: Ejemplo de De Bruijn $DB_{2,2}$

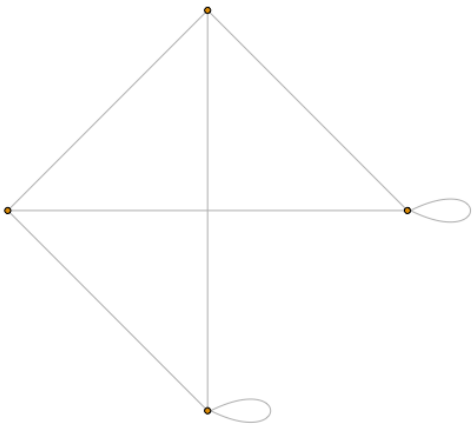
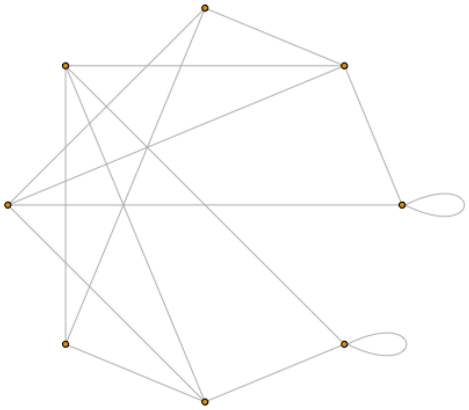


Figura 6.10: Ejemplo de De Bruijn $DB_{2,3}$



Finalmente, siguiendo lo que se presenta en [14] en la Tabla 6.4 y la Tabla 6.5 se puede observar que el algoritmo $TreeBlock$ logra resultados competitivos con la heurística que allí se propone para los casos analizados. Ejemplos de estos grafos se presentan en las Figuras 6.9, 6.10, 6.8 y 6.7.

Tabla 6.4: De Bruijn. RH, TBA, NTBA [14]

N	m	n	RH	TBA	NTBA	ILP	TB
8	2	3	4	4	4	3	4
16	2	4	5	5	5	4	5
32	2	5	7	6	7	6	7
64	2	6	8	8	8	7	8
128	2	7	9	9	10	-	10
256	2	8	11	11	12	10	12
512	2	9	12	12	13	-	14
1024	2	10	14	14	15	-	15
2048	2	11	15	15	17	-	17
4096	2	12	17	17	19	-	19
8192	2	13	18	18	20	-	21

Tabla 6.5: Hypercube. C, TBA, NTBA [14].

N	l	dim	O	C	TBA	NTBA	TB
16	2	4	4	-	-	-	4
32	2	5	5	5	5	5	5
64	2	6	6	7	6	7	6
128	2	7	7	8	7	9	7
256	2	8	8	9	9	11	8
512	2	9	9	10	10	14	9
1024	2	10	10	12	11	15	10
2048	2	11	11	-	12	18	11
4096	2	12	12	-	13	20	12

Luego de probados el método exacto *ILP* y la heurística *TreeBlock* en diferentes topologías, se resalta que la fórmula ILP, si bien no es capaz de resolver todas las instancias ejecutadas, alcanza los valores óptimos para muchos de los casos.

Por otro lado, la heurística *TreeBlock*, que fue desarrollada sin especializarse en ningún tipo de grafo en particular y con foco en grafos con más de una componente biconexa, logra resultados muy competitivos.

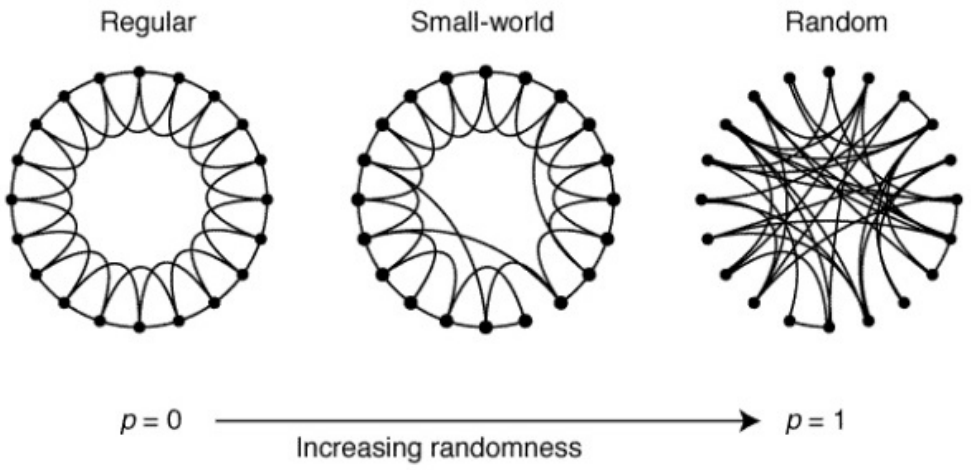
6.2. Grafos Watts-Strogatz

Las redes reales (redes sociales, biológicas, de transporte, comunicación) no son topológicamente ni regulares ni aleatorias, sino que suelen encontrarse en una

posición intermedia entre ambos tipos, es decir, poseen propiedades características de los dos grandes grupos de grafos. Watts y Strogatz han estudiado estas redes y han buscado modelos que se ajusten a ellas, creándolos finalmente a partir de "provocar un cierto desorden" en un grafo regular [32] [30]. Han demostrado que estas redes tienen un alto índice de agrupamiento, como los grafos regulares, además de una longitud de camino característica pequeña como los grafos aleatorios. Por las analogías de estas investigaciones con otras realizadas en psicología por Stanley Milgram en torno a 1960, en las cuales se experimentaba con las redes sociales norteamericanas, demostrando que cualquier par de personas estaba conectada mediante lazos de amistad por no más de seis personas intermedias, llamando a este efecto "pequeño mundo"; este tipo de redes han sido denominadas redes Small World. Este fenómeno es además particularmente cierto en las redes de telecomunicaciones, donde el problema en estudio se aplica.

El modelo propuesto por Watt y Strogatz en 1998 es el siguiente: se comienza a partir de un anillo de n nodos y k aristas por nodo. Luego se van considerando las distintas aristas, y con una probabilidad p se mueve un extremo de ella de un nodo a otro nodo seleccionando este último aleatoriamente, con la salvedad de que no se pueden crear dos aristas entre un par de nodos ni tampoco lazos. Así, cuando $p = 0$, se obtiene un grafo regular, mientras que cuando $p = 1$ el grafo es completamente aleatorio.

Figura 6.11: Grafo Watts-Strogatz (pequeño mundo) [32]



Con el objetivo de que el análisis experimental se ciña lo más posible a la realidad, es que en las siguientes etapas se experimenta con redes de este tipo, variando los parámetros para su construcción. En particular, se estudia en las diferentes subsecciones tanto la calidad de los resultados arrojados por la heurística $TreeBlock$, siempre que sea posible; como los tiempos de ejecución en la medida que los parámetros de construcción de las diferentes instancias vayan cambiando.

6.2.1. Comparativa de resultados con método exacto

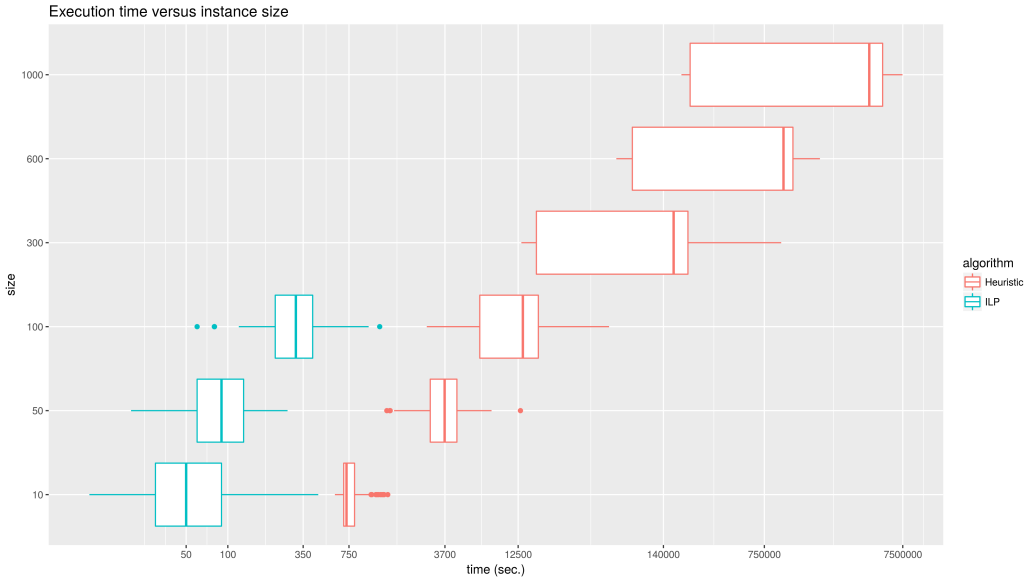
Esta primera subsección busca principalmente estudiar de forma comparativa los resultados que se obtienen con la heurística, frente a los que la fórmula exacta ILP puede encontrar. Es de especial interés para acercarse a una noción de correctitud de la heurística construida, por lo que se busca explorar diferentes tipos de instancias variando tanto tamaño como densidad de aristas, siempre en la medida que ILP sea capaz de resolver.

Para la muestra fueron construidos 301 instancias de grafos pequeño mundo de 10 nodos, y además 100 instancias de grafos del mismo tipo para cada una de las siguientes cantidades de nodos: 50, 100, 300, 600 y 1000. Para todos los casos, la estrategia fue elegir la probabilidad de recableado p de forma uniformemente independiente debido a que es esta variable la encargada de la aleatoriedad del grafo, y la variable *neighborhood* porcentual a la cantidad de nodos, a modo de cubrir diferentes densidades de aristas, pero sin prestar atención a cómo se lograba. El detalle de las instancias utilizadas en esta subsección puede verse en el Apéndice A.1.

Para cada instancia, se ejecutaron tanto la heurística construida como la fórmula ILP para hacer una comparación de los resultados.

Figura 6.12: Comparativa: tiempo de ejecución versus tamaño de la instancia

Tiempo de ejecución en función del tamaño de las instancias en términos de nodos, para los algoritmos *ILP* y *TreeBlock*



Cabe destacar que de toda la muestra creada sólo las instancias de grafos de 10, 50 y 100 nodos pudieron ser resueltas mediante la fórmula exacta. Por lo tanto, la muestra efectiva sobre la que se puede analizar en esta subsección termina siendo la correspondiente a estos casos y no la inicial.

Se puede apreciar este aspecto en la Figura 6.12, donde queda en evidencia

que no fue posible arrojar una solución con la fórmula ILP, principalmente para las instancias más grandes. Esto se debe a la complejidad que supone el algoritmo, consecuencia de la complejidad del problema.

Con respecto a los tiempos de procesamiento, en los casos donde ILP encontró una solución óptima, lo hizo en menos tiempo que la heurística. De todos modos, este aspecto no se considera del todo relevante debido a que:

i) ILP se ejecuta en CPLEX, un paquete comercial de código altamente optimizado, mientras que la heurística es un código de investigación no optimizado.

ii) CPLEX se ejecuta en un servidor de muy alto rendimiento mientras que la heurística se ejecuta en equipos domésticos.

Asimismo, se observa en las Figuras 6.13 y 6.14 que los tiempos de ejecución de la heurística también están estrechamente relacionados con el tamaño de la instancia, tanto por el número de nodos, como también como por el número de aristas. Esto se desprende como consecuencia del cambio de color de los puntos del gráfico.

Figura 6.13: Tiempo de ejecución de heurística I

Se representa el tiempo de ejecución en función al resultado encontrado. Además, se visualiza con colores el tamaño de la instancia en términos de nodos. No se presenta relación entre la solución y el tiempo de ejecución, pero sin embargo, si se visualiza que el tamaño de la instancia está fuertemente relacionado. A mayor cantidad de nodos, mayor el tiempo de ejecución de la heurística (queda claro como el color de los puntos va modificándose en tanto se crece en las ordenadas)

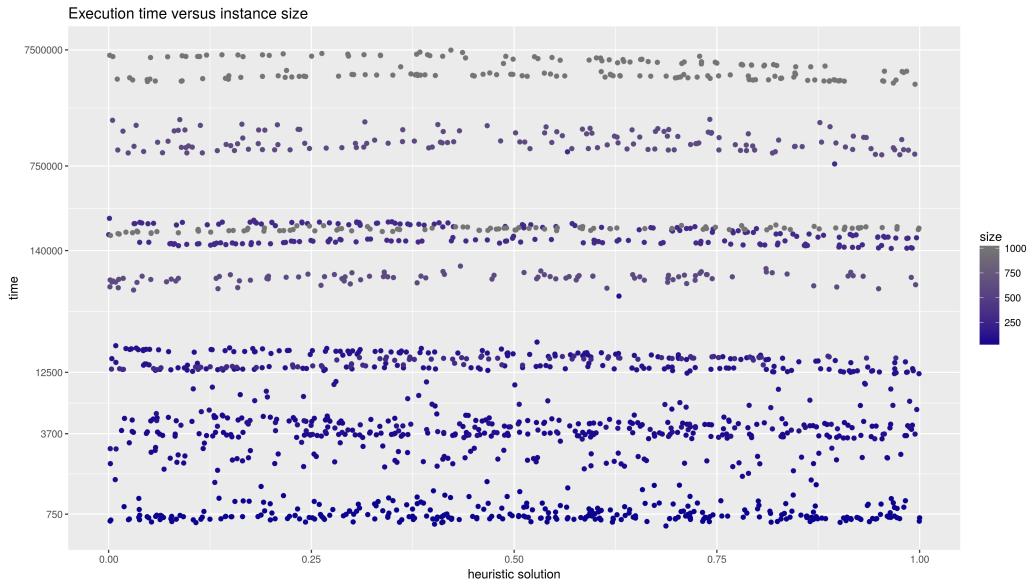
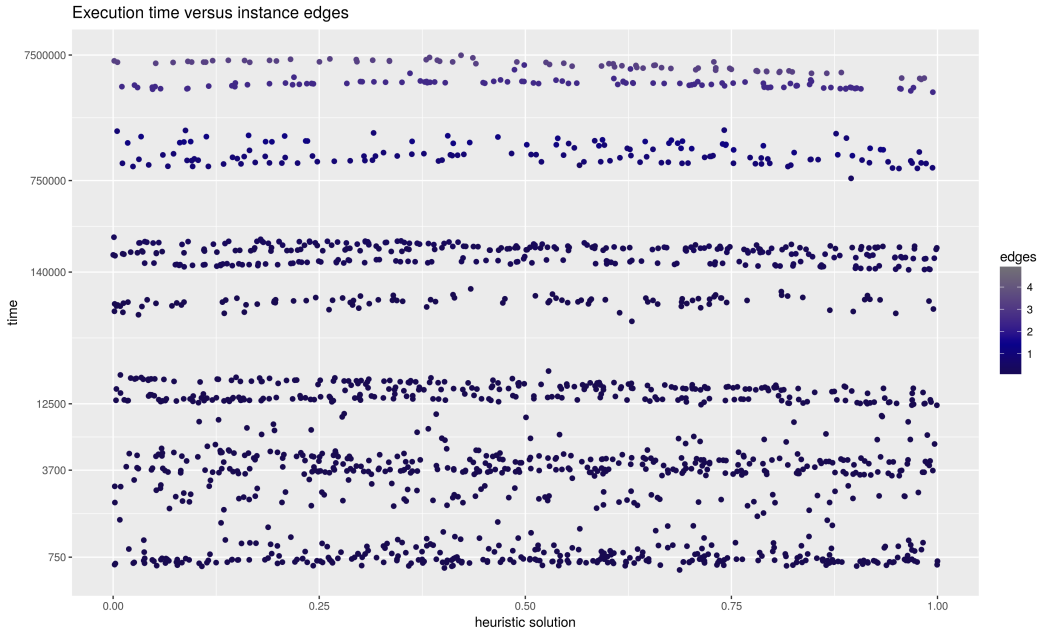


Figura 6.14: Tiempo de ejecución de heurística II

Se representa el tiempo de ejecución en función al resultado encontrado. Además, se visualiza con colores el tamaño de la instancia en términos de aristas. No se presenta relación entre la solución y el tiempo de ejecución, pero sin embargo, si se visualiza que el tamaño de la instancia está fuertemente relacionado. A mayor cantidad de aristas, mayor el tiempo de ejecución de la heurística (queda claro como el color de los puntos va modificándose en tanto se crece en las ordenadas). Las aristas están en una escala de $1e^{-5}$



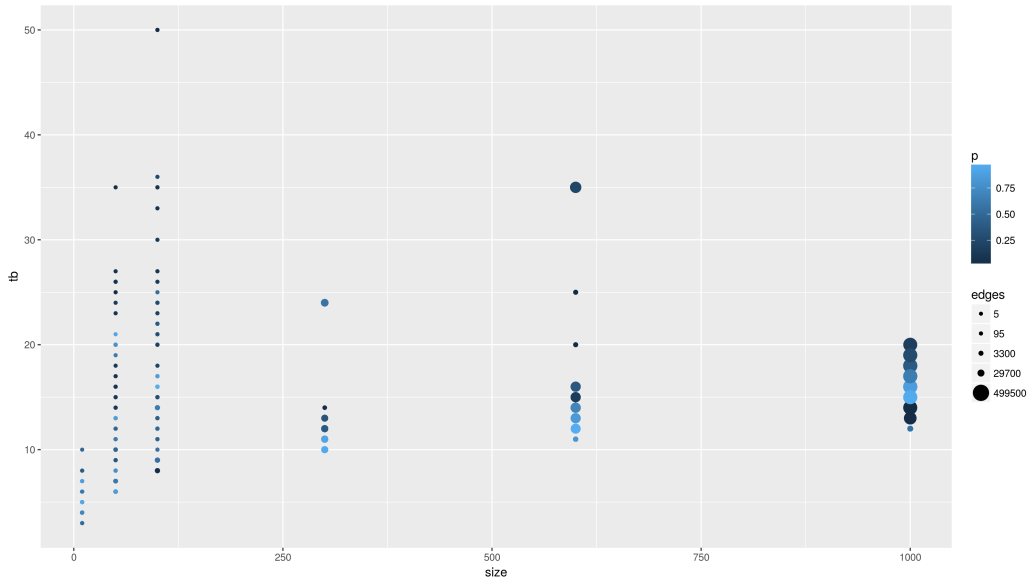
Considerando grupos de instancias de igual tamaño (nodos y aristas), en la Figura 6.15 se ve una tendencia a alcanzar un tiempo de transmisión más alto cuando la probabilidad de recableado es baja. Esta observación parece ser correcta teniendo en cuenta el proceso de creación de los grafos del tipo pequeño mundo, porque con una baja probabilidad de recableado, los grafos serían más similares al anillo inicial. En el peor de los casos, un anillo, con un tiempo de transmisión igual al número de nodos -1.

Finalmente, para las instancias en las que la fórmula ILP logró un resultado óptimo, se puede hacer una comparación en términos de la diferencia en el valor logrado por cada algoritmo.

En la Tabla 6.6 se muestra para cada grupo de instancias, cuál fue el rango de valores óptimos encontrados por la fórmula ILP y por la heurística. La columna "GAP" indica cuál es la diferencia promedio entre las soluciones arrojadas por cada método. Además, la columna "no GAP" indica el número de instancias para las cuales la heurística encontró el efectivamente óptimo.

Figura 6.15: Solución de la heurística en función de características de las instancias

Se muestra el resultado de la heurística en relación al tamaño de las instancias en cuanto a cantidad de nodos, agrupando por cantidad de aristas y probabilidad de recableado. La probabilidad p se representa por el coloreado, y la cantidad de aristas por el tamaño del punto en el gráfico.



En la Tabla 6.7, se puede ver en qué porcentaje los resultados de la heurística se aproximan al óptimo. Se destaca el segmento de 10 nodos, donde el 87 % de las instancias se resolvieron con una precisión superior al 90 %. Para el resto de los casos, no es menor enfatizar que el tamaño del conjunto de pruebas para el que fue posible que los dos algoritmos encontraran una solución es considerablemente menor.

Tabla 6.6: Rango de resultados por ILP y heurística

	size	edges	TB	ILP	GAP	no GAP	#instances
1	10	20.01	$4{,}39 \pm 0{,}91$	$4{,}22 \pm 0{,}65$	$0{,}04 \pm 0{,}12$	263	301
2	50	46.52	$12{,}85 \pm 5{,}33$	$11{,}51 \pm 5{,}27$	$0{,}14 \pm 0{,}17$	33	100
3	100	94.04	$16{,}9 \pm 7{,}4$	$14{,}46 \pm 7{,}13$	$0{,}19 \pm 0{,}21$	22	100

Tabla 6.7: Diferencia porcentual de resultados ILP y heurística

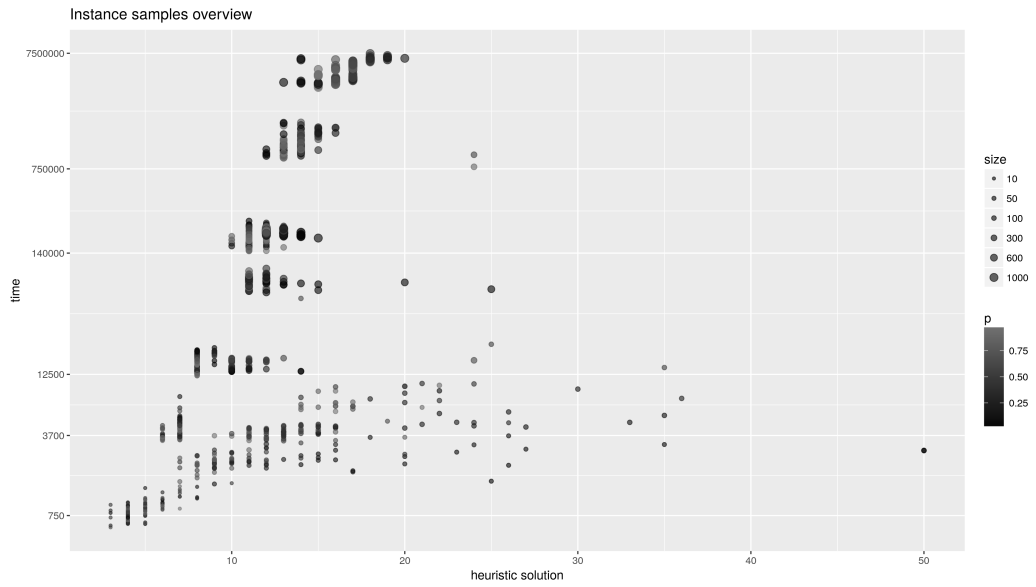
	size	edges	GAP $\leq 10\,\%$	GAP $\leq 25\,\%$	#instances
1	10	20.01	263	287	301
2	50	46.52	47	87	100
3	100	94.04	43	77	100

La Figura 6.16 muestra un resumen de la distribución de los datos en términos de tamaño y probabilidad de recableado en comparación con los tiempos y resultados.

Como novedad, se puede observar que a priori no existe una relación directa entre la solución y el tiempo de ejecución necesario para alcanzarla. Es decir, no necesariamente es requerido más tiempo de ejecución para que la heurística encuentre soluciones mayores. Queda en evidencia también en esta figura, la relación entre el tamaño de los grafos y los tiempos requeridos para la ejecución de la heurística.

Figura 6.16: Tiempo vs resultado, agrupado por tamaño y probabilidad

Se muestra el tiempo de ejecución de la heurística en función del resultado encontrado, agrupando por cantidad de nodos y probabilidad de recableado. La probabilidad p se representa por el coloreado, y la cantidad de nodos por el tamaño del punto en el gráfico.



6.2.2. Escalamiento de pruebas en cantidad de nodos

En esta subsección se estudia la relación entre la cantidad de nodos y el tiempo en hallar el resultado óptimo utilizando la heurística.

Para esto, se construye un conjunto de 1400 instancias de grafos pequeño mundo, fijando la variable k (*neighborhood*) en 1 y 3 y variando la cantidad de nodos; todos con una probabilidad de recableado p tomada de forma uniformemente independiente. Las cantidades de nodos que se utilizaron para la muestra fueron 100, 300, 600, 1000, 2000, 5000 y 10000, construyéndose 200 instancias para cada caso. Esta estrategia permite poder estudiar el potencial de la heurística en función del tamaño en cantidad de nodos, sin que los resultados sean impactados por la densidad de enlaces. El detalle de las instancias utilizadas en esta subsección puede verse en el Apéndice A.2.

Es preciso destacar en este punto, que el tamaño de instancias elegidas para este análisis es sensiblemente mayor que los ofrecidos hasta el momento en la literatura, donde se le llama *tamaño grande* a grafos en el orden de los dos mil

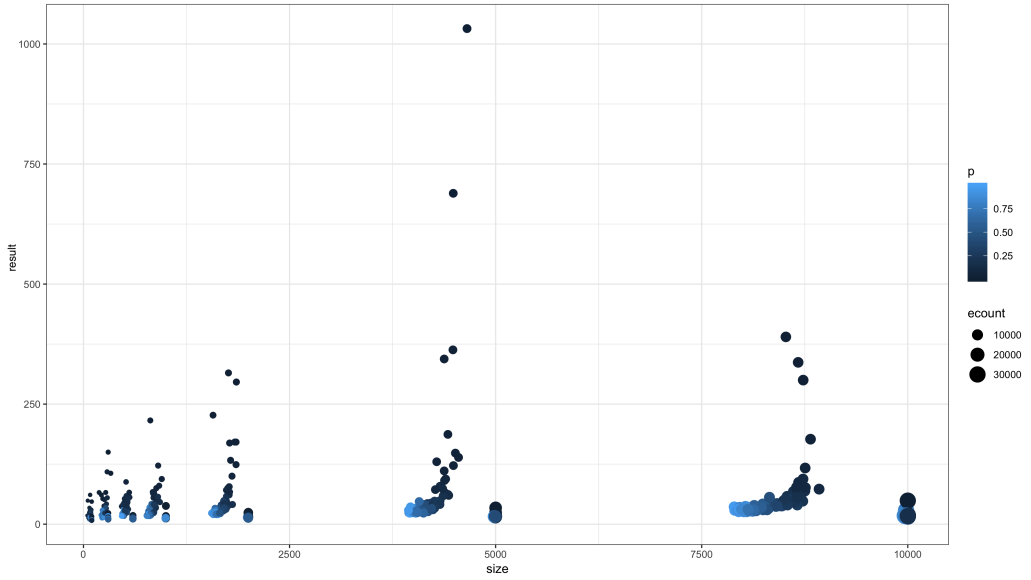
nodos [14].

Pasando a analizar los resultados obtenidos, en primer lugar, se destaca que debido a no contar con un método exacto que sea capaz de solucionar el problema en instancias de grafos de esta magnitud, no es posible afirmar el nivel de correctitud de los resultados alcanzados.

No obstante, es interesante hacer una revisión sobre la dispersión en base a las características de la muestra. Estos datos se observan en la Figura 6.17. Así como se menciona en la subsección anterior, en esta muestra también se percibe una tendencia a que el tiempo de transmisión sea más alto cuando la probabilidad de recableado es baja (en la figura se visualiza una mayor concentración de puntos oscuros en la parte inferior del gráfico). Se puede ver también que el resultado encontrado por la heurística es mayor en tanto crece la cantidad de nodos. Sin embargo, el incremento no es sustancial ni en la misma escala. Por último, se desprende del gráfico que algunas instancias en el orden de los 5000 nodos tuvieron un valor resultado bastante mayor al resto de la muestra. Según como se construyeron, esta diferencia se puede atribuir únicamente a la forma en que resultó el grafo luego del recableado.

Figura 6.17: Solución de la heurística en función de características de las instancias

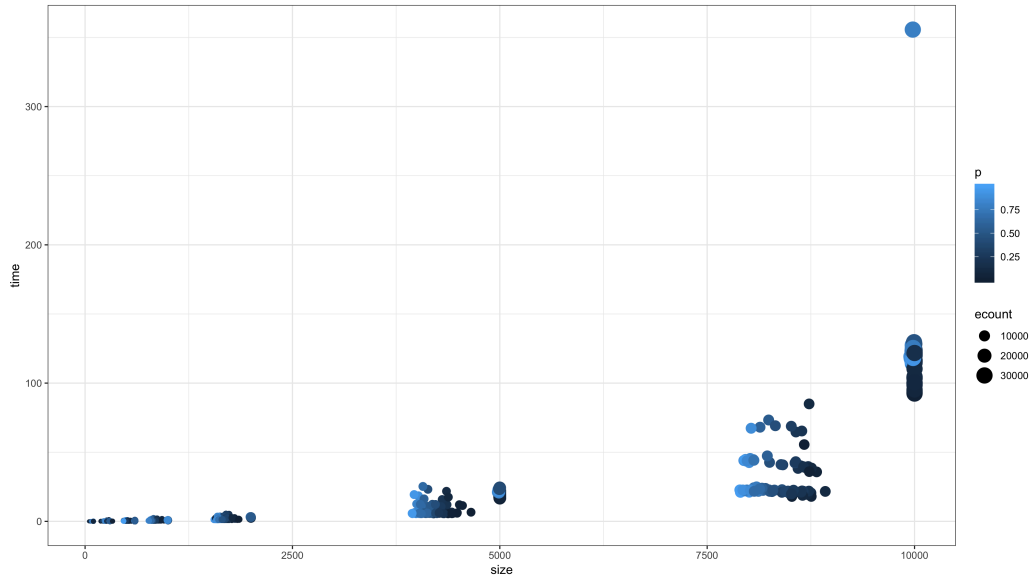
Se muestra el resultado de la heurística en relación al tamaño de las instancias en cuanto a cantidad de nodos, agrupando por cantidad de aristas y probabilidad de recableado. La probabilidad p se representa por el coloreado, y l a cantidad de aristas por el tamaño del punto en el gráfico.



Finalmente, se observa el comportamiento de la heurística en cuanto al tiempo de ejecución requerido para solucionar cada instancia. La Figura 6.18 muestra específicamente esta característica.

Figura 6.18: Tiempo de ejecución de la heurística en función de características de las instancias

Se muestra el tiempo de ejecución de la heurística en minutos en relación al tamaño de las instancias en cuanto a cantidad de nodos, agrupando por cantidad de aristas y probabilidad de recableado. La probabilidad p se representa por el coloreado, y la cantidad de aristas por el tamaño del punto en el gráfico.



Una primera conclusión y la más rápida, es que a medida que se aumenta la cantidad de nodos, aumenta el tiempo que se demora en hallar el resultado óptimo. Esto es natural, debido a que el algoritmo explora todos los nodos, por tanto si el grafo es más grande, hay más nodos que explorar.

De cualquier forma, se destaca que para los grafos de mayor magnitud, el resultado se alcanza en el orden de las dos horas; esto no es un tiempo exagerado. Se recuerda que el método exacto que sirvió de apoyo en la subsección anterior, cortaba la ejecución a las dos horas, y fue capaz de resolver instancias sólo de hasta mil nodos. En el mismo tiempo, la heurística alcanza un valor, que se estima cercano al óptimo, en instancias diez veces mayores en cuanto a cantidad de nodos.

Es menester resaltar, que dentro de cada rango de cantidad de nodos, los tiempos requeridos para la resolución son similares. Se prueba empíricamente entonces que el algoritmo sí responde en función del tamaño de la instancia. No obstante, se puede observar claramente que en cada uno de los rangos, la dispersión se los tiempos no es idéntica. Estas diferencias solo responden a la forma de cada una de las instancias, que quedaron determinadas por la probabilidad de recableado.

Un claro y muy básico ejemplo del impacto que tiene p sobre los tiempos, es imaginando un mejor caso para el algoritmo, donde al recablear la instancia de grafo signifique transformarlo en un árbol. En esa situación, es mucho menor el

procesamiento requerido para encontrar la solución, y por lo tanto los tiempos serán mejores que comparándolo con un grafo arbitrario.

Asimismo, se puede ver que cuánto menor sea el valor de la variable p, más similitud tendrá una instancia de grafo con otra, debido a que la distorsión será mínima. Esto sin dudas repercute en que los tiempos de resolución (y probablemente también los resultados arrojados) sean similares.

6.2.3. Escalamiento de pruebas en densidad de aristas

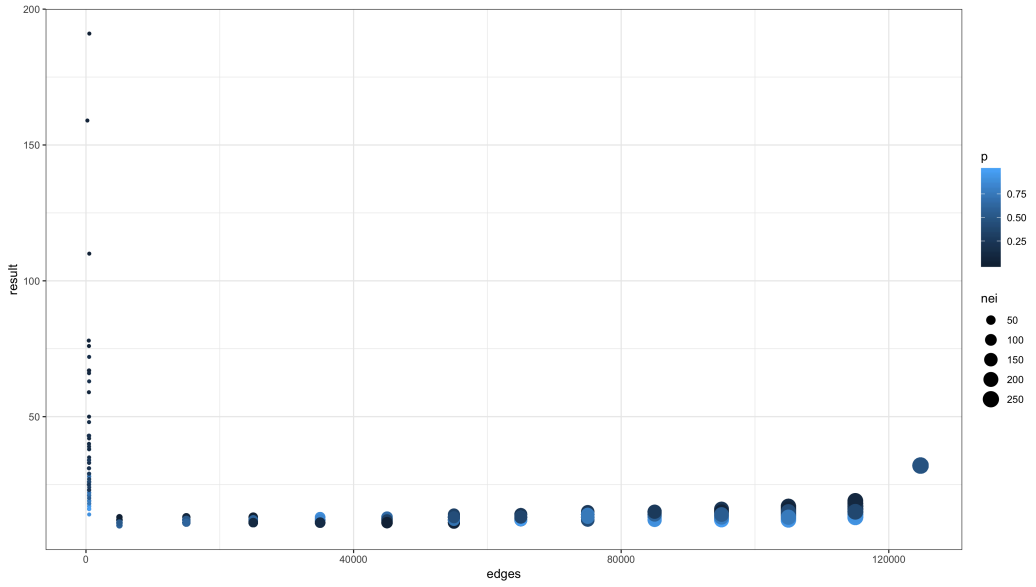
En esta subsección se estudia la relación entre la cantidad de aristas y el tiempo en hallar el resultado óptimo utilizando la heurística.

Para esto, se construye un conjunto de 1400 instancias de grafos pequeño mundo, fijando la variable n (*nodos*) en 500 y variando k (*neighborhood*) entre los valores 1, 10, 30, 50, 70, 90, 110, 130, 150, 170,190, 210, 230 y 250. Todas las instancias se crearon con una probabilidad de recableado p tomada de forma uniformemente independiente. Esto permite poder estudiar el potencial de la heurística en función de la densidad de conectividad, sin que los resultados sean impactados por el tamaño en las instancias en término de nodos. El detalle de las instancias utilizadas en esta subsección puede verse en el Apéndice A.3.

En cuanto al análisis de los resultados obtenidos, se recuerda que como en la subsección anterior, por no contar con un método exacto que sea capaz de solucionar el problema en instancias de grafos de esta magnitud, no es posible afirmar su nivel de correctitud.

Figura 6.19: Solución de la heurística en función de características de las instancias

Se muestra el resultado de la heurística en relación al tamaño de las instancias en cuanto a cantidad de aristas, agrupando por *neighborhood* y probabilidad de recableado. La probabilidad p se representa por el coloreado, y el *neighborhood* por el tamaño del punto en el gráfico.

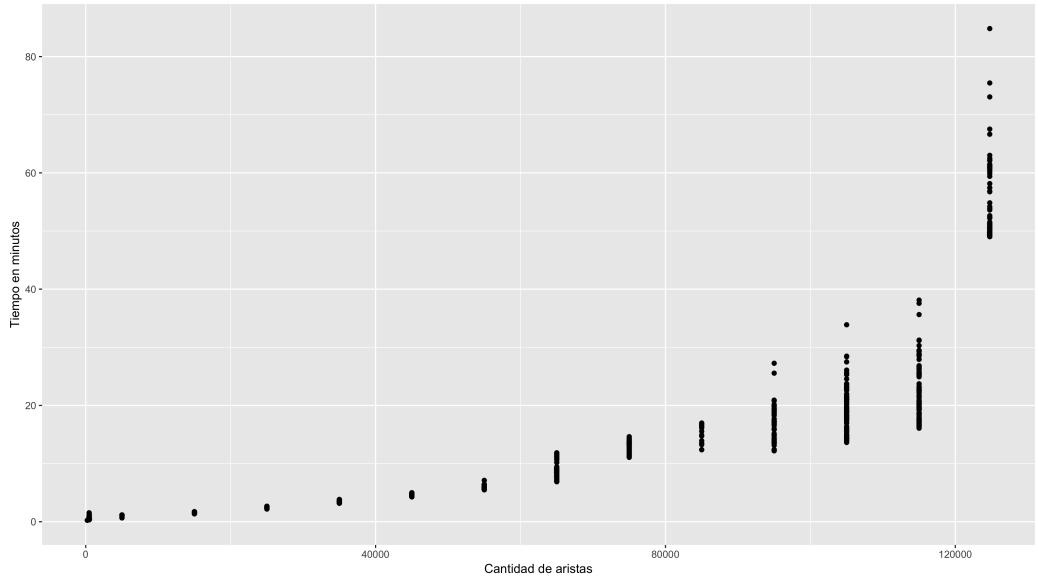


Igualmente, en la Figura 6.19 se puede observar cómo se distribuyen los resultados en función de la cantidad de aristas y la probabilidad de recableado.

Se desprende de allí en primer lugar, que para grafos con cantidad de nodos similares, una baja densidad de aristas afecta mucho a la solución de la heurística (en realidad, del problema en general). Esto tiene sentido, en consecuencia de que los canales disponibles para la difusión son muchos menos. Sin embargo, en la medida que la densidad de conexiones se incrementa, la solución de la heurística resulta ser bastante similar. También a este aspecto se le encuentra sentido, debido a que como al final, los reenvíos se realizan en un árbol (no todos los nodos reenvían por todos sus canales), cuando hay muchas aristas conectando, también hay muchas de ellas que no son utilizadas para la difusión.

Figura 6.20: Cantidad de aristas vs tiempo de resolución

Tiempo de ejecución de la heurística en minutos en función del tamaño de las instancias en términos de aristas.



Por otro lado, en la Figura 6.20 se puede visualizar el tiempo requerido para alcanzar el resultado mediante la heurística en función de la cantidad de aristas. Queda claro entonces la relación que existe entre esta variable y los tiempos, en el tanto que a mayor densidad de aristas, mayor es el tiempo requerido para la resolución. Se debe recordar que en esta muestra el tamaño del grafo en cuanto a cantidad de nodos es exactamente igual para cada instancia.

Se aprecia también que no solo la cota superior aumenta, también lo hace la cota inferior. Esto se interpreta en el sentido de que el algoritmo explora en una primera parte enlaces ociosos. En esta línea, las posibilidades para crear los árboles de recubrimiento mínimo son mayores, y por lo tanto se demora más en esa etapa.

Por otra parte, se intuye que la concentración de tiempos para las instancias de menor cantidad de aristas, responde a que son estructuras más simples y similares;

donde p no afecta tanto como en instancias de mayor magnitud. Habiendo más variación en la forma de los grafos, entonces los tiempos terminan distribuyéndose más.

Capítulo 7

Conclusiones y trabajo a futuro

Este proyecto estudia el problema de encontrar el tiempo mínimo de difusión en un grafo. Se trabaja en base a dos algoritmos para encontrar el resultado, uno a través de un método exacto y el otro a través de una heurística.

El método exacto, como podría anticiparse, es interesante para la resolución de instancias de menor tamaño, tanto en número de nodos como en densidad de aristas; arrojando para estos casos resultados en tiempos bastante pequeños (menos de dos horas).

Por su parte, la heurística, siendo su intención abarcar un universo más amplio de instancias, se prueba en tamaños más grandes y también arroja resultados en tiempos razonables.

En cuanto a la desviación de los resultados alcanzados por la heurística con respecto al óptimo se concluye que existe un porcentaje de éxito importante en los casos donde se conoce éste valor; es decir, instancias para las cuáles el método exacto arrojó resultados. Sin embargo, existe un error considerable para atacar en trabajos futuros; aunque no supera el 25 % en la mayoría de los casos.

Así mismo, se destaca que en comparación con otras heurísticas presentadas en la literatura, que además nacen enfocadas a topologías específicas, los resultados son sumamente competitivos.

Para mejorar la precisión de los resultados de la heurística, es necesario trabajar en algunas mejoras. Primero, en la decisión de cada nodo de transmitir a otro bloque o continuar transmitiendo dentro del que se encuentra. Por otro lado, el armado del árbol inicial de cada bloque realizado mediante el algoritmo $DynamicTree$ no genera el óptimo para el problema a resolver. En este caso, un árbol óptimo sería aquel que aprovechase toda la capacidad de comunicación de cada uno de los nodos (o sea, todos los enlaces) en la medida de lo posible. Mejorar este paso del algoritmo implicaría también mejoras directas en la solución encontrada.

Apéndice A

Instancias de Grafos Watts-Strogatz

A.1. Instancias para comparativa con método exacto

La muestra está constituida por grafos Watts-Strogatz con diferentes cantidades de nodos: 50, 100, 300, 600 y 1000. La variable *neighborhood*, que es la encargada de agregar aristas al anillo inicial, fue elegida de forma porcentual a la cantidad de nodos, buscando alcanzar distintas densidades de aristas. La probabilidad de recableado p, que es la que define la aleatoriedad del grafo, fue tomada de forma uniformemente independiente. En total, la muestra cuenta con 301 instancias de grafos pequeño mundo de 10 nodos, y 100 instancias de grafos del mismo tipo para cada una de las siguientes cantidades de nodos: 50, 100, 300, 600 y 1000.

La Figura A.1 muestra la distribución de instancias generadas en función de la probabilidad de recableado p y la cantidad de nodos. Se observa que se abarca de forma homogénea todo el universo de posibles valores para p, dejando claro que la elección de p se hizo de forma uniformemente independiente.

Para probar la desviación de la heurística alterando el tamaño; además de haber generado instancias con diferentes números de nodos, se utilizó para algunos casos la segunda variable llamada *neighborhood* (k en la definición), lo que afecta directamente el número de aristas de los grafos, aumentando la conectividad entre los nodos. En la Figura A.2, se observa la distribución de las instancias considerando por un lado probabilidad p y por otro la relación entre el número de nodos y la cantidad de aristas. La elección de mostrar las instancias en función de cómo se relacionan la cantidad de nodos y la cantidad de aristas de cada una, es a los efectos de visualizar que la heurística es probada para diferentes densidades de conectividad y que además, la probabilidad de recableado sigue siendo homogénea.

Figura A.1: Distribución de las instancias en términos de nodos

Cada instancia generada es mostrada en relación a la probabilidad de recableado y la cantidad de nodos que tiene.

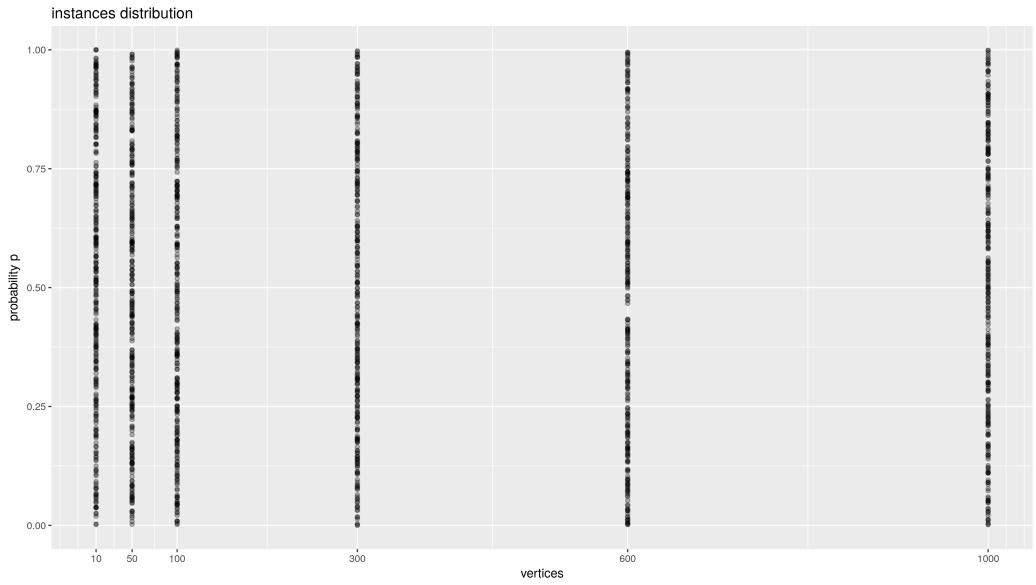
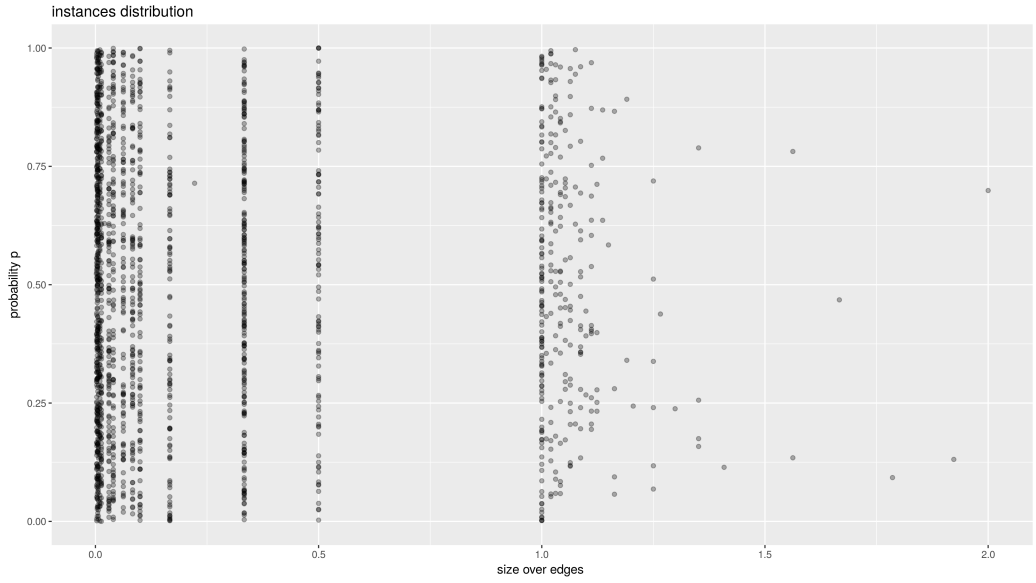


Figura A.2: Distribución de las instancias en términos de densidad de conexiones

Se eligen instancias que para las distintas cantidades de nodos, existan diferentes porcentajes de conectividad, y además la probabilidad p sea uniformemente independiente.



A.2. Instancias para pruebas en cantidad de nodos

La muestra elegida para la ejecución de esta subsección es de 200 instancias de grafos para cada una de las siguientes cantidades de nodos: 100, 300, 600, 1000, 2000, 5000 y 10000. Siendo el foco estudiar el comportamiento de la heurística según la cantidad de nodos, se decidió es que la variable *neighborhood* fuese relativamente fija, estableciéndose en los valores 1 y 3. Esto dio como resultado que para cada subconjunto de 200 grafos, 100 instancias tienen la característica de contar previo al recableado con dos enlaces, y las restantes 100 contar con seis.

Las Figuras A.3 y A.4 presentan la distribución de las variables de construcción de las instancias para esta prueba, mostrando p en función de la cantidad de nodos para el primer caso; y p en función de la variable *neighborhood* para el segundo. Asimismo, como la variable *neighborhood* termina determinando la cantidad de enlaces del grafo en función de la cantidad de nodos, en la Figura A.6 se muestra concretamente como quedan las instancias distribuidas según esta relación.

Figura A.3: Distribución de las instancias en términos de la variable nodos

Cada instancia generada es mostrada en relación a la probabilidad de recableado y la variable cantidad de nodos.

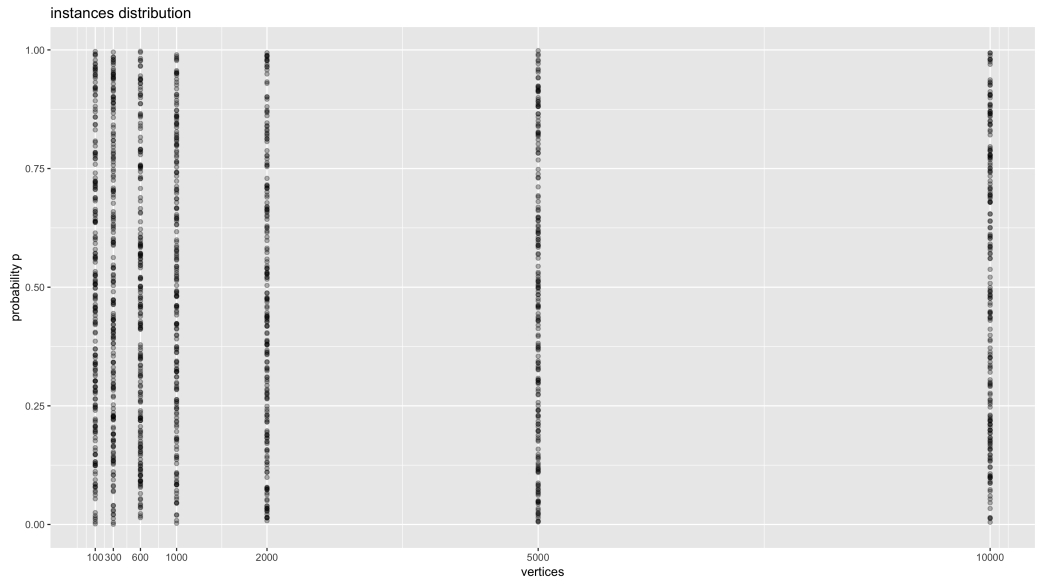


Figura A.4: Distribución de las instancias en términos de la conectividad

Cada instancia generada es mostrada en relación a la variable *neighborhood*.

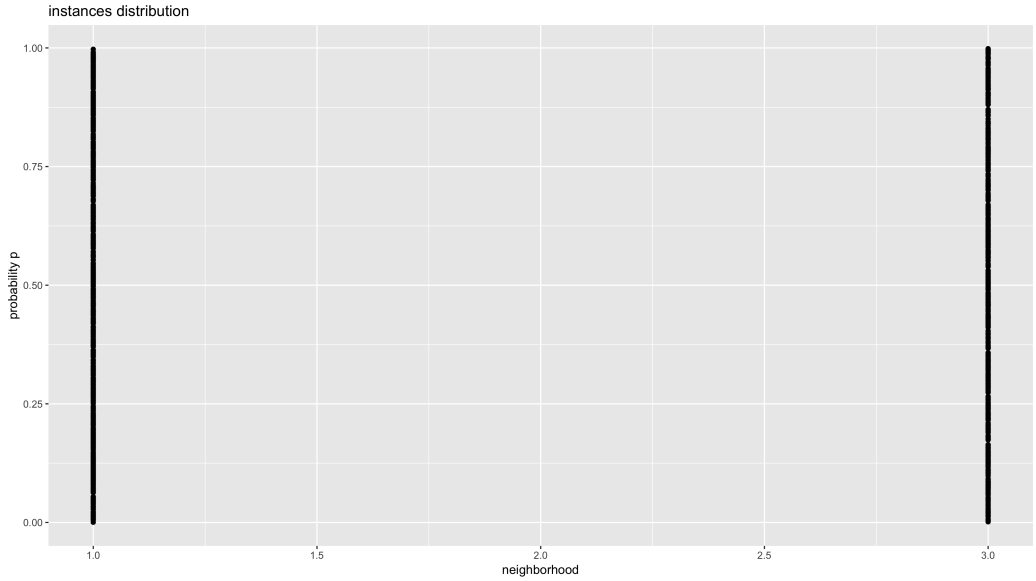
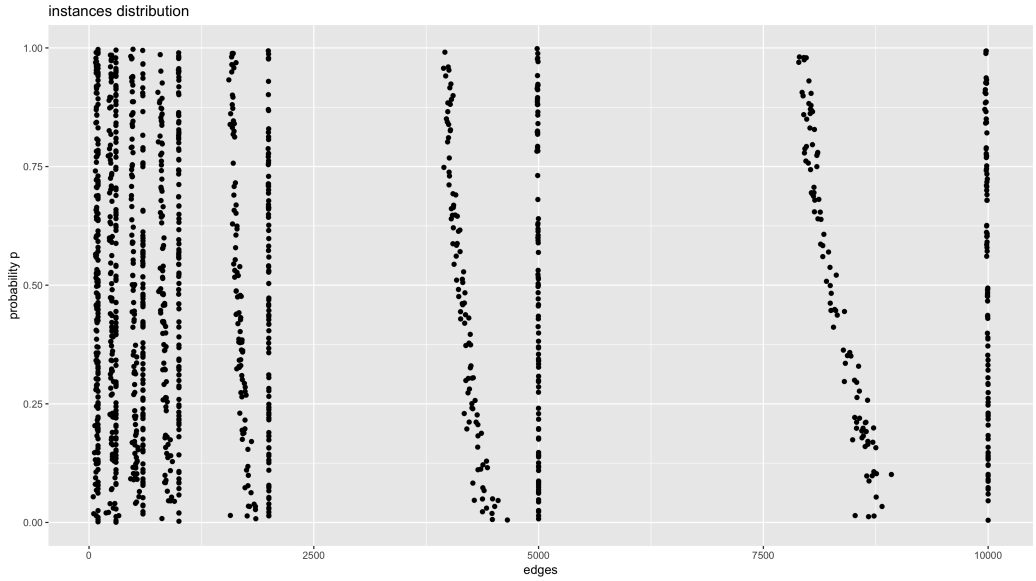


Figura A.5: Distribución real de las instancias en términos de nodos efectivos

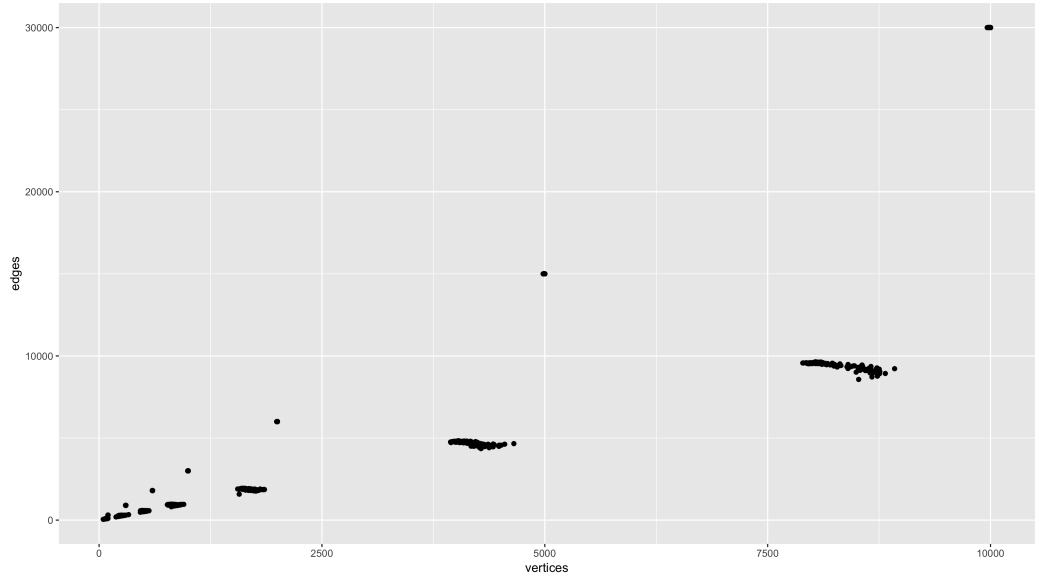
Cada instancia generada es mostrada en relación a la probabilidad de recableado y la cantidad de nodos que tiene.



Una aclaración pertinente respecto a la construcción de las instancias, refiere a la condición de necesitar que cada una sea efectivamente un grafo conexo. En particular, para los casos generados con *neighborhood* igual a 3, esta condición fue dada naturalmente. Sin embargo, para los casos con *neighborhood* igual a 1,

sucedió que luego del recableado, varias de las instancias perdieron esta característica. En función de subsanarlo; se procedió a tomar para la muestra, la mayor componente conexa de cada uno de los grafos generados. Esto representa en promedio un cubrimiento de aproximadamente el 80 % de la cantidad de nodos. La Figura A.5 muestra entonces la distribución de las instancias generadas, no en función de la cantidad de nodos que se eligió como variable como lo hace la Figura A.3, sino en función de la cantidad de nodos efectiva de cada una. Las consecuencias de esta maniobra quedan a la vista en la subsección 6.2.2 cuando se analizan los resultados.

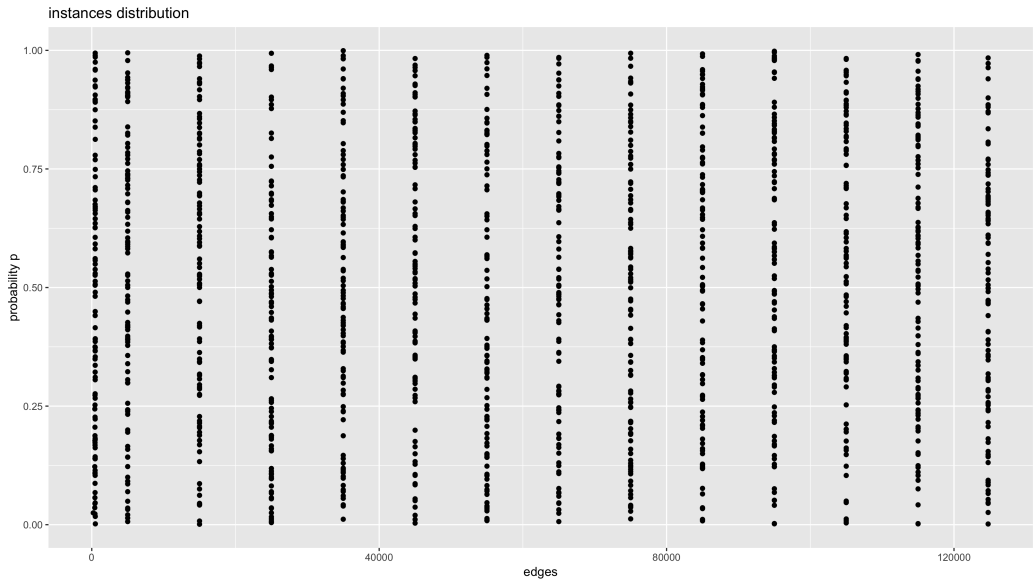
Figura A.6: Relación entre nodos y enlaces



A.3. Instancias para pruebas en densidad de aristas

La muestra para esta subsección fue construida fijando la cantidad de nodos y variando la variable *neighborhood*; todas las instancias con una probabilidad de recableado p uniformemente independiente. Se cuenta con 1400 instancias, cada una de ellas con 500 nodos. Para constituir la muestra, se construyeron 100 instancias para cada una de los siguientes *neighborhoods*: 1, 10, 30, 50, 70, 90, 110, 130, 150, 170,190, 210, 230 y 250.

Figura A.7: Distribución de las instancias en términos de aristas



La Figura A.7 presenta la distribución de las variables de construcción de las instancias para esta prueba, mostrando p en función de la cantidad de aristas, consecuencia directa de la variable *neighborhood* y los nodos.

Apéndice B

Publicación para ALIO/EURO 2018

Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in DISCRETE MATHEMATICS

Electronic Notes in Discrete Mathematics 69 (2018) 165–172 www.elsevier.com/locate/endm

Heuristics for the Minimum Broadcast Time

Amaro de Sousa [1]

Instituto de Telecomunicações
Campus Universitário de Santiago
Aveiro, Portugal

Gabriela Gallo, Santiago Gutierrez [2]
Franco Robledo, Pablo Rodríguez-Bocca, Pablo Romero

Instituto de Matemática y Estadística
Facultad de Ingeniería, Universidad de la República
Montevideo, Uruguay.

Abstract

The problem under study is the Minimum Broadcast Time (MBT). We are given a simple graph and a singleton that owns a message. The goal is to disseminate the message as soon as possible, where the communication takes place between neighboring-nodes in a selective fashion and each forwarding takes one time-slot. The MBT serves as an inspirational problem for the design of delay-sensitive forwarding schemes. Since the problem belongs to the $\mathcal{NP}$-Hard class, the literature offers heuristics, approximation algorithms and exact exponential-time solutions.

The contributions of this paper are two-fold. First, an ILP formulation for the problem is provided. Second, a competitive heuristic is developed. A fair comparison between TreeBlock and previous heuristics highlights the effectiveness of our proposal.

Keywords: Minimum Broadcast Time, Computational Complexity, Heuristics.

https://doi.org/10.1016/j.endm.2018.07.022

1 Motivation

The identification of an ideal forwarding scheme is a challenging problem in telecommunications, not well understood. The scientific literature offers fluid models for massive communication systems [1], tree-like or one-to-one forwarding schemes [13], the tit-for-tat communication paradigm for incentivess [11], among others. The *Minimum Broadcast Time*, or MBT, finds original applications in telephonic services [5]. However, it serves as an inspirational problem for the design of current delay-tolerant forwarding schemes in modern communication systems like Content Delivery Networks (CDN) and Peer-to-Peer (P2P) [3] networks, or Cognitive Radio (CR) [12] networks.

A natural description of the problem is in terms of phone-calls (we can identify forwarding or gossiping, depending on the context). Suppose that a member originates a message which is to be communicated to all other members of the network. This is to be accomplished as quickly as possible by a series of calls placed over lines of the network. In the MBT, there are three strict broadcasting rules:

i Each phone-call requires one time-slot,

ii a member can participate in only one call per slot, and

iii a member can only call a neighbor member.

The problem under study is the following: given a connected graph G and a originator vertex v_0, what is the minimum number of time-slots required to complete broadcasting starting from v_0 and following the previous broadcasting rules?

There is a rich scientific literature available for this problem. Garey and Johnson include the MBT in the list of $\mathcal{NP}$-Complete problems [6]. The MBT of a complete graph $G = K_n$ is clearly $b(K_n) = \lceil log_2(n) \rceil$, yet K_n is not minimal with this property. The minimum number of links to achieve $b(K_n)$ is a relevant reverse-engineering problem with valuable progress. For instance, he hypercube Q_m has $m2^{m-1}$ links and optimum broadast time. The optimal forwarding in trees is recursively achieved in a greedy-like manner [15]. Grids accept an intuitive optimal forwarding scheme [10]. Several other particular graph families are studied, such as butterflies and Harary graphs [2]. A recent heuristic for the MBT is offered in [9]. The reader is invited to consult a literature-review in [14] and [8].

[1] Email: asou@ua.pt

[2] Email: (gabriela.gallo,santiago.gutierrez,frobledo,prbocca,promero)@fing.edu.uy

This article is organized in the following manner. An Integer Linear Programming formulation is introduced in Section 2. *TreeBlock* heuristic is fully described in Section 3. A fair comparison with the most competitive heuristics is carried out in Section 4. Section 5 presents concluding remarks and trends for future work.

2 ILP for the MBT

For convention, we denote $V = \{0, 1, \ldots, n-1\}$ to the node-set, and we fix $v_0 = 0$ as the source-node. Denote $V(i)$ to the set of neighboring nodes of i. Let T be an upper-bound for the broadcast time ($T = n$ is a trivial one). Consider the set of binary variables x_{ij}^t that is set to 1 iff the message is transmitted from node i to node j during time-slot t. The ILP for the MBT is expressed as follows:

$$\min z \tag{1}$$

$$s.t$$

$$\sum_{j \in V(i)} x_{ij}^1 = \delta_0,\ \forall i \in \{0, \ldots, n-1\} \tag{2}$$

$$\sum_{j \in V(i)} \sum_{t=1}^{T} x_{ji}^t = 1,\ \forall i \in \{1, \ldots, n-1\} \tag{3}$$

$$\sum_{j \in V(i)} x_{ij}^\tau \leq 1,\ \forall i \in \{0, \ldots, n-1\}, t \in \{2, \ldots, T\} \tag{4}$$

$$x_{ij}^t \leq \sum_{\tau=1}^{t-1} \sum_{k \in V(i) \setminus \{j\}} x_{ki}^\tau,\ \forall (i,j) \in E, t \in \{2, \ldots, T\} \tag{5}$$

$$\sum_{t=1}^{T} t \cdot x_{ij}^t \leq z\, \forall (i,j) \in E \tag{6}$$

$$z \in \mathbb{N};\ x_{ij}^t \in \{0,1\}\, \forall (i,j) \in E, t \in \{1, \ldots, T\} \tag{7}$$

Constraints 2 state that onle the source-node 0 sends the message when $t = 1$.
Constraints 3 state that the message is received by each node exactly once.
Constraints 4 state that a node transmits the message to a single neighbor.
Constraints 5 state that a transmitter must receive the message first.
Constraints 6 state that the broadcast time dominates any transmission.
Finally, Constraints 7 set the variable domain. The computational order rests on a smart upper-bound T, which will be found using a dedicated heuristic.

3 Heuristic for the MBT

Recall that a node $v \in V(G)$ is a *cut-point* if $G - v$ has more components than G. A block is a maximal subgraph with no cut-points. Every connected graph accepts a decomposition into tree-blocks, where the blocks are linked in a tree-like structure [16].

The two main building blocks of our algorithm are Functions $BroadcastTree$ and $BroadcastBlock$. Essentially, $BroadcastTree$ is the optimal *interblock* forwarding in the tree-block structure, while $BroadcastBlock$ serves as an *intrablock* forwarding. A full forwarding strategy is obtained by a cooperation of both functions.

Algorithm 1 $F = TreeBlock(G, v_0)$

$U \leftarrow CutPoints(G)$
$U_0 \leftarrow \{v_0\} \cup U$
$d \leftarrow Distances(U_0)$
$T \leftarrow MST(U_0, d)$
$B \leftarrow Blocks(G)$
$L \leftarrow LeafBlocks(B)$
for all $B_i \in L$ **do**
  $z_i \leftarrow B_i \cap CutPoints(G)$
  $e_i \leftarrow max_{v \in B_i}\{d(w_i, v)\}$
  $w(z_i, B_i) \leftarrow e_i$
end for
$T \leftarrow T \cup \{(z_i, B_i)\}_{i=1,\ldots,|L|}$
$F \leftarrow BroadcastTree(T, v_0)$
for all $B_i \in B$ **do**
  $F_i \leftarrow BroadcastBlock(B_i, F(B_i))$
  $F \leftarrow F \cup F_i$
end for
return F

A step-by-step description of the algorithm $TreeBlock$ follows. All the cut-points U of G are determined in Line 1. The source-node is added to this special set in U_0 (Line 2). A complete weighted graph is built in Lines 3, where the corresponding weights are the distance between every pair of nodes in U_0. A minimum spanning tree T is found using Kruskal algorithm in Line 4. This tree serves as the *skeleton* of the interblock forwarding scheme. The weights in the links represent a *delay* between different cut-points. Observe that there is an additional delay with leaf-blocks, that should be added to reach all nodes

in the graph. Therefore, in Lines 5 and 6, the blocks and leaf-blocks from G are found. Recall that a leaf-node has a single cut-point. In the block of Lines 7-11, the eccentricity e_i from the cut-point $z_i \in B_i$ is found for any leaf-block $B_i \in L$. This additional latency is considered by a tree-augmentation, in Line 12. The result is that the new nodes z_i are leaf-nodes in T, and the corresponding links have weights e_i. An inter-block forwarding scheme F is performed using Function $BroadcastTree$ with source-node v_0 in the tree T. Function $BroadcastTree$ returns the optimum forwarding scheme in this weighted tree. Finally, the intra-block forwarding is performed using Function $BroadcastBlock$ (see the last for-loop; Lines 14-17).

Algorithm 2 $F_i = BroadcastBlock(B_i, v'_1, v'_2, \ldots, v'_k)$

1: $T \leftarrow MST(v'_1, B_i)$
2: **for** $i = 1$ **to** k **do**
3: $\quad w_i \leftarrow \{|B_i|/i\}$
4: $\quad w(u_i, v'_i) \leftarrow w_i$
5: $\quad T \leftarrow T \cup w(u_i, v'_i)$
6: **end for**
7: $F_i \leftarrow BroadcastTree(T, v'_1)$
8: **return** F_i

Let us finally consider $BroadcastBlock$. It receives a block B_i and the expected forwarding order $F(B_i)$ of all its k cut-points, to know, $v'_1, v'_2, \ldots, v'_k$. Therefore, node v'_1 is the root-node. In Line 1, Kruskal algorithm is applied into B_i to find the minimum spanning tree, rooted at v'_1. In order to force the correct forwarding order in the cut-points, the weights in their incident links are modified as follows. The incident link to v'_i in T will be assigned a weight $w_i = |B_i|/i$ (Lines 3-4). The resulting tree is updated in Line 5. Finally, Function $BroadcastTree$ is applied into this tree in order to define an intra-block forwarding scheme (Line 7), and the result is returned in Line 8.

4 Proof of Concept

An extensive experimental analysis was carried-out in order to understand the effectivess of $TreeBlock$ heuristic and the capacity of our ILP formulation. Here we highlight a comparison with the latest heuristic we found from the literature [9]. The reader is invited to consult our technical report, with a performance analysis of the heuristic under Harary, Chord Harary, Butterflies, de Bruijn and Small World networks [4].

The broadcast in different grid graphs is successfully found using $TreeBlock$. In a celebrated work from Frank Harary, a family of graphs $H_{n,k}$ with maximum connectivity k is provided for any fixed number of links [7]. The authors from [9] confirm optimality in half of Harary instances under test. We found the optumum broadcast in 10 out of 16 instances, and unit-gap in the others. There are three instances of dense graphs where the exact ILP formulation cannot find the globally optimum broadcast (by a CPU time constraint of 2 hours).

Table 1
Harary. Broadcast scheme [2]

N	k	n	ILP	$TreeBlock$
17	2	17	9	9
17	3	17	5	5
17	5	17	5	5
17	6	17	5	5
17	7	17	5	5
30	2	30	15	15
30	3	30	9	9
30	8	30	5	6
30	9	30	5	6
30	10	30	5	6
50	2	50	25	25
50	3	50	14	14
50	11	50	-	7
50	20	50	-	8
50	21	50	-	7
100	2	100	50	50

5 Conclusions and Trends for Future Work

In this paper, the Minimum Broadcast Time is studied. An efficient ILP formulation and a heuristic is introduced. *TreeBlock* combines the optimal forwarding scheme in trees and an efficient block decomposition of a connected graph. The exact ILP works for small-sized graphs, while *TreeBlock* finds small gaps with respect to the globally optimum solution whenever it is available. Further analysis confirms its wide applicability for large instances as well, with a reasonable CPU time for graphs with thousands of nodes. The heuristic is competitive with respect to previous sub-optimal algorithms from the literature.

As future work, we want to find the broadcast in real-life topologies and have a better understanding of its applications in delay-tolerant systems.

Acknowledgment

This work is partially supported by Project CSIC I+D 395 entitled *Sistemas Binarios Estocásticos Dinámicos*.

References

[1] Anulova, S., "Fluid Limit for Switching Closed Queueing Network with Two Multi-servers," Springer International Publishing, Cham, 2017 pp. 343–354.

[2] Bhabak, P., H. Harutyunyan and P. Kropf, *Efficient broadcasting algorithm in harary-like networks*, in: *2017 46th International Conference on Parallel Processing Workshops (ICPPW)*, 2017, pp. 162–170.

[3] Chuang, Y.-T., *Protecting against malicious and selective forwarding attacks for p2p search & retrieval system*, Peer-to-Peer Networking and Applications **10** (2017), pp. 1079–1100.

[4] de Sousa, A., G. Gallo, S. Gutierrez, F. Robledo, P. Rodríguez-Bocca and P. Romero, *Exact Algorithm and Heuristics for the Minimum Broadcast Time*, Technical report, Universidad de la República (2018).
URL https://www.fing.edu.uy/~frobledo/Paper_MBT_2018.pdf

[5] Farley, A. M., *Broadcast time in communication networks*, SIAM Journal on Applied Mathematics **39** (1980), pp. 385–390.

[6] Garey, M. R. and D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness," W. H. Freeman & Company, New York, NY, USA, 1979.

[7] Harary, F., *The maximum connectivity of a graph*, Proceedings of the National Academy of Sciences of the United States of America **48** (1962), pp. 1142–1146.

[8] Harutyunyan, H., A. Liestman, J. Peters and D. Richards, *Broadcasting and gossiping*, in: J. L. Gross, J. Yellen and P. Zhang, editors, *Handbook of Graph Theory, Second Edition*, Chapman & Hall/CRC, 2013, 2nd edition p. 18.

[9] Harutyunyan, H. A. and S. Kamali, *Efficient broadcast trees for weighted vertices*, Discrete Applied Mathematics **216** (2017), pp. 598 – 608, levon Khachatrian's Legacy in Extremal Combinatorics.

[10] Hedetniemi, S. M., S. T. Hedetniemi and A. L. Liestman, *A survey of gossiping and broadcasting in communication networks*, Networks **18** (1988), pp. 319–349.

[11] Liu, W., D. Peng, C. Lin, Z. Chen and J. Song, *Enhancing tit-for-tat for incentive in bittorrent networks*, Peer-to-Peer Networking and Applications **3** (2010), pp. 27–35.

[12] Liyana Arachchige, C. J., S. Venkatesan, R. Chandrasekaran and N. Mittal, "Minimal Time Broadcasting in Cognitive Radio Networks," Springer Berlin Heidelberg, Berlin, Heidelberg, 2011 pp. 364–375.

[13] Pazzi, R. W., A. Boukerche, R. E. Grande and L. Mokdad, *A clustered trail-based data dissemination protocol for improving the lifetime of duty cycle enabled wireless sensor networks*, Wirel. Netw. **23** (2017), pp. 177–192.

[14] Peleg, D., "Time-Efficient Broadcasting in Radio Networks: A Review," Springer Berlin Heidelberg, Berlin, Heidelberg, 2007 pp. 1–18.

[15] Slater, P. J., E. J. Cockayne and S. T. Hedetniemi, *Information dissemination in trees*, SIAM Journal on Computing **10** (1981), pp. 692–701.

[16] West, D., "Introduction to Graph Theory," Featured Titles for Graph Theory Series, Prentice Hall, 2001.

Apéndice C

Reporte técnico Minimum Broadcast Time

Heuristics for the Minimum Broadcast Time

Amaro de Sousa [1]

Instituto de Telecomunicações
Campus Universitário de Santiago
Aveiro, Portugal

Franco Robledo, Pablo Rodríguez-Bocca, Pablo Romero
Gabriela Gallo, Santiago Gutierrez [2]

Instituto de Matemática y Estadística
Facultad de Ingeniería, Universidad de la República
Montevideo, Uruguay.

Abstract

The problem under study is the Minimum Broadcast Time (MBT). We are given an undirected connected graph and a singleton that owns a message. The goal is to broadcast this message as soon as possible, where the communication takes place between neighboring-nodes in a selective fashion and each forwarding takes one time-slot. Historically, the MBT finds applications in telephonic services; however, it serves as an inspirational problem for the design of current delay-tolerant forwarding schemes in modern communication systems like content delivery networks and peer-to-peer networks. The problem belongs to the class of $\mathcal{NP}$-Complete class. As a consequence, the literature offers heuristics, approximation algorithms and exact exponential-time solutions.

The contributions of this paper are two-fold. First, an efficient Integer Linear Programming (ILP) formulation for the problem is provided. Second, a competitive heuristic called *TreeBlock*, is developed. TreeBlock exploits the optimality for the MBT in tree-graphs, and the fact that arbitrary connected graphs accept

a decomposition into a tree-block structure, where the building blocks are maximally biconnected subgraphs. A fair comparison between TreeBlock and previous heuristics highlights the effectiveness of our proposal.

Keywords: Minimum Broadcast Time, Computational Complexity, Integer Linear Programming, Heuristics.

1 Introduction

The identification of an ideal forwarding scheme is a challenging problem in telecommunications. The scientific literature offers fluid models for massive communication systems [1], tree-like or one-to-one forwarding schemes [23], the tit-for-tat communication paradigm for incentivess [21], among many others.

A fundamental fluid model for a complete peer-to-peer network was posed for the first time by Qiu and Srikant [31]. In this fluid model, the authors consider a complete network with identical users, where simultanous forwarding takes proportionally more time. The authors claim a counterintuitive result, where *the one-to-one is a good forwarding strategy*. Even though the authors study the service capacity of a file sharing peer-to-peer system, its formulation is general enough. They find a closed formula for the average waiting time following a one-to-one forwarding scheme when the population is a power of two. In [25], a formal proof that the one-to-one forwarding scheme achieves the minimum waiting time is included, when the population is a power of two. Furthermore, recent works confirm that the optimality holds for arbitrary (non-complete) graphs as well. Incidentally, the one-to-one forwarding scheme is open, in the sense that the neighboring selection strategy must be determined. This means that the optimal forwarding scheme is equivalent to a well-known historical problem, known as the *Minimum Broadcast Time*, or MBT for short [9]. It finds original applications in telephonic services. However, it serves as an inspirational problem for the design of current delay-tolerant forwarding schemes in modern communication systems like Content Delivery Networks (CDN) and Peer-to-Peer (P2P) [5] networks, or Cognitive Radio (CR) [22] networks.

A natural description of the problem is in terms of phone-calls (we can

[1] Email: asou@ua.pt
[2] Email: (gabriela.gallo,santiago.gutierrez,frobledo,prbocca,promero)@fing.edu.uy

identify forwarding or gossiping, depending on the context). Suppose that a member originates a message which is to be communicated to all other members of the network. This is to be accomplished as quickly as possible by a series of calls placed over lines of the network. In the MBT, there are three strict broadcasting rules:

i Each phone-call requires one time-slot,

ii a member can participate in only one call per slot, and

iii a member can only call a neighbor member.

The problem under study is the following: given a connected graph G and a originator vertex v_0, what is the minimum number of time-slots required to complete broadcasting starting from v_0 and following the previous broadcasting rules? Here the graph represents an communication network, where vertices V are the members, and the edges E are the possible communication links between them. Defining an originator vertex v_0, the *minimum broadcast time (MBT)*, denoted $b(v_0, G)$ is the minimal number of time-slots necessary to disseminate a message from v_0 to all the other vertices of G. Most works study the MBT of a graph, $b(G)$, as the worst time among all potential originator members in the graph G, i.e. $b(G) = max\{b(v_0, G)|v_0 \in V\}$. There must be at least one *forwarding strategy* that achieves $b(v_0, G)$, and it could be specified as a sequence of parallel calls that follows the broadcasting rules and informs all the nodes in the network G in $b(v_0, G)$ time-slots.

This article is organized as follows. The main body of related work is covered in Section 2. An Integer Linear Programming formulation is introduced in Section 2.1. TreeBlock heuristic is fully described in Section 3. A fair comparison with the most competitive heuristics is carried out in Section 4. Section 5 presents concluding remarks and trends for future work.

2 Minimum Broadcast Time and Related Work

Garey and Johnson include the MBT in the list of $\mathcal{NP}$-Complete problems [11]. They consider an arbitrary starting set V_0 with possible more than a singleton, subject to the three previous broadcasting rules. Observe that a completion time in $T = 1$ time-slot exists if and only if V_0 accepts a perfect matching in bipartite graphs, which accepts a polynomial-time algorithm [7]. However, the case $T = 2$ accepts a reduction from 3-dimensional matching [27].

The MBT of a complete graph $G = K_n$ is clearly $b(K_n) = \lceil log_2(n) \rceil$, yet K_n is not minimal with this property. The minimum number of links to

achieve $b(K_n)$ is an inverse design problem with valuable progress. Indeed, when $n = 2^m$ for some natural m, the hypercube Q_m has $m2^{m-1}$ links and the same broadast time is $b(Q_m) = m = log_2(n)$. Hence, the hypercube is the most economical broadcasting structure when the population is a power of two.

The optimum forwarding scheme is available only for specific families of graphs. The optimality in trees is recursively achieved in a greedy-like manner [27]. The idea is a bottom-to-top process, choosing members with maximum delay. 2D Grid graphs accept an optimal forwarding scheme as well [17]. If we are given an $R \times C$ grid, the broadcast time is precisely the diameter $D = R - 1 + C - 1 = R + C - 2$. The forwarding is intuitive, as the reader can appreciate starting from a corner and ending in the opposite one. Several other particular graph families are studied, to mention: cube-connected cycles [20], Butterfly and DeBruijn networks [18], and Harary networks [3].

In the general case, for an arbitrary graph G and an arbitrary originator vertex, Elkin et. al. [8] present an algorithm to obtain a forwarding strategy that found the best theoretical upper bound knowns [26], with $O(\frac{\log(|V|)}{\log\log(|V|)}b(G))$ time-slots. Several extensions to the original problem was studied. We will not cover them here, please read [24] and [13] for a review in MBT and related problems.

The literature offers several metaheuristics and approximation algorithms for the MBT, see for instance: [8], [19], [10], [2], [15], [14] and [6].

2.1 Exact Formulation for the MBT

In this section, an exact Integer Linear Programming formulation for the MBT is introduced. Let $G = (V, E)$ be a connected graph, where $V = \{0, 1, \ldots, n-1\}$ is the node-set. Denote $V(i)$ to the set of neighboring nodes of i. For simplicity, we choose $v_0 = 0$ as the source-node.

Recall that in every time-slot, a node i that owns the message can forward it to a single member of its neighbor-set $V(i)$. Furthermore, this node did not receive the message yet. The broadcast time is the number of time-slots required for the reachability of the message to the whole set V. The goal is to minimize the broadcast time.

In order to find an efficient ILP formulation, we consider an upper-bound T for the minimum completion time. Consider the set of binary variables x_{ij}^t

such that, when equal to one, mean that the message is transmitted from node i to node j during time-slot t. In addition, we use an integer variable z whose value represents the broadcast time (measured in time-slots).

The Integer Linear Programming (ILP) model for the MBT is expressed as follows:

$$\begin{aligned} &\text{Minimize } z \\ &\text{Subject to:} \end{aligned} \tag{1}$$

$$\sum_{j \in V(i)} x_{ij}^1 = \begin{cases} 1\,, i = 0 \\ 0\,, i \neq 0 \end{cases} \quad , i = 0, ..., n-1 \tag{2}$$

$$\sum_{j \in V(i)} \sum_{t=1}^{T} x_{ji}^t = 1 \quad , i = 1, ..., n-1 \tag{3}$$

$$\sum_{j \in V(i)} x_{ij}^t \leq 1 \quad \begin{matrix} , i = 0, ..., n-1 \\ , t = 2, ..., T \end{matrix} \tag{4}$$

$$\sum_{\tau=1}^{t-1} \sum_{k \in V(i) \setminus \{j\}} x_{ki}^{\tau} \geq x_{ij}^t \quad \begin{matrix} , (i,j) \in A : i \neq 0 \\ , t = 2, ..., T \end{matrix} \tag{5}$$

$$\sum_{t=1}^{T} t \cdot x_{ij}^t \leq z \quad , (i,j) \in A \tag{6}$$

$$x_{ij}^t \in \{0, 1\} \quad \begin{matrix} , (i,j) \in A \\ , t = 1, ..., T \end{matrix} \tag{7}$$

$$z \in \mathbb{N} \tag{8}$$

The objective function 1 is the broadcast time. Constraints 2 state that the message is transmitted only once by the source-node 0 during the first time-slot $t = 1$. Constraints 3 guarantee that the message is received by each node exactly once from some neighboring-node. Constraints 4 guarantee that on each time slot $t = 2, ..., T$ (the case of $t = 1$ is already constrained in 2) a node can only transmit the message to at most one neighboring node. Constraints 5 guarantee that the message is transmitted from node i (except when it is the source node 0) to node j on time slot t only if the message

has been received by node i on a previous time slot from another neighboring node of i (*i.e.*, not from node j). Constraints 6 guarantee that the completion time is not lower than the time slot of any message transmission. Finally, constraints 7–8 are the variable domain constraints.

The runtime required to solve the optimization problem by using a standard solver is significantly shortened with the introduction of the following improvements. First, consider the minimum number of hops of any path in G from source node 0 to node i given by h_i (this can be easily obtained by a standard shortest path algorithm). For a given number of hops h, consider set N_h composed by all nodes such that $h_i = h$.

Then, a node $i \in N_h$ cannot transmit the message on any of the time slots $t = 1, ..., h_i$. So, for each node $i \in N_h$, we can set to 0 all variables x_{ij}^t, for $t = 1, ..., h$ (resulting, in practice, in a reduction of the number of variables). Moreover, a node $i \in N_h$ can transmit at most one message on time slot $t = h + 1$, and, generalizing, at most π messages on time slots $t = h + 1 + \pi$. So, we can define the following valid inequalities:

$$\sum_{i \in N_h} \sum_{j \in V(i)} x_{ij}^{h+1+\pi} \leq 1 + \pi \qquad \begin{array}{l} , h = 1, ..., T-1 \\ , \pi = 0, ..., T-h+1 \end{array} \tag{9}$$

Our computational tests have showed that constraints 9 do not improve significantly the model when $\pi > 0$ and, for this reason, the computational results were obtained by adding inequalities 9 only for $\pi = 0$.

Finally, we can reduce even further the number of variables in the following way. Consider d as the degree of source node 0. Without any lack of generality, we can say that if the d neighboring nodes are used by source node 0 to transmit the message, than, these transmissions will happen in the first d time slots. So, we can set to 0 all variables x_{0j}^t, for $(0, j) \in A$ and $t = d + 1, \ldots, T$.

3 Heuristic for the MBT

In this section we introduce a novel heuristic for the MBT. The key idea is a decomposition of the graph into two-node connected blocks, and an efficient routing in the tree that is defined by these blocks, inspired by the optimum routing in trees. Recall that the MBT accepts an optimal forwarding scheme in trees. This optimal routing is called MakespanTree here.

A node $v \in V(G)$ is called a *cut-point* if $G - v$ has more components than

G. A block is a maximal subgraph of G that has no cut-points. Clearly, if G is connected and has no cut-points, then G is a block. Every connected graph G accepts a tree-block descomposition, where each block B_i is an induced subgraph with no cut-points, and every cut-point $v \in V(G)$ is incident to the blocks that include it. Formally, given an arbitrary connected graph $G = (V, E)$, the tree-block graph is $B(G) = (B \cup U, E')$, where $U = \{v_1, \ldots, v_r\}$ is the set of cut-points in G, $B = \{B_1, \ldots, B_s\}$ the set of blocks of G, and $E' \subseteq B \times U$ consists of links between v_i and B_j whenever the cut-point belongs to B_j. By its definition, tree-block is a tree, where some nodes are blocks. Observe that if G has no cut-points, then $B(G)$ has a single isolated node, and it is a tree as well. Additionally, in a non-trivial tree, there exists some *leaf-blocks*, that are incident with a single cut-point. The reader is invited to consult [30] for further details.

The two main building blocks of our algorithm are Functions $BroadcastTree$ and $BroadcastBlock$. Essentially, $BroadcastTree$ serves as an *interblock* forwarding in the tree-block structure, while $BroadcastBlock$ serves as an *intra-block* forwarding. A full forwarding strategy is obtained by a cooperation of both functions.

Algorithm 1 $F = TreeBlock(G, v_0)$

$U \leftarrow CutPoints(G)$
$U_0 \leftarrow \{v_0\} \cup U$
$d \leftarrow Distances(U_0)$
$T \leftarrow MST(U_0, d)$
$B \leftarrow Blocks(G)$
$L \leftarrow LeafBlocks(B)$
for all $B_i \in L$ **do**
  $z_i \leftarrow B_i \cap CutPoints(G)$
  $e_i \leftarrow max_{v \in B_i}\{d(w_i, v)\}$
  $w(z_i, B_i) \leftarrow e_i$
end for
$T \leftarrow T \cup \{(z_i, B_i)\}_{i=1,\ldots,|L|}$
$F \leftarrow BroadcastTree(T, v_0)$
for all $B_i \in B$ **do**
  $F_i \leftarrow BroadcastBlock(B_i, F(B_i))$
  $F \leftarrow F \cup F_i$
end for
return F

A step-by-step description of the algorithm $TreeBlock$ follows. All the

cut-points U of G are determined in Line 1. The source-node is added to this special set in U_0 (Line 2). A complete weighted graph is built in Lines 3, where the corresponding weights are the distance between every pair of nodes in U_0. A minimum spanning tree T is found using Kruskal algorithm in Line 4. This tree serves as the *skeleton* of the interblock forwarding scheme. The weights in the links represent a *delay* between different cut-points. Observe that there is an additional delay with leaf-blocks, that should be added to reach all nodes in the graph. Therefore, in Lines 5 and 6, the blocks and leaf-blocks from G are found. Recall that a leaf-node has a single cut-point. In the block of Lines 7-11, the eccentricity e_i from the cut-point $z_i \in B_i$ is found for any leaf-block $B_i \in L$. This additional latency is considered by a tree-augmentation, in Line 12. The result is that the new nodes z_i are leaf-nodes in T, and the corresponding links have weights e_i. An inter-block forwarding scheme F is performed using Function $BroadcastTree$ with source-node v_0 in the tree T. Function $BroadcastTree$ returns the optimum forwarding scheme in this weighted tree. Finally, the intra-block forwarding is performed using Function $BroadcastBlock$ (see the last for-loop; Lines 14-17).

In the following paragraphs we provide details of Functions $BroadcastTree$ and $BroadcastBlock$ respectively.

Algorithm 2 $F = BroadcastTree(T, v_0)$

```
1:  H ← FindHeight(T, v_0)
2:  for all v ∈ T : height(v) = 0 do
3:     l(v) ← 0
4:  end for
5:  for h = 1 to H do
6:     T(h) = {v ∈ T : height(v) = h}
7:     for all v ∈ T(h) do
8:        (d_1, ..., d_r) ← SortChildren(v)
9:        l(v) ← max_{1≤i≤r}{d_i + i}
10:    end for
11: end for
12: F ← LargestLabel(v_0)
13: return F
```

The height H of the tree with respect to the source-node as root is found in Line 1. In the block of Lines 2-4, all leaf-nodes are labelled with $l(v) = 0$. This label represent the *priority* to forward this node (the default label to leaves mean that the forwarding order in leaf-nodes is indifferent). All the remaining nodes are labelled during the for-loop of Lines 5-11. We iteratively consider

different heights, represented by the node-subset $T(h) \subseteq T$ (see Line 6). Consider an arbitrary node $v \in T(h)$, with r children. These children are ordered from the largest label d_1 to the lowest label d_r (Line 8), and the label of v is found in Line 9. It is worth to remark how this labelling process works: the message is transmitted to node with the most stringent global timing constraint (i.e., the largest label) first. This greedy-like forwarding scheme (Line 12) achieves optimality in trees, and it is returned in Line 13.

Algorithm 3 $F_i = BroadcastBlock(B_i, v'_1, v'_2, \ldots, v'_k)$

1: $T \leftarrow MST(v'_1, B_i)$
2: **for** $i = 1$ **to** k **do**
3: $\quad w_i \leftarrow \{|B_i|/i\}$
4: $\quad w(u_i, v'_i) \leftarrow w_i$
5: $\quad T \leftarrow T \cup w(u_i, v'_i)$
6: **end for**
7: $F_i \leftarrow BroadcastTree(T, v'_1)$
8: **return** F_i

Let us finally consider *BroadcastBlock*. It receives a block B_i and the expected forwarding order $F(B_i)$ of all its k cut-points, to know, $v'_1, v'_2, \ldots, v'_k$. Therefore, node v'_1 is the root-node. In Line 1, Kruskal algorithm is applied into B_i to find the minimum spanning tree, rooted at v'_1. In order to force the correct forwarding order in the cut-points, the weights in their incident links are modified as follows. The incident link to v'_i in T will be assigned a weight $w_i = |B_i|/i$ (Lines 3-4). The resulting tree is updated in Line 5. Finally, Function *BroadcastTree* is applied into this tree in order to define an intra-block forwarding scheme (Line 7), and the result is returned in Line 8.

4 Experimental Analysis

In this section, we evaluate the performance of our heuristic for the MBT. The problem was previously studied for well-known topologies, therefore we first compare our heuristic with the known results of the problem for these graphs. Later, we explore the performance of our heuristic over a large dataset of synthetic graphs, following the Watts-Strogatz small-world model [29], that it is an acceptable model for real communication networks [28]. For this dataset, whenever it is possible, we compare results with an IPL solver.

4.1 Particular Topologies

Following previous studies, we will present the results of our heuristic for five well-known topologies: *Lattice*, *De Bruijn*,*Hypercube*, *Harary*, and *Chord Harary*. Tables 1-5 show the number of nodes (N), the optimal result according to the ILP formulation (ILP), and the $TreeBlock$ heuristic result (TB). Other columns in tables are specific of the particular topology.

As a particular lattice topology, we studied the 2D grid graphs. In this case, a problem instance is defined by the number of rows R and number of columns C of the grid, and by the source node defined as (r, c), where $1 \leq r \leq R$ is the row number and $1 \leq c \leq C$ is the column number of the source node. Following [17], when source node is $(1, 1)$ then the broadcast time is precisely the diameter $D = R-1+C-1 = R+C-2$. Table 1 summirizes the results for 16 random 2D grid instances. The heuristic $TreeBlock$ reaches the optimum result in all of them.

In [12], Frank Harary introduced the Harary graph, noted $H_{k,n}$. It is a k-connected graph on n vertices which have a degree of at least k and $kn2$ edges. The structure of $H_{k,n}$ depends on the parities of k and n.

The algorithm analyzed in [4] ensures to find the exact result of broadcasting time for half of the cases. The results are mostly optimal in the instances studied with the TB heuristic; in the other cases they turn out to be very close, being a timeslot the difference.

Chord-Harary graph is a variation of the Harary graph, $CH_{k,n}$, introduced in [3]. This topology is proposed by having managed to find an algorithm where low values of broadcasting time can be reached. It was also shown that it reaches the optimum for a subset of these graphs, where k maintains a certain relation with n. Table ref tab: chordharary shows that the results obtained by the TB heuristic are still competitive, reaching the exact result in some cases, for this topology. The chosen instances did not take into account the relationship between the parameters for which the algorithm of cite bhabak2017 achieves optimality precisely in order to observe how it behaves in generality.

Finally, following what is presented in [16] in Table 4 and Table 5 it can be observed that the TB algorithm achieves competitive results with the previous heuristics for the analyzed cases .

Having tested different topologies it is found that the ILP formula, although it is not capable of solving all the instances, reaches the optimal values for many cases. On the other hand, the TB heuristic was developed without specializing in any type of graph in particular and with focus on graphs with

more than one biconnex component. However, it achieves acceptable results, with the gap between TB and the heuristics presented very small in almost all cases.

Table 1
Lattice

R	C	r	c	N	ILP	TB
7	10	1	1	70	15	15
7	10	2	3	70	12	12
7	10	4	1	70	13	13
7	10	4	5	70	9	9
9	15	1	1	135	22	22
9	15	3	4	135	17	17
9	15	5	1	135	19	19
9	15	5	8	135	13	13
11	20	1	1	220	29	29
11	20	3	5	220	23	23
11	20	6	1	220	25	25
11	20	6	10	220	16	16
13	30	1	1	390	41	41
13	30	4	8	390	31	31
13	30	7	1	390	36	36
13	30	7	15	390	22	22

Table 2
Harary. Broadcast scheme S. [3]

N	k	n	ILP	TB
17	2	17	9	9
17	3	17	5	5
17	5	17	5	5
17	6	17	5	5
17	7	17	5	5
30	2	30	15	15
30	3	30	9	9
30	8	30	5	6
30	9	30	5	6
30	10	30	5	6
50	2	50	25	25
50	3	50	14	14
50	11	50	-	7
50	20	50	-	8
50	21	50	-	7
100	2	100	50	50

Table 3
Chord-Harary. Broadcast Scheme B. [3]

N	k	n	ILP	TB
10	3	10	4	4
10	5	10	4	4
30	3	30	9	9
30	5	30	5	6
30	7	30	5	6
50	5	50	6	7
50	7	50	-	7
50	9	50	-	7
50	11	50	-	6
100	7	100	7	8
100	11	100	-	8
100	13	100	-	8
300	7	300	-	15
300	11	300	-	10
300	17	300	-	10

Table 4
De Bruijn. RH, TBA, NTBA [16]

N	m	n	RH	TBA	NTBA	ILP	TB
8	2	3	4	4	4	3	4
16	2	4	5	5	5	4	5
32	2	5	7	6	7	6	7
64	2	6	8	8	8	7	8
128	2	7	9	9	10	-	10
256	2	8	11	11	12	10	12
512	2	9	12	12	13	-	14
1024	2	10	14	14	15	-	15
2048	2	11	15	15	17	-	17
4096	2	12	17	17	19	-	19
8192	2	13	18	18	20	-	21

Table 5
Hypercube. C, TBA, NTBA [16].

N	l	dim	O	C	TBA	NTBA	TB
16	2	4	4	-	-	-	4
32	2	5	5	5	5	5	5
64	2	6	6	7	6	7	6
128	2	7	7	8	7	9	7
256	2	8	8	9	9	11	8
512	2	9	9	10	10	14	9
1024	2	10	10	12	11	15	10
2048	2	11	11	-	12	18	11
4096	2	12	12	-	13	20	12

4.2 Large Dataset of Watts-Strogatz Graphs

An analysis was carried out on small-world networks for different number of nodes, and with different levels of connectivity, in order to be able to identify if it performs for graphs of more significant sizes and closer to reality.

In small world instances, we expect to compare quality and execution time.

Watts and Strogatz identified that many real networks have high levels of clustering, but small distances between most nodes. And it is particulary true in communication networks, where our problem applies. In order to create a network graph with both of these properties, Watts and Strogatz creates a graph model that begings with a lattice structure, and then randomly rewires a small percentage of the edges (with an independent probability p, and with attention to avoid the construction of loops and multi-edges). As expected, this model has high level of clustering and low average distances between nodes.

Fig. 1. Instances Distribution

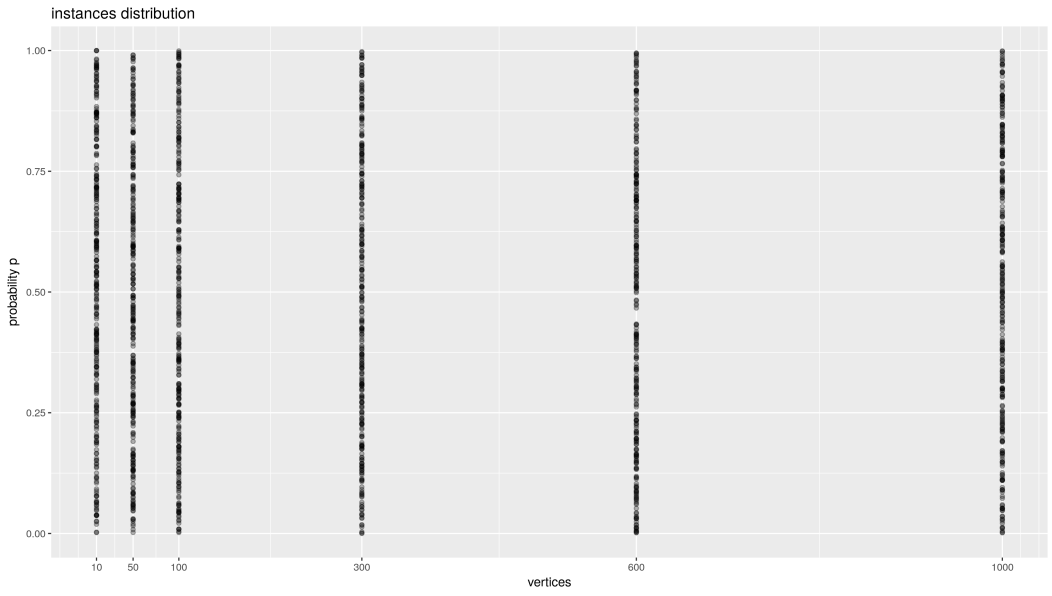
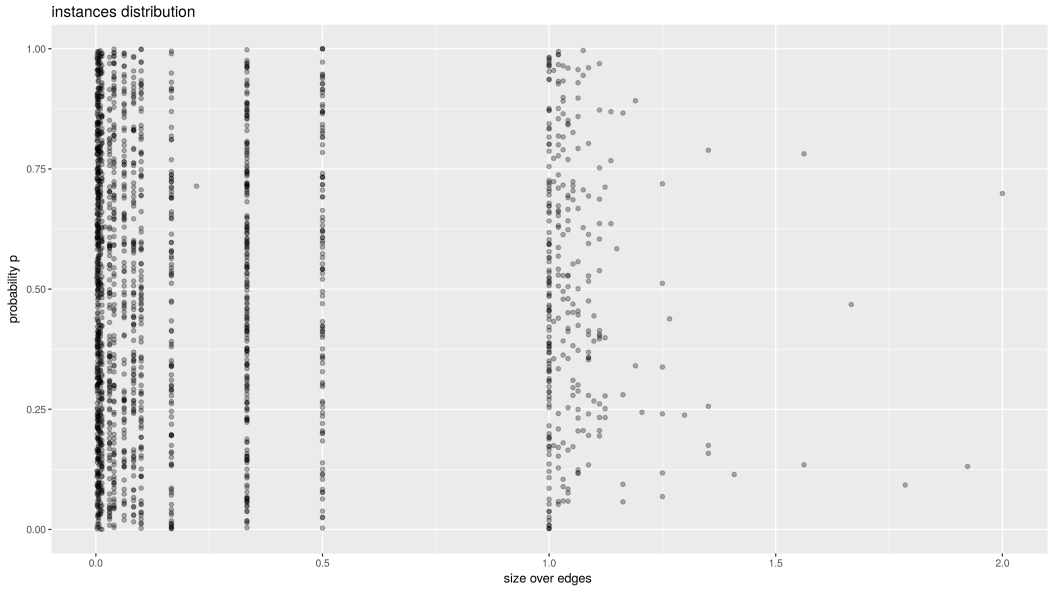


Fig. 2. Instances Distribution considering edges



The figure 1 shows the distribution of instances generated based on the probability of rewirng p and the number of nodes, clarifying that the choice of p was made uniformly independent.

In order to prove the deviation of the heuristic by altering the size; Besides having generated instances with different numbers of nodes, a second variable called *neighboorhood* was used for some cases, which directly impacts the number of edges of the graphs, increasing connectivity between nodes. In figure 2 the distribution is observed taking into account the probability and also the relationship between the number of nodes and the amount of edges.

For each instance, both the constructed heuristic and the ILP formula were executed in order to make a comparison of the results.

Regarding the processing times, it is evidenced in the figure 3 that for many cases it was not possible to throw a solution using the ILP formula, mostly for large instances. This is due to the complexity that the algorithm supposes, consequence of the complexity of the problem.

However, in cases where ILP found an optimal solution, it did so in less time than the heuristic. But this is not relevant since: 1. ILP is run on CPLEX, a highly optimized code commercial package while the heuristic is a research non-optimized code. 2. CPLEX is run on a very high performing server while heuristic is run on home-computers.

Likewise, it is observed in the figures 4 and 5 that the execution times of

the heuristic are also strongly related to the size of the instance, both by the number of nodes, as well as by the number of edges. However, it is visualized that p does not seem to affect at least the times.

Fig. 3. Comparative: execution time versus instance size

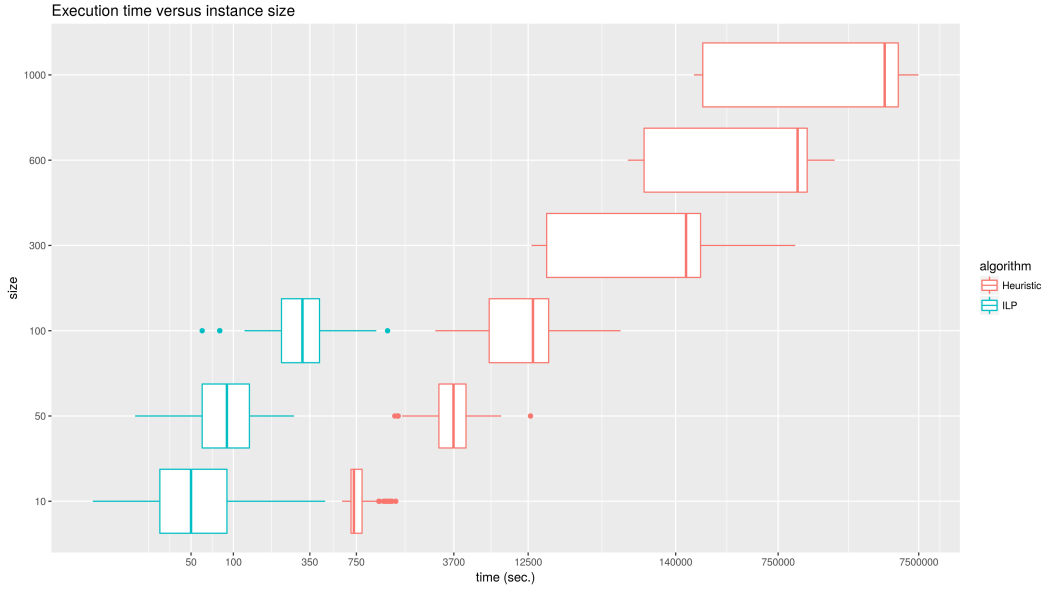


Fig. 4. Heuristic execution time versus instance size

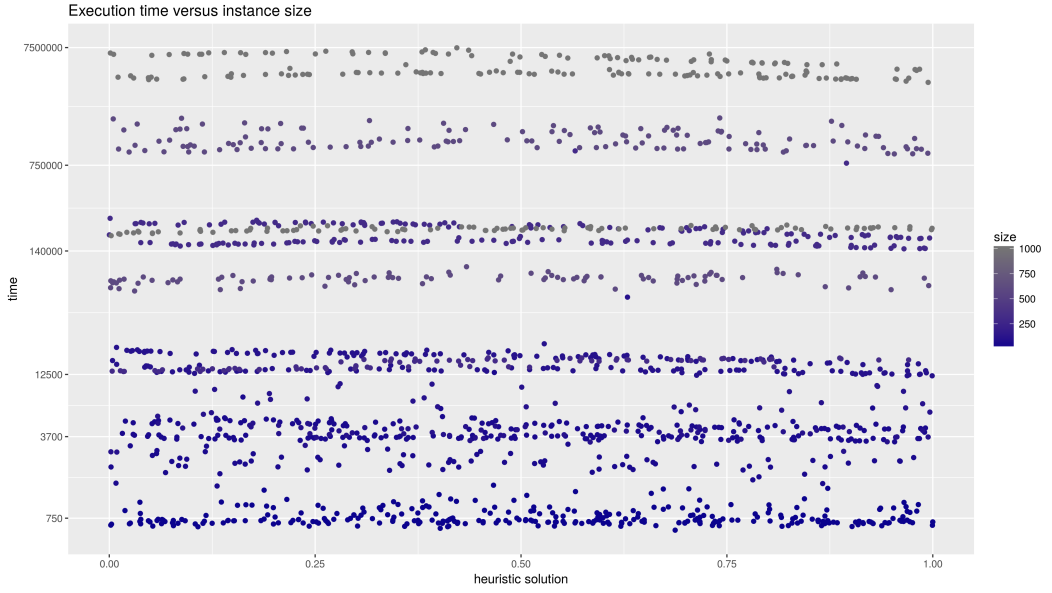
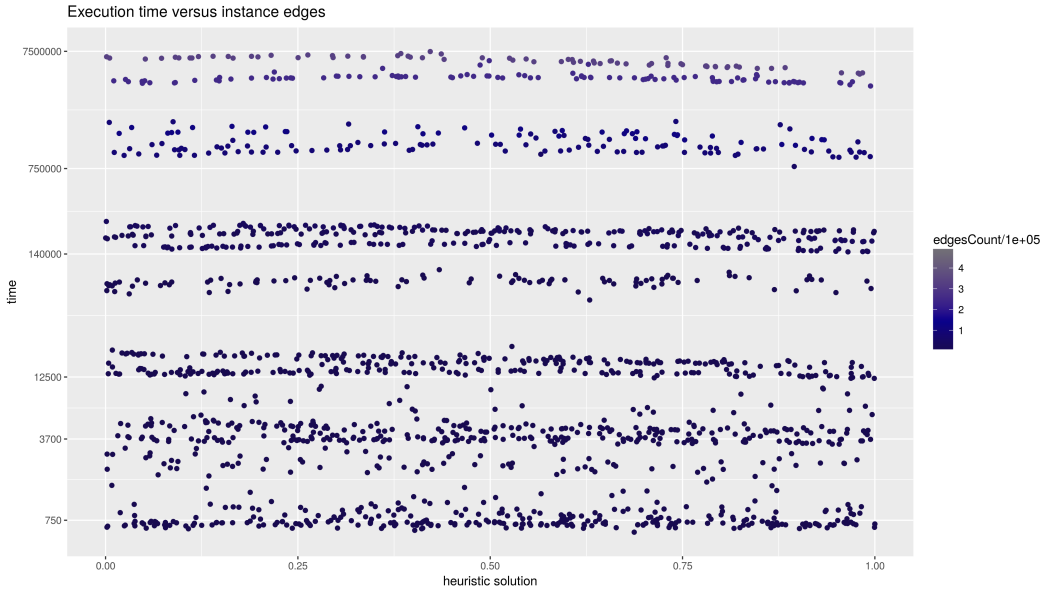
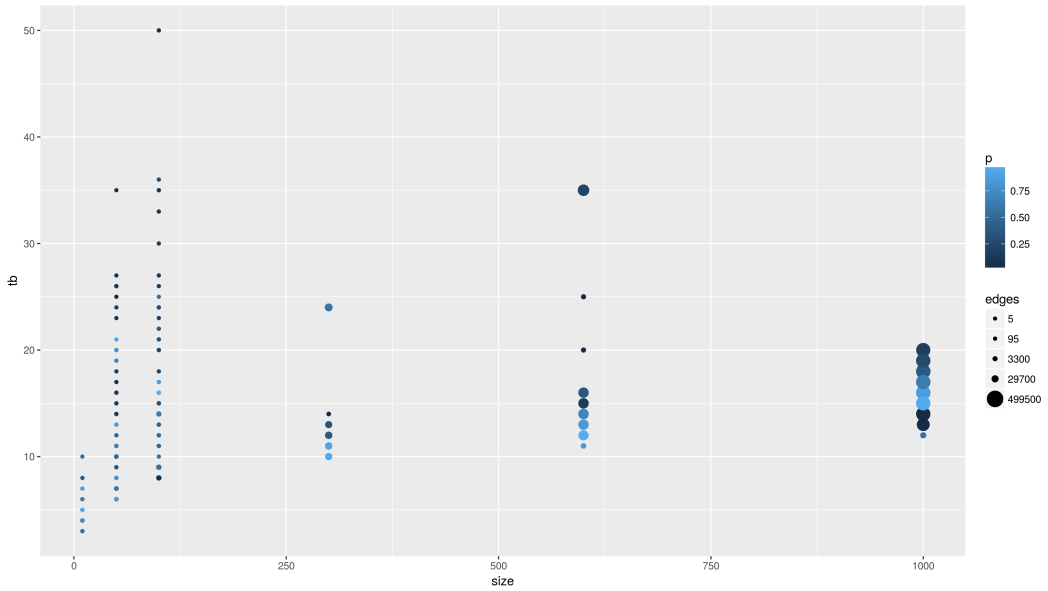


Fig. 5. Heuristic execution time versus instance edges count



Considering groups of instances of equal size (nodes and edges), in the figure 6 we see a tendency to reach a higher broadcasting time when the probability of rewiring is low. This observation seems to be accurate considering the process of creation of the small-worlds, because with a low probability of rewiring, the graphs would be more similar to the initial lattice. In the worst case a ring, with broadcasting time equal to the number of nodes -1.

Fig. 6. Heuristic solution versus p and edges grouped by size



Finally, for the instances in which the ILP formula achieved an optimal result, a comparison can be made in terms of the difference in value achieved in each algorithm.

In the table 6 is shown for each group of instances, which was the range of optimal values found by the formula ILP and which by the heuristic. The column 'GAP' indicates what is the average difference between the solution achieved with the heuristic and the ILP formula. In addition, the column 'no GAP' indicates the number of instances for which the heuristic found the optimum.

In the table 7 you can see in what percentage the results of the heuristic approach the optimum. The 100-size segment stands out, where 87% of the instances were resolved with a precision of less than 10%. For the rest of the cases it is not minor to emphasize that the size of the set of tests for which it was possible for the two algorithms to find a solution is considerably smaller.

Table 6

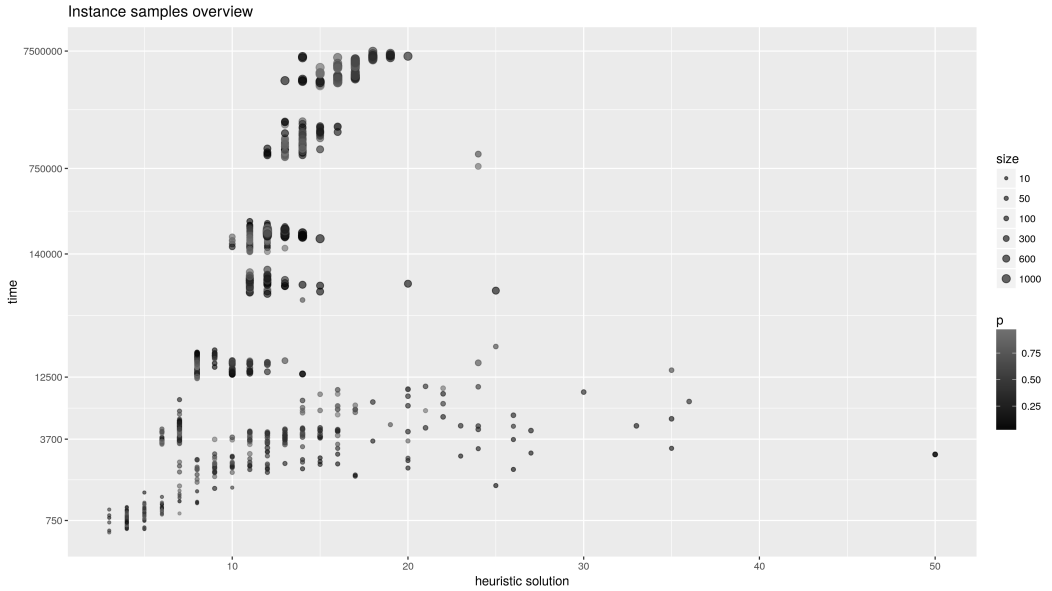
	size	edges	TB	ILP	GAP	no GAP	#instances
1	10	20.01	4.39 ± 0.91	4.22 ± 0.65	0.04 ± 0.12	263.00	301.00
2	50	46.52	12.85 ± 5.33	11.51 ± 5.27	0.14 ± 0.17	33.00	100.00
3	100	94.04	16.9 ± 7.4	14.46 ± 7.13	0.19 ± 0.21	22.00	100.00

Table 7

	size	edges	GAP $\leq 10\%$	GAP $\leq 25\%$	#instances
1	10	20.01	263.00	287.00	301.00
2	50	46.52	47.00	87.00	100.00
3	100	94.04	43.00	77.00	100.00

La figura 7 muestra la distribucin de los datos en terminos de tamao y probabilidad de rewiring en comparacion con los tiempos y resultados.

Fig. 7.



5 Conclusions and Trends for Future Work

This paper studies the problem Minimum Broadcast Time. Two algorithms are presented to find the result, one through an exact method and the other through a heuristic. The exact method, as could be anticipated, is interesting for the resolution of instances of smaller size, both in number of nodes and edges; throwing for these cases results in time quite small (less than two hours). For its part, the heuristic, being its intention to cover a larger universe of instances, was tested in larger sizes also yielding results in a reasonable time for up to 1000 nodes.

Regarding the deviation of the results achieved by the heuristic against

the optimum; It is found that there is an important success in cases where the optimum is known. The margin of error exists and is considerable; it does not exceed 25 % in most cases. Likewise, in comparison to other known heuristics, focused mostly on specific topologies, the results are competitive.

In order to improve the accuracy of the results for the heuristic, it is expected to work on some improvements. First, work on the decision of each node to transmit to another block or continue in it. The assembly of the initial tree of each block does not yield the most convenient and does not take advantage of the maximum transmission capacity of each node for all cases. Improving this step of the algorithm would incur direct improvements to the solution found.

Regarding the data set used, we understand it opportune to generate a larger one, in order to confirm some trends. In addition, it would be interesting to be able to test the methods presented in other known topologies, to validate the behavior in them.

Acknowledgment

This work is partially supported by Project CSIC I+D 395 entitled *Sistemas Binarios Estocásticos Dinámicos*.

References

[1] Anulova, S., "Fluid Limit for Switching Closed Queueing Network with Two Multi-servers," Springer International Publishing, Cham, 2017 pp. 343–354.

[2] Beier, R., R. Beier, J. F. Sibeyn and J. F. Sibeyn, *A powerful heuristic for telephone gossiping* (2000).

[3] Bhabak, P., H. Harutyunyan and P. Kropf, *Efficient broadcasting algorithm in harary-like networks*, in: *2017 46th International Conference on Parallel Processing Workshops (ICPPW)*, 2017, pp. 162–170.

[4] Bhabak, P., H. A. Harutyunyan and S. Tanna, *Broadcasting in harary-like graphs*, pp. 1269–1276.
URL http://ieeexplore.ieee.org/document/7023754/

[5] Chuang, Y.-T., *Protecting against malicious and selective forwarding attacks for p2p search & retrieval system*, Peer-to-Peer Networking and Applications **10** (2017), pp. 1079–1100.

[6] Crescenzi, P., P. Fraigniaud, M. Halld?rsson, H. A. Harutyunyan, C. Pierucci, A. Pietracaprina and G. Pucci, *On the complexity of the shortest-path broadcast problem*, Discrete Applied Mathematics **199** (2016), pp. 101 – 109, sixth Workshop on Graph Classes, Optimization, and Width Parameters, Santorini, Greece, October 2013.
URL http://www.sciencedirect.com/science/article/pii/S0166218X15002267

[7] Edmonds, J., "Paths, Trees, and Flowers," Birkhäuser Boston, Boston, MA, 1987 pp. 361–379.

[8] Elkin, M. and G. Kortsarz, *A combinatorial logarithmic approximation algorithm for the directed telephone broadcast problem*, SIAM Journal on Computing **35** (2005), pp. 672–689.

[9] Farley, A. M., *Broadcast time in communication networks*, SIAM Journal on Applied Mathematics **39** (1980), pp. 385–390.

[10] Fraigniaud, P. and S. Vial, *Approximation algorithms for broadcasting and gossiping*, J. Parallel Distrib. Comput. **43** (1997), pp. 47–55.
URL http://dx.doi.org/10.1006/jpdc.1997.1318

[11] Garey, M. R. and D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness," W. H. Freeman & Company, New York, NY, USA, 1979.

[12] Harary, F., *The maximum connectivity of a graph*, Proceedings of the National Academy of Sciences of the United States of America **48** (1962), pp. 1142–1146.
URL http://www.jstor.org/stable/71730

[13] Harutyunyan, H., A. Liestman, J. Peters and D. Richards, *Broadcasting and gossiping*, in: J. L. Gross, J. Yellen and P. Zhang, editors, *Handbook of Graph Theory, Second Edition*, Chapman & Hall/CRC, 2013, 2nd edition p. 18.

[14] Harutyunyan, H. A. and S. Kamali, *Efficient broadcast trees for weighted vertices*, Discrete Applied Mathematics **216** (2017), pp. 598 – 608, levon Khachatrian's Legacy in Extremal Combinatorics.
URL http://www.sciencedirect.com/science/article/pii/S0166218X16304358

[15] Harutyunyan, H. A. and B. Shao, *An efficient heuristic for broadcasting in networks*, Journal of Parallel and Distributed Computing **66** (2006), pp. 68 – 76.
URL http://www.sciencedirect.com/science/article/pii/S0743731505001681

[16] Harutyunyan, H. A. and W. Wang, *Broadcasting algorithm via shortest paths*, pp. 299–305.
URL http://ieeexplore.ieee.org/document/5695616/

[17] Hedetniemi, S. M., S. T. Hedetniemi and A. L. Liestman, *A survey of gossiping and broadcasting in communication networks*, Networks **18** (1988), pp. 319–349.
URL http://dx.doi.org/10.1002/net.3230180406

[18] Klasing, R., B. Monien, R. Peine and E. A. St?hr, *Broadcasting in butterfly and debruijn networks*, Discrete Applied Mathematics **53** (1994), pp. 183 – 197.
URL
http://www.sciencedirect.com/science/article/pii/0166218X94901848

[19] Kortsarz, G. and D. Peleg, *Approximation algorithms for minimum-time broadcast*, SIAM Journal on Discrete Mathematics **8** (1995), pp. 401–427.

[20] Liestman, A. L. and J. G. Peters, *Broadcast networks of bounded degree*, SIAM Journal on Discrete Mathematics **1** (1988), pp. 531–540.

[21] Liu, W., D. Peng, C. Lin, Z. Chen and J. Song, *Enhancing tit-for-tat for incentive in bittorrent networks*, Peer-to-Peer Networking and Applications **3** (2010), pp. 27–35.

[22] Liyana Arachchige, C. J., S. Venkatesan, R. Chandrasekaran and N. Mittal, "Minimal Time Broadcasting in Cognitive Radio Networks," Springer Berlin Heidelberg, Berlin, Heidelberg, 2011 pp. 364–375.

[23] Pazzi, R. W., A. Boukerche, R. E. Grande and L. Mokdad, *A clustered trail-based data dissemination protocol for improving the lifetime of duty cycle enabled wireless sensor networks*, Wirel. Netw. **23** (2017), pp. 177–192.

[24] Peleg, D., "Time-Efficient Broadcasting in Radio Networks: A Review," Springer Berlin Heidelberg, Berlin, Heidelberg, 2007 pp. 1–18.

[25] Romero, P., "Mathematical Analysis of Scheduling Policies in Peer-to-Peer video streaming networks," Ph.D. thesis, Universidad de la República, Montevideo, Uruguay (2012).

[26] Schindelhauer, C., *On the inapproximability of broadcasting time*, in: *Proceedings of the Third International Workshop on Approximation Algorithms for Combinatorial Optimization*, APPROX '00 (2000), pp. 226–237.
URL http://dl.acm.org/citation.cfm?id=646688.702973

[27] Slater, P. J., E. J. Cockayne and S. T. Hedetniemi, *Information dissemination in trees*, SIAM Journal on Computing **10** (1981), pp. 692–701.

[28] Watts, D. J., "Six Degrees: The Science of a Connected Age," W. W. Norton and Company, Inc., 2004, 1st edition.

[29] Watts, D. J. and S. H. Strogatz, *Collective dynamics of 'small-world' networks*, Nature **393** (1998), pp. 440–442.

[30] West, D., "Introduction to Graph Theory," Featured Titles for Graph Theory Series, Prentice Hall, 2001.

[31] Yang, X. and G. de Veciana, *Service Capacity of Peer to Peer Networks*, , **4**, 2004, pp. 2242–2252.

Referencias

[1] Anulova, S., "Fluid Limit for Switching Closed Queueing Network with Two Multi-servers," Springer International Publishing, Cham, 2017 pp. 343–354.

[2] Averbuch, A., R. Hollander Shabtai and Y. Roditty, *Efficient construction of broadcast graphs*, Discrete Applied Mathematics **171** (2014), pp. 9–14.
URL `https://linkinghub.elsevier.com/retrieve/pii/S0166218X14000614`

[3] Beier, R. and J. F. Sibeyn, *A Powerful Heuristic for Telephone Gossiping*, The Seventh International Colloquium on Structural Information & Communication Complexity (2000), pp. pp. 17–36.

[4] Bhabak, P., H. Harutyunyan and P. Kropf, *Efficient broadcasting algorithm in harary-like networks*, in: *2017 46th International Conference on Parallel Processing Workshops (ICPPW)*, 2017, pp. 162–170.

[5] Bhabak, P., H. A. Harutyunyan and S. Tanna, *Broadcasting in harary-like graphs*, pp. 1269–1276.
URL `http://ieeexplore.ieee.org/document/7023754/`

[6] Chuang, Y.-T., *Protecting against malicious and selective forwarding attacks for p2p search & retrieval system*, Peer-to-Peer Networking and Applications **10** (2017), pp. 1079–1100.

[7] Crescenzi, P., P. Fraigniaud, M. Halld?rsson, H. A. Harutyunyan, C. Pierucci, A. Pietracaprina and G. Pucci, *On the complexity of the shortest-path broadcast problem*, Discrete Applied Mathematics **199** (2016), pp. 101 – 109, sixth Workshop on Graph Classes, Optimization, and Width Parameters, Santorini, Greece, October 2013.
URL `http://www.sciencedirect.com/science/article/pii/S0166218X15002267`

[8] de Sousa, A., G. Gallo, S. Gutierrez, F. Robledo, P. Rodríguez-Bocca and P. Romero, *Heuristics for the Minimum Broadcast Time*, Electronic Notes in Discrete Mathematics **69** (2018), pp. 165–172.
URL `https://linkinghub.elsevier.com/retrieve/pii/S1571065318301665`

[9] Elkin, M. and G. Kortsarz, *A combinatorial logarithmic approximation algorithm for the directed telephone broadcast problem*, SIAM Journal on Computing **35** (2005), pp. 672–689.

[10] Farley, A. M., *Broadcast time in communication networks*, SIAM Journal on Applied Mathematics **39** (1980), pp. 385–390.

[11] Garey, M. R. and D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness," W. H. Freeman & Company, New York, NY, USA, 1979.

[12] Harary, F., *The maximum connectivity of a graph*, Proceedings of the National Academy of Sciences of the United States of America **48** (1962), pp. 1142–1146.
URL `http://www.jstor.org/stable/71730`

[13] Harutyunyan, H. A. and B. Shao, *An efficient heuristic for broadcasting in networks*, Journal of Parallel and Distributed Computing **66** (2006), pp. 68 – 76.
URL `http://www.sciencedirect.com/science/article/pii/S0743731505001681`

[14] Harutyunyan, H. A. and W. Wang, *Broadcasting algorithm via shortest paths*, pp. 299–305.
URL `http://ieeexplore.ieee.org/document/5695616/`

[15] Hedetniemi, S. M., S. T. Hedetniemi and A. L. Liestman, *A survey of gossiping and broadcasting in communication networks*, Networks **18** (1988), pp. 319–349.
URL `http://dx.doi.org/10.1002/net.3230180406`

[16] Hoelting, C. J., D. A. Schoenefeld and R. L. Wainwright, *A genetic algorithm for the minimum broadcast time problem using a global precedence vector*, in: *Proceedings of the 1996 ACM symposium on Applied Computing - SAC '96* (1996), pp. 258–262.
URL `http://portal.acm.org/citation.cfm?doid=331119.331187`

[17] Kortsarz, G. and D. Peleg, *Approximation algorithms for minimum-time broadcast*, SIAM Journal on Discrete Mathematics **8** (1995), pp. 401–427.

[18] Kruskal, J. B., *On the shortest spanning subtree of a graph and the traveling salesman problem*, Proceedings of the American Mathematical Society **7** (1956), pp. 48–50.
URL `http://www.jstor.org/stable/2033241`

[19] Lin, H.-C., T.-M. Lin and C.-F. Wu, *Constructing application-layer multicast trees for minimum-delay message distribution*, Information Sciences **279** (2014), pp. 433–445.
URL `http://linkinghub.elsevier.com/retrieve/pii/S0020025514004277`

[20] Liu, W., D. Peng, C. Lin, Z. Chen and J. Song, *Enhancing tit-for-tat for incentive in bittorrent networks*, Peer-to-Peer Networking and Applications **3** (2010), pp. 27–35.

[21] Liyana Arachchige, C. J., S. Venkatesan, R. Chandrasekaran and N. Mittal, "Minimal Time Broadcasting in Cognitive Radio Networks," Springer Berlin Heidelberg, Berlin, Heidelberg, 2011 pp. 364–375.

[22] Marathe, M. V., R. Ravi, D. J. Rosenkrantz, R. Sundaram, S. S. Ravi and H. B. H. Iii, *Bicriteria Network Design Problems* , p. 26.

[23] Minoux, M., *Efficient greedy heuristics for steiner tree problems using reolptimization and super modularity*, INFOR: Information Systems and Operational Research **28** (1990), pp. 221–233.
URL https://doi.org/10.1080/03155986.1990.11732136

[24] Pazzi, R. W., A. Boukerche, R. E. Grande and L. Mokdad, *A clustered trail-based data dissemination protocol for improving the lifetime of duty cycle enabled wireless sensor networks*, Wirel. Netw. **23** (2017), pp. 177–192.

[25] Ravi, R., *Rapid rumor ramification: Approximating the minimum broadcast time*, in: *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, SFCS '94 (1994), pp. 202–213.
URL https://doi.org/10.1109/SFCS.1994.365693

[26] Robledo, F., P. Rodrıguez-Bocca and P. Romero, *Analysis and Performance of Complete Homogeneous Communication Networks*, Technical report.
URL https://www.colibri.udelar.edu.uy/jspui/bitstream/123456789/18403/1/Art%C3%ADculo_de_Performance_en_P2P.pdf

[27] Romero, P., "Mathematical Analysis of Scheduling Policies in Peer-to-Peer video streaming networks," Ph.D. thesis, Universidad de la República, Montevideo, Uruguay (2012).

[28] Shang, W., P. Wan and X. Hu, *Approximation algorithms for minimum broadcast schedule problem in wireless sensor networks*, Frontiers of Mathematics in China **5** (2010), pp. 75–87.
URL http://link.springer.com/10.1007/s11464-009-0050-4

[29] Slater, P. J., E. J. Cockayne and S. T. Hedetniemi, *Information dissemination in trees*, SIAM Journal on Computing **10** (1981), pp. 692–701.

[30] Watts, D. J., *Networks, Dynamics, and the Small-World Phenomenon*, American Journal of Sociology **105** (1999), pp. 493–527.
URL https://www.journals.uchicago.edu/doi/10.1086/210318

[31] Watts, D. J., "Six Degrees: The Science of a Connected Age," W. W. Norton and Company, Inc., 2004, 1st edition.

[32] Watts, D. J. and S. H. Strogatz, *Collective dynamics of 'small-world' networks*, Nature **393** (1998), pp. 440–442.

[33] Weng, M. X. and J. A. Ventura, *A doubling procedure for constructing minimal broadcast networks*, Telecommunication Systems **3** (1994), pp. 259–293.
URL `http://link.springer.com/10.1007/BF02110308`

[34] West, D., "Introduction to Graph Theory," Featured Titles for Graph Theory Series, Prentice Hall, 2001.

[35] Yang, X. and G. de Veciana, *Service Capacity of Peer to Peer Networks*, , **4**, 2004, pp. 2242–2252.

[36] Čevnik, M. and J. Žerovnik, *Broadcasting on cactus graphs* **33**, pp. 292–316.
URL `http://link.springer.com/10.1007/s10878-015-9957-8`

Índice de tablas

Índice de figuras

Índice de figuras

Esta es la última página.
Compilado el jueves 13 septiembre, 2018.
https://www.fing.edu.uy/inco