A tree-block decomposition-based heuristic for the minimum broadcast time

Amaro de Sousa

Departamento de Eletrónica,
Instituto de Telecomunicações,
Telecomunicações e Informática,
Universidade de Aveiro,
Aveiro, Portugal
Email: asou@ua.pt

Gabriela Gallo, Santiago Gutiérrez,
Franco Robledo, Pablo Rodríguez-Bocca and
Pablo Romero*

Facultad de Ingeniería,
Instituto de Computación,
Universidad de la República,
Montevideo, Uruguay
Email: gabriela.gallo@fing.edu.uy
Email: santiago.gutierrez@fing.edu.uy
Email: frobledo@fing.edu.uy
Email: prbocca@fing.edu.uy
Email: promero@fing.edu.uy
*Corresponding author

Abstract: The problem under study is the minimum broadcast time. 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 neighbouring-nodes in a selective fashion and each forwarding takes one time-slot. Historically, this problem 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 $\mathcal{NP}$-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 formulation for the problem is provided. Second, a competitive heuristic called *TreeBlock*, is developed. A fair comparison between TreeBlock and previous heuristics highlights the effectiveness of our proposal.

Keywords: minimum broadcast time; MBT; computational complexity; integer linear programming; ILP; heuristics.

Reference to this paper should be made as follows: de Sousa, A., Gallo, G., Gutiérrez, S., Robledo, F., Rodríguez-Bocca, P. and Romero, P. (2020) 'A tree-block decomposition-based heuristic for the minimum broadcast time', *Int. J. Metaheuristics*, Vol. 7, No. 4, pp.379–401.

Biographical notes: Amaro de Sousa received his five-year BSc in Electronics and Telecommunications Engineering from the University of Aveiro, Portugal in 1989, MSc in Telecommunications Engineering from the University College of North Wales, UK in 1991, and PhD in Electrical Engineering from the University of Aveiro in 2001. He is currently a Professor in the Department of Electronics, Telecommunications and Informatics, University of Aveiro, Portugal, and a senior researcher in the Applied Mathematics Group, Institute of Telecommunications, Aveiro. His research interests include advanced services and protocols for telecommunications, traffic engineering and network design, and optimisation algorithms for efficient network resource management.

Gabriela Gallo received her five-year BSc in Computer Science at the Universidad de la República, Uruguay, in 2008. She finished her end-project career studying the Minimum Broadcast Time, advised by Franco Robledo, Pablo Rodríguez-Bocca and Pablo Romero. She is currently studying in order to obtain her Master's in Operations Research, advised by Dr. Pablo Romero and Dr. Héctor Cancela. Her main interests cover decision science and operations research.

Santiago Gutiérrez received his five-year BSc in Computer Science at the Universidad de la República, Uruguay, in 2008. He finished her end-project career studying the Minimum Broadcast Time, advised by Franco Robledo, Pablo Rodríguez-Bocca and Pablo Romero. He is currently involved in industrial applications. His main interests cover network optimisation.

Franco Robledo is a Head Professor at the Operational Department and Mathematical Institute, Universidad de la República, Uruguay, where he received his five-year BSc in Computer Science and MSc in Computer Science. In 2005 he received his PhD in Computer Science from the University of Rennes I, France. He has published around 120 journal and/or conference refereed papers. He is currently the Head of the Laboratory of Probability and Statistics. His main interests cover network reliability, topological network design and optimisation.

Pablo Rodríguez-Bocca is a Full-time Professor in the Institute of Computer Science, Universidad de la República, Uruguay, where he received his five-year BSc in Computer Science and in Electrical Engineering, and MSc in Computer Science. In 2008, he received his PhD in Computer Science from the University of Rennes I, France. He has published around 50 journal and/or conference refereed papers. He has been involved in the organisation of several conferences and journal special issues. His main interests are network analysis, machine learning, and optimisation.

Pablo Romero is a Full-time Professor in the Operational Research Department and Mathematical Institute, Universidad de la República, Uruguay, where he received his five-year BSc in Electrical Engineering in 2008, MSc in Mathematical Engineering in 2009 and PhD in Computer Science in 2012. He has published around 80 journal and/or conference

refereed papers. He concluded the direction of three PhD in Computer Science and six MSc in Mathematical Engineering and Computer Science. His main interests cover system reliability analysis and optimisation.

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 (Anulova, 2017), tree-like or one-to-one forwarding schemes (Pazzi et al., 2017), the tit-for-tat communication paradigm for incentives (Liu et al., 2010), among many others. A fundamental fluid model for a complete peer-to-peer network was posed for the first time by Yang and de Veciana (2004). In this fluid model, the authors considered a complete network with identical users, where simultaneous forwarding takes proportionally more time. The authors claimed a counter-intuitive result, where *the one-to-one is a good forwarding strategy*. Even though Yang and de Veciana studied the service capacity of a file sharing peer-to-peer system, its formulation is general enough. They found a closed formula for the average waiting time following a one-to-one forwarding scheme when the population is a power of two. In Romero (2012), it was formally proved that the one-to-one forwarding scheme achieves the minimum waiting time is included, when the population is a power of two. Furthermore, we recently confirmed that the optimality holds for arbitrary (non-complete) graphs as well (Bhabak et al., 2020). Incidentally, the best one-to-one forwarding scheme is still an open problem, in the sense that the best neighbouring 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 (Farley, 1980). The MBT 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 and peer-to-peer networks (Chuang, 2017), or cognitive radio networks (Liyana Arachchige et al., 2011).

Section 3 contains a formal definition of the MBT in terms of integer linear programming (ILP). A natural description of the problem under study 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:

1 each phone-call requires one time-slot

2 a member can participate in only one call per slot

3 a member can only call a neighbour member.

The problem under study is the following: given a connected graph $G = (V, E)$ and a originator node $v_0 \in V$, 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 a communication network, where the nodes V are the members, and the edges E are the possible communication links between them. Defining an originator node v_0, the *MBT*, denoted $b(v_0, G)$, is the minimum number of time-slots necessary to disseminate a message from v_0 to all the other nodes of G.

This work is motivated by the potential applications of the MBT, and the importance of forwarding schemes in cooperative environments on the Internet. The main contributions of this article can be summarised by the following items:

- A novel heuristic called $TreeBlock$ is developed. $Treeblock$ combines two ingredients: all connected graphs accept a decomposition into blocks connected in a tree-like structure (Harary, 1962), and the globally optimum for the MBT in trees is already known (Slater et al., 1981).
- The effectiveness of our proposal is highlighted with a fair comparison with a previous heuristic.

This article is organised as follows. The main body of related work is covered in Section 2. A novel ILP formulation for the MBT is introduced in Section 3. An efficient heuristic that exploits a tree-block decomposition structure shared by all connected graphs is described in Section 4. A fair comparison with the most competitive heuristics is carried out in Subsection 5.1, while a testing on largest graphs is covered in Subsection 5.2. Section 6 presents the concluding remarks and some trends for future work.

2 Related work

Most works (Elkin and Kortsarz, 2005; Bhabak et al., 2017; Harutyunyan and Kamali, 2017; Bhabak et al., 2017) studied 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. Garey and Johnson included the MBT in the list of $\mathcal{NP}$-complete problems (Garey and Johnson, 1979). They considered an arbitrary starting set V_0 with possible more than a singleton, subject to the three previous broadcasting rules. Observe that the broadcast time equals one if and only if V_0 accepts a perfect matching in bipartite graphs, which accepts a polynomial-time algorithm (Edmonds, 1987). However, the case $T = 2$ accepts a reduction from 3-dimensional matching (Slater et al., 1981).

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 hyper-cube Q_m has $m2^{m-1}$ links and the same broadcast time is $b(Q_m) = m = log_2(n)$. Hence, the hyper-cube 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 optimal in trees is recursively achieved in a greedy-like manner (Slater et al., 1981). The idea is a bottom-to-top process, choosing members with maximum delay. 2D grid graphs accept an optimal forwarding scheme as well (Hedetniemi et al., 1988). 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 (Liestman and Peters, 1988), butterfly and DeBruijn networks (Klasing et al., 1994), and Harary networks (Bhabak et al., 2017).

In the general case, for an arbitrary graph G and an arbitrary originator node, Elkin and Kortsarz (2005) presented an algorithm to obtain a forwarding strategy that found the best theoretical upper bound known, with $O(\frac{\log(|V|)}{\log\log(|V|)}b(G))$ time-slots (Schindelhauer, 2000). The literature offers several metaheuristics and approximation algorithms for the MBT, see for instance: Elkin and Kortsarz (2005), Kortsarz and Peleg (1995), Fraigniaud and Vial (1997), Beier et al. (2000), Harutyunyan and Shao (2006), Harutyunyan and Kamali (2017), Harutyunyan and Wang (2010) and Crescenzi et al. (2016). In particular, Harutyunyan and Wang (2010) proposed a new heuristic called new tree-based algorithm (NTBA), and compared it with their previously developed tree-based algorithm (TBA) (Harutyunyan and Shao, 2006) and with the round heuristic (RH) (Beier and Sibeyn, 2000)) over a set of particular topologies. In Subsection 5.1, we will compare our proposal with these heuristics. Furthermore, several extensions to the original problem were studied. The reader is invited to consult (Peleg, 2007; Harutyunyan et al., 2013) for a literature review on the MBT and related problems.

3 Exact formulation for the MBT

In this section, an exact ILP formulation for the MBT is introduced. Let $G = (V, E)$ be a connected graph, where $V = \{0, 1, ..., n-1\}$ is the node-set and E is the set of pairs of nodes that can communicate directly. Consider also A as the arc set where each pair of nodes i and j in E is represented by two elements in A: (i, j) represents a message forwarded by i to j. For simplicity, we consider $v_0 = 0$ as the source-node (i.e., the singleton that owns the message at the beginning).

Recall that in every time-slot, a node i that owns the message can forward it to a single member of its neighbour-set $V(i)$ among the ones that did not yet receive the message. The broadcast time is the number of time-slots required for the message to reach the whole set V and the goal is to minimise the broadcast time.

Observe that the globally optimum solution for the MBT never exceeds the order of the graph. In order to define the ILP formulation, we consider the parameter $T = n$, which reflects an upper-bound for the maximum number of time-slots.

Then, we consider the set of binary variables x_{ij}^t such that, when equal to one, mean that the message is forward from node i to node j in time-slot t. Finally, we consider an integer variable z whose value represents the broadcast time (in number of time-slots). The ILP model for the MBT is expressed as follows:

Minimise:

$$z \tag{1}$$

Subject to:

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

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

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

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

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

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

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

The objective function (1) is the broadcast time to be minimised. In the first time-slot $t = 1$, constraint (2) guarantee that the message is forwarded only by the source-node $i = 0$ to one its its neighbouring nodes. Constraint (3) guarantee that the message is received by each node i (except the source-node 0) exactly once from one neighbouring node. Constraint (4) guarantee that each node i can only forward the message to at most one neighbouring node $j \in V(i)$ on each time slot $t = 2, ..., T$ [the case of $t = 1$ is already dealt with by constraint (2)]. Constraint (5) guarantee that the message is forwarded 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 $\tau = 1, ..., t-1$ from another neighbouring node of i (i.e., not from node j). Constraint (6) guarantee that the broadcast time is not lower than the time-slot of any message forward. Finally, constraints (7)–(8) are the variable domain constraints.

The ILP formulation has n^3 binary variables, where n is the number of nodes in the input graph. In order to reduce the number of variables involved in our formulation, several observations are in order. First, consider the minimum number of hops h_i in a shortest path from 0 to i. For a given number of hops h, consider set V_h composed by all nodes such that $h_i = h$. Therefore, a node $i \in V_h$ cannot forward the message on any of the time-slots $t = 1, ..., h$, since the time is not enough for the message to reach node i. So, for each node $i \in V_h$, we can set to 0 all variables x_{ij}^{t}, for $t = 1, ..., h$. We can further reduce the number of variables in the following way. Let d be the degree of the source-node 0. Note that, even if all d neighbouring nodes are used by source node 0 to forward the message, an optimal solution always exists where these message forwards are in the first d time slots. So, we can set to 0 all variables x_{0j}^{t}, for $(0, j) \in A$ and $t = d + 1, ..., T$.

Besides the elimination of these variables, we can also identify the following valid inequalities. Consider again the set of nodes V_h for a given h. At the end of h time-slots, at most one of these nodes can receive the message and, so, at most one message forward can happen from these nodes in time-slot $t = h + 1$. So, the following valid inequalities that can be added to the ILP model:

$$\sum_{i \in V_h} \sum_{j \in V(i)} x_{ij}^{h+1} \leq 1, \qquad h = 1, ..., T-1 \tag{9}$$

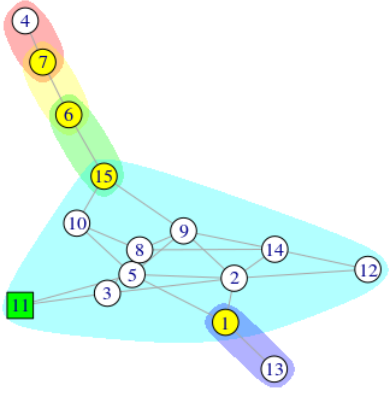
Our computational tests show that both improvements. i.e., eliminating the identified variables and adding constraint (9), indeed shorten the run-time required to solve the resulting ILP and enable to find the optimal solutions for larger problem instances.

One key aspect is the upper-bound parameter T which has an huge impact on the total number of variables and this value should be as close as possible to the optimal solution value. In the experimental results presented in Section 5, parameter T is set with the result of the *TreeBlock* heuristic. So, the efficiency of the heuristic also contributes to the capacity of the exact method to compute the optimal solutions for the largest problem instances that can be solved by the exact method.

4 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.

Figure 1 Example of two-node connected blocks for a graph instance (see online version for colours)



First, we will describe some formal aspects of our algorithm. A node $v \in V(G)$ is 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 decomposition, 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. Given the importance of this concept, we consider the following:

Tree-block graph: Given a connected graph $G = (V, E)$, the tree-block graph is $B(G) = (B \cup U, E')$, where $U = \{v_1, ..., v_r\}$ is the set of cut-points in G, $B = \{B_1, ..., B_s\}$ 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.

The following property of tree-blocks is detailed in the classical book on graph theory (West, 2001):

Proposition 1: The tree-block is always a tree.

The key is to observe that a cut-point in G is also a cut-point in $B(G)$. On the other hand, $B(G)$ has *leaf-blocks*, that are incident with a single cut-point. The reader can find further properties of tree-blocks in West (2001).

In the following, we describe our heuristic for the MBT, called $TreeBlock$. The two main building blocks of our algorithm are the functions $BroadcastTree$ and $BroadcastBlock$. Essentially, $BroadcastTree$ serves as an *inter-block* 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.

4.1 $TreeBlock$ algorithm

The full heuristic $TreeBlock$ is presented in Algorithm 1. A step-by-step description of the $TreeBlock$ algorithm 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 line 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 inter-block 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. The reader can see these steps represented at Figures 2 and 3 for the instance graph shown at Figure 1. 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. Finally, the blocks of lines 13–18 build the forwarding scheme. Specifically, the globally optimum forwarding scheme $BroadcastTree$ is applied to the skeleton tree T as the inter-block forwarding scheme (line 14). Then, an intra-block forwarding scheme takes effect for each block B_i (lines 15–18), using function $BroadcastBlock$.

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

Figure 2 Complete weighted graph built in line 3 of algorithm $TreeBlock$ (see online version for colours)

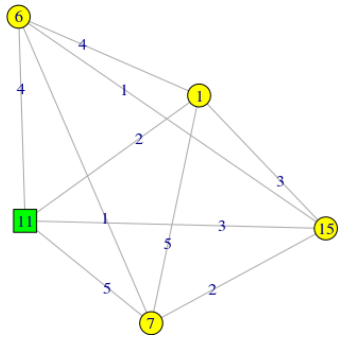


Figure 3 MST of the complete weighted graph built in line 4 of algorithm $TreeBlock$ (see online version for colours)

Figure 4 MST with leaf blocks (see online version for colours)

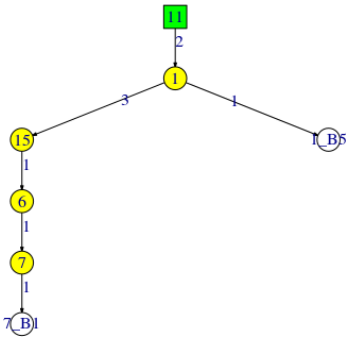
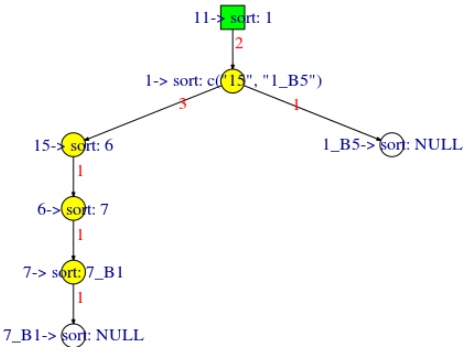


Figure 5 Result from BroadcastTree (see online version for colours)



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. A representation of these steps is shown at Figures 4 and 5. Finally, the intra-block forwarding is performed using function $BroadcastBlock$, which combines a strategy of recursive exploration of the whole graph, and also the intra-block forwarding. The function $BroadcastBlock$ is called with source-node v_0 and for the blocks where v_0 is into, called $B(v_0)$.

The following subsections provide details of the respective functions $BroadcastTree$ (Algorithm 2) and $BroadcastBlock$ (Algorithm 3).

4.2 $BroadcastTree$ function

In Algorithm 2, $BroadcastTree$, 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 optimal in trees, and it is returned in line 13.

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

1: $H \leftarrow FindHeight(T, v_0)$
2: **for all** $v \in T : height(v) = 0$ **do**
3: $\quad l(v) \leftarrow 0$
4: **end for**
5: **for** $h = 1$ **to** H **do**
6: $\quad T(h) = \{v \in T : height(v) = h\}$
7: $\quad$ **for all** $v \in T(h)$ **do**
8: $\quad\quad (d_1, ..., d_r) \leftarrow SortChildren(v)$
9: $\quad\quad l(v) \leftarrow max_{1 \leq i \leq r}\{d_i + i\}$
10: $\quad$ **end for**

11: **end for**
12: $F \leftarrow LargestLabel(v_0)$
13: **return** F

4.3 $BroadcastBlock$ function

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, $v'_1, v'_2, ..., v'_k$. Therefore, node v'_1 is the root-node. In line 1 of Algorithm 3, $DynamicTree$ algorithm is applied into B_i to find a minimum spanning tree, rooted at v'_1. This step is important to try to find a better spanning tree than the one that returns the Kruskal method, which does not worry about minimising times, but only the number of re-sends that are made. A clear instance that shows that need is a complete graph (Figure 6), where a minimum spanning tree is the one shown at Figure 7. Certainly, the number of re-sends is optimal but the time-slots needed to complete the broadcast in the block is not.

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

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

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

1: **for all** $v \in B_i$ **do**
2: $T \leftarrow T \cup shortestPath(B_i, v_0, v)$
3: **end for**
4: $TB_i \leftarrow BroadcastTree(T, v_0)$
5: **for all** $v \in B_i$ **do**
6: $N'(v) = \{n \in B_i : n \in N_{B_i}(v)\ and\ n \notin N_{TB_i}(v)\}$
7: $t_v = receivedTime(v, TB_i)$
8: $sendsDone = \#(N_{TB_i}(v))$
9: **for all** $n \in N'(v)$ **do**
10: $t_n = receivedTime(n, TB_i)$
11: $t'_n = t_v + sendsDone + 1$
12: **if** $t_n > t'_n$ **then**
13: $TB_i \leftarrow TB_i - E(to(n))$
14: $TB_i \leftarrow TB_i + (v, n)$
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

Then, $DynamicTree$ (Algorithm 4) is applied in the following way. Firstly, in the block of lines 1–3, a tree is built by adding to an empty one the shortest paths from root node v'_1 to all the other nodes in the block. In order to find for each node the time in which the message would arrive, $BroadcastTree$ is used in line 4. After that, in the block of lines 5–20, for each node v and in sequential order, algorithm explores its idle capacity. That means, all unused communication channel that could potentially improve the overall broadcast time in the block. That is, neighbours of v in the block B_i $(N_{B_i}(v))$, which are not neighbours in the spanning tree TB_i $(N_{TB_i}(v))$. We call that set of nodes $N'(v)$.

Figure 6 Block – complete graph (see online version for colours)

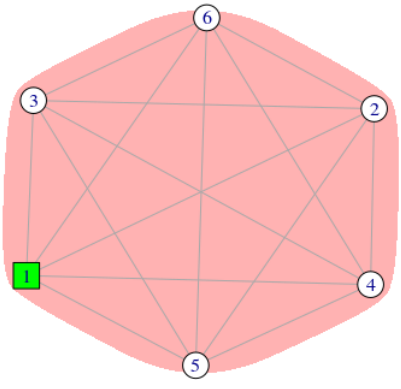
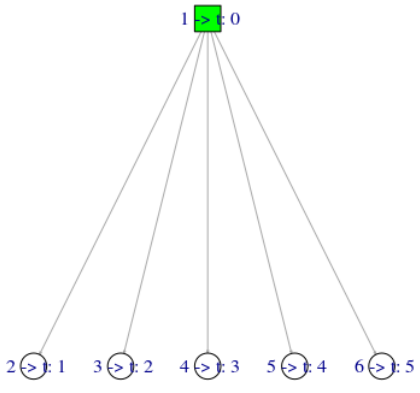


Figure 7 Dynamic tree: step 0 (see online version for colours)



Next, for each node n in $N'(v)$, if the time in which the message could be sent to n from v is less than the time in which it was originally sent, we proceed to alter the TB_i. That means that after v finishes sending the message in the original tree, it can take advantage of its capacity in the block by sending to n to reduce the times. Therefore, forwarding tree is alter as follows. Firstly, erasing the edge by which the message originally arrived at n (line 13). This represents that the message is not arriving

from the old node anymore. Secondly, adding the edge from v to n (line 14), because the new used channel is from v. It should be noted that the sub-tree of TB_i routed in n is maintained in the same way, since after the message arrives n, it forwards it as before. However, when the algorithm iterates over n, exploring its idle capacity, it has the opportunity to change it. Finally, the forwarding times are updated for the sub-tree from TB_i rooted at n (line 16) after the previous change. As a consequence, node v forwards the message to a different neighbour (line 17).

Figure 8 Dynamic tree: iteration 1 (see online version for colours)

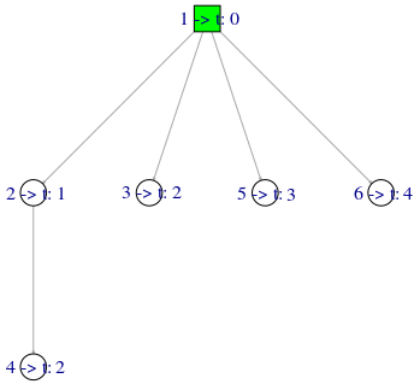
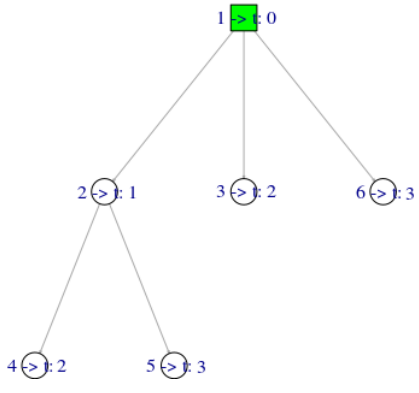


Figure 9 Dynamic tree: iteration 2 (see online version for colours)



This procedure is performed only once for each node belonging to the block B_i. Figures 7, 8, 9, 10, and 11 show each iteration from the $DynamicTree$ algorithm applied to the block shown in Figure 6.

Considering again $BroadcastBlock$ algorithm, two important features stand out: firstly, that the algorithm must return the forwarding tree for the nodes of the block and secondly, that within the block there are possibly other cut points, which means the possibility of going out to explore other blocks. As long as it is more profitable, it is better to explore new blocks before an internal forwarding scheme. Nevertheless, there are cases where this strategy is not optimal. This is the reason why the exploration is done recursively. An artificial procedure is included before the definition of a particular order. This artificial procedure is explained in the following.

Figure 10 Dynamic tree: iteration 3 (see online version for colours)

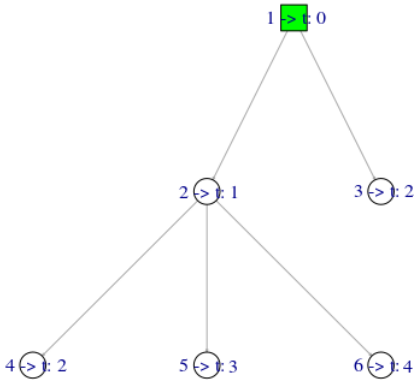
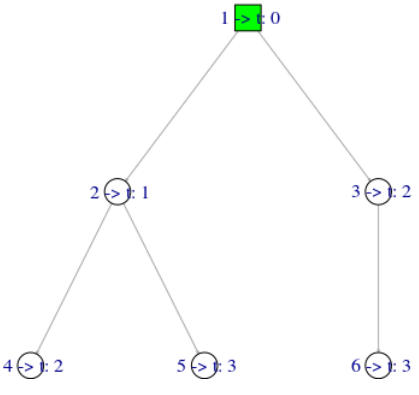


Figure 11 Dynamic tree: iteration 4 (see online version for colours)



For each cut-point belonging to the block, $BroadcastBlock$ is recursively applied to get the time-slots required to cover the blocks which are connected by these nodes. This is precisely the broadcast time for that direction. For each cut-point, an artificial node is added to the block from the cut point, and the corresponding link has the broadcast time F_j as its corresponding weight (lines 2–7). Finally, $BroadcastTree$ is applied into this edited tree in order to define an intra-block forwarding scheme (line 9). The result is returned in line 10.

Observe that if the block has no cut-point, this artificial procedure is not performed, and $BroadcastTree$ is applied on the tree found by $DynamicTree$.

Let us explain how $BroadcastTree$ works for the artificial nodes. In line 9 of the algorithm $BroadcastTree$, the label is found by adding the value i, which corresponds to the relative position in which each node was notified. For ordinary nodes, this is still the case. However, for the new artificial nodes, i does not mean just one, but the time required to notify each child of the cut-point in the block that is being represented. Therefore, i no longer corresponds to an incremental one, but its number in the block.

Combining the presented algorithms we can find the broadcast time of our proposal, and its corresponding forwarding tree.

5 Experimental analysis

In this section we evaluate the performance of our heuristic for the MBT. As we summarise in Section 2 of related work, the problem was previously studied in well-known topologies, in some cases, an exact solution is guaranteed and in other cases, a performant heuristic was developed. Therefore, in Subsection 5.1 we first compare our heuristic with the known results of the problem (exact or well-known heuristic) for these particular topologies.

As most of the previous work was done over small topologies, and there is not a dataset of large instances. Therefore, in Subsection 5.2 we analyse the performance of our heuristic over a large dataset of synthetic graphs. These dataset is created following the Watts-Strogatz small-world model (Watts and Strogatz, 1998), which is widely accepted as a generator of real-like communication networks (Watts, 2004).

In both subsections, whenever is possible, we compare the heuristic results with the optimal results provided by an ILP solver, using the formulation described in Section 3.

5.1 Particular topologies

Following previous studies, we present the results of our heuristic for five well-known topologies: *Lattice*, *DeBruijn*, *hypercube*, *Harary*, and *Chord Harary*. For each topology, Tables 1–5 show the optimal result according to the ILP formulation, and the $TreeBlock$ heuristic result (TB). The remaining columns specify the parameters of the graph instance (as the number of nodes, N). And, when the exact solution is unknown, the best heuristic result is shown.

As a particular Lattice topology, we study 2D grid graphs. In this case, a problem instance is defined by the number of rows R and 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 (Hedetniemi et al., 1988), when the source node is $(1, 1)$ then the broadcast time is precisely the diameter $D = R - 1 + C - 1 = R + C - 2$. Table 1 summarises the results for 16 random 2D grid instances. The heuristic $TreeBlock$ reaches the optimum result in all of them.

In Harary (1962), introduced the Harary family of graphs, noted $H_{k,n}$. The member $H_{k,n}$ is k-node connected with n nodes. The structure of $H_{k,n}$ depends on the parities of k and n. The MBT is achieved in Bhabak et al. (2014) for half of the cases. The results are mostly optimal in the instances studied with the $TreeBlock$ heuristic; in the other cases the gap is a single time-slot (see Table 2).

Chord-Harary graph is a variation of the Harary graph, $CH_{k,n}$, introduced in Bhabak et al. (2017). The authors proposed an algorithm where low values of broadcasting time can be reached for those graphs. Further, he globally optimum solution is met for a subset of these graphs, when k is a specific function of n. Table 3 shows that the results obtained by the $TreeBlock$ heuristic are still competitive, reaching the globally optimum solutions in some cases. The chosen instances did not take into account the relationship between the parameters for which the algorithm of Bhabak et al. (2017) achieves optimal precisely in order to observe how it behaves in general.

Finally, we compare results in another two families of graphs, DeBruijn networks (Klasing et al., 1994) and hypercubes (Liestman and Peters, 1988). We can observe from Tables 4 and 5 that our proposal achieves competitive results in comparison with

previously proposed heuristics: RH (Beier and Sibeyn, 2000), TBA (Harutyunyan and Shao, 2006) and NTBA (Harutyunyan and Wang, 2010).

Table 1 Comparison of optimal (ILP) and $TreeBlock$ (TB) heuristic results

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*

Notes: Random instances of the Lattice topology, where R is the number of rows, C is the number of columns, and (r, c) is the (row number, column number) of the source node.

Table 2 Comparison of optimal (ILP) and $TreeBlock$ (TB) heuristic results

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*

Notes: Random instances of the Harary topology, where N is the number of nodes, and k and n are the parameters of the family $H_{k,n}$.

Source: Bhabak et al. (2017)

Table 3 Comparison of optimal (ILP) and $TreeBlock$ (TB) heuristic results

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

Notes: Random instances of the Chord-Harary topology, where N is the number of nodes, and k and n are the parameters of the family $CH_{k,n}$.

Source: Bhabak et al. (2017)

Table 4 Comparison of optimal (ILP), $TreeBlock$ (TB) heuristic, and previous heuristics (RH, TBA, and NTBA) results

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
1,024	2	10	14	14	15	-	15
2,048	2	11	15	15	17	-	17
4,096	2	12	17	17	19	-	19
8,192	2	13	18	18	20	-	21

Notes: Random instances of the DeBruijn topology, where N is the number of nodes, and m and n are the parameters of the family.

Source: Harutyunyan and Wang (2010)

From the analysis of all these graph families, it is worth to remark that the ILP formulation achieves the globally optimum solution for most of the tested topologies (in the others, computation constraints limited the results). ILP could be computed in 71% of the 58 instances of Tables 1–4. On the other hand, our $TreeBlock$ heuristic shows competitive results even if the instance under study has multiple blocks, showing small gaps with respect to ILP and other previously developed heuristics. $TreeBlock$

shows a GAP lower than 10% with respect to ILP in 29 of the 41 evaluated instances of Tables 1–4.

Table 5 Comparison of $TreeBlock$ (TB) heuristic and previous heuristics (TBA and NTBA) results

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
1,024	2	10	10	12	11	15	10
2,048	2	11	11	-	12	18	11
4,096	2	12	12	-	13	20	12

Notes: Random instances of the hypercube topology, where N is the number of nodes, and l, dim, O and C are the parameters of the family.

Source: Harutyunyan and Wang (2010)

5.2 Large dataset of Watts-Strogatz graphs

In this Subsection, we present the results of our heuristic for a large dataset of large instances. The dataset is created with instances of small-world networks, with different number of nodes and with different levels of connectivity. The final objective of this subsection is to evaluate the performance of our heuristic in large and realistic graphs, where we measure the performance in terms of quality and execution time. For quality, we compare our heuristic result with the ILP solution.

5.2.1 About the dataset construction

Watts and Strogatz identified that many real networks have high levels of clustering, but small distances between most nodes. This is particularly true in communication networks which is the motivation of this study. In order to create a graph with both properties, Watts and Strogatz consider a Lattice structure, and then randomly 'rewire' a small percentage of its edges (with an independent probability p, avoiding loops and multi-edges). This model has high level of clustering and low average distances between nodes (Watts and Strogatz, 1998). The instance creation process is presented in Figure 12.

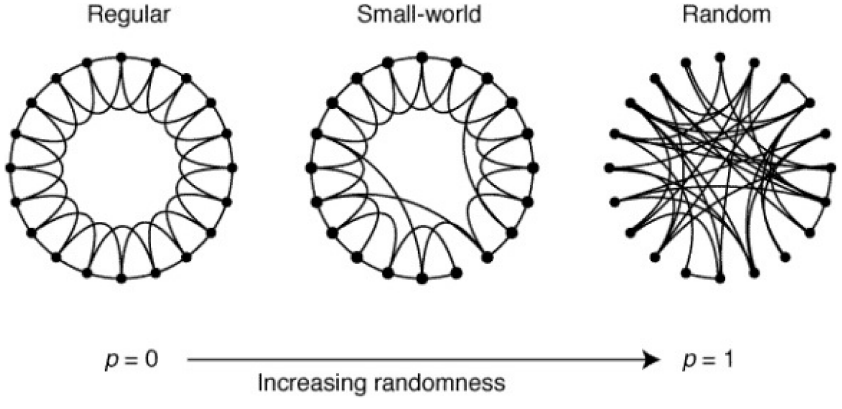
The dataset is made up of 301 instances of small world graphs of 10 nodes, and 100 instances of small world graphs for each of the following number of nodes: 50, 100, 300, 600 and 1,000. That is, the dataset has 801 instances. For all cases, the probability of rewiring p is drawn randomly and uniformly in [0, 1].

5.2.2 Execution time results

For each instance, both the constructed heuristic and the ILP solver were executed in order to make a comparison of the results. Regarding the CPU times, Figure 13 shows

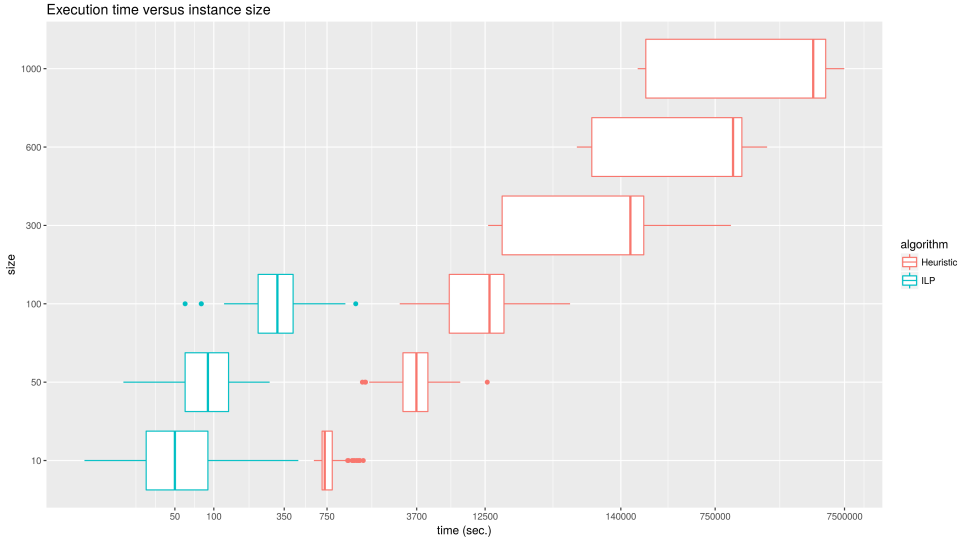
that the elapsed time is reached for the ILP solver for large instances. This is due to the hardness of the MBT. Another example of this is that only the graph instances of 10, 50 and 100 nodes could be solved using the ILP solver (in the others the execution was interrupted after one hour without result).

Figure 12 Small-world graph creation, using the Watts and Strogatz model



Source: Watts and Strogatz (1998)

Figure 13 Comparative: execution time versus instance size (see online version for colours)



As expected, CPU time for the heuristic, and ILP solver, is strongly related to the size of the instance. However, the rewiring probability p does not affect the CPU times (not shown in figure).

It is important to mention, that computation time can not be compared between the heuristic and the ILP solver in this experiment because ILP is executed on a cluster, while the heuristic is on home-PC.

5.2.3 Quality results

For the instances in which the ILP solver achieved an optimal result (graph instances of 10, 50 and 100 nodes), a comparison can be made in terms of the difference in value achieved by each algorithm. In Table 6 is shown for each group of instances, which is the range of optimal values found by the ILP solver and which by the heuristic. The 'GAP' column indicates what is the average difference between the solution achieved with the heuristic and the ILP exact solution. In addition, the 'no GAP' column indicates the number of instances for which the heuristic found the optimum. In Table 7 you can see in what percentage the results of the heuristic approach the optimum. The 10-size segment stands out, where 87% of the instances are resolved with a GAP lower than 10%. Considering all the instances, the GAP is lower than 25% for more than 90% of them. It is important to note that few instances are not included in the comparison because of execution constraints of ILP solver, and this is more relevant in instances of larger sizes (50 and 100). Finally, Figure 14 shows the distribution of the data in terms of size and probability of rewiring in comparison with the times and results. It shows a large dispersion in quality and computation time between instances of the same size and probability of rewiring p, especially in small instances.

Figure 14 CPU-time versus quality in the objective

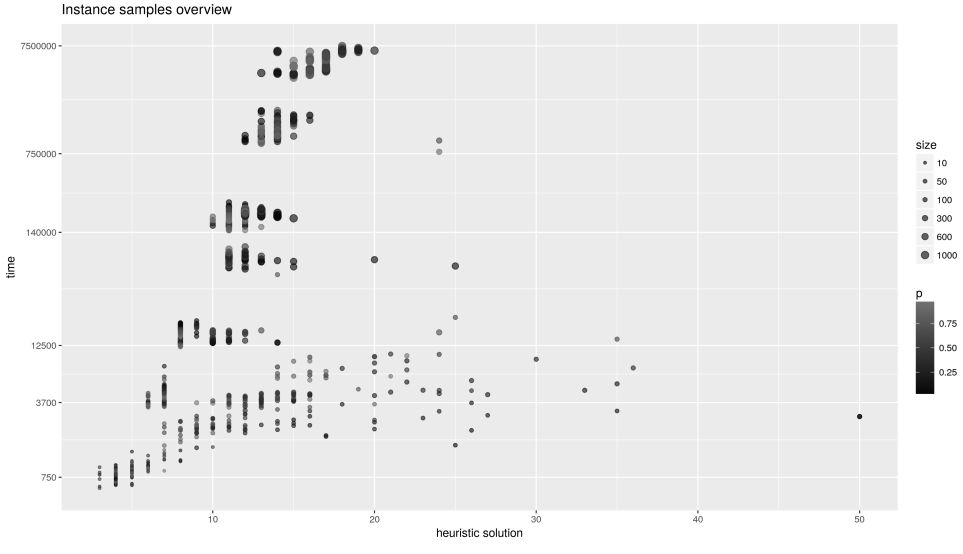


Table 6 Average ILP solver and heuristics results

	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	301
2	50	46.52	12.85 ± 5.33	11.51 ± 5.27	0.14 ± 0.17	33	100
3	100	94.04	16.9 ± 7.4	14.46 ± 7.13	0.19 ± 0.21	22	100

Note: It includes the average GAP between both results.

In summary, as expected, the problem MBT is complex, where the exact solution can be reached only for small instances, while the TB heuristic scales much better. We test

heuristic with several instances, and its solution shows large dispersion in quality and computation time. In cases where an exact solution is known, the heuristic shows results close to optimal.

Table 7 GAP comparison between ILP solver and heuristics results

	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

Note: For more than 90% of the instances, the GAP is lower than 25%.

6 Conclusions and trends for future work

In this work, the MBT is studied. The MBT belongs to the class of $\mathcal{NP}$-complete problems. As a consequence, the related literature offers exact solutions for specific families of graphs, heuristics and approximation algorithms. Here, a novel heuristic called *TreeBlock* is introduced. Furthermore, an ILP formulation for the MBT is presented. Given the hardness of the problem, the exact ILP shows its potential for small-sized instances. On the other hand, the heuristic aims to solve a larger universe of instances, and it yields reasonable CPU times for up to 1,000 nodes.

It is worth to mention that our heuristic is also competitive with previous proposals in terms of optimality. The GAP is small or null when the globally optimum solution is known. The GAP is lower than 25% in more than 90% of the instances under study.

We would like to determine in which graphs the exploration of new blocks is better than the optimum forwarding scheme for the internal nodes belonging to a specific block. This trade-off is not well understood yet. The capacity (i.e., degree) of the intermediate nodes should be also considered in a smarter heuristic. We expect to include larger instance for future work.

Acknowledgements

This work is partially supported by Project MATHAMSUD 19-MATH-03 *RareDep* and ANII FCE_1_2019_1_156693 *Teoría y Construcción de Redes de Máxima Confiabilidad*.

References

Anulova, S. (2017) *Fluid Limit for Switching Closed Queueing Network with Two Multi-servers*, pp.343–354, Springer International Publishing, Cham, ISBN: 978-3-319-66836-9.

Beier, R. and Sibeyn, J.F. (2000) *A Powerful Heuristic for Telephone Gossiping*, Forschungsberichte des Max-Planck-Instituts für Informatik, MPI Informatik, Bibliothek & Dokumentation.

Beier, R., Beier, R., Sibeyn, J.F. and Sibeyn, J.F. (2000) 'A powerful heuristic for telephone gossiping', *Proceedings of the International Colloquium on Structural Information and Communication Complexity*.

Bhabak, P., Harutyunyan, H.A. and Kropf, P. (2017) 'Efficient broadcasting algorithm in Harary-like networks', in *2017 46th International Conference on Parallel Processing Workshops (ICPPW)*, August, pp.162–170.

Bhabak, P., Harutyunyan, H.A. and Kropf, P. (2020) 'Analysis and performance of complete homogeneous communication networks (submitted)', in *Proceedings of the Sixth International Conference on Machine Learning, Optimization, and Data Science*.

Bhabak, P., Harutyunyan, H.A. and Tanna, S. (2014) 'Broadcasting in Harary-like graphs', *IEEE*, pp.1269–1276.

Chuang, Y-T. (2017) 'Protecting against malicious and selective forwarding attacks for P2P search & retrieval system', *Peer-to-Peer Networking and Applications*, July, Vol. 10, No. 4, pp.1079–1100.

Crescenzi, P., Fraigniaud, P., Halldórsson, M., Harutyunyan, H.A., Pierucci, C., Pietracaprina, A. and Pucci, G. (2016) 'On the complexity of the shortest-path broadcast problem', *Discrete Applied Mathematics*, Vol. 199, No. Supplement C, pp.101–109.

Edmonds, J. (1987) *Paths, Trees, and Flowers*, pp.361–379, Birkhäuser Boston, Boston, MA, ISBN: 978-0-8176-4842-8.

Elkin, M. and Kortsarz, G. (2005) 'A combinatorial logarithmic approximation algorithm for the directed telephone broadcast problem', *SIAM Journal on Computing*, Vol. 35, No. 3, pp.672–689.

Farley, A.M. (1980) 'Broadcast time in communication networks', *SIAM Journal on Applied Mathematics*, Vol. 39, No. 2, pp.385–390.

Fraigniaud, P. and Vial, S. (1997) 'Approximation algorithms for broadcasting and gossiping', *J. Parallel Distrib. Comput.*, May, Vol. 43, No. 1, pp.47–55, ISSN: 0743-7315.

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

Harary, F. (1962) 'The maximum connectivity of a graph', *Proceedings of the National Academy of Sciences of the United States of America*, Vol. 48, No. 7, pp.1142–1146.

Harutyunyan, H.A. and Kamali, S. (2017) 'Efficient broadcast trees for weighted vertices', *Discrete Applied Mathematics*, Vol. 216, No. Part 3, pp.598–608.

Harutyunyan, H.A. and Shao, B. (2006) 'An efficient heuristic for broadcasting in networks', *Journal of Parallel and Distributed Computing*, Vol. 66, No. 1, pp.68–76.

Harutyunyan, H.A. and Wang, W. (2010) 'Broadcasting algorithm via shortest paths', in *2010 IEEE 16th International Conference on Parallel and Distributed Systems*, pp.299–305.

Harutyunyan, H.A., Liestman, A.L., Peters, J.G. and Richards, D. (2013) 'Broadcasting and gossiping', in Gross, J.L., Yellen, J. and Zhang, P. (Eds.): *Handbook of Graph Theory, Second Edition*, Chapter 12, 2nd ed., p.18, Chapman & Hall/CRC, London, UK.

Hedetniemi, S.M., Hedetniemi, S.T. and Liestman, A.L. (1988) 'A survey of gossiping and broadcasting in communication networks', *Networks*, Vol. 18, No. 4, pp.319–349.

Klasing, R., Monien, B., Peine, R. and Stöhr, E.A. (1994) 'Broadcasting in butterfly and DeBruijn networks', *Discrete Applied Mathematics*, Vol. 53, No. 1, pp.183–197.

Kortsarz, G. and Peleg, D. (19995) 'Approximation algorithms for minimum-time broadcast', *SIAM Journal on Discrete Mathematics*, Vol. 8, No. 3, pp.401–427.

Liestman, A.L. and Peters, J.G. (1998) 'Broadcast networks of bounded degree', *SIAM Journal on Discrete Mathematics*, Vol. 1, No. 4, pp.531–540.

Liu, W., Peng, D., Lin, C., Chen, Z. and Song, J. (2010) 'Enhancing tit-for-tat for incentive in bittorrent networks', *Peer-to-Peer Networking and Applications*, March, Vol. 3, No. 1, pp.27–35.

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

Pazzi, R.W., Boukerche, A., Grande, R.E. and Mokdad, L. (2017) 'A clustered trail-based data dissemination protocol for improving the lifetime of duty cycle enabled wireless sensor networks', *Wirel. Netw.*, January, Vol. 23, No. 1, pp.177–192.

Peleg, D. (2007) *Time-Efficient Broadcasting in Radio Networks: A Review*, pp.1–18, Springer, Berlin, Heidelberg.

Romero, P. (2012) *Mathematical Analysis of Scheduling Policies in Peer-to-Peer Video Streaming Networks*, PhD thesis, November, Universidad de la República, Montevideo, Uruguay.

Schindelhauer, C. (2000) 'On the inapproximability of broadcasting time', in *Proceedings of the Third International Workshop on Approximation Algorithms for Combinatorial Optimization*, pp.226–237, London, UK.

Slater, P.J., Cockayne, E.J. and Hedetniemi, S.T. (1981) 'Information dissemination in trees', *SIAM Journal on Computing*, Vol. 10, No. 4, pp.692–701.

Watts, D.J. (2004) *Six Degrees: The Science of a Connected Age*, 1st ed., W.W. Norton and Company, Inc.

Watts, D.J. and Strogatz, S.H. (1998) 'Collective dynamics of 'small-world' networks', *Nature*, June, Vol. 393, No. 6684, pp.440–442, DOI: 10.1038/30918.

West, D.B. (2001) *Introduction to Graph Theory*, Featured Titles for Graph Theory Series, Prentice Hall, ISBN: 9780130144003.

Yang, X. and de Veciana, G. (2004) 'Service capacity of peer to peer networks', in *Twenty-Third Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'04)*, Vol. 4, pp.2242–2252.