An Iterative Algorithm for the Minimum Broadcast Time Problem

Conference Paper · November 2004

CITATIONS
2

READS
25

2 authors, including:

Calin D Morosan
INRO
14 PUBLICATIONS **49** CITATIONS

SEE PROFILE

AN ITERATIVE ALGORITHM FOR THE MINIMUM BROADCAST TIME PROBLEM

Hovhannes A. Harutyunyan
Department of Computer Science
Concordia University
1455 de Maisonneuve Blvd. West
Montreal, QC, H3G1M8, Canada
email: haruty@cs.concordia.ca

Calin D. Morosan
Department of Computer Science
Concordia University
1455 de Maisonneuve Blvd. West
Montreal, QC, H3G1M8, Canada
email: cd_moros@cs.concordia.ca

ABSTRACT
Broadcasting is the problem of dissemination of information in which one piece of information needs to be transmitted to a group of individuals connected by an interconnection network. Finding an optimum strategy for broadcasting, such that this process is accomplished in minimum time, has been proven to be NP-complete, for an arbitrary network. Therefore, some heuristic algorithms have been developed for finding the minimum broadcast time of a network, usually modelled as a simple graph.

To our knowledge, the best known heuristic works in $O(Rnm)$, for the entire graph, where R is the approximate broadcasting time, n is the number of vertices, and m is the number of edges. In this paper we propose a global, iterative heuristic for finding the minimum broadcast time for the entire graph, being the first algorithm of this kind in the area. The algorithm uses a new interesting property of the optimal broadcast schemes that we have found.

The algorithm behaves well in practice, especially for strongly irregular topologies, working in $O(knm)$ time and $O(n^2)$ space, where k is the number of iterations, n the number of vertices, and m the number of edges.

KEY WORDS
Networks, minimum broadcast time, iterative algorithms.

1 Introduction

Broadcasting is the problem of dissemination of information in an interconnection network in which, one member, called originator, knows a piece of information and has to transmit it to all other members of the network. This problem finds applications in parallel and distributed computing, Internet, rumours and virus spreading, etc.

Some models have been proposed and analysed for broadcasting. In this paper we will consider the 1-*port*, *constant time* model. That is, once a member of the network knows the information, he can transmit it only to one neighbour at a time, and the transmitting time is constant.

We will model the interconnected network by a simple undirected connected graph G, in which the members of the network will be the vertices of G and the communication lines, the edges of G.

Given a graph G and an originator v, the broadcast time of v, $b(v)$, will be the minimum time necessary to inform all the vertices of G, starting from v. The broadcast time of the graph G will be:

$$b(G) = \max\{b(v) | v \in G\} \quad (1)$$

We define the average broadcast time of a graph G:

$$\overline{b(G)} = \frac{1}{n}\sum_{v=1}^{n} b(v) \quad (2)$$

The minimum broadcast time problem consists in finding a strategy, for a given graph G and an originator v, such that the broadcast time of v is minimum. This problem has been proved to be NP complete in [1] by reduction to another NP-complete problem: the three-dimension matching (3DM) [2]. Good surveys on broadcasting can be found in [3] and [4].

Due to the NP-completeness, some heuristic algorithms have been developed, in order to give an approximate solution for this problem ([5], [6], [7], [8], [9], [10], [11]).

There are two main criteria to compare between these heuristics: the complexity of the algorithm and the "optimality" of the solution. Since the first criterion is relatively easy to compute, the second one raises some questions, especially for random generated graphs, for which an optimal solution is generally not known a priory. There are three approaches to overcome this problem: to obtain theoretical results regarding the inapproximability of the problem, to run the algorithm on graphs with known broadcast time, mainly regular ones, or to do benchmarks on random graphs, generated using common generators. Unfortunately, only in [11] such benchmarks are provided, for different random graph topologies.

To our knowledge, the best result regarding the inapproximability of the minimum broadcast time problem is due to Schindelhauer, who showed in [12] that the broadcast time cannot be approximated within a factor of $\frac{57}{56} - \varepsilon$, for any $\varepsilon > 0$.

Most of the designed heuristics are based on graph matching techniques, yielding a relatively high complexity, fact that can be a serious impediment, especially for large

networks. The best known complexity is $O(Rnm)$, for the entire graph [11], where R is the approximate broadcasting time, n is the number of vertices, and m is the number of edges. In this paper we describe an algorithm which works in $O(knm)$ time for the entire graph, where k is the number of iterations, n is the number of vertices, and m is the number of edges.

The algorithm is iterative and global, being the first algorithm of this kind in the area. It is iterative in the sense that, at each iteration, it attempts to find a better broadcast scheme than the previous known one. It is global in the sense that it gives the broadcast schemes for all the vertices of the graph and not only for one vertex. We have experimentally found that during the first $\log n$ iterations, the approximate solution has the greatest rate of convergence, which yields to a $O(nm \log n)$ average complexity, for the entire graph. We have extensively tested this algorithm on a wide range of graph topologies and we present here the results, which are similar to those from [11].

The paper is organized as follows: section 2 presents the algorithm, section 3 analyses the algorithm, section 4 presents the experimental results, and section 5 draws the conclusions.

2 Algorithm description

We use the adjacency lists as a data structure for keeping G in memory, each list being organized as a double-linked hash table of length n. To each vertex $v \in G$ we attach a label, corresponding to its known broadcast time. Since, at the beginning, there is no broadcast time available, we assign to all vertices a broadcast time $b(v) = n$, the number of vertices, which is clearly an upper bound for $b(v)$. Also, for each adjacency list, denoted by $adj(v)$, we keep a pointer to the neighbour with the smallest label.

The algorithm repeatedly parses all the vertices from G, considering each of them at a time as originators. The number of iterations will correspond to how many times we parse the whole set of vertices of G.

Once an originator is chosen, say v, we construct the adjacency lists for all vertices, picking up the neighbours in random order and inserting them in the hash tables according to their broadcast time. Then we include v in the set S of informed vertices and we start to broadcast the message from the vertices belonging to S to the rest of vertices from $G - S$. The politic of choosing one neighbour to announce by a vertex, say u, is: at each time an informed vertex will call its uninformed neighbour, say w, with the smallest known broadcast time corresponding to its label, $b(w)$. If more than one vertex has the smallest label, the algorithm will pick up the first one. The edge used for this call will be deleted from both adjacency lists: $adj(u)$ and $adj(w)$. The new informed vertices will be included in *NI*, the set of new informed vertices after a round of calls, only if their adjacency list is not empty. If $adj(u)$ is also empty, we delete u from S. When all the vertices from S have made a call, we increment the broadcast time by one and we attach the set *NI* to the set S, setting $NI = \emptyset$.

This process stops when all the vertices from G are informed. In this moment, we have a new broadcast scheme available for the vertex v, and a new broadcast time. If this new broadcast time is smaller then the previous known one, we modify accordingly its label $b(v)$, and we continue by choosing the next vertex as originator. One iteration is considered finished when we have considered all the vertices from G as originators.

We denote by *MAX_ITER*, the number of desired iterations. This is considered as a free parameter for our algorithm and will control the "optimality" of the resulted broadcast schemes. This can be also automatically set to be the number of iterations after which there are no changes in the broadcast time of any of the vertices form G. As you will see in the next section, this approach can be tricky since the rate of convergence of the algorithm is rapidly decreasing at the beginning but can be stationary for a long time after a certain number of iterations, until decreases again.

Formally, the algorithm can be written as follows:

for i = 0 **to** *MAX_ITER*
  for each $v \in G$
    build the adjacency lists of all vertices from G
    $S = \{v\}$
    informed_vertices = 1
    b_time = 0; $NI = \emptyset$
    while informed_vertices $< |G|$
      for each $u \in S$
        w = neighbor of u with $b(w) = \min$.
        delete u from adjacency list of w
        delete w from adjacency list of u
        if $adj(w)$ is not empty
          $NI = NI \cup \{w\}$
        if $adj(u)$ is empty
          $S = S - \{u\}$
        informed_vertices = informed_vertices + 1
      end for
      b_time = b_time + 1
      $S = S \cup NI$
    end while
    if $b(v) >$ b_time
      $b(v) =$ b_time
  end for
end for

3 Algorithm analysis

The crux of the algorithm is the way of choosing the next neighbor to inform: the one with the minimum known broadcast time, which corresponds to its label. This idea was coming during the study of optimal broadcast schemes in simple graphs such as paths, trees, etc. Particularly for trees, we proved somewhere else [13] that given a tree T, an originator v, and a label attached to each vertex corresponding to its minimum broadcast time, during an optimal

broadcast process starting from v, an informed vertex will always call one uniformed neighbor with the smallest label. Indeed, let us denote by u an informed vertex at time t, and by w and z two uninformed neighbors of u, such that $b(w) > b(z)$. We will denote by T_w the subtree induced by w and by T_z the subtree induced by z, both in an optimal broadcast scheme having v as originator. We denote by T_u, the subtree rooted in u remaining after deleting T_w and T_z from T (Figure 1). Also we denote by:

- $b(w \rightarrow T_w)$ the broadcast time needed for w to broadcast in the subtree T_w
- $b(z \rightarrow T_z)$ the broadcast time needed for z to broadcast in the subtree T_z
- $b(u \rightarrow T_u)$ the broadcast time needed for u to broadcast in the subtree T_u

It can be shown that $b(w) > b(z)$ implies:

$$b(w \rightarrow T_w) < b(z \rightarrow T_z) \tag{3}$$

Equation (3) shows that that u must call z before w in order for a broadcast scheme to be optimal.

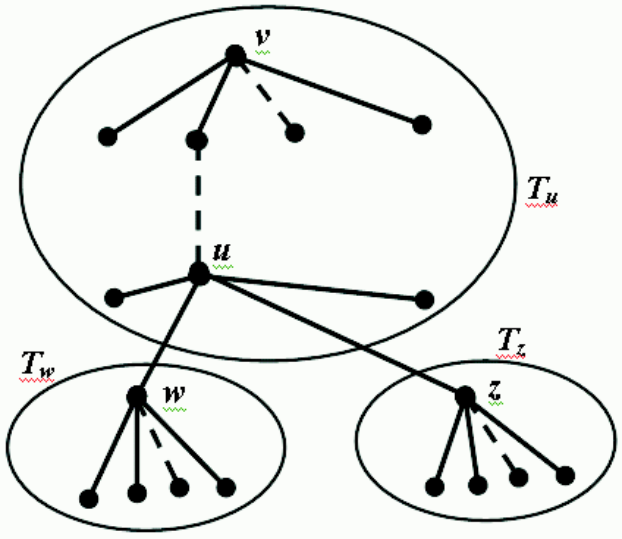

Figure 1. The tree T and the subtrees generated by u, w, and z

In order to analyse the time-complexity of the algorithm we have to analyse the following operations:

1. The outer **for** loop represents the number of iterations, *MAX_ITER = k*. This is a free parameter for the algorithm and is controlling the "solution's optimality". Even there is still no theoretical support for the necessary number of iterations in order to exactly converge to the optimal solution, experimental results suggest that during the first $\log n$ iterations, the convergence rate is the highest, and the solution is near optimal (see section 4, Figure 2). Nevertheless, for some graphs, as full binary trees, or binomial trees, the algorithm was able to find the optimal solution after only one iteration. This number also relays on the starting vertex and the parsing order but, for large graphs and bigger number of iteration, this dependency is very weak.
2. The middle **for** loop, which gives n operations, corresponding to the number of vertices.
3. The adjacency lists building operation, which takes $O(m)$ time, where m is the number of edges from G.
4. The combination between the inner **for** and **while** loops, corresponding to the total number of calls, which is n.
5. The number of operations necessary to retrieve the neighbor with the smallest label. This can be done in $O(1)$ by keeping a hash table and a pointer to the first non-empty entry in this table.
6. The number of operations necessary to delete the vertices u and t from the adjacency lists. This can be also done in $O(1)$ by keeping an an n^2 array with the vertices addresses in the adjacency lists.

From the above analysis, the driving operations for complexity are in the steps 1), 2), and 3), yielding an $O(kmn)$ time complexity.

One could expect that the complexity will increase by increasing the vertices degree in graph. Nevertheless, experimental results show that, once the graph density increases and the graph tends to the complete graph, the number of necessary iterations needed decreases. For the complete graph or for trees there is only one iteration needed. The experimental results show that the graphs with average density require a greater number of iterations than those with densities tending to either 0 or 1.

We are using the adjacency list as data structure for keeping the graph in memory, yielding an $O(n^2)$ space complexity.

Since, at the beginning, all the vertices have the same label, there is no difference between a random algorithm (see [14]) and this one, for the first originator. Once this originator has a smaller broadcast time assigned, during the rest of calls there will be already an available choice to make. Even this "warm up" part of the algorithm seems to be somehow "uncontrollable", the experimental results show that the value of labels decreases from the very beginning with a very high rate (Figure 2).

4 Experimental results

As we mentioned in introduction, there are two main types of experiments, characteristic to such heuristics: to run them on graphs with known broadcast time, or to do benchmarks on random graphs, generated using common generators. We studied the algorithm behavior on both types:

- Graphs with theoretically known broadcast time: full binary trees *bT*, binomial trees *BT*, hypercubes on dimension d (H_d), Knödel graphs on 2^d vertices and dimension d ($W_{d,2^d}$), cube-connected-cycles (*CCC*), de Brujin graphs (*HB*), butterfly (*BF*), and shuffle exchange (*SE*), all of them up to dimension 10.
- Two types of random graphs: GT-ITM pure random and GT-ITM transit-stub random graphs [15].

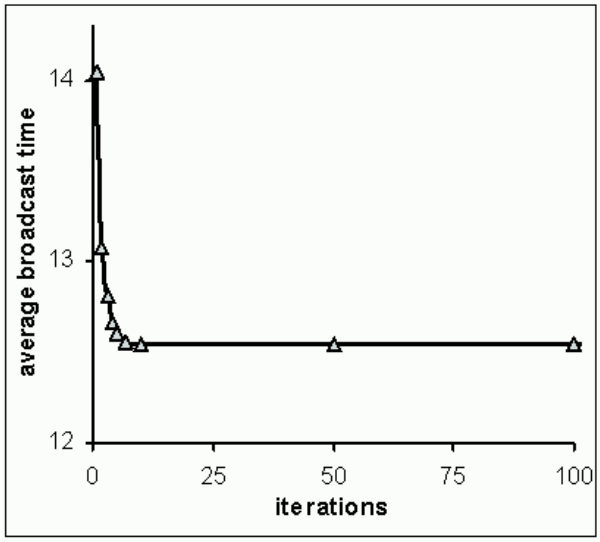

Figure 2. The average broadcast time function of the number of iteration steps for a transit-stub random generated graph on 100 vertices. The majority of vertices got their broadcast time approximation after the first 6 steps.

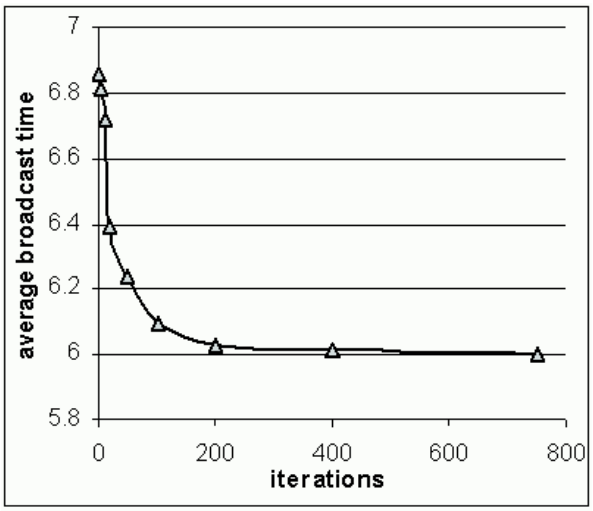

Figure 3. The average broadcast time function of number of iteration steps for the hypercube on dimension 6.

Two type of tests were done regarding the "solution's optimality": what is the average broadcast time after $\log n$ iterations for the whole graph and what is the number of iterations needed in order to obtain the optimal solution for the graphs, where this is known (hypercubes and Knödel graphs) and where this number is achievable in a reasonable amount of time.

Experimentally, we observe a rapid decreasing of the average broadcast time during the first $\log n$ iterations (Figure 2). More than that, for example, for the hypercube in dimension 6, the solution converges to the optimal broadcast scheme after 750 iterations (Figure 3).

The tests on regular graphs are not very relevant for this algorithm since the main idea of choosing the neighbour with the minimum broadcast time is not exploited due to the uniform distribution of $b(v)$ function among graph's vertices. From this point of view, once all the vertices get almost the same label, there is no big difference between this algorithm and a random one. This can be also seen comparing the figures 2 and 3. We observe that the curve from Figure 2 decreases more sharply than the one from Figure 3.

The algorithm is really powerful on the irregular networks, with a significant irregularity of the $b(v)$ function. This was the reason of testing this algorithm on transit-stub random networks, which are more "real", miming the cluster-type networks encountered in the real world (table 1). Also this is the reason of the high performance of this algorithm on trees.

In the tables 2, 3, and 4 we present the testing results for some regular networks, considering the average broadcast time, as it is defined in (2), for the whole graph, after $\log n$ iterations.

Table 1. The average broadcast time after $\log n$ iterations, for pure random graphs and transit-stub random graphs

type	**vertices *n***	**iterations log *n***	**average b(G)**
pure-random	100	7	10.10
transit-stub	100	7	12.06
transit-stub	600	10	18.77

Table 2. The average broadcast time after $\log n$ iterations, for hypercubes in dimensions 3 to 10

dimension	**iterations**	**average b(G)**
3	3	3
4	4	4
5	5	5.1
6	6	6.7
7	7	8
8	8	9
9	9	10.4
10	10	11.5

Table 3. The average broadcast time after d iterations, for Knödel graphs on 2^d vertices and degree d

dimension	iterations	average b(G)
3	3	3
4	4	4.1
5	5	5.9
6	6	7
7	7	8
8	8	9
9	9	10
10	10	11

Table 4. The average broadcast time after d iterations for shuffle-exchange graphs on 2^d vertices

dimension	iterations	average b(G)
3	3	4.12
4	4	6.12
5	5	8.12
6	6	10.07
7	7	12.07
8	8	14.29
9	9	16.37
10	10	18.75

5 Conclusions

In this paper we present a new heuristic algorithm for finding the minimum broadcast time, being the first global and iterative algorithm in this area. The algorithm is based on a new property of optimal broadcast schemes that we have found studying known broadcast schemes in simple graphs.

The algorithms works in $O(knm)$ time and $O(n^2)$ space and the testing results show a good behaviour in practice, especially for graphs with strong irregularities. Also, testing results show a continuous self improvement of the solution and a great rate of convergence during the first $\log n$ iterations.

6 Acknowledgements

We thank Bin Shao for his valuable help during the testing phase.

References

[1] P.J. Slater, E.J. Cockayne, & S.T. Heditniemi, Information dissemination in trees. *SIAM J.Comput. 10*(4), 1981, 692-701.

[2] M. R. Garey & D. S. Johnson, *Computers and intractability: a guide to the theory of NP-completeness* (W. F. Freeman, San Francisco, 1979).

[3] S.M Hedetniemi, S.T. Hedetniemi & A.L. Liestman, A survey of gossiping and broadcasting in communication networks. *Networks*, *18*, 1988, 319-349.

[4] J. Hromkovicbreve, R. Klasing, B. Monien, & R. Peine, Dissemination of information in interconnection networks (broadcasting and gossiping), D.-Z. Du, F. Hsu (Eds.), *Combinatorial network theory* (Kluwer Academic Publishers, Dordrecht), 1995, 125-212.

[5] P. Scheuermann and G. Wu, Heuristic algorithms for broadcasting in point-to-point computer network. *IEEE Transactions on Computers*, *C-33*(9), 1984, 804-811.

[6] R. Ravi, Rapid rumor ramification: approximating the minimum broadcast time. 35^{th} *Symposium on Foundation of Computer Science*, Montreal, QC, Canada, 1994, 202-213.

[7] Guy Kortsarz & David Peleg, Approximation algorithms for minimum-time broadcast, *SIAM Journal on Discrete Mathematics*, *8*(3), 1995, 401-427.

[8] Cory J. Hoelting, Dale A. Schoenefeld, Roger L. Wainwright, A genetic algorithm for the minimum broadcast time problem using a global precedence vector, *Proc. of the 1996 ACM symposium on Applied Computing*, Philadelphia, Pennsylvania, USA, 1996, 258-262.

[9] P. Fraigniaud & S. Vial, Approximation algorithms for broadcasting and gossiping, *Journal of Parallel and Distributed Computing*, *43*(1), 1997, 47-55.

[10] R. Beier & J. F. Sibeyn, A Powerful Heuristic for Telephone Gossiping. *The 7th International Colloquium on Structural Information & Communication Complexity (SIROCCO 2000)*, L'Aquila, Italy, 2000, 17-36.

[11] H. A. Harutyunyan & B. Shao. A heuristic for broadcasting. *International Conference Communications and Computer Networks (CCN)'2002*, Cambridge, MA, USA, 2002, 360-365.

[12] C. Schindelhauer, Broadcasting time cannot be approximated within a factor of 57/56-epsilon, *ICSI Technical Report TR-00-002*, 2002.

[13] H. A. Harutyunyan & C.D. Morosan, On a property of optimal broadcast schemes, *manuscript*.

[14] U. Feige, D. Peleg, P. Raghavan & E. Upfal, Randomized broadcast in networks, *Random Struct. Algorithms*, *1*(4), 1990, 447-460.

[15] E. W. Zegura, K. Calvert, and S. Bhattacharjee, How to model an internetwork, *IEEE INFOCOM*, San Francisco, CA, USA, 1996.