2008 14th IEEE International Conference on Parallel and Distributed Systems

Efficient Broadcasting in Networks with Weighted nodes

Hovhannes A. Harutyunyan and Shahin Kamali
Department of Computer Science
Concordia University
Montreal, PQ, H3G 1M8
Canada

September 8, 2008

Abstract: *In this paper the problem of broadcasting in networks with weighted nodes is considered. This model is defined on networks in which nodes are assigned some weights representing the internal process or delay that they should perform before sending data to their neighboring nodes. This model has real world applications in parallel computation and satellite terrestrial networks. The problem is shown to be NP-complete. In this paper we present three algorithms to find near optimal broadcast time of an arbitrary network with weighted nodes. We also present our simulation results to compare the performance of these algorithms.*

Keywords: broadcasting, weighted-vertex graphs, networks, heuristics, simulations

1 Weighted-Vertex Model

The classical model of broadcasting models a network by an n-vertex graph $G = (V, E)$. The communication in the network is assumed to be synchronous, i.e., occurs in discrete rounds, and in every round every processor is allowed to pick one of its neighbors, and send it a message. There has been plenty of research on the classical model during the last decades. To review some of the existing results, see [10, 8, 7].

The classical model of broadcasting does not always fit the requirements of real-world situations. This is the reason that several other models are presented each having its own properties (see for example [3, 6, 2]).

Here we enhance the classical model to the graphs with weighted vertices resulting a new model called weighted-vertex model. The weights assigned to the vertices may have different meanings. Each node may represent a smaller network such that the assigned weight be an upper bound for the broadcast time of that network. In a parallel machine, the weights assigned to processors can be interpreted as a kind of internal process each processor needs to perform before starting informing its neighbors (sometimes we regard the weights of the nodes as *internal process* or *internal delay*). The weights can also be considered as the delay for receiving the message due to input device limitations.

Another application of weighted-vertex model is in Satellite-Terrestrial (ST) networks. These are networks in which a large number of nodes are interconnected by both satellite and terrestrial networks [1]. ST networks can effectively support multicast services. However the performance of these networks is not high when they are modeled by regular unweighted graphs. The reason is that the cost of satellite nodes is more than terrestrial nodes. To resolve this problem, ST networks has been modeled by weighted-vertex graphs in which the vertices representing satellite nodes have positive integer weights, while the vertices representing terrestrial nodes have 0 as their costs. The weight assigned to the satellite nodes is the sum of the costs of up-links and down-links connecting them to the terrestrial network. By including these vertices in the broadcast tree, their costs are included in the costs of trees [1, 12].

In the rest of this paper, whenever we use term weighted graphs, we mean graphs with weights on the vertices (and not on the edges).

Let $G = (V, E)$ be an undirected graph where V is a set of n nodes and E is the set of communication links and let $u \in V$ be the originator node. We associate with each node $v \in V$ a non-negative integer weight $w(v)$. These weights represent the delay of a node to perform some process after receiving the message. Suppose that node v *'receives'* the message at time t from one of its neighbors. After receiving the message, v should wait for $w(v)$ rounds and then start informing its uninformed neighbors. This is the time in which v *'completes'* handling its delay. In this way, the first neighbor of v receives the message at time unit $t(v) + w(v) + 1$.

It is not hard to verify that weighted-vertex model is

1521-9097/08 $25.00 © 2008 IEEE
DOI 10.1109/ICPADS.2008.27

a novel model which differs from all previously studied models [11]. Note that the weights assigned to the nodes are always considered to be non-negative integers. When the weights of all vertices are 0, the model is equivalent to the classical model.

In weighted graphs, similar to unweighted graphs, any broadcast scheme can be represented by a spanning tree which is rooted at the originator. As shown in [11], there is an effective algorithm for broadcasting in trees under weighted-vertex model. This algorithm enables us to represent broadcast schemes with spanning trees.

Note that in the broadcast tree of an instance (u, G) of broadcast problem, the root (originator) and the leaves have some weights that are considered as a part of the broadcast time. So when a leaf v receives the message, the broadcasting process is not complete before an additional $w(v)$ time units.

Consider Figure 1 in which a broadcast tree is depicted and node A is the originator. In this scheme A receives the message at time $t = 0$. It is busy during next 5 rounds until it completes its delay at time $t = 5$. Then A informs B at time $t = 6$, C at time $t = 7$, D at time $t = 8$ and E at time $t = 9$. The same way B is busy in time period $[6, 8]$. Afterward, F receives the message through B at time $t = 9$ and completes after 3 rounds delay when it sends the message to K. The broadcasting completes when K completes its internal delay at time $t = 18$.

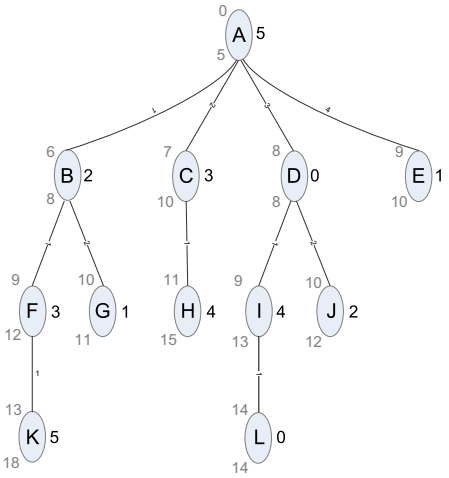


Figure 1: A broadcast tree rooted at originator A. The numbers in the right side of nodes indicate their weights; while the numbers in left indicate the times in which nodes receive the message (top) and complete their delay (bottom).

Definition 1. *A broadcast (multicast) scheme is a* non-lazy *scheme iff any vertex v after receiving the message and completing its internal delay, in all rounds before completing broadcast informs one of its uninformed neighbors.*

It is easy to see that for any broadcast scheme, there is an equivalent non-lazy scheme with lower or equal broadcast time. From now whenever we talk about a scheme, we mean a non-lazy scheme.

Note that in many concepts the weighted-vertex model is similar to classical model of broadcasting:

Fact 1. *In the weighted-vertex model of broadcasting:*

- *The communication mode is two-way, in each round a link can be used by both of its endpoints to send data through it.*
- *Message transition pattern is 1-port (whispering), in each round a completed vertex can send data to at most one of its neighbors.*
- *The model can be considered as a sample of constant model in the sense that the time needed to transmit a message through a link is the constant 1; although the time needed to handle the message varies for different nodes.*

Note that the classical model is also a two-way, 1-port and constant model. This is the reason that sometimes instead of using the term 'weighted-vertex model of broadcasting' we use 'broadcasting in graphs with weighted vertices'.

2 Heuristic Algorithms for Weighted Vertex Broadcasting

We present three heuristic algorithms for the broadcast problem in weighted vertex graphs. In Section 2.1, a modified version of Dijkstra's algorithm is presented which creates weighted shortest path trees as potential good answers to the problem. In Section 2.2, a greedy algorithm is described; while in Section 2.3, an evolutionary algorithm is proposed for the problem. In Section 3, the performance of these methods is compared using computer simulation. It should be mentioned that in designing these algorithms we used the idea behind the algorithms presented in [4]. There, the authors were interested in heuristics for broadcasting in complete graphs under generalized postal model.

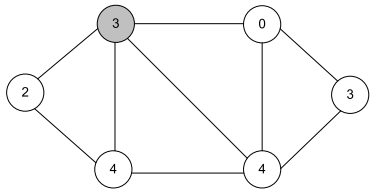
2.1 Modified Dijkstra's Algorithm

Dijktra's algorithm solves the single-source shortest-path problem in an arbitrary graph. A version of this algorithm for creating shortest path trees in weighted-vertex graphs is studied in [5]. The cost of a path between two

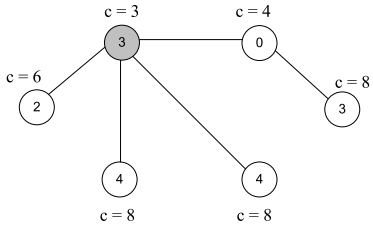
vertices is considered to be the length of the path plus the total weights of the vertices on that path. Note that originator has a cost as well, which is equal to its weight. Figure 2(a) shows a weighted vertex graph; and Figure2(b) denotes the shortest path tree created by Dijkstra's algorithm. Note that $c(v)$ denotes the cost of a node v in the shortest path tree.

Here we apply Dijkstra's algorithm to solve the broadcast problem in weighted-vertex graphs. Having originator u as the source of the tree, the shortest path tree rooted at u can be assumed to be a good solution to the problem.

The algorithm is the same as classical Dijkstra's algorithm. The only difference is that while adding a node v to the output tree, the weight of v is also added to the cost of v. Algorithm 1 is a pseudocode for the weighted vertex Dijkstra's algorithm. To present the output tree, the algorithm determines the parent of each node (except originator) in the tree.



(a) A weighted weight graph



(b) The output tree of Dijkstra's algorithm

Figure 2: Applying the modified Dijkstra's algorithm for the weighted-vertex broadcasting. The dark node is the originator.

In [5], it is proved that, for any weighted vertex graph $G(V, E)$, the time complexity of Dijkstra's algorithm, is $\mathcal{O}(|E| + |V| \log |V|)$.

2.2 The Greedy Algorithm

In Dijkstra's algorithm, the cost of a node remains constant once it has been set, ignoring the fact that sending message to each neighbor requires a separate unit of time. Here we present a greedy algorithm in which this issue is considered. The algorithm is based on the modified Dijkstra's algorithm. Let I represents the set

Algorithm 1 Modified Dijkstra's Algorithm

Input: A weighted graph $G = (V, E)$, the originator $u \in V$

Output: The shortest path tree from u as a scheme for broadcasting from u in G.

1: **for** any vertex $v_i \in V$ **do**
2: $c(v_i) \leftarrow \infty$
3: $p(v_i) \leftarrow NULL$
4: **end for**
5: $I \leftarrow \{u\}$ {the set of Informed vertices}
6: $c(u) \leftarrow w(u)$
7: $U \leftarrow V - \{u\}$ {the set of Uninformed vertices}
8: **while** $U \neq \phi$ **do**
9: Find $v_\alpha \in I, v_\beta \in U$, $(v_\alpha, v_\beta) \in E$ such that $c(v_\alpha) + w(v_\beta) = \min(c(v_i) + w(v_j))$ $[v_i \in I, v_j \in U, (i, j) \in E]$
10: $p(v_\beta) \leftarrow v_\alpha$
11: $c(v_\beta) = c(v_\alpha) + w(v_\beta) + 1$
12: $I \leftarrow I \cup \{v_\beta\}$
13: $U \leftarrow U - \{v_\beta\}$
14: **end while**
15: **return** $p(v_0), p(v_1), \cdots p(v_n)$

of informed vertices in each iteration of the algorithm, while U is the set of uninformed vertices. When a link (v, w) $[v \in I, w \in U]$ is attached to the broadcast tree, after setting the cost of w, the cost of v increases one unit. As a result, an extra cost would be applied in the cost of any other neighbor of v which is going to be informed in future iterations. Algorithm 2 is a pseudocode for this approach.

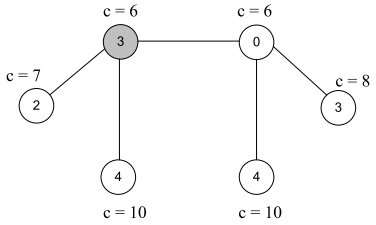


Figure 3: Applying the greedy algorithm on the graph of Figure 2(a). The dark node is the originator.

Note that the only difference between Algorithm 2 and Algorithm 1 is the additional step in line 12 of Algorithm 2. However this *small* difference dramatically enhances the greedy approach in comparison with Dijkstra's algorithm (as will be discussed later). Figure 3 shows the output by the greedy method for the graph of Figure 2(a). It is easy to verify that the broadcast time of this tree is equal to 9; while the broadcast time of the tree generated by the modified Dijkstra's algorithm (Figure2(b)) is

Algorithm 2 Greedy Algorithm

Input: A weighted graph $G = (V, E)$, the originator $u \in V$

Output: The shortest path tree from u as a scheme for broadcasting from u in G.

1: **for** any vertex $v_i \in V$ **do**
2: $c(v_i) \leftarrow \infty$
3: $p(v_i) \leftarrow NULL$
4: **end for**
5: $I \leftarrow \{u\}$ {the set of Informed vertices}
6: $c(u) \leftarrow w(u)$
7: $U \leftarrow V - \{u\}$ {the set of Uninformed vertices}
8: **while** $U \neq \phi$ **do**
9: Find $v_\alpha \in I, v_\beta \in U$, $(v_\alpha, v_\beta) \in E$ such that $c(v_\alpha) + w(v_\beta) = \min(c(v_i) + w(v_j))$ $[v_i \in I, v_j \in U, (i, j) \in E]$
10: $p(v_\beta) \leftarrow v_\alpha$
11: $c(v_\beta) = c(v_\alpha) + w(v_\beta) + 1$
12: $c(v_\alpha) = c(v_\alpha) + 1$
13: $I \leftarrow I \cup \{v_\beta\}$
14: $U \leftarrow U - \{v_\beta\}$
15: **end while**
16: **return** $p(v_0), p(v_1), \cdots p(v_n)$

10. Later we generalize this result to state that the performance of the greedy algorithm, in average, is much better than Dijkstra's algorithm.

It is rather easy to see that the complexity of two algorithms is the same (because the difference is just in one step taking constant time). So we can apply the result of [5] to conclude that Algorithm 2 produces a broadcast scheme for any graph $G = (V, E)$ in time $\mathcal{O}(|E| + |V| \log |V|)$.

2.3 The Evolutionary Algorithm

There are another type of heuristic algorithms that can be applied for the broadcast problem based on evolutionary processes. In [9] a genetic algorithm is proposed for the problem in classical model. Also a simulated annealing approach is presented in [4] for broadcasting in generalized postal model. Here we introduce an evolutionary algorithm for broadcasting in weighted-vertex graphs. The algorithm is desired to be simple enough for future theoretical analysis.

To obtain a good scheme for an instance $G = (u, V)$ of the broadcasting problem, We start with a candidate solution (broadcast tree) which is initialized randomly. Then the solution is improved in an evolutionary process. In each iteration of the evolution, a small change, called *mutation*, is applied on the tree. If the broadcast time of the mutated tree is better than the original one, the new tree will be *accept*ed; otherwise the algorithm *reject*s it. This process continues up to the point that the broadcast times of mutated trees do not improve after a series of iterations. In this situation we say the system is *freezed* and stop the evolution process.

In this way, the process starts at a random solution and gradually moves to better and better states until it cannot improve any more. Algorithm 3 precisely describes the evolution process. The algorithm works based on the following assumptions:

- A mutation is applied by removing a random edge of the tree, thus splitting a subtree off it. Then the subtree is rejoined to the main tree by inserting an edge between the root of subtree and a random node of the main tree.
- The algorithm always keeps the *best solution* to returns it, if at the end of the process no better solution was found.
- An *Equilibrium* is reached if no improvement occurs after a certain number of iterations (In Algorithm 3 this number is stored in valuable ctr). In this situation we reset the *current solution* with the preserved *best solution*.
- After a certain number of repeated equilibriums, the system is considered to be freezed and the algorithm terminates (in Algorithm 3 this number is stored in valuable $freezLevel$).

Note that the time complexity of the algorithm is a matter of $freezBond$. Higher values for this value may result in better solutions, however it may dramatically increase the time complexity of the algorithm.

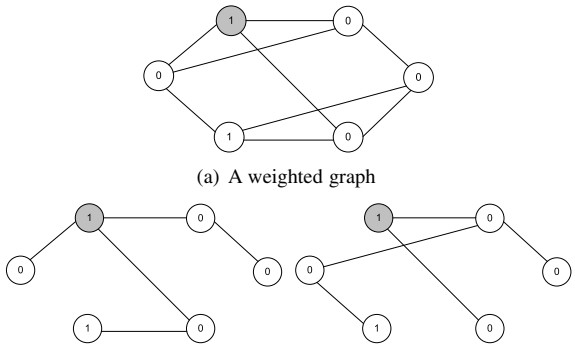
3 Simulations and Analysis

In this section we compare the performance of the heuristics introduced in the previous section based on computer simulation. Note that we can not say that any of the heuristics always performs better than another one. It may come to mind that the output by the greedy algorithm is always better than the modified Dijkstra's algorithm. However it is not true. Figure 4 shows a graph and output trees of the two algorithms. It can be verified that broadcasting in the scheme generated by Dijkstra's algorithm completes after 4 rounds; while the scheme created by the greedy method needs 5 rounds to complete.

Note that many parameters may affect the performance of the algorithms; each of them needs rounds of simulation to be studied separately. Here, we consider graphs with specific limitations on their size, degree, and weight. Although the result of this work can not be extended to all graphs, it gives an intuition on the performance of the

heuristics. Theoretical analysis of the performance of the algorithms is a much more appealing topic which is left as a future work.

We implemented the algorithms presented in the previous sections, as well as an algorithm for creating random spanning trees as the baseline solution. We applied each algorithm 50 times on 100 different randomly generated graphs.



(a) A weighted graph

(b) The output tree of Dijkstra algorithm

(c) The output tree of greedy algorithm

Figure 4: A graph in which the modified Dijkstra's algorithm performs better ($br = 4$) than the greedy algorithm ($br = 5$).

While creating random graphs, it is assumed that a specific number of edges is distributed uniformly between nodes of graph. Also it is assumed that the weights assigned to the nodes obey a normal distribution.

Here, it is studied how the average weight and the average degree of vertices affect the efficiency of the algorithms.

- **Average weight of vertices:**

  In this experiment, we create random graphs on $n = 200$ vertices and 4000 edges (so the average degree of each node is 40) to study the performance of the algorithms when the average weight of the vertices changes. Note that the topology of graphs is preserved when the weights change. Table 1 contains the broadcast times of the algorithms in each case.

  As expected, in average, the results of the greedy algorithm are much better than Dijkstra's algorithm. Also, we can see when the weights grow up, the gap between the performance of the greedy algorithm and Dijkstra's algorithm decreases. A simple explanation is that when the weights increase, this is the weight (and not degree) of the vertices lying on the paths from originator to the leaves, that mainly determines the broadcast time; while the the strength of the greedy algorithm is in producing schemes with lower degrees. Note that the performance of evolutionary algorithm decreases when the weights grow up. We should recall that the efficiency of the evolutionary algorithms is a matter of *freezBound*. In this experiment, this parameter is set to be 10 which is relatively low. However, as Table 1 shows, this 'fast' algorithm evolves randomly made trees in a reasonable rate.

Algorithm 3 Evolutionary Algorithm

Input: A weighted graph $G = (V, E)$, the originator $v_0 \in V$, $FreezBond$ [the freezing bound indicating termination condition].
Output: A tree rooted at v_0 as a scheme for broadcasting from v_0 in G.

1: $curTree \leftarrow$ a random schedule (spanning tree) of G {the current solution that is evolving }
2: $bestTree \leftarrow curTree$, $bestTime \leftarrow br_{v_0}(curTree)$ {the best solution to the problem is kept in these variables}
3: $freezLevel = 0$
4: $cnt = 0$
5: **while** $freezLevel < freezBond$ **do**
6: $curTree \leftarrow mutation(curTree)$
7: **if** $br_{v_0}(curTree) < bestTime$ **then**
8: $bestTree \leftarrow curTree$
9: $bestTime = br_{v_0}(curTree)$
10: $freezLevel = 0$
11: **else**
12: $cnt = cnt + 1$
13: **if** $cnt > freezBond$ **then**
14: $curTree \leftarrow bestTree$
15: $freezLevel = freezLevel + 1$
16: **end if**
17: **end if**
18: **end while**
19: **return** $bestTree$

- **Average degree of vertices:**

  In this experiment, we create graphs on $n = 200$ vertices and average weight of $\bar{w} = 10$. We change the number of edges to study the performance of the algorithms when the average degree of nodes changes. Table 2 contains the broadcast times of the algorithms in each case.

  Theoretically, the broadcast time of the optimum scheme improves when the average degree increases. However in Table 2 it can be seen that the performance of Dijkstra's algorithm decreases when the number of edges grows. The reason is that in this algorithm, the cost caused by the degree of the vertices does not come to account; the out-

$\bar{w}$	$Dijkstra$	$Greedy$	$Random$	$Evolutionary$
0	42	8	12	11
1	46	9	21	17
2	53	13	29	26
5	53	25	75	62
10	60	40	121	107
20	69	70	255	214
40	154	157	573	457
60	230	234	766	690
80	266	267	1034	791
100	340	339	1188	999

Table 1: The broadcast time of the heuristics, when the average weight of vertices ($\bar{w}$) changes; The number of vertices is 200 and the number of edges is 4000.

$\bar{deg}$	$Dijkstra$	$Greedy$	$Random$	$Evolutionary$
30	72	55	140	133
40	82	46	135	131
50	78	50	135	128
60	83	44	133	125
70	82	51	131	123
80	92	45	128	123
90	103	46	122	115
100	193	44	121	114
150	196	43	107	102

Table 2: The broadcast time of the heuristics, when the average degree of vertices ($\bar{deg}$) changes; The number of vertices is assumed to be 200 and the average weight is 10.

put of Dijkstra's algorithm in large-degree graphs are trees with high-degree and small diameter. As an extreme example, in complete graphs, the output of Dijkstra's algorithm is a star tree; which is the worst possible scheme (much wost than a random scheme). The greedy approach is also based on minimizing the length of paths. So in dense graphs the algorithm tends to create trees with small diameter. However the cost applied for extra children compromises this drawback to some extent. In the case of random trees and trees generated by evolutionary algorithms, the broadcast time reduces in high-degree graphs; which is quite natural.

References

[1] T. Asaka, T. Miyoshi, and Y. Tanaka. Multicast routing in satellite-terrestrial networks. *Fifth Asia-Pacific Conference on Communications and Fourth Optoelectronics and Communications Conference*, 1:768–771 vol.1, 1999.

[2] A. Bar-Noy and C-T. Ho. Broadcasting multiple messages in the multiport model. *IEEE Transactions on Parallel and Distributed Systems*, 10(5):500–508, 1999.

[3] A. Bar-Noy and S. Kipnis. Designing broadcasting algorithms in the postal model for message-passing systems. In *ACM Symposium on Parallel Algorithms and Architectures*, pages 13–22, 1992.

[4] A. Bar-Noy and U. Nir. The generalized postal model-broadcasting in a system with non-homogeneous delays. *Electrotechnical Conference, 1998. MELECON 98., 9th Mediterranean*, 2:1323–1327 vol.2, 18-20 May 1998.

[5] M Barbehenn. A note on the complexity of dijkstra's algorithm for graphs with weighted vertices. *IEEE Transactions on Computers*, 47(2):263, 1998.

[6] D. Culler, R. Karp, D. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian, and T. Von Eicken. Logp: towards a realistic model of parallel computation. *SIGPLAN Not.*, 28(7):1–12, 1993.

[7] P. Fraigniaud and E. Lazard. Methods and problems of communication in usual networks. *Discrete Applied Math.*, 53(1-3):79–133, 1994.

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

[9] Cory J. Hoelting, Dale A. Schoenefeld, and Roger L. Wainwright. A genetic algorithm for the minimum broadcast time problem using a global precedence vector. In *SAC '96: Proceedings of the 1996 ACM symposium on Applied Computing*, pages 258–262, New York, NY, USA, 1996. ACM.

[10] J. Hromkovic, R. Klasing, A. Pelc, P. Ruzicka, and W. Unger. *Dissemination of Information in Communication Networks: Broadcasting, Gossiping, Leader Election, and Fault-Tolerance*. Springer, Berlin, 2005.

[11] S. Kamali. Broadcasting in weighted-vertex graphs. Master Thesis, Faculty of Engineering & Computer Science, Concordia University, 2008.

[12] E. H-k. Wu and C. Chang. Adaptive multicast routing for satellite-terrestrial network. *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*, 3:1440–1444 vol.3, 2001.