Quorumcast Routing by Multispace Search[†]

Bin Du[1], Jun Gu[1], Danny H.K. Tsang[2], and Wei Wang[2]
[1]Department of Electrical and Computer Engineering
University of Calgary, Calgary, Canada T2N 1N4
gu@enel.ucalgary.ca
[2]Dept. of Electrical and Electronic Engineering
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong

Abstract

In this paper, we present a multispace search algorithm, *MUSQ*, for the quorumcast routing problem. We first prune the candidate set to reduce the original search space. Some of the unqualified links for the optimal tree are removed from the network. Then a multispace search algorithm is used to solve the quorumcast routing problem. By altering the original objective function, the original problem instance is transformed into a series of gradually more simplified problem instances with smoother terrain surfaces. A local search algorithm is used to solve each problem instance, from the simplest structure to the original structure, and the solutions of the more simplified problem instances are used to guide the search of more complicated ones. Experimental results showed that this new method improved the performance of the existing heuristic algorithms.

1 Introduction

Many networking applications require multicast connections in addition to conventional unicast connections. Multicasting delivers a message from a single source node to a subset of nodes, the *multicast destinations*, in the network. Such could be the case of multipoint video conference, the distribution of a document to selected number of persons via a computer network, or the request for certain information from a distributed database. The problem of connecting a subset of nodes in an undirected graph using the minimal cost subgraph, such that there is a path between each pair of nodes, is known to be NP-complete [6, 11, 14]. A variety of connection schemes, such as steiner tree and spanning tree methods, can be used to multicast connections.

Quorumcasting delivers a message from a single source node to any subset of $|Q|$ nodes in an n-node network ($|Q| < n$). The problem of quorumcast routing is important in a number of distributed applications and the management of replicated data [4]. In distributed systems, for example, *quorum consensus* gives a general class of synchronization protocols. An operation proceeds to completion only after it coordinates with nodes that constitute a *quorum set* [2]. The collection of quorum groups used by an operation is called a *quorum set*. A well-known method for defining quorum sets is *voting* [1]. In voting, each member in the process group is assigned one vote and critical operations that require synchronization must obtain a *quorum* before proceeding further.

Although the multicasting problems have been researched actively in the past [5, 12, 10, 15, 3], the quorumcasting problem has only been addressed recently. In this paper, we propose a multispace search algorithm to solve the quorumcast routing problem.

The rest of the paper is organized as follows: In the next section, we will give an overview of some existing quorumcast routing methods. In Section 3, we introduce the basic idea of scrambling in the objective function space for quorumcast routing. A multispace search algorithm for the quorumcast routing problem is introduced in Section 4. The experimental results of the multispace search algorithm for quorumcast routing are shown in Section 5. Section 6 concludes the paper.

2 Previous Work

Quorumcast routing is a generalization of multicast routing. It can be stated as follows:

In an n-node communication network, a quorum set Q consists of any $|Q|$ members ($|Q| < n$). A message is to be sent from a source node in the network to the quorum set, such that the routing tree spanning the source node and the quorum set has the minimum communication cost, determined as the sum of costs of the links in the routing tree.

For an n-node communication network, there are $\begin{pmatrix} n \\ |Q| \end{pmatrix}$ ways of selecting $|Q|$ nodes from the group of n nodes. The number of routing trees that span the source node and the quorum set can be very large. The optimum routing tree is the one that has the minimum cost. As the multicast routing problem is NP-complete, efficient optimal solutions to the quorumcast routing problem are unlikely to be found [4].

Cheung and Kumar [4] presented three heuristics: the Minimum Cost Path (MPH) heuristic, the Improved Minimum Path (IMP) heuristic, and the Modified Average Distance (MAD) heuristic, to find low cost solutions for the quorumcast routing problem. In the MPH heuristic, initially, only the source node s is put in the quorum set, and the size of the quorum set is 1. In each step, the algorithm examines all the nodes that do not belong to the quorum set and selects one nearest to the quorum set until the required size of the quorum set is reached. The IMP heuristic and the MAD heuristic are two improved heuristic methods presented in [4]. Extensive experimental results showed that,

[†]This work was supported in part by NSERC Strategic Grant MEF0045793 and NSERC Research Grant OGP0046423 and is presently supported in part by NSERC Strategic Grant STR0167029, the Federal Micronet Research Grant, and Hongkong Telecom Institute of Information Technology Grant HKTIIT 93/94.EG01.

among the three heuristics, the IMP heuristic produces the best solution.

Recently the *multispace search* approach has been developed for solving general optimization problems [7, 9, 8]. In this paper, we apply the multispace search to solve the quorumcast routing problem. The technique is capable of smoothing the rugged terrain surface of the search space. It interplays structural operations on the objective function in conjunction with the local search algorithms. Structural operations on the objective function disturb the environment of forming local minima, which makes multispace search a very natural approach to handle the quorumcast routing problem.

3 Scrambling in the Objective Function Space for Quorumcast Routing

A network is modeled as a graph $G(V, E)$, where V represents the set of nodes (or vertices) and E the set of interconnecting links (or edges). Links are assumed to be undirected. In the following discussion, let:

- n be the number of nodes in the network, $n = |V|$;
- m be the number of links, $m = |E|$;
- s be the source node;
- c_i be the cost of link i, $i \in E$, and c be the cost vector;
- link i, $i \in E$, connecting nodes a and b, be represented by (a, b);
- Q be the quorum set ($|Q| < n$);
- A and B be two subsets of the link set E, such that $A \cup B = E$ and $A \cap B = \phi$, where A represents the *chosen* link set and B the *unchosen* link set; and
- C be the objective function value, i.e., the total communication cost of the routing tree spanning the quorum set and the source node:

$$C = \sum_{j=1}^{|Q|} c_{l_j}, \text{ where } l_j \in A. \tag{1}$$

To search for an optimal solution efficiently, combinatorial optimization procedures must overcome local minimum/maximum points which normally exist. Multispace search not only alters values in the value space, but also scrambles across the variable space and other active spaces. It has been proven to be efficient to overcome the pathological phenomena in combinatorial optimization problems. In this paper, the proposed multispace search algorithm will scramble in the objective function space [9].

For a given network, the costs of different links may be different, which makes the search space rugged. We use a *smoothing factor*, α, to characterize the degree of a smoothing operation on the objective function and the smoothness of the resulting search space. For the quorumcast routing problem, one uniform initial state is the *average link cost*. By summing the costs of all links in the network and normalizing it, we can get the average link cost $c_{avg} = \frac{1}{m}\sum_{i=1}^{m} c_i$.

The costs of all links in the network are set to the average link cost c_{avg}. As the costs of all links are the same, we can directly connect any $|Q|$ nodes with the source node, and the objective value C (total cost of the routing tree) is same, i.e., $C = c_{avg} \times |Q|$.

Then, a *simplified* quorumcast routing problem instance can be defined by a specified smoothing factor, α. For each problem instance, the cost of the ith link is determined by:

$$c_i(\alpha) = c_{avg} + (c_i - c_{avg}) \times \alpha, \tag{2}$$

where $0 \leq \alpha \leq 1$. When α increases gradually from 0 to 1, a series of gradually more complicated problem instances, from the simplest problem instance to the original problem instance, are generated.

A search space generated from a smaller α exhibits a smoother terrain surface, and a search space generated from a larger α exhibits a more rugged terrain surface. Two extreme cases in the series of the problem instances are:

- If $\alpha = 0$, then $c_i(\alpha) \to c_{avg}$, which is the *simplest case*;
- If $\alpha = 1$, then $c_i(\alpha) \to c_i$, which is the original problem instance.

Once we have a series of smoothed problem instances, we can use any existing local search algorithm to solve them. We start from the simplest problem instance ($\alpha = 0$), i.e., the case with a *flat* search space. A solution to the simplest problem can easily be found. The solution of this problem instance is then taken as the initial solution to the next problem instance that has a slightly more complicated search space. The problem is again solved using the local search algorithm. The above procedure is repeated until the final problem instance having the original search space (i.e., $\alpha = 1$) is solved.

4 A Multispace Search Algorithm for Quorumcast Routing

The *MU*ltispace *S*earch algorithm for *Q*uorumcast routing, *MUSQ*, is shown in Figure 1. It consists of two stages, a reducing candidate set stage and a multispace search stage.

4.1 Reducing Candidate Set

We propose two procedures (*remove_unqualified_set* and *choose_unavoided_set*) to reduce the search space by reducing the candidate set. The procedures reduce the feasible space while maintaining the optimality. They restrict the number of links/nodes to be selected in a constrained range instead of the complete link/node set.

The procedure *heuristic_search*() initially uses some heuristic algorithm to obtain an approximate solution C_0. For example, the simple heuristic algorithm can start with the source node. Each time a node nearest to the quorum set and the source node is selected until the size of the quorum set is reached.

The first procedure to reduce the candidate set, *remove_unqualified_set*(), makes use of the approximate solution C_0. We explain the procedure through an example.

procedure MUSQ()
begin
  /* reduced candidate set stage */
  C_0 = heuristic_search(G);
  G' = remove_unqualified_set(G, C_0);
  R = choose_unavoided_set(G');
  if the optimal routing tree is found
    return the optimal routing tree;

  /* multispace search stage */
  A_{opt} = create_an_initial_route(G', R);
  $\alpha = 0$; δ = set_threshold(G);
  while $\alpha < 1$ **do**
  begin
    α = increment(α);
    c = set_link_cost(α);
    $C_0 = \sum_{j=1}^{|Q|} c_{l_j}$, where $l_j \in A_{opt}$;
    /* local search */
    A_{opt} = local_search(A_{opt}, C_0, δ);
  end;
  return the optimal routing tree A_{opt};
end;

Figure 1: **MUSQ**: A multispace search algorithm for quorumcast routing.

Example. A simplified NSFNet T3 backbone [10] is shown in Figure 2. Suppose the source node s is node *Argonne* and $|Q|$ is 5. We need to find a minimum cost tree spanning a set of 5 nodes and the source node *Argonne*.

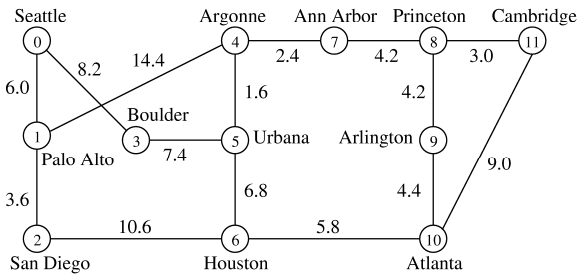

Figure 2: An example of the 12-node communication network NSFNet T3.

At first, a simple heuristic algorithm is used to find an approximate solution. Each time the nearest node to the quorum set and the source node is chosen. In the network, the nearest node to node *Argonne* is *Urbana*. So node *Urbana* is included in the quorum set. The process continues until we find 5 nodes: *Urbana*, *Ann Arbor*, *Princeton*, *Cambridge*, and *Arlington*. The routing cost is $C_0 = 15.4$. To reduce the candidate set, we first sort the link set in an ascending order on the value of the link cost c_i, (i.e., c_1 is the minimum link cost and c_m is the maximum link cost). For the first $(|Q| - 1)$ links with the minimum costs, the total cost is

$$sum = \sum_{j=1}^{4} c_j = 1.6 + 2.4 + 3.0 + 3.6 = 10.6.$$

Since C_0 is 15.4, any link whose cost is greater than $C_0 - sum$ (i.e., 4.8) can not be a link in the candidate set for the minimum cost quorumcast tree. This is because that if any link whose cost is greater than 4.8 is selected in conjunction with the first 4 minimum cost links, their total costs will exceed C_0. The obtained cost tree will not be a minimum cost quorumcast tree. In this example, there are 8 links whose costs are greater than 4.8. These 8 links are removed from the link set, giving a simplified network (Figure 3).

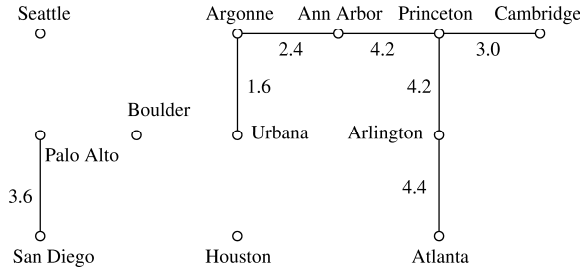

Figure 3: The simplified NSFNet after removing 8 links whose costs are greater than 4.8.

The simplified network may be a disconnected graph. Among all the disconnected trees, we just keep the one connecting with the source node for later search. Originally, there are 15 links in the network. We should search 5 links among these 15 links and there are 3,003 choices. Now the size of the candidate link set has been reduced to 6. We only need to select 5 links among these 6 links.

The second procedure to reduce the candidate set, *choose_unavoided_set*(), chooses some nodes which must be selected to the quorum set with the minimum cost. Denote the nodes that have been included in the quorum set in the procedure as set R. We give two methods to choose the qualified nodes to the quorum set.

Method 1: At first, only the source node s is chosen and the current quorum set is empty. We examine all unchosen nodes which are directly connected to s or the current quorum set. If link *i, (a, b)*, for which node a is in the candidate node set and node b is in the current quorum set or the source node, has the minimum cost in the unchosen link set, then node a must be included in the optimal quorum set and link i is removed from the unchosen link set B and added to the chosen link set A. Now node a is in the quorum set. We continue this process until either the size of the final quorum set is reached (in this case, the optimal solution is found) or none of the links directly connected to the source node or the current quorum set have the minimum cost in the unchosen link set.

Method 2: If there is only one link (a, b), in which node a is in the candidate node set and node b is in the current quorum set or the source node, node a must be included in the optimal quorum set and the size of the quorum set

increases by 1. Accordingly, the link is removed from the unchosen link set B and added to the chosen link set A.

We can use the above two methods to construct the quorum set. If the size of the quorum set is reached, the procedure *choose_unavoided_set*() has found the optimum quorum set with the minimum cost. If not, we will use a multispace search algorithm to continue the search until the final quorum set is found. We explain these two methods through the above example.

Example (continued). There are two links connected with the source node s, and their costs are 1.6 and 2.4, respectively. The minimum link cost in the network is 1.6. Using method 1, node *Urbana* should be added to the current quorum set and the link between *Argonne* and *Urbana* is removed from the unchosen link set and added to the chosen link set.

There is only one link, i.e., the link between *Argonne* and *Ann Arbor*, connecting the current quorum set and the source node to other candidate nodes. Using method 2, node *Ann Arbor* should be added to the current quorum set. In the same way, nodes *Princeton, Cambridge*, and *Arlington* are added to the quorum set, giving the size of the quorum set $|Q| = 5$. Since the methods for reducing the candidate set do not compromise the optimality, the solution obtained in this example is optimal.

In the above example, we can see that for some simple problems, the two-step algorithm for reduced candidate set is so efficient that the optimal solutions can be obtained. In this case, the algorithm just terminates with the optimal solution and the following multispace search is not needed. For more complicated problems, although the algorithm may not find an optimal solution, it can efficiently reduce the original search space. Consequently, the subsequent multispace search can find good solutions much more efficiently.

4.2 Multispace Search Stage

After the reduction of candidate set, any unqualified nodes and links have been removed from the original graph $G(V, E)$. The remaining nodes and links compose the simplified graph $G'(V', E')$. Usually, the sets V' and E' are smaller than the original sets V and E.

Procedure *create_an_initial_route*() in Figure 1 gives an initial quorumcast routing assignment. It chooses $|Q|$ nodes which are connected together with the source node s. Note that $|R|$ nodes have been chosen by procedure *choose_unavoided_set*(). In this paper, we use three heuristic methods to perform the initial routing assignment. The first method randomly chooses $|Q| - |R|$ nodes which are directly connected to the source node and the current quorum set R as the initial routing assignment. The second and third methods use the solutions given by the Minimum Cost Path (MPH) heuristic and the Improved Minimum Path (IMP) heuristics, respectively. If there is a contradiction between the solutions and the partial tree containing R, we discard some nodes in the solution in contradiction to the partial tree. Procedure *set_threshold*() sets the threshold parameter δ used for the later local search, which will be discussed later.

The initial value of the smoothing factor α is set to 0. In each iteration, α is increased gradually from 0 to 1 in procedure *increment*(). Each time we increase α by 0.2. During each iteration, a quorumcast routing instance closer to the original problem instance with a more rugged search space is generated. This is done by function *set_link_cost*() which re-computes the costs of all candidate links by Equation (2). The quorumcast routing cost C_0 is obtained by summing all the chosen links.

Procedure *local_search*() in Figure 1 is used to find a routing tree having a locally minimum cost. For an unchosen node a and a chosen node b, if the cost of the routing tree obtained by swapping node a and node b does not exceed the sum of C_0 and δ, node a and node b are swapped. Node a is removed from the quorum set to the candidate set and node b is removed from the candidate set to the quorum set. The local search procedure repeats swapping nodes in the two sets in a sequential order until no further improvement (i.e., a routing tree with locally minimum cost is found).

In our local search procedure, the threshold δ is a parameter used to handle local minima. Since the link cost has an effect on δ, we set δ according to a simple formula

$$\delta = \frac{1}{3} \times \sqrt{\frac{\sum_{i=1}^{m} c_i^2}{m}}.$$

In the *local_search*() procedure, we use a simple but efficient local handler, called *climbing* heuristic, to avoid being stuck at local minima. In the *climbing* heuristic, two nodes are swapped even if the value of the objective function increases, as long as the new routing cost does not exceed $(C_0 + \delta)$. *Climbing* heuristic in the local search procedure can effectively improve the convergent performance of the *MUSQ* algorithm.

5 Numerical Results

Our experimental results were performed on a SUN SPARC 20 workstation. All CPU times are in seconds. The first network is the 12-node NSFNet network [10] (Figure 2). Table 1 gives the performance comparison of the multispace search algorithm (*MUSQ*) and the MPH and the IMP heuristics for the NSFNet network. The first two columns give the source node s and the size of the quorum set $|Q|$. From the third to the tenth columns, we give the results obtained by the random initial assignment. As the results depend on the random initial assignment, we ran the program for 100 times to get the probabilistic behavior of the algorithm. For each routing case, the minimum cost (*Best*) and the average cost (*Avg*) in the 100 runs are given in columns 3 and 4. Columns 5, 6, and 7 give the minimum, mean, and maximum CPU time in the 100 runs (Min, $Mean$, and Max), respectively. In columns 8, 9, and 10, the minimum, mean and maximum iterations in the 100 runs (Min, $Mean$, and Max) are given respectively. Out of the 100 trials, the *MUSQ* algorithm with the random initial assignment was able to locate the optimal solution almost every time (the optimal solution was obtained by an exhaustive search algorithm).

Table 1: Performance of the *MUSQ* algorithm in term of quorumcast routing cost for the 12-node NSFNet network.

		MUSQ algorithm										*Cheung et al. [4]*	
		Random initialization								Initialized by			
s	$\|Q\|$	100 Trials		Execution Time (s)			Iterations			MPH	IMP	MPH	IMP
		Best	Avg	Min	Mean	Max	Min	Mean	Max				
8	9	39.8	39.80	<0.001	0.010	0.020	18	19.29	21	39.8	39.8	41.2	41.2
0	8	35.4	35.40	0.010	0.013	0.020	19	19.30	22	35.4	35.4	36.4	36.4
2	5	25.0	25.02	0.010	0.011	0.020	18	19.24	21	25.0	25.0	26.8	26.2
2	7	32.2	32.20	0.010	0.014	0.030	19	19.72	22	32.2	32.2	33.4	33.4

Table 2: Performance of the *MUSQ* algorithm in term of quorumcast routing cost for the 100-node network.

		MUSQ algorithm										*Cheung et al. [4]*	
		Random initialization								Initialized by			
s	$\|Q\|$	100 Trials		Execution Time (s)			Iterations			MPH	IMP	MPH	IMP
		Best	Avg	Min	Mean	Max	Min	Mean	Max				
6	10	133.2	133.64	1.150	1.335	1.670	19	20.91	23	133.2	133.2	142.0	137.6
9	12	194.2	195.30	1.570	1.938	2.420	19	21.45	26	194.2	194.2	213.1	206.7
25	10	162.0	164.59	1.170	1.428	1.820	19	21.11	25	162.0	162.0	185.8	171.4
59	16	227.0	229.21	2.340	3.151	4.300	19	22.39	29	227.0	227.0	232.1	232.1

In our algorithm, we also use the solutions obtained by the MPH heuristic and the IMP heuristic as the initial assignments for the *MUSQ* algorithm. The results are shown in columns 11 and 12, respectively. The last two columns in Table 1 show the results of the MPH heuristic and the IMP heuristic by Cheung et al. [4].

The second network is the 100-node network [13]. Table 2 shows the performance comparison of the multispace search algorithm (*MUSQ*) and the MPH and IMP heuristics for the 100-node network. For 100 executions, the average costs obtained by the multispace search algorithm are very close to the minimum cost obtained in the 100 runs. For example, for the first case, our algorithm can find cost 133.2 almost every time, while the MPH and IMP algorithms find costs 142.0 and 137.6, respectively.

Several observations can be made from these results. Using the random initialization, the average costs of the 100 runs are very close to the minimum cost obtained in the 100 runs. Since we can not obtain the optimal solution for this network, we would use the minimum cost obtained from the 100 runs as a reference. If we use the solutions obtained by either the MPH heuristic or the IMP heuristic as the initial assignments, we find that the *MUSQ* algorithm can always find the minimum costs obtained by the solution of the random initial assignment in the 100 runs.

6 Conclusion

In this paper, we give a multispace search algorithm to minimize the quorumcast routing cost of the network. Experimental results indicate that when compared to existing heuristic approaches, the multispace search algorithm gives significant performance improvements by achieving a lower routing cost. In addition, it is insensitive to the initial solution point and is computationally efficient. High performance is achieved by reducing the candidate set, by using an efficient multispace search algorithm, and by using a novel climbing heuristic which is able to handle local minima effectively. Additionally, the algorithm is scalable since it can be used for multicast and quorumcast routing in large size networks.

References

[1] M. Ahamad, M.H. Ammar, and S.Y. Cheung. Multi-dimentional voting. *ACM Transactions on Computer Systems*, 9(4):399–431, Nov. 1991.

[2] D. Barbara and H. Garcia-Molina. Mutual exclusion in partitioned distributed systems. *Distrib. Comput.*, 1:119–132, 1986.

[3] F. Bauer and A. Varma. Degree-constrained multicasting in point-to-point networks. In *Proc. IEEE INFOCOM*, pages 369–376, Boston, Massachusetts, Apr. 1995.

[4] S.Y. Cheung and A. Kumar. Efficient quorumcast routing algorithms. In *Proc. INFOCOM*, pages 840–847, Toronto, Ontario, Canada, Jun. 1994.

[5] C.H. Chow. On multicast path finding algorithms. In *Proc. INFOCOM*, pages 1274–1283, Bal Harbour, FL, Apr. 1991.

[6] E.N. Gilbert and H.O. Pollak. Steiner minimal trees. *SIAM Journal of Applied Mathematics*, 16(1):1–29, 1968.

[7] J. Gu. *Multispace Search: A New Optimization Approach (Summary). Lecture Notes in Computer Science, Vol. 834*, pages 252–260. Springer-Verlag, Berlin, 1994.

[8] J. Gu. *Optimization by Multispace Search.* Kluwer Academic Publishers, Massachusetts, 1996.

[9] J. Gu and X. Huang. Efficient local search with search space smoothing. *IEEE Trans. on Systems, Man, and Cybernetics*, 24(5):728–735, May 1994.

[10] C.A. Noronha Jr. and F.A. Tobagi. Optimum routing of multicast streams. In *Proc. INFOCOM*, pages 865–873, Toronto, Ontario, Canada, Jun. 1994.

[11] R.M. Karp. *Reducibility among Combinatorial Problems. Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[12] H. Tode, Y. Sakai, M. Yamamoto, H. Okada, and Y. Tezuka. Multicast routing algorithm for nodal load balancing. In *Proc. INFOCOM*, pages 2086–2095, Florence, Italy, May 1992.

[13] J.E. Wieselthier, C.M. Barnhart, and A. Ephremides. A neural network approach to routing without interference in multihop radio networks. *IEEE Trans. on Communications*, 42(1):166–176, Jan. 1994.

[14] P. Winter. Steiner problem in networks: A survey. *Networks*, 17:129–167, 1987.

[15] Q. Zhu, M. Parsa, and J.J. Garcia-Luna-Aceves. A source-based algorithm for delay-constrained minimum-cost multicasting. In *Proc. IEEE INFOCOM*, pages 377–385, Boston, Massachusetts, Apr. 1995.