Tabu Search Algorithm for Core Selection in Multicast Routing

Satya Prakash Sahoo  
Dept. of Comp Sc.& Engineering  
VSS University of Technology, Burla  
Sambalpur, India  
e-mail: sa_typaprakash@rediffmail.com

Manas Ranjan Kabat  
Dept. of Comp Sc.& Engineering  
VSS University of Technology, Burla  
Sambalpur, India  
e-mail: manas_kabat@yahoo.com

Ashis Kumar Sahoo  
Dept. of Comp Sc.& Engineering  
VSS University of Technology, Burla  
Sambalpur, India  
e-mail: ashiskumar123@gmail.com

Abstract— With the development of network multimedia technology, more and more real-time multimedia applications need to transmit information using multicast. The basis of multicast data transmission is to construct a multicast tree. The main problem concerning the construction of a shared multicast tree is the selection of the root of the shared tree or the core point. In this paper, we propose an algorithm based on Tabu Search for core selection in multicast routing. This algorithm selects core point by considering both delay and inter-destination delay variation. The simulation results show that the proposed algorithm performs better than the existing algorithms in terms of delay variation subject to the end-to-end delay bound.

Keywords— QoS routing; Multicast routing; delay variation; end-to-end delay

I. INTRODUCTION

With the development in network multimedia technology, more and more real-time multimedia applications need to transmit information, which usually has various quality of service (QoS) performance requirements. These applications often require the underlying network to provide multicast capabilities. Multicast refers to the delivery of packets from a single source to multiple destinations. Therefore, it is important and urgent research problem to set up multicast routing with high QoS. The central problem of QoS multicast routing is to set up a multicast root tree that can satisfy certain QoS parameter. The prime problem of constructing a shared multicast tree is to determine the position of the root of the shared tree, which is the center selection problem. The center is called core point in core based tree (CBT)[1] or Rendezvous Point(RP) in PIM-SM[2]. This problem is first proposed by Wall [3]. The selection of core directly affects the performance of multicast. A poor selection may lead to such performance problems as high cost, high delay and high congestion. Therefore, it is very important to select a good core to have effective multicast. But the core selection problem is an NP-Complete problem [1, 3, 4], which needs to be solved using heuristic algorithm. Researchers have already proposed several solutions to this problem [5-15]. However, how to find a better or best core node has not been completely solved.

In this paper, we propose a Tabu search based Core Selection Algorithm (TCSA) by introducing the previous research findings and the basic tabu search algorithm. The tabu search algorithm has already been used in [18] for RP selection in PIM-SM multicast routing that simultaneously minimizes the delay and cost of the multicast tree. Our algorithm attempts to find the best core using the same method with a different fitness function. The algorithm begins the search with a random initial node as core belonging to the network. Then the neighborhood solutions are evaluated via the fitness function. The solution giving minimum fitness value is considered as a new starting point of next search. This search terminates when the solution is converged.

The rest of the paper is organized as follows. The mathematical notation used to model a computer network is presented in section 2. The existing core selection algorithms are presented in section 3. The proposed tabu search algorithm for core selection is presented in section 4. The simulation results of our algorithm are presented in section 5. Finally, we present the conclusion.

II. MATHEMATICAL NOTATION

A network is modeled as a directed, connected graph $G=(V, E, v_s, M)$, where $V$ is a finite set of vertices (network nodes) and $E$ is the set of edges (network links) representing connection of those vertices. Let $|V|$ be the number of network nodes and $|E|$ is the number of network links. $v_s$ represents the source node and $M$ is the set of destination nodes. The link $e=(v_i, v_j)$ from node $v_i \in V$ to node $v_j \in V$ implies the existence of a link and $e'=(v_j, v_i)$ from $v_j$ to $v_i$. Each link is associated with a positive real value: delay $D(e) : E \rightarrow R^+$. The link delay function $D(e)$ is considered to be the sum of queuing delay, transmission delay and propagation delays.

A multicast tree $T(v_s, M)$ is a sub-graph of $G$ spanning the source node $v_s \in V$ and the set of destination nodes $M$. Let $P_f(v_s, v_j)$ be a unique path in the tree $T$ from the source node $v_s$ to a destination node $v_j \in M$. The total delay of the path $P_f(v_s, v_j)$ is simply the sum of delay of all links $D(P_f(v_s, v_j)) = \sum_{e \in P_f(v_s, v_j)} D(e)$.

The other parameter multicast delay-variation $dv$, is the maximum difference between the end-to-end delays along the paths from the source to any two destination nodes and is defined as follows:

$$dv = \max \left\{ \left| \sum_{e \in P_f(v_r, v_j)} D(e) - \sum_{e \in P_f(v_r, v_s)} D(e) \right| \quad \forall \ v_j, v_s \in M \right\}$$

Thus, based upon the above definition we can now state mathematically the multicasting routing problem in our paper as for a given weighted graph $G=(V, E)$, a source node $v_s \in V$, a destination node set $M \subseteq V-{v_s}$, a link-delay
function $D(e) : E \rightarrow R$, $e \in E$ and a constant $\Delta$, determine an optimal multicast tree $T$ such that

$$D(v_s,M) = \max_{v_i \in M} \sum_{e \in P_i(v_i,v_j)} D(e) \leq \Delta$$

$$\delta(v_s,M) = \max \left| \sum_{e \in P_i(v_i,v_j)} D(e) - \sum_{e \in P_j(v_i,v_j)} D(e) \right| \forall v_i, v_j \in M \leq \delta$$

III. RELATED WORKS

There are many well-known approaches to construct core based multicast tree satisfying delay and delay-variation constraints. These are Delay Variation Multicast Algorithm (DVMA) [12], Delay and Delay Variation Constrained Algorithm (DDVCA) [13], AKBC (Ahn, Kim, Bang, Choo) algorithm [14] and AKC (Ahn, Kim, Choo) algorithm [15]. The issue of minimizing multicast delay variation problem under the multicast end-to-end delay constraints are defined and discussed in [12]. This problem is referred as the Delay and delay-variation. A variation of this algorithm is proposed by KIM et.al [6] to produce a core based multicast tree satisfying delay and delay-variation constraints. These are Delay Variation Multicast Algorithm (DVMA) [12], Delay and Delay Variation Constrained Algorithm (DDVCA) [13], AKBC (Ahn, Kim, Bang, Choo) algorithm [14] and AKC (Ahn, Kim, Choo) algorithm [15]. The issue of minimizing multicast delay variation problem under the multicast end-to-end delay constraints are defined and discussed in [12]. This problem is referred as the Delay and Delay Variation Bounded Multicast Tree (DVBMT) problem. In DVMA, it is assumed that the complete topology available at each node. The algorithm starts with a spanning tree satisfying the delay constraint only, which may not include some destination nodes. Then the algorithm searches through the candidate paths satisfying the delay and delay-variation constraints from a non-tree member node to any of the tree nodes. It works on the principle of k-shortest paths to the group of concerned destination nodes. If these paths do not satisfy a delay constraint, then it may find longer path, which is a shortcoming of DVMA. The spanning tree built by DVMA satisfies both delay and delay-variation constraints. But due to the very high time complexity it does not fit in modern high speed computer network environment.

The other approach to solve the DVBMT problem is DDVCA [13]. The DDVCA first calculates the delay of the least delay paths from each destination node to all the nodes. The node that has the minimum delay-variation is selected as the core node. The source node sends a single copy of the message to the core node. Then the core node forwards the message to all the receivers through the minimum delay path. In comparison with the DVMA, the DDVCA possesses a significant lower time complexity i.e. $O(mn^2)$ where $m$ represents number of destination nodes and $n$ represents the total number of nodes in the computer network.

Another efficient core selection algorithm has been proposed by KIM et.al [6] to produce a core based multicast tree under delay and delay-variation constraints. First, this algorithm finds a set of candidate core nodes that have the same associated multicast delay-variation for each destination node. Then, the final core node is selected from the set of candidate core nodes that has the minimum potential delay-variation. A variation of this algorithm is proposed in [14], which uses the MODE function to find the exact location of the core. Then the potential delay-variation associated with each candidate core node is found and the node that has the minimum potential delay-variation is considered as the best core node. However, all these algorithms only applied in the symmetric network environment that has no direction. Ahn, Kim and Choo [15] proposed an algorithm that constructs a multicast tree with low delay-variation in a realistic network environment that has two-way directions. This algorithm works efficiently in the asymmetric network.

However, these algorithms [6, 14, 15] select the best core node out of a set of candidate core nodes that have same associated delay-variation. Therefore, these algorithms are restricted to select the best core node, which may not generate an optimal delay-variation based multicast tree in many cases.

IV. TABU SEARCH ALGORITHM FOR CORE SELECTION

In this section, we present the fundamentals of tabu search and the basic features needed to implement the tabu search. The solution objects considered in this section are nodes, tabu list and the neighboring nodes of the nodes.

A. Basic Tabu Search algorithm

Tabu Search (TS) has been originated by Glover [16], which solve combinatorial optimized problem. It is the expansion of local search algorithm, which follows the neighborhood structure of local search with the addition of a tabu table. Tabu search algorithm manifests two strategies – centralization and diffusion, which respectively promise to reach and skip the local better solution. The basic idea behind tabu search is that, adding short-term memory to local search improves its ability to locate optimal solution. It is an iterative search that starts from some initial feasible solution and attempts to determine the best solution in the manner of a Hill-climbing algorithm. The algorithm keeps historical local optima for leading to the near global optimum fast and efficiently [17].

B. Fitness function

The core selection problem in multicast routing is to find the best position of core such that the multicast delay variation can be optimal subject to the condition that each routing path from source to destination node satisfies the delay constraint. Since the location of the core node greatly influences the multicast delay variation [14], the MODE function is used to know the location of core and CMP function to measure the potential delay variation [15]. The fitness function used in our paper is same as the CMP function computed in [15]. The MODE and Fitness function are defined as follows.

$$\text{MODE} \text{(core)} = \begin{cases} I & \text{if core} = s \\ II & \text{if } g \in P_{d}(s, \text{core}) \\ III & \text{if } g \in P_{d}(g, \text{core}) \\ IV & \text{if II and III} \\ V & \text{otherwise} \end{cases}$$

where, $s$ is source gateway, $g \in M$ is destination gateway and $P_{d}(x,y)$ be the least delay path from node $x$ to $y$. Let $\max^* = \max_{e \in M} \{D(P_{d}(\text{core}, g)) \}$ \forall $g \in M \{ g \ast \in M \ast \in P_{d}(s, \text{core}) \ast \in P(g, \text{core}) \}$ and $\min^* = \min_{e \in M} \{D(P_{d}(\text{core}, g)) \}$ \forall $g \in M \ast \in P_{d}(s, \text{core}) \ast \in P(g, \text{core}) \}$ then,
C. Aspiration Criteria

An important element of tabu search is the incorporation of an aspiration criteria which may lead to a better solution than the best one found so far. A tabu move becomes admissible if it yields a solution that is better than an aspiration value. An aspiration criterion is used to allow tabu moves to be released if they are judged to be worth or interesting. In other words, the aspiration criteria is to allow “excellent” tabu moves to be selected if the aspiration level is attained. In the paper, an aspiration criterion is set in the following way. Although one neighbor is found tabued, it can be aspired if the evaluation value of this neighbor is better than that of current node that is this neighboring node may be used as next search node.

D. Termination Criteria

The selection procedure of core will continue until we got an optimal value. But the process may take long time. To find a better result in short time, we define some criteria that stop the algorithm after the conditions are being satisfied. The conditions are Condition 1: After a predefined fixed number of iteration. Condition 2: If current best result is stable one, i.e. no improvement in the objective function value after some predefined number of iterations. Condition 3: When the objective reaches a pre specified threshold value. Condition 4: If there is no node to search next, the algorithm stops automatically.

E. Operation of TCSA

Our algorithm TCSA starts with establishing the relevant data structure for the network $G=(V,E,v,M)$. Then the maximum iteration number, maximum number of iteration of stable solution and tabu size are set. The delay of the least delay path from source $v_i$ to all nodes and from all nodes to the destination nodes are computed by using Dijkstra’s algorithm. Tabu search starts with randomly selecting a node from network topology as core node and fitness function value of the core is calculated. Then the tabu search is performed by selecting the best move as next move direction from the candidate moves. The best solution state is updated i.e. if the move is not tabued and lead to another solution state better than the one so far visited, or if the move is tabued but aspiration criteria is attained then update the best solution state. Next, the tabu move is set or released. The executed move is recorded as tabu move in the tabu list or the tabu move is released if the aspiration criteria attained.

If the change of the fitness function value of successive best so far solution is less than a given value, the stopping criteria is satisfied and the algorithm terminates with returning the current best core node. Otherwise, the tabu search continues with the best move as next move direction from candidate moves.

Algorithm TCSA($G,S,M,delay$)

```plaintext
1. Input $G=(V,E,M)$
2. Input MIT, MST, MC, ML
3. /* MIT is the maximum no of iterations, MST is the maximum number of iterations of stable solution. MC is the candidate set maximum number element and ML is the tabu length */
4. init_core = SelectRandom $V$
5. $H$ ← initialize tabu table
6. its ← 0 /* iterative times */
7. stds ← 0 /* counter of iterations of current situation that is not changed */
8. $G'$ ← transpose $(G)$
9. for $\forall v_i \in M$ in $G'$
10. $P_{d}(v_i,v_j)$ is the least delay path from $v_i$ to $v_j$, $v_i \in V$.
11. $P_{d}(s,v_i)$ is the least delay path from $s$ to $v_i$.
12. while (its<MIT and stds<MST) do
13. $\forall can_core \in CAN(cur_node)$ randomly choose can_core which satisfies the condition that can_core is not tabued and can_core is aspired by $S(can_core) < A(cur_core)$ where can_core $\in$ $N(cur_node)$
14. cur_best = $\min S(can_core)$
15. cur_best$\cdot$cur_best$\cdot$cur_best$\cdot$cur_best$\cdot$cur_best$\cdot$cur_best
16. cur_best = $\min S(can_core)$
17. best_core$\cdot$best_core$\cdot$best_core$\cdot$best_core$\cdot$best_core
18. if $S(best_core) < S(cur_best)$
19. $cur_best = best_core$
20. stds = stds + 1
21. cur_core = best_core
22. $H$ = update the tabu table
23. its = its + 1
24. stds = stds + 1
25. return cur_best
```
### F. Complexity Analysis of TCSA

The time complexity of TCSA is analyzed line by line as follows. The complexity of line 1 is $O(1)$. The shortest paths from source $v_i$ to all destinations are computed in line. The time complexity of line 2 is $O(|V|^2)$. In line 3, $O(|V|^2)$ is required to transpose adjacency matrix $G$ to its $G'$. The time complexity of line 4 is $O(|M| * |V|^2)$. The line 5 can be computed in $O(|M| * |V|)$. Line 6 requires $O(1)$. In line 8-9 neighboring nodes of the current node are stored in $N(cur\_core)$. Then a node is selected randomly from $N(cur\_core)$ and stored in $CAN(cur\_core)$. The complexity is $O(N(cur\_core) + \log |V|)$). Lines 10-12 are computed in $O(1)$. The complexity of lines 13-14 is $O(|CAN(cur\_core)| * |M|)$. The node with minimum fitness value is computed in line 15. The complexity is $O(|CAN(cur\_core)|)$. Lines 16 to 24 are computed in $O(1)$. Lines 8-24 are run its times under the while loop in line 7. Generally its $\leq |V|$, then the complexity of the loop is $O(it* (O(1) + O(2|E|/|V|) + O(|CAN(cur\_core)| * |M|) + O(|CAN(cur\_core)|))) \leq O(|E| + |V| + (|M| + I)*|V|^2)$ i.e. $O(|E| + |M| * |V|^2)$. Therefore, the complexity of the algorithm is $O(|E| + |M| * |V|^2)$.

### V. Simulation Results

To examine the performance of the algorithm (TCSA), we have implemented our proposed algorithm along with DDVCA [13] and AKC [15] in Visual C++. The experiments are performed on an Intel Core i3 @ 2.27 GHz and 2 GB RAM based platform.

The positions of the nodes are fixed randomly in a rectangle of size 4000 km x 2400 km. The Euclidean metric is then used to determine the distance between each pair of nodes. Edges are introduced between the pairs of nodes $u, v$ with a probability that depends on the distance between them. The edge probability is given by $P(u,v)=\beta \exp(-D(u,v)/\alpha L)$, where $D(u,v)$ is the distance from node $u$ to $v$ and $L$ is the maximum distance between any two points in the graph. The value of $\alpha$ controls the number of short links in the randomly generated network topology. The smaller the value of $\alpha$, the higher number of shorter links, $\beta$ controls the number of links in the randomly generated network topology. The lower the value of $\beta$, the larger the number of links, where $\alpha$ and $\beta$ are set to 0.8 and 0.7 respectively. The parameters of TCSA are $MIT=30$ and $MST=10$.

The link delay function $D(e)$ is defined as the propagation delay of the link, which is calculated by the equation $\frac{d(u,v)}{SCALE}$. The $SCALE$ is assumed to be 20 ms. The source node is selected randomly and destination nodes are picked up uniformly from the set of nodes chosen in the network topology. The delay bound $\Delta$ is set to be 1.5 times the minimum delay between the source and the farthest destination node. The algorithms are run on 100 randomly generated networks and the average of that is taken as the output.

#### A. Comparison on multicast delay-variations

The Fig. 1 and Fig. 2 show the simulation results of multicast delay-variations versus the number of nodes on a network. The destination nodes in a multicast group occupy 10% and 25% of the overall network nodes respectively. The multicast delay-variation of our proposed algorithm is found to be better than existing algorithms [13,15].

The Fig. 3 shows the multicast delay-variation for a network of 100 nodes. The multicast group size is between 10% and 70% of the overall nodes of the network. We observe that the multicast trees obtained by our proposed algorithm have an average multicast delay variation better than DDVCA and AKC algorithm. This is because the earlier algorithms select the best core node from a set of candidate nodes that have minimum delay variation whereas our algorithm searches the network with a dynamic neighborhood search technique to find the best core node.

---

Fig. 1 Comparison of multicast delay variation with 10 percent nodes as destinations.

Fig. 2 Comparison of multicast delay variation with 25 percent nodes as destinations.
Fig. 3 Comparison of multicast delay variation in a network of 100 nodes

Table 1: The performance efficiency of the TCSA in comparison with AKC algorithm

| No. of nodes (|V|) | 60 | 80 | 100 | 120 | 140 | 160 |
|----------------|----|----|-----|-----|-----|-----|
| |M|= 10% of |V|   |     |     |     |     |
| Reduced (%) | 50 | 51 | 50  | 43  | 38  | 40  |
| Unchanged (%)| 45 | 42 | 43  | 52  | 56  | 54  |
| Increased (%) | 5  | 7  | 7   | 5   | 6   | 6   |

As indicated in Figs.1-3, it is easily noticed that the proposed algorithm performs better than DDVCA[13] and AKC[15] when we take the average of 100 runs. However, Table 1 shows the performance efficiency of the proposed algorithm in comparison with AKC to show the performance of the proposed algorithm for each run. Table 1 shows that there exist rare cases where the multicast delay variation in TCSA becomes larger than that in AKC[15].

VI. CONCLUSION

The current core selection algorithms select the core from a set of candidate core nodes. Therefore, these algorithms sometimes fail to find the best core. To alleviate this problem, we proposed an algorithm based on tabu search for core selection in QoS multicast routing. To test the effectiveness of our algorithm we compare it with a number of other commonly used core selection algorithms. The simulation results reveal that the multicast tree generated by our algorithm has less delay-variation than that found by other algorithms. The mathematical time complexity of our proposed algorithm is comparable to the existing algorithms.

REFERENCES