EFFICIENT TREE CONSTRUCTION FOR THE MULTICAST PROBLEM

Giuseppe Di Fatta[1,2], **Giuseppe Lo Re**[1]
[1]CERE, Centro di Studio sulle Reti di Elaboratori, C.N.R.,
viale delle Scienze, 90128 Palermo, Italy
difatta, lore@cere.pa.cnr.it
[2]ICSI, International Computer Science Institute,
1947 Center Street, Suite 600, Berkeley, CA 94704-1198, USA
difatta@icsi.berkeley.edu

Resumo - Apresenta-se uma nova heurística para o problema de árvores de Steiner mínimas. O método proposto baseia-se na deteção de classes particulares de nós, os conjuntos de "nós quentes", que são usados para uma melhor aproximação das soluções ótimas. Propõe-se um algoritmo capaz de melhorar o desempenho de soluções heurísticas clássicas. Comparações são feitas com um método enumerativo e com métodos heurísticos clássicos.

Abstract - A new heuristic for the Steiner Minimal Tree problem is presented here. The method described is based on the detection of particular sets of nodes in networks, the "Hot Spot" sets, which are used to obtain better approximations of the optimal solutions. An algorithm is also proposed which is capable of improving the solutions obtained by classical heuristics, by means of node rearrangements in the solution tree. Classical heuristics and an enumerative method are used as comparison terms in the experimental analysis which demonstrates the goodness of the heuristics discussed in this paper.

Keywords: Steiner Tree Problem, Multicast Routing, Minimum Path Heuristics.

1. INTRODUCTION

New distributed applications, such as video-conference, and voice and video distribution, require simultaneous transmission of data to multiple destinations [1]. The network load caused by multimedia applications can be reduced using ad hoc multicast techniques, that is, by sending a message to its intended receivers, whenever possible through common paths. New standards for packet switched networks, such as Asynchronous Transfer Mode (ATM) and Frame Relay provide already support for multicasting.
A problem arises from these new exigencies, namely, to identify the best path capable of connecting source and destinations, minimising its total cost. This problem is well known under the name of Steiner Tree Problem [2, 8, 13], and in its classical formulation has been proved to be NP-complete. Many heuristics [9, 11, 13], capable of isolating sub-optimal solutions in polynomial time have been proposed in the past years. In this paper we present a new algorithm, which is capable of minimising the cost of the solution tree found by classical heuristics, and which improves the solution tree by arranging nodes and edges in the tree according to a minimisation criterion. The main topic of this paper is represented by a new heuristic for the Steiner Tree Problem in Networks (SPN) [13]. The heuristic proposed uses a particular set of nodes, the Hot Spot set. A hot spot vertex is a node which has intrinsic proprieties derived from network topology and terminal node distribution. An algorithm was designed with the aim of using effectively the Hot Spot set. This algorithm is based on the same incremental tree construction strategy as the Minimum Path Heuristic (MPH) algorithm [11], but it adopts a different metric and makes use of a temporary insertion of the hot spot nodes during the tree growing phase. The terminology and notation used here are described in Section 2, and some classical heuristics are summarized which will be used as comparison terms. Their performance guarantees and complexities are also presented. Section 3 deals with the process which improves the trees generated by classical heuristics (the "stirring" process). Section 4 defines the Hot Spot characteristics and section 5 describes the Hot Spot detection process, the tree construction and the complexity of our algorithm. The final part of the paper (section 6) is devoted to the experimental analysis, based on the results of a wide range of tests.

2. STEINER TREE HEURISTICS

In this section a short summary of previous Steiner Tree Heuristics is presented, and some basic notations and terms are introduced.
Given a graph $G = (V, A)$ with node set V ($|V| = n$) and edge set A ($|A| = a$), we will use $Z \subset V$ ($|Z| = m$) to indicate the subset of terminal nodes, $S = V - Z$ to indicate non-terminal nodes, and $z_0 \in Z$ the source node. Given a cost function $cost(e)$ with $e \in A$, we want to find a tree $T = (V_T, A_T)$ with $V_T \subseteq V$ and $A_T \subseteq A$, which spans the terminal nodes and has a minimal overall cost. The Steiner Tree Problem, as defined here, has been proved to be NP-complete. Many heuristics proposed in the past years have been capable of identifying sub-optimal solutions with polynomial complexities. Among these, the Pruned Dijkstra Heuristic (PDH) [13], the Minimum or Shortest Path Heuristic (SPH or MPH) [11], the Distance Network Heuristic (DNH) [13], the K-Shortest Path Heuristic (K-SPH)

[9], and so on. Some of the heuristics which we used as bases for our algorithm are briefly described here.
The simplest heuristic PDH comes from Dijkstra's Shortest Path Tree algorithm [5] which is capable of isolating the shortest paths from a given source to all the other nodes on the network. The PDH simply prunes all the S-vertices of degree one (one at time) from the shortest path tree (SPT) until none are left.
The DNH builds the distance network K_z induced by Z. It constructs the minimum spanning tree (MST) on the K_z network and replaces the virtual links with the real paths (the nodes and links of the initial network), thus obtaining G_z, a subgraph of the initial network. It then computes the minimum spanning tree (MST) on G_z and finally prunes all the S-vertices of degree one. The MPH builds a subtree of G in an incremental fashion: it starts off by selecting an arbitrary node among the terminal nodes (typically the source node) and then progressively adds the terminal node nearest to the tree, including the nodes and edges of the connecting path. The K-SPH is an improvement of the MPH algorithm. It builds a forest of subtrees joining together the closest nodes or subtrees until a single solution tree has been obtained.
The efficacy of these heuristics can be measured in terms of the ratio between the cost of the solutions they are able to identify, and the cost of the optimal solutions. Many algorithms capable of identifying the optimal solution tree have been proposed in the past. All of them are characterised by an exponential complexity. Among these, the Spanning Tree Enumeration Algorithm (STEA) [13] has been adopted to produce the optimal solution tree we have considered as comparison term.
Table 1. summarises the complexities and performance for the algorithms discussed above.

Algorithm	Performance Guarantee	Complexity
PDH	m	$O(n^2)$
DNH	2	$O(mn^2)$
MPH (SPH)	2	$O(mn^2)$
K-SPH	2	$O(mn^2)$
STEA	1	$O(m^2 2^{(n-m)} + n^3)$

Table 1. Performance Guarantee and Complexity

3. THE STIRRING PROCESS

The tree construction process of the MPH algorithm involves the addition of the nearest Z-node z_i to the transitory sub-tree T_i. Nevertheless, the minimum path between the selected node z_i and the successive sub-trees T_j (with $j > i$) does not remain constant during the tree growing process, because the insertion of further nodes into the tree could reduce the cost of the connecting path of a previously inserted node. For this reason, we considered the possibility of recalculating the "grafting" point of each node of the tree at the end of the MPH algorithm. By the term "grafting point" of a node n_i in a tree, we mean the closest ancestor of n_i which has at least two children ($degree \geq 3$). For any node n_i, we check the existence of a node n_k in the tree with $d(n_i, n_k) < d(n_i, n_a)$ where n_a is the current grafting point, and n_k does not belong to the path $< n_i, n_a >$ and is not a descendant of n_i. If such a node n_k is found in the tree, the path $< n_i, n_a >$ is replaced by the $< n_i, n_k >$ one. This procedure is executed for each node which is a leaf of the tree ($degree = 1$) or a grafting point ($degree \geq 3$). Path replacement may provoke the insertion of new nodes into the tree. Increasing the subset of nodes extends the solution space explored by the heuristic. The stirring process is repeated until no further improvement can be obtained in the solution tree. The process described here has been used in an improved version of the MPH algorithm, which will be referred to below as SMPH. In section 6.2 a comparison is made between the results obtained by the SMPH and the ones obtained by classical heuristics.

4. HOT SPOT DEFINITION

Classical heuristics use the terminal node set as their starting point in the construction of the solution tree. The solution trees generated by the various heuristics reveal significant differences in internal nodes. Most of the terminal nodes are leaves in the tree, i.e. they are nodes of degree one in the solution tree. Some of the nodes in networks have a high probability of representing relevant switching points to cover the multicast subset. In other words, such a node allows lots of terminals in a solution tree to be reached. If we could define a value for each node in a network, providing a measure of the number of terminal nodes which can be efficiently reached through it, then high value nodes would have a high probability of being present in the optimal solution tree. The definition of the characteristics of this kind of node is important in identifying them in the network, where each node can generally be used to reach any other, even if not very conveniently. The value attributed to each node should be related to the number of terminal nodes accessible from it at low cost, and this can be done using the sum of all the shortest path costs towards each terminal node. Value attribution determines node classification in the network. Nodes which have the highest values are selected to act as switching points, and they will be referred to below as "Hot Spots" (HS). The definition given above means that the Hot Spot set is heavily dependent on the terminal node set. Different terminal sets will generate different Hot Spot sets.
In Figure 1., a simple example is shown in which the classical heuristics MPH and KSPH fail to find the optimal solution, whilst the usage of Hot Spot nodes is shown to be effective. The network has four nodes, the source is the node s and the terminals are the nodes i and j. The execution of MPH on this network yields a solution tree with a cost equal to 200. However, if we consider the node h as Hot Spot, and require its inclusion in the solution tree, we obtain a lower cost, equal to 153. Other similar situations can obviously be imagined in more complex networks and topologies. In the following section we will discuss a method employed to detect and use the Hot Spot nodes.

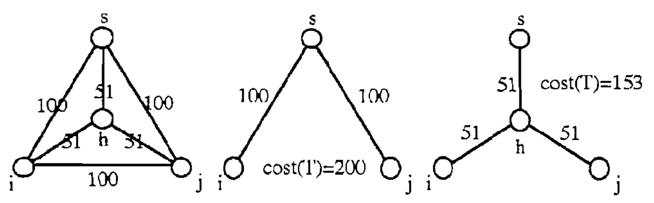

Figure 1. A four node graph and two different multicast trees

5. THE HOT SPOT HEURISTIC

The idea which is the foundation stone for our heuristic is based on the concept of score propagation among the nodes of the whole network. The concept which inspired this approach is related to the diffusion of a message from some intensive producers (sources) towards other network entities. These target entities receive messages from all sources and collect them to determine their own score. The nodes with the biggest scores will be eligible as Hot Spots. This simple idea has been adapted to suit network characteristics, and to the SMT problem. In this case, the terminal nodes represent the sources which generate the signals. The signal intensity I of the terminal nodes is determined as the maximum among the costs of the shortest paths between each terminal and all the other nodes of the network. Basically, signal propagation from the terminal m to the generic node v is carried out decreasing the source signal intensity by the cost of the shortest path between m and v. The score of the generic node v is evaluated as the collective strength of the signals from all the Z-nodes:

$$score(v) = \sum_{m \in Z} I - d(v,m) \qquad (1)$$

$$\text{where: } I = \max_{m \in Z, v \in V} \{d(m,v)\}$$

and $d(v,m)$ is the cost of the shortest path from node v to node m. In this way, a node which is powerfully influenced by lots of other terminal nodes can collect a high score, and this implies a high probability of being a critical switching point towards its nearest terminal nodes. Of all the nodes, we identify a subset of those which have the first highest h scores in the network. The h value depends on the number of terminal nodes and the network size. This method produces a sequence of Hot Spots (HS) which can be used in the construction of the solution tree. It should be noted that some of the selected Hot Spots may belong to the same neighborhood of the network because such an area might be characterised by the strong influence of lots of multicast nodes. The selection of more HS belonging to the same neighborhood allows more accurate scanning of the space of local solutions.

5.1. TREE CONSTRUCTION

In the previous section we discussed Hot Spot detection in a network. Now we are going to define a method for establishing a solution tree using these particular points. We adopted an incremental construction method as used by the Minimum Path Heuristic. Let E be the set of points given by the union of Z and H, where Z is the set of multicast nodes and H is the set of the Hot Spot points. We call E the set of the multicast nodes expanded by means of the H set. First, the algorithm selects an arbitrary node among the nodes of the expanded set E (typically the source) and then it progressively adds the other nodes of E to the tree according to the rule of the closest node. When a node is selected to be included into the transitory solution tree, a different approach is followed for the Z and the H-nodes. Terminal nodes m_i are inserted on a permanent basis. In contrast, the H-nodes h_i are inserted on a temporary basis. Analogously, the nodes and the edges of the respective connecting paths are considered either permanent or provisional. When a terminal node m_i is inserted into the tree by means of a grafting point h_i which is a temporary node, this latter node h_i becomes permanent together with all its temporary ancestors (Figure ??). In this way, the permanent setting of a H-vertex depends on its current usage in the solution tree. The insertion of a temporary node is not immediately advantageous for the transitory solution tree T. Nevertheless, there is a high probability of using that node as the grafting point for lots of terminal nodes. However, when a terminal node is added to the tree through a temporary node, the computation of its insertion cost must also take into account the cost of the temporary path. The adopted metric M performs this job, and defines the insertion cost of a node v through a temporary node h as:

$$d(v,T) = d(v,h) + f(h) \qquad (2)$$

where $f(h)$ depends on the cost of the temporary path connecting h to the tree. The insertion overcost $f(h)$ due to the temporary path avoids the usage of H-vertices which aren't actually cost-effective.
As previously discussed, the H-nodes have a high probability of reaching lots of terminals in a solution tree with a low cost path. In order to exploit this characteristic, we assume a value, called HIT, which is an estimation of the minimal number of terminal nodes reached through each of them in the final solution tree. This means that we expect a minimal performance from the Hot Spot vertices. The greater the value of HIT, the greater the risk of an over-valuation. We adopt the metric M to detect the next closest node to insert during tree construction. In order to take advantage of the H-vertices, the metric M uses the HIT constant. The temporary path overcost of the H-node h has to be subdivided among the Z-vertices we expect to reach through it, and it is expressed by:

$$f(h) = \frac{d(h,T)}{\text{HIT}}\,. \qquad (3)$$

Consequently, to calculate the distance between a node v and the tree T we need to calculate the minimum between two different distances d_1 and d_2 (4). The first one, d_1, is the classical node-to-tree distance calculated between the node v and the subset of the permanent nodes in the tree (5); the other one, d_2, is the distance calculated between the node v

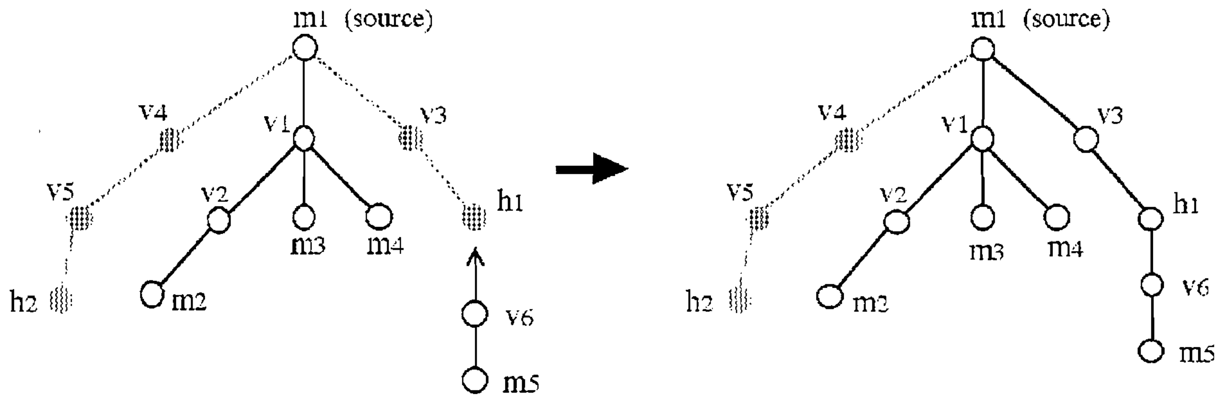

Figure 2. Permanent and temporary nodes in transitory trees

and the subset of the temporary nodes, as in (6).

$$d(v,T) = \min\{d_1(v,T_p), d_2(v,T_T)\} \quad (4)$$

where:

$$d_1(v,T_p) = \min_{p \in T_p}\{d(v,p)\} \quad (5)$$

and

$$d_2(v,T_t) = \min_{t \in T_t}\left\{d(v,t) + \frac{d(t,p_t)}{\text{HIT}}\right\} \quad (6)$$

$T_p \subseteq T$ is the subset of the permanent nodes in T; $T_t \subset T$ is the subset of the temporary nodes in T; p_t is the closest permanent ancestor of the temporary node t. The value of the constant HIT can be selected from the range between 2 and the number of all the terminal nodes, because a value of 1 would mean no actual usage of Hot Spots (as SMPH algorithm).
The insertion process ends when all the terminal vertices in the extended set E have been included. The construction process continues with the stirring phase, as described for the SMPH algorithm in section 3. In this case, the stirring process adopts the metric M and propagates the permanent or temporary node state at each change of the tree. Finally, all temporary nodes are pruned, thus obtaining the final solution tree.
The algorithm of the Hot Spot Heuristic (HSH) is presented as pseudocode in the next section.

5.2. HOT SPOT HEURISTIC ALGORITHM

```
Input:
    a graph $G = (V, A)$,
    a terminal node set $Z \subset V$ with $|Z| = m$,
    a source $z_0 \in Z$.
Output:
    a Steiner Tree $T = (V_T, A_T)$ rooted at $z_0$.

1. Score Propagation
    Determination of the Hot Spot set H ($|H| = h$)
    from the graph G and the terminal set Z.
2. Tree Construction
    Let be $E = Z \cup H$;
    Init $V_T \leftarrow \{z_0\}$ and $A_T \leftarrow \{\}$;
    while $E \cap Z \neq \{\}$
        find $v \in E$, $d(v,T) = \min\{d(e,T)\}_{e \in E}$;
        if ($v \in Z$) then insert in T nodes and edges
            of $ShortestPath(v,T)$ as "permanent";
        else insert in T nodes and edges
            of $ShortestPath(v,T)$ as "temporary";
        $E \leftarrow E - \{v\}$;
    end while.
3. Stirring
    while some change in $T = (V_T, A_T)$ occurs
        for each node $v \in V_T$ where $degree(v) \neq 2$
            let be $n_a$ its current grafting point
            if a better grafting point $n_k$ exists: i.e.
                $d(v, n_k) < d(v, n_a)$
            then replace the path $< v, n_a >$ with $< v, n_k >$;
        end for;
    end while.
4. Pruning
    while some change in $T = (V_T, A_T)$ occurs
        for each node $v \in V_T$
            if ($degree(v) = 1$) and ($v \notin Z$)
            then delete $v$ from $V_T$ and its edge from $A_T$;
    end while.
```

5.3. HOT SPOT HEURISTIC COMPLEXITY

Here we consider the operational complexity of the Hot Spot Heuristic algorithm. The Hot Spot generation process involves computation of the shortest path from each terminal to every vertex in the network. This can be done by computing a priori the SPT rooted in each terminal, in $O(mn^2)$.
The score propagation process involves a complexity of
The tree construction phase involves a complexity which is similar to the MPH one. The set of nodes to be inserted is expanded by the h Hot Spot nodes, so we obtain $O((m+h)^2 n + a)$.
The last phase in the algorithm is the stirring process. As described in the pseudocode, it searches for a better grafting point among the tree nodes (at most n). This is done for each

node in the tree which has not degree equal to two, i.e. in leaves and grafting points, which are at most $(2(m+h)-1)$ nodes. The whole process is repeated until no further changes occur (in the worst case n times). This results in $O((m+h)n^2)$, which is also the main complexity of the algorithm.

6. EXPERIMENTAL ANALYSIS

To generate the networks for our experiments we adopted the Tiers [6, 7] tool. It is capable of randomly generating graphs whose characteristics are those of real computer networks. It is important to ensure that the network models being used to test the algorithm bear some resemblance to the real networks for which the algorithm is designed.
The Tiers tool uses the classical Waxman model [12] for generating graphs, i.e. the nodes in the network are distributed at random across a Cartesian coordinates grid. The generic nodes u and v are connected by a link which is set according to the following probability law:

$$P(u,v) = \beta e^{\frac{-d(u,v)}{L\alpha}} \tag{7}$$

where $d(u,v)$ is the Euclidean distance between u and v, L is the maximum distance between two nodes, and α and β are parameters in the range $0 < \alpha, \beta < 1$. A large value of α increases the number of connections to nodes further away, whilst a large value of β increases the number of edges from each node. Furthermore, a series of modifications to the Waxman model make the generated networks more similar to real networks by using different hierarchies of nodes. Parameters are required to define the topological characteristics, such as the number of WAN, MAN and LAN, the number of nodes in each subnetwork, and the different connection degrees [7].
We generated different network sets to test our algorithms. These sets are characterised by different sizes (from 200 to 1000 network nodes), different topologies (only a wide WAN and multiple LAN), and different link redundancies. A high redundancy degree involves the presence of more complex solutions. We tested different terminal node locations choosing them from nodes with high or low connectivity degrees.

6.1. ENUMERATION METHOD

The knowledge gained about the optimal solution allows us to make a precise comparison of the different heuristics. This is necessary to understand how many times the classical heuristics fail in terms of optimal solution detection, meaning the times when it is possible to obtain better solutions. We therefore implemented the Spanning Tree Enumeration algorithm [13]. Furthermore, because the NP-complete nature of the Steiner Minimal Tree problem makes the optimal solution search a computationally heavy task, we used the reduction rules, as described in [13], in order to increase the size of the explored networks. By means of these rules a particular instance of the STP can often be reduced to a smaller one by examining local properties of the networks. Networks with two hundred nodes and ten terminal nodes are reduced to networks with 50 nodes and 8 terminals on average.

6.2. THE STIRRING PROCESS AND THE HOT SPOT HEURISTIC VS CLASSICAL HEURISTICS

The first experiment was devoted to determining the actual performance gain over the Minimum Path Heuristic introduced by the Stirring process, as described in section 3. Furthermore, we tested the performance of the Hot Spot Heuristic in comparison with the MPH and SMPH. We selected networks and terminal sets for which we were able to find the optimal solution in a short time. We carried out a series of tests over three hundred networks, each with 200 nodes and 10 terminal nodes. Among the classical heuristics to be found in the literature, we chose and implemented the following three: the Distance Network Heuristic (DNH), the Minimum Path Heuristic (MPH) and the Kruskal-based Shortest Path Heuristic (KSPH).

The experiment shows that our Stirring method as applied to the MPH algorithm (SMPH) and the Hot Spot heuristic (HSH) give the best results of all heuristics considered in the most of cases. The results are summarized in Figure 3.. The left bars were obtained considering only the optimal solutions for each heuristic. The right ones were obtained taking into consideration the sub-optimal ones, where the sub-optimal is the minimal cost solution among those found by the heuristics. As shown in the chart, the Stirring method (SMPH) was able to find the optimal solution in 72% of the cases as compared with the 53% MPH score and the 58% KSPH score. In 19% of the cases, SMPH manages to find the optimal solution which the MPH couldn't identify, and to improve the MPH solution in about 27% of the tested networks. These differences are to be attributed exclusively to the Stirring process. HSH computed and adopted a set of special nodes in the network, the Hot Spot node set. In this case we used at most 20 Hot Spot nodes and $HIT = 3$. As clearly shown by the histograms in the Figure 3., the HSH algorithm was able to isolate the optimal solution in 87% of the sample networks, which means the HSH algorithm detected the optimal solution 34% more often than MPH. In other words, our heuristic failed to find the optimal solution in only 13% of the tested networks. We have to take into account that the HSH algorithm is an intrinsic improvement of the MPH and SMPH. To enlarge the space of the sub-optimal solutions found, we also considered the KSPH and the DNH algorithms. Because of the different approach used, these algorithms were able to find a better solution than HSH in 4% of the tested networks. Nevertheless, HSH achieved a sub-optimal solution in a good 96% of the cases against the KSPH's 64%, which is the best result among the classical heuristics we tested.

6.3. THE HOT SPOT NUMBER

In the chart in Figure 4., the results obtained when the number of the Hot Spot nodes adopted is varying in the set $\{0, 5, 10, 20\}$ are shown. Optimal and sub-optimal cases are plotted.

The more effective improvement 9% (from 72% to 81%)

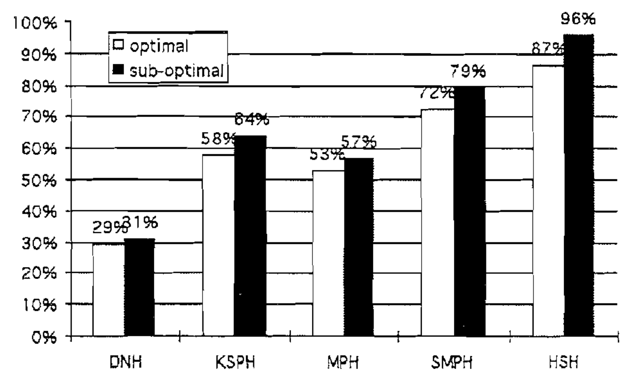

Figure 3. dnh, ksph, mph, smph and hsh optimal and suboptimal solutions

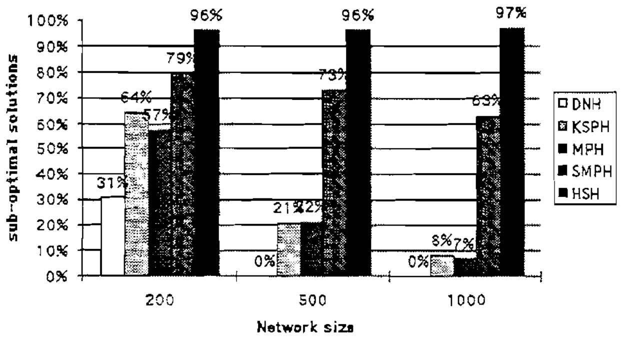

Figure 5. Suboptimal solutions in 200, 500 and 1000 node networks

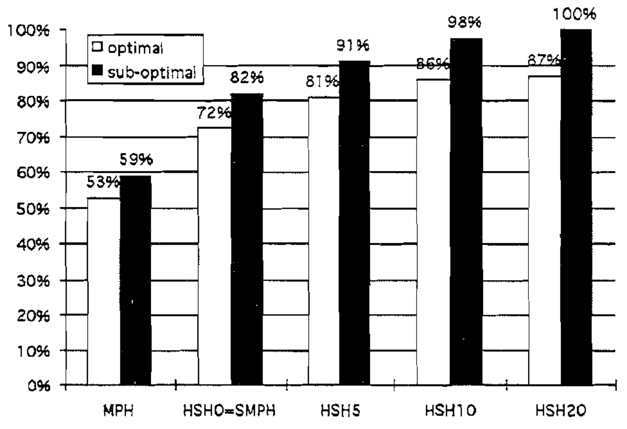

Figure 4. Varying the Hot Spots number: optimal and suboptimal solutions

was achieved by using the five HS node set (HSH5), while further improvements were respectively 14% (from 72% to 86%) and 15% (from 72% to 87%).

The experimental results show that the usage of a few Hot Spot nodes is sufficient to achieve a high percentage of improved cases.

6.4. THE NETWORK SIZE

The previous experiments have been carried out over networks of limited complexity due to the small number (200) of nodes and terminals in the networks. Their main aim was to compare the optimal solution with the heuristic ones.

Heuristic performances in more realistic networks, characterised by a bigger size, are compared here. Obviously, in such a case we are not able to find the optimal solution, so only sub-optimal solutions are compared.

In this experiment we varied the network size from 200 to 1000 nodes, and the terminal node number from 10 to 50. We fixed the value of HIT to 3 and the maximum HS number to 20. The results related to these tests are reported in Figure 5.. The Hot Spot Heuristic performance was almost constant when the network size increased, whilst the classical heuristics showed a severe failure in the bigger networks. When the network size increased, both the contribution of the Stirring process and Hot Spot insertion became more effective.

The big gap (7%-63%) between MPH and SMPH in the 1000 node networks provides excellent evidence of the effectiveness of the Stirring method. However, since HSH finds better solutions than SMPH in 34% of cases, it seems clear that HSH is particularly suitable for real networks.

6.5. THE HOT SPOT GOODNESS

Given the intrinsic nature of the Hot Spot nodes, there is a high probability that they belong to Minimal Steiner Trees. The detection of such nodes prior to the tree construction process is the basis of our heuristic. To verify the validity of the Hot Spot detection process, and therefore the actual effectiveness of the Hot Spots, we carried out the following experiment to determine how many Hot Spots were present in the optimal solution tree. The network size is 200 nodes, for which we are able to find the optimal tree by means of the enumeration method. In the test, we varied the size of the Hot Spot set from 1 to 20 nodes. The different lines in the chart represent the different sizes tested (see Figure 6.). The x axis represents the actual number of Hot Spots found in the optimal tree. The y axis represents the percentage of cases where the optimal tree contains that number of Hot Spot nodes. For example, the "3HS" line shows that in 80% of the cases, all three Hot Spot nodes were found in the optimal tree, in 19% only two of the three nodes were found, and in a percentage near to zero only one node was found, whilst a situation in which none was present in the solution tree never occurred.

Looking at the chart in Figure 6. it is easy to observe that in the smaller Hot Spot sets (until "9HS") there is a probability not equal to zero that all the nodes in the set are present in the optimal tree. This is not true in the bigger sets, where the probability of a high number of nodes belonging to the optimal tree is null.

The HSH algorithm can be considered merely as an improvement of the SMPH one because it introduces new nodes during the tree construction phase. A test of Hot Spot goodness is to use this algorithm with a set of randomly

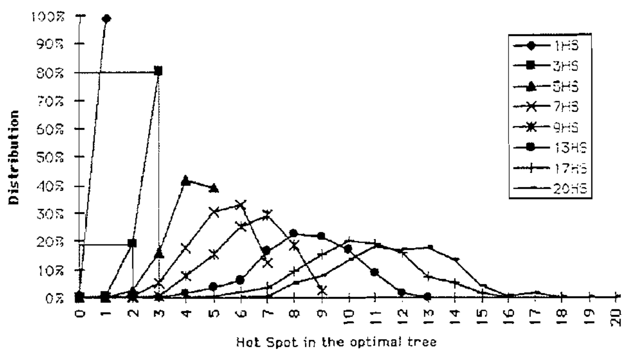

Figure 6. HS distribution in the optimal solution trees.

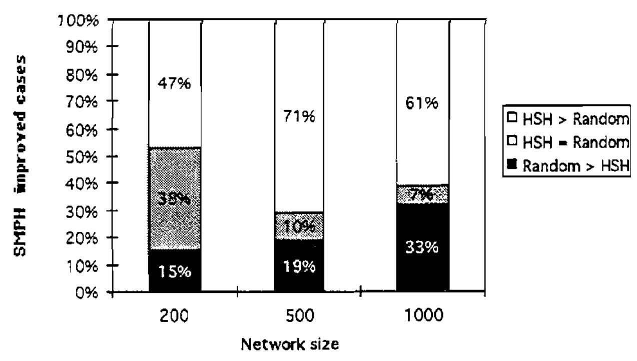

Figure 7. Hot Spot set vs random set.

generated nodes. The next experiment was carried out running the SMPH and two versions of the HSH. The first one generated the Hot Spot set according to the propagation method (see section 5), whilst the second one produced the Hot Spot set in a random way. The columns in Figure 7. show the percentage of the SMPH solutions which were improved by the two different versions (HSH and Random). The lower areas indicate the cases improved by the random set, the middle areas illustrate the cases where both found the same sub-optimal solution, and the upper areas those cases in which the Hot Spot set involved the best solution detection. Finally, we determine the influence of a Hot Spot node with respect to its position in the Hot Spot set. The propagation method generated the Hot Spot set as an ordered list of nodes according to their scores (see section 5). The chart (Figure 8.) shows the percentage of SMPH solutions improved by the hot spot in the position reported along the x axis. The figure highlights the performance of the first hot spot vertex which alone would guarantee a good percentage of improvements. The result reported above confirms the effectiveness of the Hot Spot set detection process and allows the usage of few nodes to improve the solutions of previous heuristics.

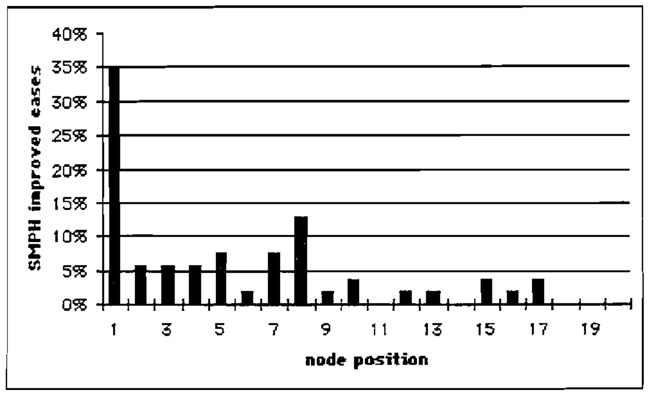

Figure 8. Effectiveness distribution in the Hot Spot set.

7. CONCLUSIONS

We have performed experimental investigations into the problem of multicast tree generation by introducing a new heuristic. It introduces the concept of the Hot Spot node which is a point of the network with a relevant topological role. A selection method for such nodes and a tree construction process capable of taking advantage of them are proposed. We carried out comparative experiments and the results demonstrated the goodness of the method. The effectiveness of the Hot Spot selection process has been demonstrated by means of an "a posteriori" analysis of their presence in the optimal solutions and also by means of a comparison with the solutions obtained using random sets of nodes.

REFERENCES

[1] K. Bharath-Kumar and Jaffe, "Routing to Multiple Destinations in Computer Networks", IEEE Trans on Commun. vol. COM-31, n.3, pp.343 - 351, Mar. 1983

[2] F. Bauer, A. Varma, "Distributed Algorithms for Multicast Path Setup in Data Networks", IEEE/ACM Transactions on Networking, Vol. 4, N. 2, pp. 181 - 191, April 1996

[3] F. Bauer, A. Varma, "ARIES: A Rearrangeable Inexpensive Edge-Based On-Line Steiner Algorithm", IEEE Journal on Selected Areas in Communications, vol. 15, N.3, pp. 382 - 397, April 1997

[4] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. G. Liu, L. Wei, "The PIM Architecture for Wide-Area Multicast Routing", IEEE/ACM Transactions on Networking, Vol. 4, N. 2, pp. 153 - 162, April 1996

[5] E.W. Dijkstra, "A Note on Two Problems in Connection with Graphs", Numer. Math., vol. 1 1959 , 269 - 271

[6] M. B. Doar, I. Leslie, "How Bad is naive multicast routing?", Proc. of IEEE Infocom, pp. 82 - 89, San Francisco CA, April1993,

[7] M. B. Doar, "TIERS User Manual", v. 1.1 (ftp://ftp.nexen.com/pub/papers/)

[8] V. P. Kompella, J. C. Pasquale, G. C. Polyzos, "Multicast Routing for Multimedia Communication", IEEE/ACM Transactions on Networking, Vol. 1, N. 3, pp. 286 - 292, June 1993

[9] J. Kruskal, "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem", Proc. Amer. Math. Soc., vol. 7, pp. 48 - 50, 1956.

[10] S. Ramanathan, "Multicast Tree Generation in Networks with Asymmetric Links", IEEE/ACM Transactions on Networking, Vol. 4, N. 4, pp. 558 - 568, August 1996

[11] H. Takahashi, A. Matsuyama, "An Approximate Solution for the Steiner Problem in Graph", Math. Japonica, vol. 24, n. 6, pp. 573 - 577, 1980.

[12] B. Waxman, "Routing of Multipoint Connections", Journal on Selected Areas in Communications, vol. 6, N.9, pp. 1617 - 1622, Dec. 1988.

[13] P. Winter, "Steiner Problem in Networks: A Survey", Networks vol. 17 (1987) pp. 129 - 167

Giuseppe Di Fatta received the Laurea degree in electronic engineering with specialization in computer science from the University of Palermo, Italy, in 1995. In 1996 he received a two years grant from the Italian National Research Council. Presently, he is affiliated as research fellow at the International Computer Science Institute, Berkeley California, and he is also working toward a Ph. D. at the University of Palermo. His research interests are in Network Protocols, Distributed Systems, and Multicast Routing.

Giuseppe Lo Re received the Laurea degree in computer science from the University of Pisa, Italy, in 1990, and the Ph.D. degree in Electronic, Computer Science and Telecommunication Engineering at the University of Palermo in 1999. In 1991, he joined the Italian National Research Council (CNR) at the "Study Centre for Computer Networks" of Palermo, where he currently holds a researcher position. His current research interests are in the area of computer communication networks and distributed systems. He is a member of the Association for Computer Machinery (ACM), and of the Institute of Electrical and Electronics Engineers (IEEE).