Multicast Routing for Multimedia Communication

Vachaspathi P. Kompella
Joseph C. Pasquale
George C. Polyzos

Computer Systems Laboratory
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093-0114

Abstract

We present heuristics for multicast tree construction for communication that depends on: (i) bounded end-to-end delay along the paths from source to each destination, *and* (ii) minimum cost of the multicast tree, where edge cost and edge delay can be independent metrics. This problem of computing such a constrained multicast tree is NP-complete. We show that the heuristics demonstrate good average case behavior in terms of cost, as determined through simulations on a large number of graphs.
Keywords: Multicast, Steiner tree, delay-bounded communication

1 Introduction

Multicasting is the simultaneous transmission of data to multiple destinations. Although support for multicasting has been slow, it is now being viewed as a very important facility in networks especially because video and audio applications can use such a feature effectively. The most popular solutions to multicast routing involve tree construction. There are two reasons for basing efficient multicast routes on trees: (i) the data can be transmitted in parallel to the various destinations along the branches of the tree, and (ii) a minimum number of copies of the data are transmitted, with duplication of data being necessary only at forks in the tree.

Algorithms for constructing multicast trees have been developed with two optimization goals in mind. The first is the minimum average path delay, $\mathcal{D}_T$, which is the average of the minimum path delays from the source to each of the destinations in the multicast group. A minimum average path delay tree can be constructed in $O(n^2)$ time using Dijkstra's shortest path algorithm [2], where n is the number of nodes in the graph. The second measure of efficiency is in terms of the cost of the multicast tree, $\mathcal{C}_T$, which is the sum of the costs on the edges in the multicast tree. The least cost tree is called a *Steiner tree* [5], and the problem of finding a Steiner tree is known to be *NP*-complete [7]. Furthermore, the problem remains *NP*-complete, even if edges have unit cost [4].

Several algorithms that construct low-cost multicast routes [1, 6, 13, 14] are based on heuristics for approximate Steiner trees [10, 11, 12]. Empirical observations show that the heuristics produce near optimal trees quickly. The algorithms take polynomial time, ranging from $O(n^3)$ to $O(n^4)$. Furthermore, they produce solutions that are provably within twice the cost of the optimal solution.

The two measures $\mathcal{C}_T$ and $\mathcal{D}_T$ are individually insufficient to characterize a good multicast route for multimedia communication. The performance of such a multicast route is determined by two factors: (i) bounded end-to-end delay along the individual paths from source to each destination, *and* (ii) minimum cost of the multicast tree, for example, in terms of network bandwidth utilization. In our formulation, a multicast tree is a *constrained Steiner tree*, *i.e.*, a delay-bounded minimum cost tree, where the delay bound is specified by the application performing the multicast.

Kadaba and Jaffe [6] studied a problem that involves optimizing on both the cost and delay measures of the multicast tree. They investigated how a compromise could be struck between minimizing $\mathcal{D}_T$ and $\mathcal{C}_T$. There are two primary differences between our approach and that of [6]. The fundamental difference is that we assume that edge cost and edge delay are different functions. For example, edge cost could be a measure of the amount of buffer space or channel bandwidth used, and edge delay could be a combination of propagation, transmission, and queueing delay. A second major difference is that we are trying to construct a constrained minimum cost tree, where the constraint is on the individual path delay, rather than trying to minimize the average path delay to all destinations.

2 The Constrained Steiner Tree

We now formulate the constrained Steiner tree (CST) problem as follows. Given a graph[1] $G = (V, E)$ with node set V and edge set E, we define two weight functions, $\mathcal{C}(e)$ and $\mathcal{D}(e)$, on edge e. $\mathcal{C}(e)$ is a positive real cost function on e, and $\mathcal{D}(e)$ is a positive integer delay function on e. On this graph, we have a source node s and a set of destination nodes S, called the multicast group. Given a delay tolerance Δ, a *constrained spanning tree* T is a tree, rooted at s, that spans the nodes in S, such that for each node v in S, the delay on the path from s to v is bounded above by Δ. Formally, for each $v \in S$, if $P(s,v)$ is the path in T from s to v, $\sum_{e \in P(s,v)} \mathcal{D}(e) < \Delta$. We shall assume that Δ is a bounded integer value. The constrained Steiner tree can now be described as a constrained spanning tree such that $\sum_{e \in T} \mathcal{C}(e)$ is minimized. We note that this problem is *NP*-complete since it reduces to the standard Steiner tree problem [8].

[1] We consider only simple graphs, i.e., graphs with only one edge between two nodes.

3 Heuristics for the Constrained Steiner Tree

In the following, we assume that the source has all the information necessary to construct the multicast tree. We call these algorithms *source-based* routing algorithms. We define the following terms. A *constrained cheapest path* between v and w is the least cost path from v to w that has delay less than Δ. We denote the cost on such a path by $\mathcal{P}_C(v,w)$, and the delay on it by $\mathcal{P}_D(v,w)$. A *closure graph* G' on a set of nodes N is a complete graph on the nodes in N with edge cost between nodes $v, w \in N$ equal to $\mathcal{P}_C(v,w)$ and edge delay $\mathcal{P}_D(v,w)$.

In order to compute the closure graph, G', we first determine the constrained cheapest paths between all pairs of nodes in the set $S \cup \{s\}$. Although this problem is *NP*-complete [4] for arbitrary values of Δ, we assume Δ to be a bounded integer, and therefore the solution takes polynomial time in the size of the graph. We compute the all-pairs constrained cheapest paths using a dynamic programming approach similar to Floyd's shortest path algorithm [3]. We define $\mathcal{C}_d(v,w)$ to be the cost of the cheapest path from v to w with delay exactly d. If there are multiple cheapest constrained paths with the same cost, then the one with the least delay is chosen. We can formulate $\mathcal{C}_d(v,w)$ and $\mathcal{P}_C(v,w)$ as follows,

$$\mathcal{C}_d(v,w) = \min_{u \in V} \{\mathcal{C}_{d-\mathcal{D}(u,w)}(v,u) + \mathcal{C}(u,w)\} \tag{1}$$
$$\mathcal{P}_C(v,w) = \min_{d < \Delta} \mathcal{C}_d(v,w) \tag{2}$$

$\mathcal{P}_D(v,w)$ is then determined by the constrained cheapest path that corresponds to $\mathcal{P}_C(v,w)$. Thus, we can construct the closure graph G' on the nodes in the set $S \cup \{s\}$. Note that G' is also a simple graph.

The second step is to construct a constrained spanning tree of G'. We use a greedy approach to add edges to a subtree of the constrained spanning tree until all the destination nodes are covered. Assume v is in the tree constructed thus far, and that we are considering whether to include some edge adjacent to v. We have considered the following two selection functions:

$$f_{CD}(v,w) = \frac{\mathcal{P}_C(v,w)}{\Delta - (\mathcal{P}(v) + \mathcal{P}_D(v,w))}$$
$$f_C(v,w) = \mathcal{P}_C(v,w)$$

provided the delay from source to w is within the delay bound, where $\mathcal{P}(v)$ = delay on the path from s to v in the spanning tree constructed thus far[2]. The third step consists of expanding the edges of the constrained spanning tree into the constrained cheapest paths they represent, and remove any loops that may be caused by this expansion. f_{CD} and f_C give rise to two source-based heuristics, CST_{CD}, and CST_C, respectively.

CST_{CD}, uses the selection function f_{CD}, which explicitly uses both cost and delay in its functional form. It tries to choose low cost edges, but modulates the choice by trying to pick edges that maximize the residual delay. This increases the chances of extending the path through this edge, and beyond to another destination. The idea is to reduce the cost of the tree through path sharing. However, this heuristic has a tendency to optimize on delay also, in that it may find paths with delays far lower than Δ, at the expense of added cost to the tree.

CST_C, minimizes f_C, thereby trying to construct the cheapest tree possible, while ensuring that the delay bound is met. This tends to minimize the cost of the tree without unduly minimizing the delay.[3]

[2] One of the problems with G' is that it presents a distorted view of G. For example, in Fig.2, the constrained cheapest path from F to H passes through E. The closure graph does not reflect this, and has two edges, FE and FH. This inaccuracy leads to spurious conflicts of choices between edges that share paths in G. We can improve this picture by keeping track of the nodes on a constrained cheapest path. Then we can mark all these nodes as being in the tree when that constrained cheapest path is chosen. This leads to improved performance, because these conflicts about edge cost are resolved. Both the heuristics take advantage of this optimization.

[3] We present below simulation results that show that the CST_C has better average performance than CST_{CD}. We have

Figs. 2 through 5 show the workings of the two heuristics, which we compare with the tree of shortest delay paths to the destinations, shown in Fig. 6.

We note that CST_{CD} and CST_C always produce a constrained spanning tree, if one exists. The following lemma is true for source-based techniques using a closure graph.

Lemma: 1 A constrained multicast tree exists if and only if there are k edges incident on s with finite cost in the constrained closure graph, where $k = |S|$.

Proof: For a solution to exist, there must be at least one path from s to each destination v. Thus, the closure graph has to have one edge from s to v, for each destination v, representing the constrained cheapest path between those two nodes. Since there are k destinations, there must be k edges out of s. On the other hand, if there are k edges out of s, then, since there are exactly k destinations, there is one edge to each destination from s. Since each edge represents a constrained cheapest path, there is a path of delay less than Δ from s to each destination, and hence a solution exists.

Theorem: 1 CST_{CD} and CST_C find a solution if and only if a solution exists.

Proof: If either heuristic is able to cover all the k vertices, then we know two facts. Firstly, the algorithm produces a tree that spans all the k vertices. Secondly, no delay condition has been violated. Thus, there must be a path from s to each destination under the delay constraint, which means that a solution exists. On the other hand, if a solution exists, then by Lemma 1, there is at least one edge from s to each destination, such that the edge has bounded delay. Thus, there is at least one tree in G' such that its edges yield finite values using the selection function. This is the star out of s to each destination. Hence, the heuristics will find at least one tree, the star graph.

4 Performance Analysis of the Heuristics

We can define the CST for a given problem as follows. Of all the possible constrained spanning trees that can be constructed, the one with the least cost is the constrained Steiner tree. The optimal algorithm, OPT, enumerates all constrained spanning trees, and finds the least cost tree. We describe the performance of heuristic H in terms of δ_H, the deviation from the optimal, defined as follows:

$$\delta_H \triangleq \frac{T_H - T_O}{T_O}$$

where T_H is the cost of the tree using heuristic H and T_O is the cost of the optimal tree.

4.1 Empirical Performance Analysis

In order to fairly evaluate these heuristics, we decided to run them on randomly generated graphs with low average degree, which would better represent the topologies of common point-to-point networks, *e.g.*, the NSFNET. Initial measurements with small graphs showed that OPT and CST_C had comparable performance (see Fig.7, and [8]). For large graphs, finding the optimal solution is impractical, so we used CST_C as a point of reference, and defined the metric $\widehat{\delta}_H$ as follows:

$$\widehat{\delta}_H = \frac{T_H - T_{CST_C}}{T_{CST_C}}$$

We also compared the performance of the shortest delay tree created by Dijkstra's shortest path algorithm, SPT, with unnecessary branches pruned.

devised two distributed algorithms for constructing constrained Steiner trees which use the same two selection functions. Interestingly, for the distributed algorithms, we find that the heuristic that uses f_{CD} works better than the one using f_C[9].

The nodes in the graphs were randomly placed in a unit square, and the edge delays were proportional to the Euclidean distance separating the endpoints. We used both unit edge costs and random edge cost generated uniformly from the set $\{1, \ldots, 10\}$. The random edge costs match typical values for costs used in the NSFNET backbone network. However, the results indicate that the properties of the heuristics and their relative performance remain the same for both unit costs and random costs. The graphs we considered in the evaluation of the heuristics had between 50 and 100 nodes, and an average degree between 5 and 15. Each experiment generated R graphs with identical parameters: number of nodes, maximum degree of each node, delay bound, and size of the multicast group. Each graph was then checked to ensure that a solution existed. The confidence interval for $\widehat{\delta}_H$ at the 95% confidence level is about 1-2% for all the graphs shown.

Fig. 8 and Fig. 9 show that CST_{CD} performs marginally worse than CST_C. As the multicast group increases in size, the algorithms CST_{CD} and CST_C converge to the minimum spanning tree. This is borne out by Fig. 10. This is a desirable property, since it means that the heuristics converge to the optimal solution for large group sizes. Finally, Fig. 11 shows that as the delay tolerance increases, the performance of both source-based heuristics converge. This is because when Δ is far larger than the path delays, then f_{CD} converges to f_C.

The main point to be made here is that *SPT*, the shortest delay tree algorithm, produces trees with consistently high costs, between 70 and 80% more than CST_C, as seen in Figs. 8, 9 and 10. Thus, we believe that when interactive multipoint communication becomes more critical, it will be essential to have a routing policy that is not based purely on delay, but also takes into account other factors.

4.2 Complexity of the Heuristics

An analysis of the source-based heuristics shows that the majority of the time is spent in computing the constrained cheapest paths between all nodes. Eqns. (1) and (2) take $O(n^3\Delta)$ time to compute, since the computation loops over all pairs of nodes, and over all intermediate nodes, like Floyd's shortest path algorithm. However, it also has to loop over all the possible values of delay from 1 to Δ. Constructing the constrained minimum spanning tree on the closure graph with k nodes takes $O(k^3)$ time, and expanding the tree into the constrained spanning tree solution takes $O(k\,n)$ time, for k edges each to be expanded into at most n edges, and to remove loops. For comparison, Dijkstra's shortest paths algorithm takes $O(n^2)$ time [2].

5 Conclusions

Multimedia communications researchers are more and more aware of the inadequacies of existing network protocols and algorithms to handle continuous media such as audio and video. We have addressed one aspect of the problem, by presenting routing algorithms that take into account quality of service parameters required to make multimedia communication successful.

The analysis presented in this paper shows that while the problem of optimal constrained multicast routing we have considered is intractable, there are fast heuristics that produce good solutions. They can scale to large sized graphs and, on the average, can still provide near optimal routes that minimize cost (in terms of network resources expended to support a multicast) and limit path delays. Assuming accurate network status is available to the source, we devised two heuristics that select edges to construct a constrained multicast tree. One heuristic uses an explicit function of edge cost and edge delay, while the other uses a function of the cost of an edge.

One important result of this study is that if adequate global information is available to the source, then the source-based heuristics we have proposed produce much better results than shortest delay routing.

References

[1] C.-H. Chow. On multicast path finding algorithms. *Proc. of IEEE INFOCOM '92*, pages 1274–1283, 1990.

[2] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[3] R.W. Floyd. Algorithm 97: Shortest paths. *Comm. of the ACM*, 5:345, 1962.

[4] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Co., New York, 1979.

[5] S.L. Hakimi. Steiner's problem in graphs and its implications. *Networks*, 1:113–133, 1971.

[6] B.K. Kadaba and J.M. Jaffe. Routing to multiple destinations in computer networks. *IEEE Trans. on Comm.*, COM-31(3):343–351, Mar. 1983.

[7] R.M. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[8] V.P. Kompella, J.C. Pasquale, and G.C. Polyzos. Multicasting for multimedia applications. *Proc. of IEEE INFOCOM '92*, May 1992.

[9] V.P. Kompella, J.C. Pasquale, and G.C. Polyzos. Two distributed algorithms for the constrained Steiner tree problem. Technical Report CSL-1005-92, Computer Systems Laboratory, University of California, San Diego, Oct. 1992.

[10] L. Kou, G. Markowsky, and L. Berman. A fast algorithm for Steiner trees. *Acta Informatica*, (15):141–145, 1981.

[11] V.J. Rayward-Smith. The computation of nearly minimal Steiner trees in graphs. *Intl. J. Math. Educ. Sci. Tech.*, 14(1):15–23, 1983.

[12] H. Takahashi and A. Matsuyama. An approximate solution for the Steiner problem in graphs. *Math. Japonica*, 6:573–577, 1980.

[13] D.W. Wall. *Mechanisms for Broadcast and Selective Broadcast.* PhD thesis, Elect. Engg. Dept., Stanford University, Jun. 1980.

[14] B.M. Waxman. Routing of multipoint connections. *IEEE J. on Selected Areas in Comm.*, 6(9):1617–1622, Dec. 1988.

Appendix: The Constrained Steiner Tree Algorithm

/*
 * Let $G = (V, E)$ describe the network topology
 * s = source node
 * S = multicast group
 * Δ = delay constraint
 */
Multicast $(G(V, E), s, S, \Delta)$
begin
    /* Compute the cheapest constrained paths between all nodes in $S \cup \{s\}$ */
    $V' \leftarrow S \cup \{s\}$
    for each $v, w \in V'$ **do**
    begin
        $\mathcal{P}_C[v, w] \leftarrow$ cost of cheapest constrained path from v to w
        $\mathcal{P}_D[v, w] \leftarrow$ path delay along cheapest constrained path from v to w
    end
    /* C = set of nodes already visited */
    /* $P[v]$ = path delay from s to v in the tree */
    /* T = spanning tree on the closure graph */
    $C = \{s\}$
    $\mathcal{P}[s] = 0$
    $T = \emptyset$
    /* until all nodes in V' have been spanned */
    while $(C \neq V')$ **do**
    begin
        $min = \infty$
        for each $v \in C$ **do**
        begin
            for each $w \in V' \setminus C$ **do**
            begin
                /* if delay from s to w is within limits, consider edge (v, w) as
                 * a candidate and compute the selection function
                 * $f_s(v, w) = f_{CD}(v, w)$ or $f_C(v, w)$.
                 * $f_{CD}(v, w) = \frac{\mathcal{P}_C(v,w)}{\Delta - (\mathcal{P}(v) + \mathcal{P}_D(v,w))}$ if $\mathcal{P}(v) + \mathcal{P}_D(v, w) < \Delta$
                 * $f_C = \mathcal{P}_C(v, w)$ if $\mathcal{P}(v) < \Delta$
                 * For further optimization, use a better edge cost after conflict resolution (see footnote 2)
                 */
                if $(f_s(v, w) < min)$ **then**
                begin
                    $nextedge = (v, w)$
                    $min = f_s(v, w)$
                end /* **if** */
            end /* **for** */
        end /* **for** */
        $C = C \cup \{w\}$
        $\mathcal{P}[w] = \mathcal{P}[v] + \mathcal{P}_D[v, w]$
        $T = T \cup \{nextedge\}$
    end /* **while** */
end /* Multicast */

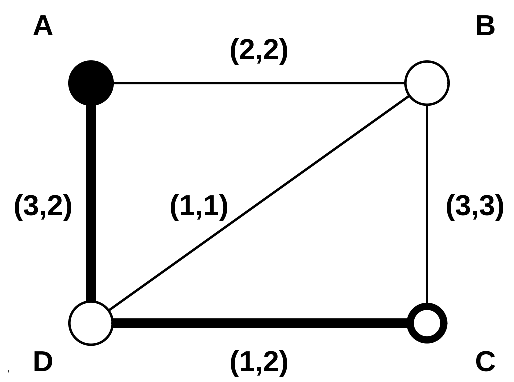

Figure 1: Constrained cheapest path from A to C with $\Delta = 6$. Figures along edges are (cost, delay).

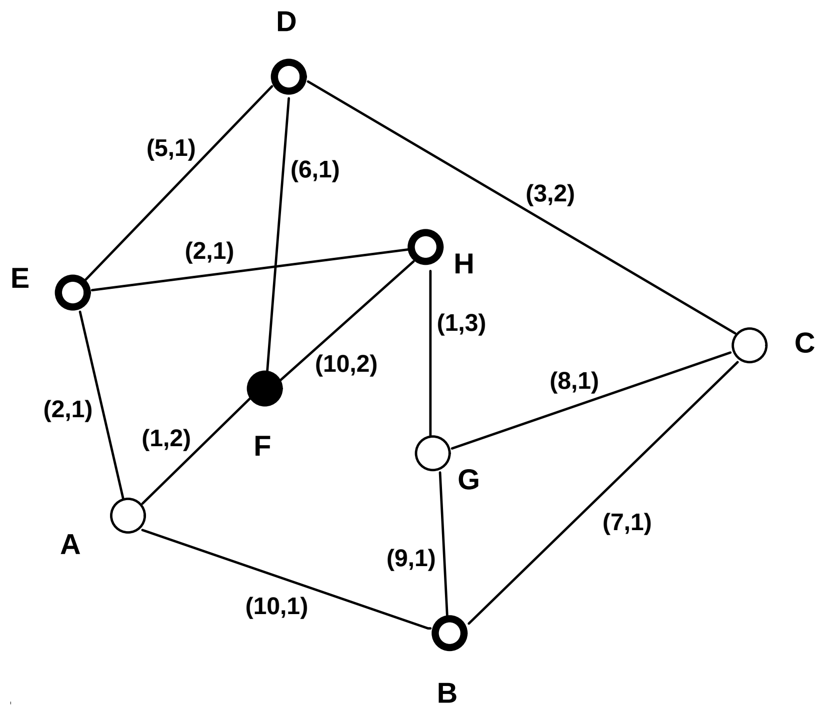

Figure 2: The graph used in presenting the heuristics.

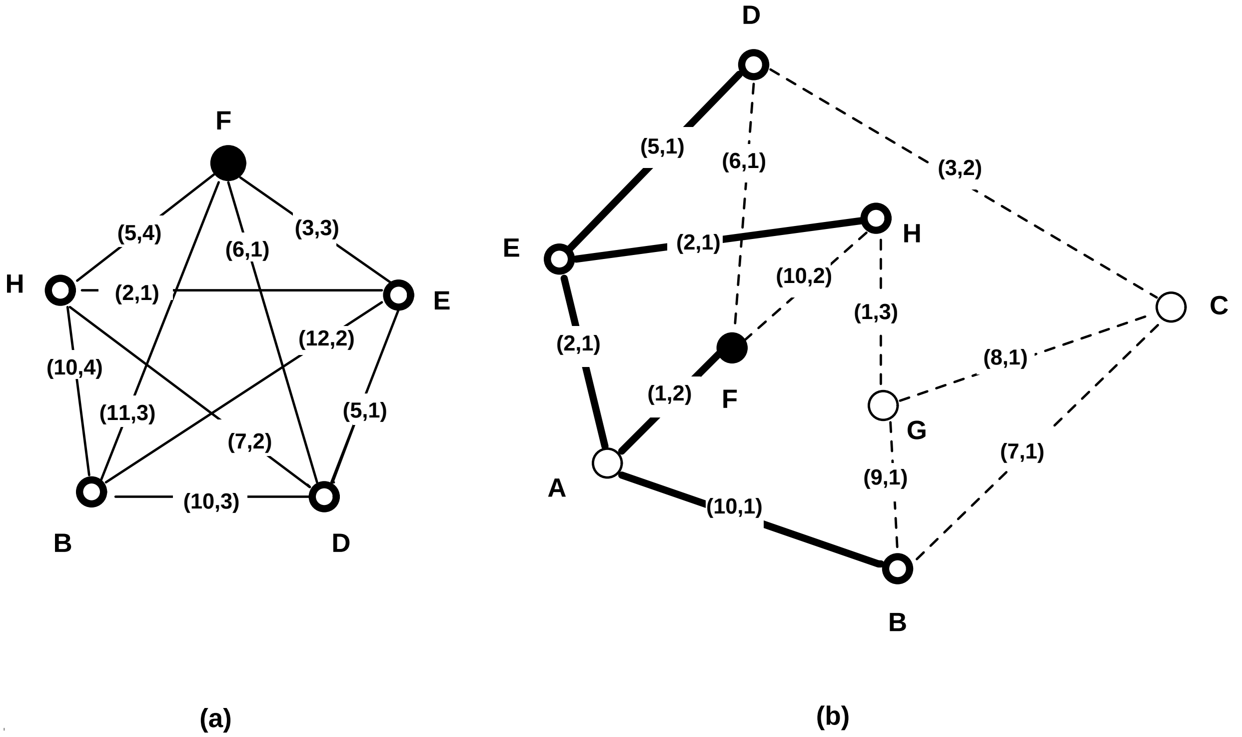

Figure 3: (a) The closure graph on S = {B,D,E,H} with $\Delta = 5$, and s = F. (b) The optimal constrained Steiner tree.

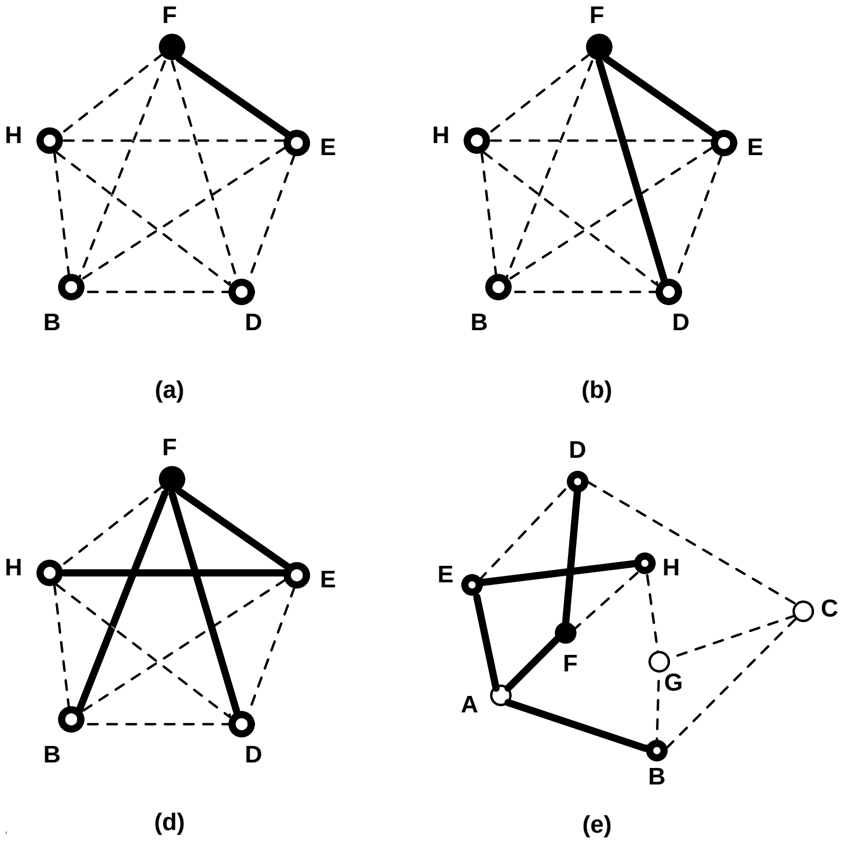

Figure 4: (a)-(d) Four stages in constructing the spanning tree using f_{CD}. (e) The constrained spanning tree using CST_{CD}.

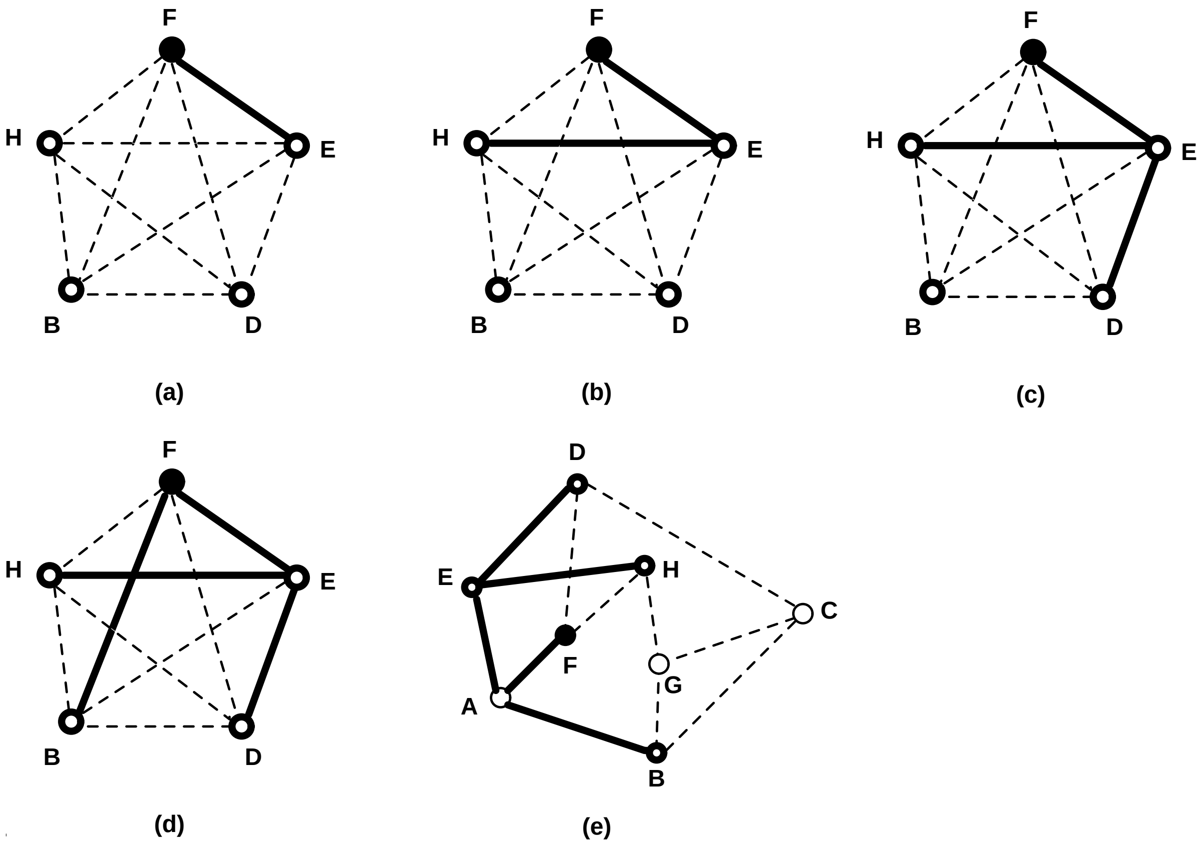

Figure 5: (a)-(d) Four stages in constructing the spanning tree using f_C. (e) The constrained spanning tree using CST_C.

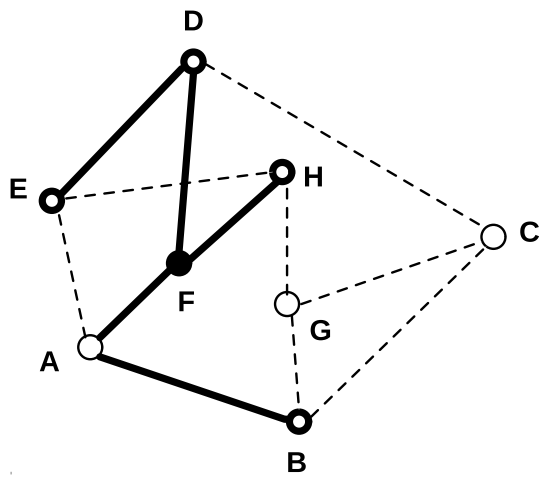

Figure 6: The tree produced by using the shortest delay paths from source to destinations.

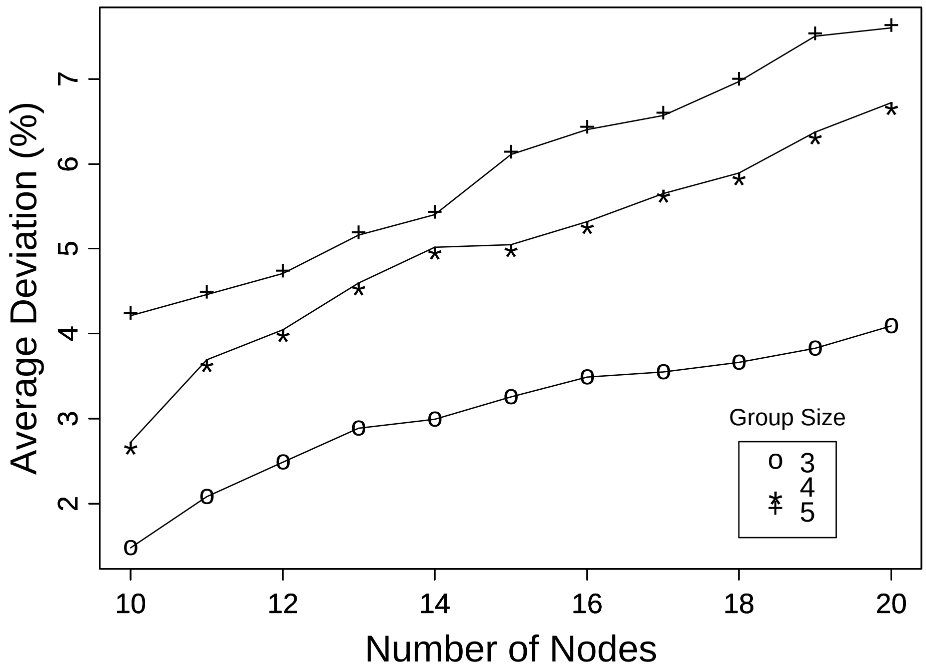

Figure 7: Average deviation δ vs. number of nodes, for three group sizes, comparing CST_C vs. OPT for small graphs, for $\Delta = 25$.

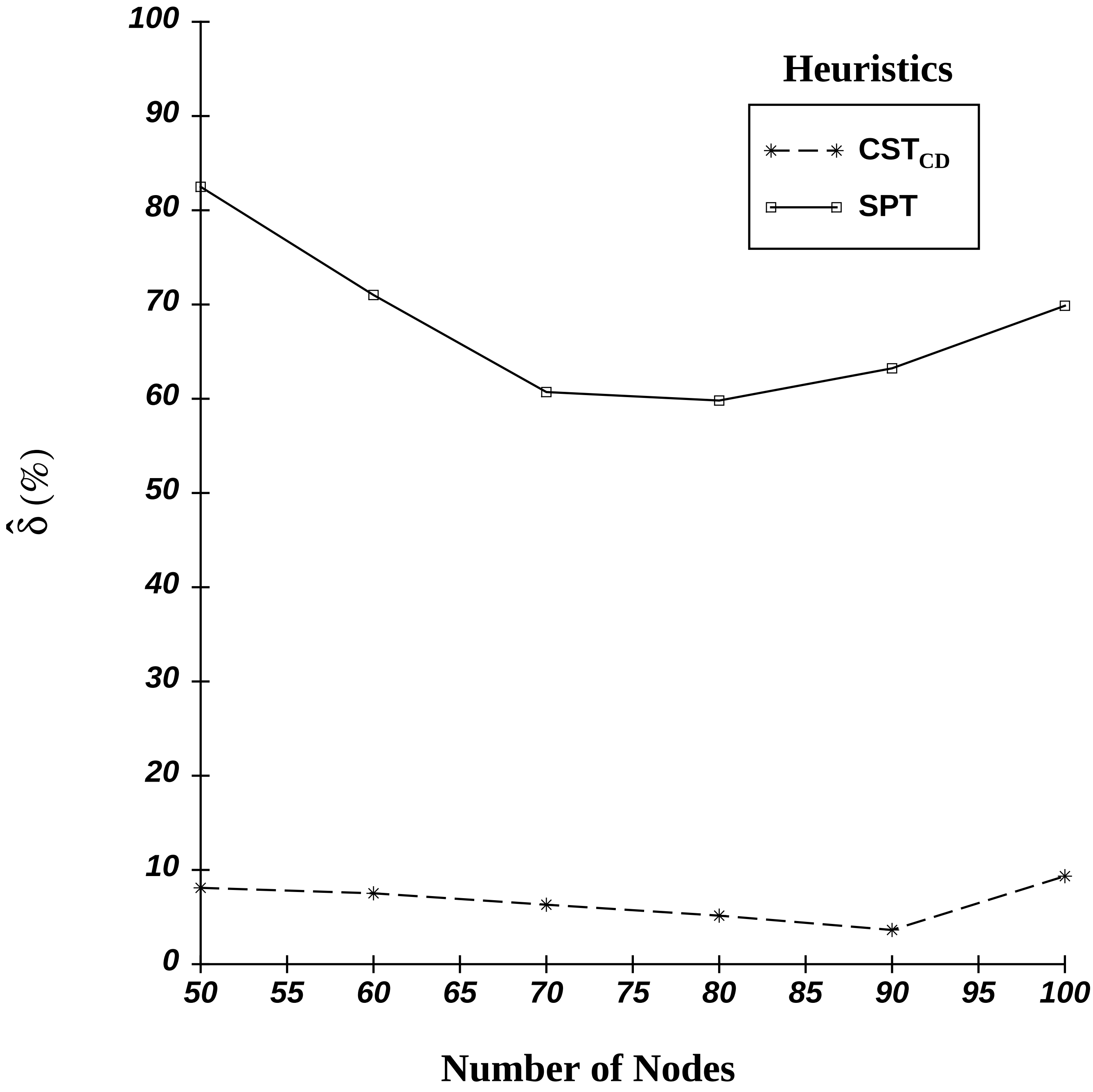

Figure 8: Average deviation $\widehat{\delta}$ vs. number of nodes, for group size = 15, $\Delta = 80$ and maximum degree = 15, w.r.t. CST_C.

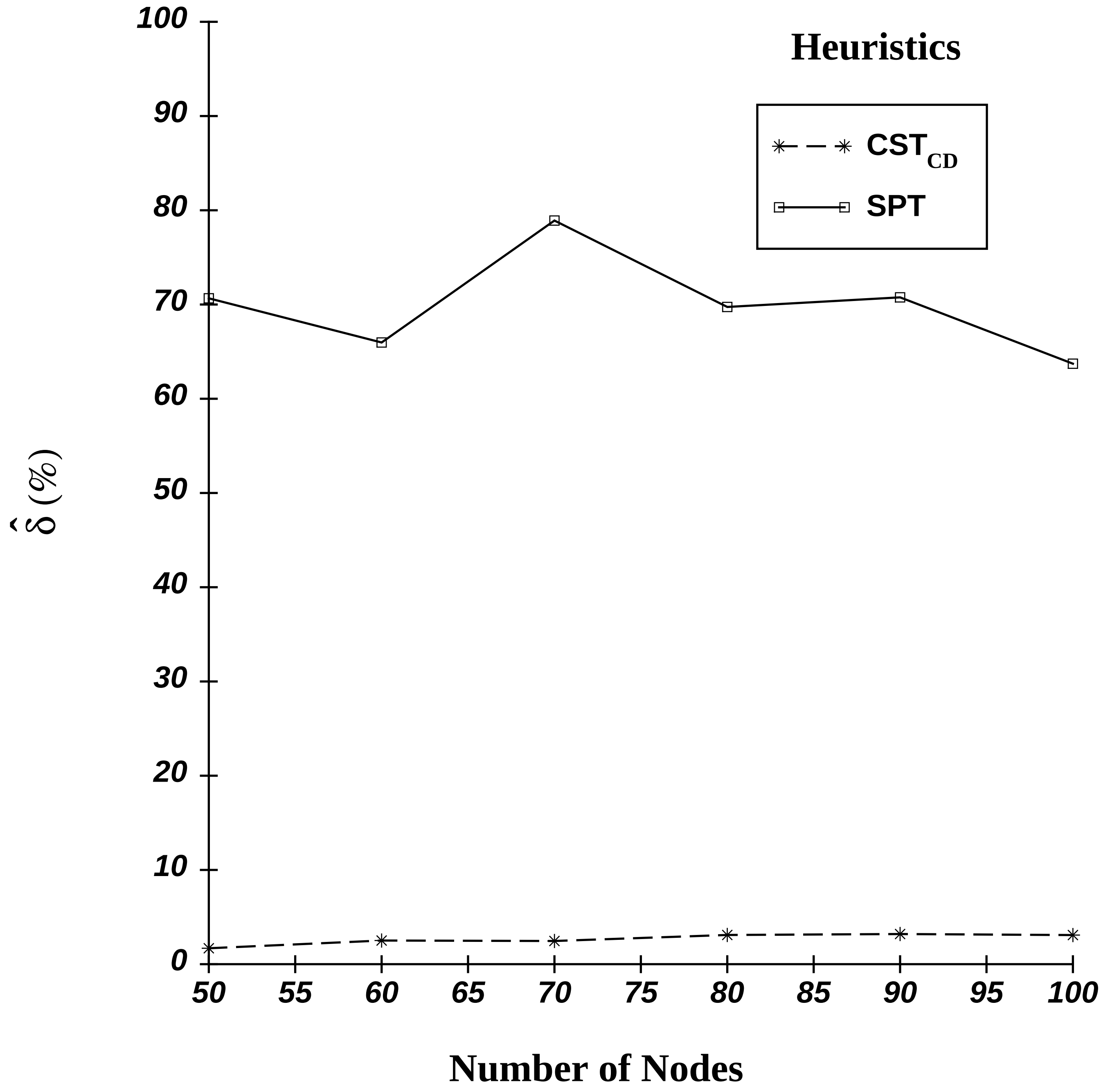

Figure 9: Average deviation $\widehat{\delta}$ vs. number of nodes, for group size = 15, Δ = 80, and maximum degree = 30, w.r.t. CST_C.

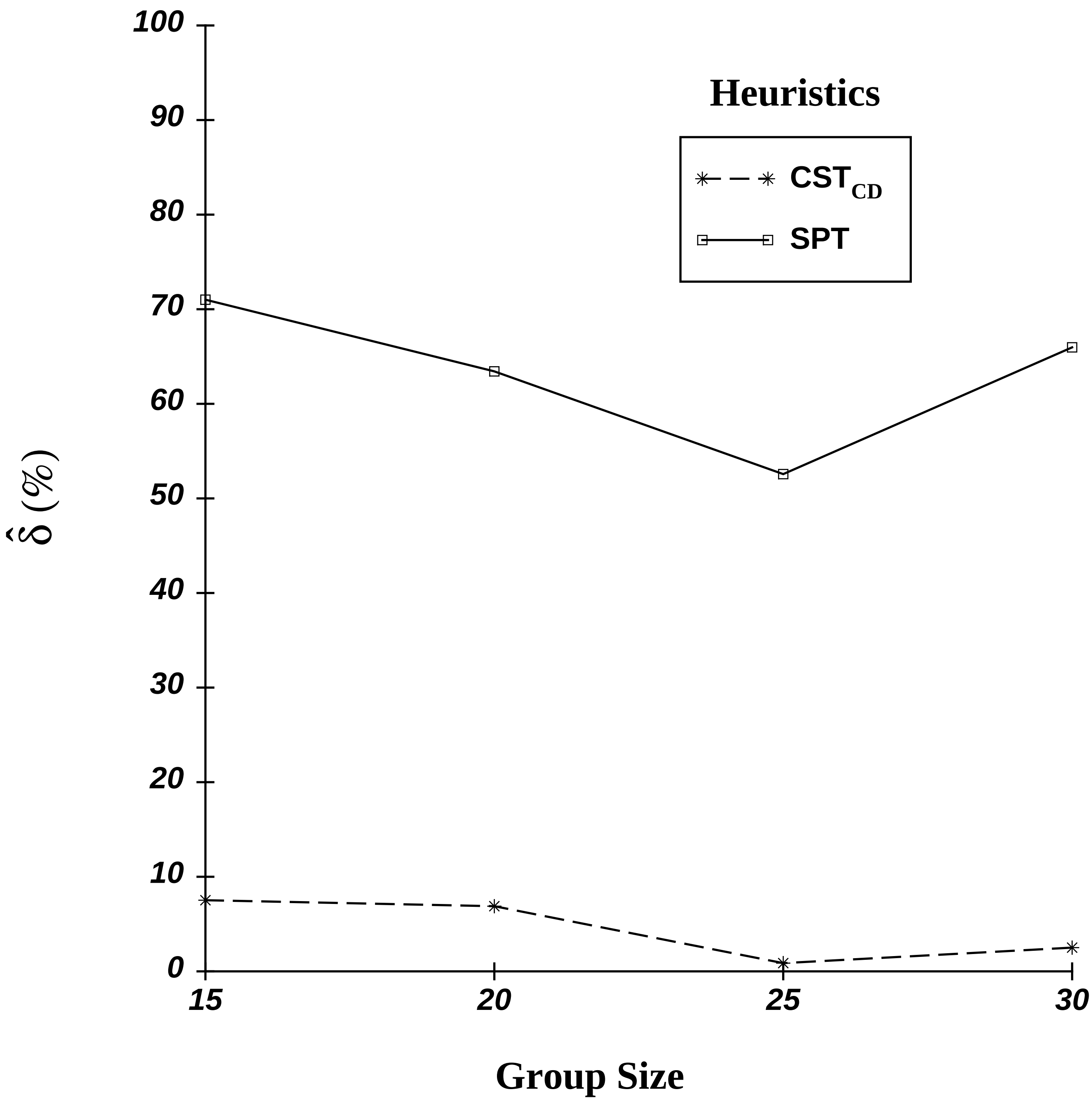

Figure 10: Average deviation $\widehat{\delta}$ vs. group size, for number of nodes = 60, and Δ = 80, with average degree of each node = 5, w.r.t. CST_C.

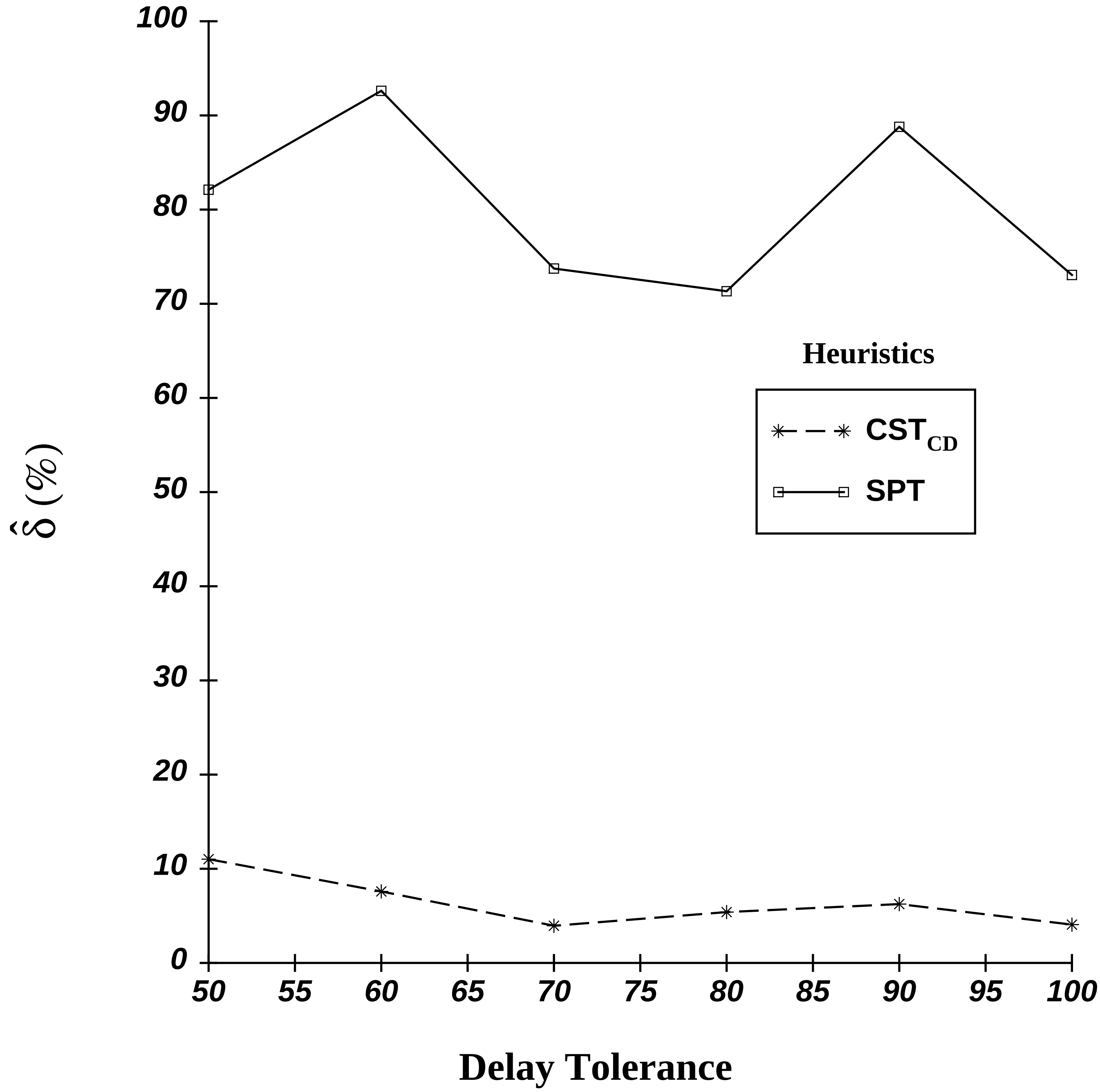

Figure 11: Average deviation $\widehat{\delta}$ vs. delay tolerance for number of nodes = 50, and group size = 20, with average degree of each node = 10, w.r.t. CST_C.