Path layout in ATM Networks - A Survey

Shmuel Zaks
Department of Computer Science
Technion, Haifa 32000, Israel
(email: zaks@cs.technion.ac.il)

Abstract

This paper surveys recent results in the area of virtual path layout in ATM networks. We present a model for the theoretical study of these layouts; the model amounts to covering the network with simple paths, under various constraints. The constraints are the *hop count* (the number of paths traversed between the vertices that have to communicate), the *load* (the number of paths that share an edge or a vertex), and the *stretch factor* (the total length traversed between pairs of vertices, reduced by the distance between them). We focus on the one-to-all (or broadcast) and the all-to-all communication problems. The results are positive (design and analysis of virtual path layout schemes, including recursive constructions, greedy and dynamic programming algorithms), and negative (lower bounds and NP-hardness). The results are presented for a variety of cases, and involving various parameters in such layouts.

1 Introduction

1.1 Motivation

The advent of fiber optic media has dramatically changed the classical views on the role and structure of digital communication networks. Specifically, the sharp distinction between telephone networks, cable television networks, and computer networks, has been replaced by a unified approach.

The most prevalent solution for this new network challenge is called *Asynchronous Transfer Mode*, or ATM for short; this is thoroughly described in the literature (see, e.g., [**?**, **?**, **?**]). ATM is based on relatively small fixed-size packets, termed *cells*. Each cell is routed independently, based on two small routing fields at the cell header, called *virtual channel index* (VCI) and *virtual path index* (VPI). At each intermediate switch, each of these fields serves as an index to a corresponding routing table; the routing is carried out in accordance to the predetermined information in the appropriate entries.

Routing in ATM is *hierarchical* in the sense that the VCI of a cell is ignored as long as its VPI is not null. This algorithm effectively creates two types of predetermined simple routes in the network - namely, routes that are based on VPIs, called *virtual paths* or VPs, and routes based on VCIs and VPIs, called *virtual channels* or VCs. VCs are used for connecting

network users (e.g., a telephone call); VP s are used for simplifying network management — in particular, routing of VCs . Thus, the route of a VC may be viewed as a concatenation of complete VP s.

As far as the mathematical model is concerned, for a given communication network, the VP s form a *virtual network* on top of the physical one, which we term the *virtual path layout* (VPL for short), on the same vertices, but with a different set of edges, which is typically a superset of the original edges. Each VC is a simple path in this virtual network.

The VP layout must satisfy certain conditions to guarantee important performance aspects of the network (see [?, ?] for technical justification of the model for ATM networks). In particular, there are restrictions on the following parameters:

The load: The number of virtual edges or vertices that share any physical edge. This number determines the size of the VP routing tables, since at each incoming port which a VP goes through, a separate entry is allocated for routing cells that belong to the VP (see [?] for a detailed description of the routing mechanism in ATM). As the ATM standard [?] limits the maximum size of VP routing tables to 4096 entries, this resource is critical in networks with a few hundreds of vertices (see [?] for a justification).

The hop count: The number of VPs which comprise the path of a VC in the virtual graph. This parameter determines the efficiency of the setup of a VC since the routing tables at the end of each VP must be updated to support the new VC. The importance of a low hop count to the efficiency of the network is very high, especially for data applications [?, ?, ?].

The stretch factor: The ratio between the length of the path that a VC takes in the physical graph and the shortest possible path between its endpoints. This parameter controls the efficiency of the utilization of the network. Most results in the literature deal with a stretch factor of one, which is the case where the connection is done along shortest paths.

In many works (e.g., [?, ?, ?, ?]), a general routing problem is solved using a simpler sub-problem as a building block. In this sub-problem it is required to enable routing between all vertices to a single vertex, rather than between any pair of vertices. This restricted problem for the ATM VP layout problem is termed the *rooted*, or *one-to-all*, *VPL* problem [?].

This paper surveys some recent results for the one-to-all and the all-to-all problems. The results are usually of two types: either bounding the (maximum or average) load $\mathcal{L}$ as a function of the other parameters (usually the number of vertices N and a bound $\mathcal{H}$ on the hop count), or bounding the maximum hop count $\mathcal{H}$ as a function of the other parameters (usually the number of vertices N and a bound $\mathcal{L}$ on the maximum load). For this latter problem, we comment that, for the one-to-all problem this is equivalent to determining the *radius* of the graph, formed by the virtual paths, that has to be embedded in the physical network; for the all-to-all problem, this is equivalent to determining the *diameter* of this graph.

1.2 Related Works

A few works have tackled the VP layout problem; some of these works used empirical techniques [?, ?], while others used theoretical analysis [?, ?]; However, none of these works has

attempted to combinatorially characterize the optimal solution, in order to achieve a tight upper bound for the problem. In addition, most of these works have considered only one of the relevant performance measures, namely the worst-case load measure. Of particular practical interest is the weighted hop count measure, since this determines the expected time for setting up a connection between a pair of users, given the relative frequency of connection requests between network vertices. A similar problem was empirically handled in [?].

The VP layout problem is closely related to graph-embedding problems, since in both cases it is required to embed one graph into another graph. However, while in most embedding problems both graphs are given, here we are given only the physical (host) graph, and we can *choose* the embedded graph, in addition to the choice of the embedding itself.

Most of the performance parameters are also different between these cases:

- While the association between the host graph and the embedded graph is made by the *dilation* parameter in embedding problems, it is made here by the stretch factor. In other words, while for embedding problems it is important to minimize the length of each individual embedded edge, for this kind of problems it is important to minimize the length of paths.

- The *hop count* parameter is closely related to the *distance* in the virtual graph. However, while the distance is defined with respect to only one of the graphs, the hop count depends on the physical graph as well, unless the stretch factor is unbounded.

- The *edge load* parameter is identical to the *congestion* in embedding problems. The different terminology is due to the loaded meaning of congestion in the communication literature.

These differences have a significant impact on the techniques and results in this model.

Another problem related to those we survey here is that of keeping small routing tables for routing in conventional computer networks. This problem has been widely studied [?, ?, ?, ?, ?, ?, ?], and has yielded interesting graph decompositions and structures. It differs, however, from the problems we treat here in some major aspects which deemed most of these solutions impractical for our purposes. The main difference stems from the fact that in our setting there is no flexibility as to the routing scheme itself since it is already determined by the ATM standard [?].

A related criterion, minimizing the *number* of VPs to achieve a required maximal hop count, is discussed in [?] mainly for database optimization purposes, where it yields very different results. For ATM, however, this criterion is of less importance since no such global constraint exists.

The notion of *forwarding index* was studied for communication networks (see, e.g., [?, ?]). The forwarding index in a given network with a given set of paths corresponds to the load in our case; in [?] the authors study load on vertices, while in [?] they study load on both vertices and edges. However, these studies (that deal with the all-to-all problem) do not consider the notion of hop count.

The rest of this survey is organized as follows. In Section 2 we present the mathematical model. Section 3 contains the survey of research in this area, while in Section 4 we present in more detail some of the recent results concerning chain networks. We conclude in Section 5 with a summary of the results and a list of open problems.

2 The Model

In this section we present the mathematical model, that was presented in [?, ?, ?]. We model the underlying communication network as an undirected graph $G = (V, E)$, where the set V of vertices corresponds to the set of switches, and the set E of edges to the set of physical links between them. In addition, we are given a set ζ of pairs of distinct nodes in V; a communication needs to be established between each of the pairs. A system $(G = (V, E), \zeta)$ has been termed a *connection network* in [?, ?]. We concentrate on two extreme cases:

- The *one-to-all* case: in this case a connection is required from one specified vertex to all others; namely, $\zeta = \{(r, u) | u \in V, u \neq r\}$, where r is the specified vertex, usually termed the *root*.
- The *all-to-all* case: in this case a connection is required between all pairs of vertices; namely, $\zeta = \{(v, u) | v, u \in V, u \neq v\}$.

In the following definitions, the network is meant to be G, while the set ζ corresponds to either of the two cases above.

Definition 2.1 *A* virtual path layout (VPL for short) Ψ *is a collection of simple paths in* G*, termed* virtual paths *(*VP*s for short).*

Definition 2.2 *The* load $\mathcal{L}(e)$ *of an edge* $e \in E$ *in a* VPL Ψ *is the number of* VP*s* $\psi \in \Psi$ *that include* e*.* [1] *The* load $\mathcal{L}(v)$ *of a vertex* $v \in V$ *in a* VPL Ψ *is the number of* VP*s* $\psi \in \Psi$ *that include* v*.*

Definition 2.3 *The* maximal edge load $\mathcal{L}_{max}(\Psi)$ *of a* VPL Ψ *is* $\max_{e \in E} \mathcal{L}(e)$*. The* maximal vertex load *of a* VPL Ψ *is* $\max_{v \in V} \mathcal{L}(e)$*.*

Unless otherwise specified, the loads referred to in this survey are edge loads.

Definition 2.4 *The* average (edge) load *of a* VPL Ψ *is*

$$\mathcal{L}_{avg}(\Psi) \equiv \frac{1}{|E|} \sum_{e \in E} \mathcal{L}(e).$$

Definition 2.5 *The* hop count $\mathcal{H}(u, v)$ *between two vertices* $u, v \in V$ *in a* VPL Ψ *is the minimum number of* VP*s whose concatenation forms a path in* G *connecting* u *and* v*. If no such* VP*s exist, define* $\mathcal{H}(u, v) \equiv \infty$*.*

Definition 2.6 *The* maximal hop count *of a* VPL Ψ *is*

$$\mathcal{H}_{max}(\Psi) \equiv \max_{(u,v) \in \zeta} \{\mathcal{H}(u, v)\}.$$

Definition 2.7 *Let* $\psi = (u, v)$ *be a* VP*. Then the* dilation *of* ψ*, denoted* $|\psi|$*, is the number of physical links that* ψ *traverses. Let* Ψ *be a* VPL*, then the* total load *of* Ψ *is* $\mathcal{L}_{tot}(\Psi) \equiv \sum_{\psi \in \Psi} |\psi|$*.*

[1] This notion is also referred to as its *congestion*, or *cutwidth* (as used in other references).

In both problems, in order to minimize the load, one can use a VPL Ψ which has a VP on each physical link, i.e., $\mathcal{L}_{max}(\Psi) = 1$, however such a layout has a large ($O(N)$) hop count. The other extreme is connecting a direct VP from the root to each other vertex, yielding $\mathcal{H}_{max} = 1$ but a large load ($O(N)$ for the one-to-all problem and $O(N^2)$ for the all-to-all problem). This paper discusses recent results for intermediate cases, which suggest trade-offs between these two extreme cases, for various network topologies.

3 Brief Survey

In this section we briefly summarize recent results in this area. All of them use the model presented in Section 2. Some of the results are described in more details in Section 4.

We discuss the problems of one-to-all and all-to-all layouts. As it will become apparent from the discussion, these problems are closely related to the radius and the diameter, respectively, of the graph that represents the virtual paths that we are embedding within our physical network (see especially the discussion in Section 4).

Unless otherwise specified, N denotes the number of vertices in a given network, and Δ denotes the maximum degree of a vertex in it; in addition, $\mathcal{H}$ and $\mathcal{L}$ denote given upper bounds for the hop count and for the load, respectively.

3.1 Chain Networks

1. One-to-all (or: broadcast):

   (a) Recursive constructions with load bounded by $\mathcal{H} \cdot N^{\frac{1}{\mathcal{H}}}$ are presented in [?].

   (b) Optimal constructions for chain networks with stretch factor of one are presented in [?], and for a general stretch factor in [?]. They are described in Sections 4.1. In both of these papers the results obtained are symmetric in the load and the hop count; a simple explanation for these symmetries is provided in Section 4.1, following [?].

2. All-to-all:

   (a) In [?] the authors define $Hops_N(\mathcal{L})$ as the maximal number of hops connecting any pair of vertices in a chain (with N vertices, where the load is bounded by $\mathcal{L}$). They prove that
   $$\sqrt{2N} - 5 < Hops_N(2) < \sqrt{2N} + 2,$$
   and that
   $$\frac{1}{2} N^{\frac{1}{\mathcal{L}}} < Hops_N(\mathcal{L}) < \mathcal{L} \cdot N^{\frac{1}{\mathcal{L}}},$$
   for any $\mathcal{L} \geq 1$.

   (b) In [?] the all-to-all path layouts is studied using a geometric approach. For a detailed discussion, see Section **??**. Stated in graph-theoretic terms, these layouts are translated into embeddings (or linear arrangements) of the vertices of a graph with N vertices onto the points $1, 2, \cdots, N$ of the x-axis. The authors pursue the existence of a graph with minimum diameter $D_{\mathcal{L}}(N)$ for which such an embedding

is possible, given a bound $\mathcal{L}$ on the cutwidth of the embedding. The results can be summarized as follows. For an N-vertex chain and for every $\mathcal{L} \geq 1$,

$$max\{\frac{1}{2}[(\mathcal{L}!\cdot N)^{1/\mathcal{L}} - \mathcal{L}], \frac{1}{2}[N^{1/\mathcal{L}} - 1], \frac{\log N}{\log(2\mathcal{L}+1)}\} \leq D_{\mathcal{L}}(N) \leq (\mathcal{L}!\cdot N)^{1/\mathcal{L}} + 2.$$

The relation of these results to those in [?, ?, ?] are summarized in Section ??.

(c) In [?] it is shown that the hop count is $\Theta(\frac{\log N}{\log \mathcal{L}})$, if $\mathcal{L} \geq \log^{1+\epsilon} N$, for any fixed $\epsilon > 0$.

3.2 Ring Networks

1. In [?] the authors study *chordal rings* (a ring network enhanced by adding non-crossing edges (*chords*)), *express rings* (chordal rings in which the chords are oriented either clockwise or counterclockwise), and *multi-rings* (in which subsidiary rings are appended to edges of a ring and, recursively, to edges of appended subrings). The authors first demonstrate the topological equivalence of these structures. They then show that for every N and $\mathcal{L}$, there exists an N-vertex express ring (with load bounded by $\mathcal{L}$) and whose diameter is bounded by $2^{-\frac{1}{\mathcal{L}}} \cdot \mathcal{L} \cdot N^{\frac{1}{\mathcal{L}}} + 1$, and that there is no construction with diameter smaller than $\frac{\mathcal{L}}{2\cdot e} N^{\frac{1}{\mathcal{L}}}$. The discussion involves geometric considerations that are similar to those later used in [?] to obtain optimal constructions (see Section ??). The relation between the diameter and the hop count is clear; see also [?] for more discussion and results. It is also shown that the insistence that the arcs in an express ring be non-crossing at most doubles the diameter of the augmented ring.

2. In [?] the all-to-all path layouts is studied for augmented ring and augmented path networks (for more details, see Section ??).

3.3 Tree Networks

1. In [?, ?], the authors describe a greedy algorithm to determine the existence of a one-to-all layout in tree networks, which, for a given bound on the hop count, determines a layout with smallest possible vertex load (for more algorithms of related problems, see also [?, ?]). This approach proved to be quite helpful for such constructions of virtual layouts.

2. In [?] the authors first describe a recursive construction for a path layout for the one-to-all problem with the load bounded by $\mathcal{H} \cdot N^{\frac{1}{\mathcal{H}}}$. A lower bound of $\Omega(\frac{1}{\Delta \cdot 2^{\frac{1}{\mathcal{H}}}} \cdot N^{\frac{1}{\mathcal{H}}})$ is also proved. This implies that for "realistic" networks the load is $\Theta(N^{\frac{1}{\mathcal{H}}})$.

3. For the all-to-all problem an upper bound of $\frac{\mathcal{H}}{2(2^{\frac{2}{\mathcal{H}}}-1)} N^{\frac{2}{\mathcal{H}}}$ and a lower bound of $\frac{1}{\Delta \cdot (8\mathcal{H})^{\frac{1}{\mathcal{H}}}} N^{\frac{2}{\mathcal{H}}}$ are shown in [?]. This proves that for "realistic" networks the load is $\Theta(N^{\frac{2}{\mathcal{H}}})$.

3.4 Mesh Networks

1. In [?] the authors study constructions for the all-to-all problem on an $a \times b$ mesh. They show a construction that achieves a load of $a^{\frac{2}{h_a}}$, where $h_a = \frac{\mathcal{H}}{\frac{\log a}{\log b}+1}$. For an $\sqrt{N} \times \sqrt{N}$ mesh this implies a bound of $\frac{\mathcal{H}}{4(2^{\frac{4}{\mathcal{H}}}-1)} N^{\frac{2}{\mathcal{H}}}$.

2. In [?] the authors study the all-to-all problem. They define $Hops_{a\times b}(\mathcal{L})$ to be the maximal number of hops connecting any pair of vertices in an $a \times b$ mesh. They show that $Hops_{k\times n}(\mathcal{L}) = \Theta(n^{\frac{1}{k\mathcal{L}}})$, for a fixed k, and that $Hops_{n\times n}(\mathcal{L}) = \Theta(\log n)$.

3. In [?] it is shown that the number of hops for the $\sqrt{N} \times \sqrt{N}$ mesh is $\Theta(\frac{\log N}{\log \mathcal{L}})$ for $\mathcal{L} \geq 2$, for the all-to-all problem.

4. In [?, ?] a lower bound of $\Omega((\frac{N}{\mathcal{H}})^{\frac{1}{\mathcal{H}}})$ is shown for the one-to-all problem, together with a construction with an upper bound of $\mathcal{H} \cdot N^{\frac{1}{\mathcal{H}}}$ on the load. For the all-to-all problem a construction with load bounded by $2 \cdot (\frac{\mathcal{H}}{3} + 1) \cdot N^{\frac{3}{2\mathcal{H}}}$ is shown.

3.5 General Networks

1. <u>All-to-all</u>:

   (a) In [?] the authors describe a recursive construction, which, for a given k, yields a layout with a stretch factor bounded by $8k$, and a vertex load bounded by $O(\mathcal{H} \cdot k \cdot \log N \cdot N^{\frac{1}{k}+\frac{2}{\mathcal{H}}})$. The technique uses ideas from [?], which seem to be quite useful for these layout designs.

   (b) In [?] the authors define $Hops_N(\mathcal{L})$ to be the maximal number of hops connecting any pair of vertices for a network with N vertices and a bound $\mathcal{L}$ for the load. They show that $Hops_N(\mathcal{L}) > \frac{\log N}{\log(\Delta\mathcal{L})} - 1$, for any $\mathcal{L} \geq 1$. We note that an identical lower bound is presented in [?].

   (c) In [?] the authors construct a virtual path layout with hop count $O(\frac{diam(G)\cdot\log\Delta}{\log\mathcal{L}})$, where $diam(G)$ is the diameter of the network G, and $\Delta \geq 3$. In the case of unbounded degree networks with diameter $O(\log N)$, these hop numbers are optimal for any $c \geq \Delta$. For any $\mathcal{L} \geq 1$ and bounded degree network with diameter $O(\log N)$ a construction is presented with hop count of $\Theta(\frac{\log N}{\log \mathcal{L}})$.

   (d) In [?] the authors describe a recursive construction for graphs with bounded treewidth (see also [?]). They show a construction with load bounded by $O(\frac{k\cdot\mathcal{H}\cdot N^{\frac{2}{\mathcal{H}}}}{2((1.5)^{\frac{2}{\mathcal{H}}}-1)})$, where k is the bound on the treewidth.

2. <u>One-to-all:</u>

   (a) In [?] a recursive construction is presented, and a bound of $\sqrt{\mathcal{H}} N^{1+\frac{1}{\mathcal{H}}}$ is shown on the maximal load.

(b) Decision Problems:

In [?], the complexity of deciding the existence of layouts of one-to-all virtual paths which have maximum hop count $\mathcal{H}$ and maximum (edge) load $\mathcal{L}$, for a stretch factor of one, has been studied. It is proved that the problem of determining the existence of such layouts is NP-complete for any given values of $\mathcal{H}$ and $\mathcal{L}$, except for the cases where $\mathcal{H} = 2$ and $\mathcal{L} = 1$, or $\mathcal{H} = 1$ and any $\mathcal{L}$. For these cases the authors give polynomial-time layout constructions; these constructions follow from polynomial time algorithms that compute maximum flow.

In [?] it was shown that the problems of determining the existence of either a one-to-all or an all-to-all layout of virtual paths, which has maximum hop count $\mathcal{H}$ and maximum vertex load $\mathcal{L}$, for an unbounded stretch factor, are both NP-complete.

3. In [?], the dynamic maintenance of virtual path layout is discussed. The authors describe methods to adjust the layout of these paths to the dynamics of changes in the usage of the networks by its end users.

4. It is certainly of interest to consider problems other than the one-to-all or the all-to-all. An interesting result, that gives a lower bound on the load, and also uses the total number of pairs to be connected as a parameter, has been recently shown in [?, ?]. The lower bound shown is $\frac{1}{\Delta}(\frac{\Delta}{4\mathcal{H}}(\frac{N_C}{|C|} - \frac{4}{\Delta}))^{\frac{1}{\mathcal{H}}}$, where C is a *cut* in the network and N_C is the number of pairs separated by this cut, between which a communication is required. For the one-to-all problem, this implies a lower bound of $\frac{1}{\Delta}(\frac{N-5}{4\cdot\mathcal{H}})^{\frac{1}{\mathcal{H}}}$.

3.6 Miscellaneous

1. In [?], the problem of path layout is considered, under the assumption that the vertices are of three types: those that can switch only virtual paths, those that can switch only virtual channels, and those that can switch both. A few solutions are presented to this problem; among them is a greedy algorithm, which optimizes the network overhead for a request/response and the utilization of bandwidth and routing table resources.

2. In [?] the authors present a measure of maximum vertex load (for more results in this venue, see also [?]), which is the total number of virtual paths that end at this vertex. They show that determining the existence of a one-to-all layout for a given network, with a vertex load bounded by $\mathcal{L}$ and a hop count bounded by $\mathcal{H}$, is an NP-complete problem for any $\mathcal{H}$ and $\mathcal{L}$, except for the cases where $\mathcal{H} = 1$ and any $\mathcal{L}$, or $\mathcal{H} = 2$ and $\mathcal{L} = 1$ (the proof techniques is borrowed from [?]). Specific bounds are given for chain, mesh and torus networks.

3. Planar graphs are studied in [?]; for the all-to-all problem, a lower bound of $\Omega(\frac{1}{\Delta}N^{\frac{3}{2\mathcal{H}}})$ on the load is shown.

4. Other specific graphs are also considered in the literature; However, the area is still far from being adequately explored. See, for example, [?] for the cases of the hypercube and the de-Bruijn graphs.

4 Detailed Description

In this section we first present in some more detail the structure of layouts for the one-to-all problem on a chain network, for a stretch factor of either one or unbounded. These layouts are optimal for the various measures. We then discuss the use of geometry in deriving optimal bounds for the all-to-all problem in chains, augmented paths and ring networks.

4.1 The One-to-All Problem on Chain Networks

This simple topology, which is also quite practical, admits a precise treatment, as summarized in this section.

We consider four performance measures, and achieve optimal solutions for each. These are the maximum hop count and average hop count, denoted $\mathcal{H}_{max}$ and $\mathcal{H}_{avg}$, respectively, and the maximum load and average load, denoted $\mathcal{L}_{max}$ and $\mathcal{L}_{avg}$, respectively. All these measures have practical implications in different ways. As the hop count is proportional to the setup time of a new connection, the worst-case hop measure represents "hard" deadlines for this overhead, which are typical to real time applications; on the other hand, the average hop measure is useful for general purpose networks. Since the load represents the utilization of the routing tables, maximum load is important in cases where the layout is large and may overflow the limited space of routing entries (4096 per routing table [**?**]), whereas average load measures are relevant for general purpose networks, in which many independent layouts may coexist in an effort to minimize local bottlenecks at any location in the network. Given an upper bound on the maximum hop count, we want to minimize the maximum load ($\mathcal{L}_{max}$) or the average load ($\mathcal{L}_{avg}$); Given an upper bound on the maximum load, we want to minimize the maximum hop count $\mathcal{H}_{max}$ or the average hop count $\mathcal{H}_{avg}$.

The following discussion presents results that use shortest path layouts. These results have been shown in [**?**]. At the end of the section we mention extensions of these results, that yield *exact* layouts for the case of a general stretch factor. In our discussion, N denotes the number of vertices in the chain, and the layout is from the leftmost vertex to all others (the extension for the case where the layout is to all vertices on the left and right is trivial; such extensions are explicitly considered in [**?**, **?**]).

We first establish a canonic form of a VPL , that will simplify the rest of the discussion.

Lemma 4.1 ([**?**]) *Given a chain network, there exists, for each optimality measure ($\mathcal{L}_{max}$, $\mathcal{L}_{avg}$, $\mathcal{H}_{max}$, or $\mathcal{H}_{avg}$) an optimal* VPL *in which every vertex $i \geq 2$ is the right-most endpoint of a single* VP *.*

In other words, this VPL induces a tree rooted at vertex 1 with the VP s corresponding to tree edges.

We proceed to define an important class of VPL 's that do not "cross" each other.

Definition 4.2 *Let $l_1 < l_2$. Two* VP *s denoted (l_1, r_1) and (l_2, r_2) constitute a* crossing *if $l_1 < l_2 < r_1 < r_2$. A* VPL *is called* crossing-free *if no pair of* VP *s constitute a crossing.*

Theorem 4.3 ([**?**]) *For each performance measure ($\mathcal{L}_{max}$, $\mathcal{H}_{max}$, $\mathcal{L}_{avg}$, and $\mathcal{H}_{avg}$) there exists an optimal* VPL *which is crossing-free.*

INDUCEVPL(T): Induce an VPL according to a tree T with N vertices.

1. Label the vertices of T in depth-first order. Let $\lambda(u)$ be the label of a vertex $u \in T$, $1 \leq \lambda(u) \leq N$.
2. For every edge $(u, v) \in T$ connect a VP between $\lambda(u)$ and $\lambda(v)$.
3. Return Ψ_T, the collection of generated VP s.

Figure 1: Procedure INDUCEVPL(T).

Lemma 4.1 implies that a VPL induces a tree. The next lemma shows that the converse holds too; namely, any tree induces a VPL . The lemma refers to procedure INDUCEVPL(T) depicted in Figure 1.

Lemma 4.4 ([?]) *Let T be an ordered tree. Then procedure* INDUCEVPL(T) *induces a crossing-free* VPL .

Figure 2 presents an example of how INDUCEVPL(T) runs on a specific ordered tree.

We now consider optimal VPL layouts for the worst-case (maximal) load and hop count measures. Specifically, if the load is required to be $\mathcal{L}_{max} \leq \mathcal{L}$, we characterize the layout with the minimal worst-case hop count; similarly, if the hop count is $\mathcal{H}_{max} \leq \mathcal{H}$, we characterize the layout with the minimal worst-case load. This is done using a new class of trees $\mathcal{T}(\mathcal{L}, \mathcal{H})$ that we define next. These trees contain all VPL s on a chain that satisfy the above constraints on maximum load and maximum hop count.

Definition 4.5 *The ordered tree $\mathcal{T}(\mathcal{L}, \mathcal{H})$ is defined recursively as follows. The root r has $\mathcal{L}$ children. The i^{th} child from the left is the root of a $\mathcal{T}(i, \mathcal{H}-1)$ subtree, for $1 \leq i \leq \mathcal{L}$. A tree $\mathcal{T}(\mathcal{L}, 0)$ or $\mathcal{T}(0, \mathcal{H})$ is a single vertex.*

See Figure 2 for an illustration of Definition 4.5.

We remarks that an internal vertex of $\mathcal{T}(\mathcal{L}, \mathcal{H})$, which is the i^{th} child (from the left) of its parent, has i children. The tree $\mathcal{T}(1, \mathcal{H})$ is a rooted chain of $\mathcal{H}+1$ vertices. Note also that $\mathcal{T}(\mathcal{L}, \mathcal{H})$ has height $\mathcal{H}$ and maximum degree $\mathcal{L}$. It is possible to show:

Lemma 4.6 ([?]) *The tree $\mathcal{T}(\mathcal{L}, \mathcal{H})$ contains $\binom{\mathcal{L}+\mathcal{H}}{\mathcal{H}}$ vertices.*

Definition 4.7 *An ordered tree T is* subsumed *in $\mathcal{T}(\mathcal{L}, \mathcal{H})$ if its root is subsumed in the root of $\mathcal{T}(\mathcal{L}, \mathcal{H})$ and the subtrees of the root's children in T are (recursively) subsumed in the subtrees of a subset of the children of the root in $\mathcal{T}(\mathcal{L}, \mathcal{H})$.*

It is easy to see that the VPL of a subsumed tree T has load and hop counts lower than those of the tree $\mathcal{T}(\mathcal{L}, \mathcal{H})$ it is subsumed in, since T may be obtained from $\mathcal{T}(\mathcal{L}, \mathcal{H})$ by deleting subtrees. The next corollary follows.

Corollary 4.8 *Let T be an ordered tree that is subsumed in $\mathcal{T}(\mathcal{L}, \mathcal{H})$, and let $\Psi_T =$* INDUCEVPL(T). *Then, $\mathcal{L}_{max}(\Psi_T) \leq \mathcal{L}$ and $\mathcal{H}_{max}(\Psi_T) \leq \mathcal{H}$.*

Figure 2: The tree $\mathcal{T}(3,3)$ and its induced VPL .

Lemma 4.9 ([?]) *For every crossing-free* VPL Ψ, *with* $\mathcal{L}_{max}(\Psi) \leq \mathcal{L}$ *and* $\mathcal{H}_{max}(\Psi) \leq \mathcal{H}$ *there exists a tree* T *which is subsumed in* $\mathcal{T}(\mathcal{L},\mathcal{H})$ *such that* $\Psi = \textsc{InduceVPL}(T)$.

Theorem 4.10 ([?]) *Given* N *and* $\mathcal{L}$, *Let* $\mathcal{H}$ *be such that*

$$\binom{\mathcal{L}+\mathcal{H}-1}{\mathcal{L}} < N \leq \binom{\mathcal{L}+\mathcal{H}}{\mathcal{L}}.$$

Then $\mathcal{H}_{\rm opt}(N,\mathcal{L}) = \mathcal{H}$.

A similar result holds for the maximum load measure:

Theorem 4.11 ([?]) *Given* N *and* $\mathcal{H}$, *let* $\mathcal{L}$ *be such that*

$$\binom{\mathcal{L}+\mathcal{H}-1}{\mathcal{H}} < N \leq \binom{\mathcal{L}+\mathcal{H}}{\mathcal{H}}.$$

Then $\mathcal{L}_{\rm opt}(N,\mathcal{H}) = \mathcal{L}$.

Given a chain with $N = N(\mathcal{L},\mathcal{H})$, there exists a unique VPL with $\mathcal{L}_{max}(\Psi) = \mathcal{L}$ and $\mathcal{H}_{max}(\Psi) = \mathcal{H}$, whereas several such VPL s exist for other values of N (that is, for values of N such that $N(\mathcal{L},\mathcal{H}-1) < N < N(\mathcal{L},\mathcal{H})$). This symmetry is not a coincidence, as explained in the sequel.

If $N = N(\mathcal{L},\mathcal{H})$ then $(\mathcal{H}!\cdot N)^{1/\mathcal{H}} - \frac{\mathcal{H}+1}{2} \leq \mathcal{L} \leq (\mathcal{H}!\cdot N)^{1/\mathcal{H}} - 1$. This is an improvement to the upper bound $\mathcal{L} \leq \mathcal{H}\cdot N^{1/\mathcal{H}}$ shown in [?], since $(\mathcal{H}!)^{1/\mathcal{H}} < \mathcal{H}$ for any $\mathcal{H} > 1$.

In [?], a greedy algorithm for finding a VPL for the more general case of tree networks is presented and shown to be optimal with respect to the $\mathcal{H}_{max}$ measure. However, the algorithm does not give insight into the structure of the obtained VPL, and, in particular, no upper bound is easily derived from it. When $N = N(\mathcal{L},\mathcal{H})$ the previous theorem gives a precise characterization of this greedy solution. (In [?] the load is measured on the vertices rather than on the edges. Thus, the greedy algorithm should be modified to use the edge-load constraint. We refer in the above to this modified algorithm in the comparison.)

We now turn to discuss the average load. We start with the case where the maximal number of hops is limited to $\mathcal{H}$, and it is required to find the layout with the smallest average load. Our next definition is justified by the fact that a layout $\Psi_{\rm opt}$ that minimizes $\mathcal{L}_{avg}$ also minimizes its total load $\mathcal{L}_{tot}(\Psi_{\rm opt})$.

Definition 4.12 *Let* $\mathcal{L}_{tot}(N,\mathcal{H})$ *denote the minimal total load of any* VPL *on* N *vertices with at most* $\mathcal{H}$ *hops; that is,*

$$\mathcal{L}_{tot}(N,\mathcal{H}) \equiv \min_{\Psi}\{\mathcal{L}_{tot}(\Psi) : \mathcal{H}_{max}(\Psi) \leq \mathcal{H}\}.$$

The rationale behind our dynamic programming algorithm for finding optimal $\mathcal{L}_{tot}$ ($\mathcal{L}_{avg}$) layouts is the following. Let $\Psi_{\rm opt}$ be the optimal VPL (that achieves $\mathcal{L}_{tot}(\Psi_{\rm opt}) = \mathcal{L}_{tot}(n,\mathcal{H})$). Let $(1,d+1)$ be the longest VP connected to the root . Since, by Theorem 4.3, we can assume that $\Psi_{\rm opt}$ is crossing-free, it follows that no VP of $\Psi_{\rm opt}$ connects a vertex $i \leq d$ with a vertex $j > d+1$; thus $\Psi_{\rm opt}$ can be split into two disjoint optimal layouts, one on vertices $1,\ldots,d$,

and the other on vertices $d+1,\ldots,N$. However, the second layout (rooted at $d+1$) may use only $\mathcal{H}-1$ hops since one hop is used to traverse the VP $(1,d+1)$. Thus, if d is known, then the total load is equal to $d+\mathcal{L}_{tot}(d,\mathcal{H})+\mathcal{L}_{tot}(N-d,\mathcal{H}-1)$, so clearly

$$\mathcal{L}_{tot}(N,\mathcal{H}) = \min_{1\le d\le N-1}\{d+\mathcal{L}_{tot}(d,\mathcal{H})+\mathcal{L}_{tot}(N-d,\mathcal{H}-1)\}. \quad (1)$$

There are two simple "boundary" cases: (i) If $1 \le N \le \mathcal{H}+1$ then clearly $\mathcal{L}_{tot}(N,\mathcal{H}) = N-1$, (ii) If $\mathcal{H}=1$ then we must connect a direct VP to each vertex, so $\mathcal{L}_{tot}(N,1) = N(N-1)/2$. The above argument leads to a natural dynamic programming algorithm, with time complexity of $O(N^2\mathcal{H})$. Moreover, the exact value of $\mathcal{L}_{tot}(N,\mathcal{H})$ can be determined as follows.

Theorem 4.13 ([?]) *Given N and $\mathcal{H}$, let $\mathcal{L}$ be the largest integer such that $N \ge \binom{\mathcal{L}+\mathcal{H}}{\mathcal{L}}$, and let $r = N - \binom{\mathcal{L}+\mathcal{H}}{\mathcal{L}}$. Then*

$$\mathcal{L}_{tot}(N,\mathcal{H}) = \mathcal{H}\binom{\mathcal{L}+\mathcal{H}}{\mathcal{L}-1} + r(\mathcal{L}+1).$$

We now turn to study the (unweighted) average hops measure, assuming a maximum bound $\mathcal{L}$ on the load. This problem can be solved by an algorithm similar to that for the case of the average load.

Definition 4.14 *Consider a crossing-free optimal* VPL $\Psi_{\rm opt}$ *for a chain with n vertices and maximum load $\mathcal{L}$ (which achieves the minimum $\mathcal{H}_{tot}(\Psi)$). Define $\mathcal{H}_{tot}(N,\mathcal{L}) \equiv \mathcal{H}_{tot}(\Psi_{\rm opt})$.*

Let $(1,d+1)$ be the longest VP connected to the root. Again, it follows that there exists no VP connecting the vertices $1,\ldots,d$ to the vertices $d+2,\ldots,N$; thus, the layouts in these two segments are disjoint and should both be optimal in $\Psi_{\rm opt}$. In this case however, the layout on the vertices $1,...,d$ should not exceed the load $\mathcal{L}-1$ (since together with the VP $(1,d)$ the load should not exceed $\mathcal{L}$). By the above discussion, it is evident that

$$\mathcal{H}_{tot}(N,\mathcal{L}) = \min_{1\le d\le N-1}\{\mathcal{H}_{tot}(d,\mathcal{L}-1)+(N-d)+\mathcal{H}_{tot}(N-d,\mathcal{L})\}. \quad (2)$$

The first and third components of the sum are the values of $\mathcal{H}_{tot}$ in the two separate segments, while the second component is the cost of an additional hop incurred by all vertices in the segment $d+1,...,N$. The boundary conditions here amount to $\mathcal{H}_{tot}(N,1) = N(N-1)/2$, since if the maximum load is 1 then the only possible VP s are identical to the network edges; also, if $N \le \mathcal{L}+1$ then $\mathcal{H}_{tot}(N,\mathcal{L}) = N-1$, since we can then afford to construct direct VP s from all vertices to the root.

A dynamic programming algorithm follows from the above recurrence relations, with time complexity $O(N^2\mathcal{L})$. Moreover, the exact value of $\mathcal{H}_{tot}(N,\mathcal{L})$ can be determined, as follows.

Theorem 4.15 ([?]) *Let N and $\mathcal{L}$ be given. Let $\mathcal{H}$ be the maximal such that $N \ge \binom{\mathcal{L}+\mathcal{H}}{\mathcal{H}}$, and let $r = N - \binom{\mathcal{L}+\mathcal{H}}{\mathcal{H}}$. Then*

$$\mathcal{H}_{tot}(N,\mathcal{L}) = \mathcal{L}\binom{\mathcal{L}+\mathcal{H}}{\mathcal{H}-1} + r(\mathcal{H}+1).$$