Optimal Layouts on a Chain ATM Network

Ornan Gerstel[*] Avishai Wool[†] Shmuel Zaks[‡]

Abstract

We study a routing problem which occurs in high-speed (ATM) networks, termed the "rooted virtual path layout problem" on chain networks. This problem is essentially a tree embedding problem on a chain host graph. We present four performance measures for the quality of such an embedding which have practical implications, and find optimal solutions for each of them. We first show that the search can be restricted to the class of layouts with no crossovers. Given bounds on the load ℓ and number of hops h in a layout, we then present a family of ordered trees $\mathcal{T}(\ell, h)$, within which an optimal solution can be found (if one exists at all); this holds for either the worst-case or average-case measures, and for a chain of length n, with $n \leq \binom{\ell+h}{\ell}$. For the worst-case measures these trees are used in characterizing, constructing, and proving the optimality of the solutions. For each average-case measure, a recursive formulation of the optimal solution is presented, from which a optimal polynomial dynamic programming algorithm is derived. Furthermore, for the unweighted average measures, these formulations are explicitly solved, and the optimal solutions are mapped to $\mathcal{T}(\ell, h)$.

[*]T.J. Watson Research Center, Hawthorne, NY 10532. E-mail: `ori@watson.ibm.com`. Part of this author's work was done while being with the Department of Computer Science, Technion, Haifa 32000, Israel.

[†]Department of Applied Mathematics and Computer Science, The Weizmann Institute, Rehovot 76100, Israel. E-mail: `yash@wisdom.weizmann.ac.il`.

[‡]Department of Computer Science, Technion, Haifa 32000, Israel. E-mail: `zaks@cs.technion.ac.il`.

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 Specifically, the sharp distinction between telephone networks, cable television networks, and computer networks, has been replaced by a unified approach.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* (ATM for short), and is thoroughly described in the literature [ITU90, HH91, Par94]. 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, these fields serve as indices to two routing tables (the VCI serves as an index to one table and the VPI to the other), and the routing is done 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 which 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); VPs are used for simplifying network management — routing of VCs in particular. Thus the route of a VC may be viewed as a concatenation of complete VPs.

As far as the mathematical model is concerned, given a communication network, the VPs 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 (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 [ATTD94, GZ94] 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 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 [CS94] for a detailed description of the routing mechanism in ATM). As the ATM standard [ITU90] limits the maximum size of VP routing tables to 4096 entries, this resource is critical in networks with a few hundreds of nodes (see [Ger95] 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 [BD91, SOT90, SS91].

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. In this work we assume that this ratio is 1, implying that a VC has to be routed along a shortest physical path.

In many works (e.g., [ABLP89, AP92, GZ94, CGZ94]), 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-many) VPL* problem [GZ94] and is the focus of the present work.

1.2 Related Work

A few works have tackled the VP layout problem, some using empirical techniques [ATTD94, LC93], and some using theoretical analysis [GZ94, CGZ94]; However, none of these works has attempted to combinatorially characterize the optimal solution, and 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, while we solve the problem for several others, equally important, performance measures too. Of particular practical interest is the weighted hop count measure, since it 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 [GS95]. To the best of our knowledge, the present work is the first to analytically tackle this problem.

The VP layout problem is closely related to graph-embedding problems since in both cases it is required to embed one graph in 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, here it is made by the *stretch factor*. In other words, in embedding problems it is important to minimize the length of each individual embedded edge, while in this model 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 depends only on one graph, the hop count also depends on the physical graph (unless the stretch factor is unbounded).

- The *load* parameter is identical to the *congestion* in embedding problems, and 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, thus none of the embedding problems we are aware of are applicable for the VP layout problem.

Another problem which is related to ours is that of keeping small routing tables for routing in conventional computer networks. This problem was widely studied [ABLP89, AP92, FJ86, FJ88, KK77, KK80, PU88] and yielded interesting graph decompositions and structures, but it differs from ours in some major aspects which deemed most of these solutions impractical for our purposes. The main difference stems from the fact that in our case there is no flexibility as to the routing scheme itself since it is determined by the ATM standard [ITU90].

A related criterion, minimizing the *number* of VPs to achieve a required maximal hop count, is discussed in [BTS94] mainly for database optimization purposes, and yields very different results. For ATM, however, this criterion is of lesser importance since no such global constraint exists.

The notion of *forwarding index* was studied for communication networks (see, e.g., [CCRS87, HMS89]). The forwarding index in a given network with a given set of paths corresponds to the load in our case; in [CCRS87] the authors study load on vertices, while in [HMS89] they study both load on vertices and on edges. However, these studies do not consider the notion of hop count, and, in addition, they study the all-to-all problem.

1.3 Summary of Results

In the paper we consider the problem of constructing a VP layout on a network with chain topology. This simple topology enables us to study the problem in greater depth than previous works. We also restrict the discussion to "rooted" layouts, in which it is required to enable connections between all vertices to a single vertex called the *root* (rather than to all vertices). As mentioned earlier, these layouts have been shown to be a useful tool in constructing more complex layouts.
We consider four performance measures, and achieve optimal solutions for each measure:

- Given an upper bound on the maximum hop count, minimize the maximum load ($\mathcal{L}_{\max}$),
- Given an upper bound on the maximum load, minimize the maximum hop count ($\mathcal{H}_{\max}$),
- Given an upper bound on the maximum hop count, minimize the average load ($\mathcal{L}_{\text{avg}}$), and
- Given an upper bound on the maximum load, and vertex weights (representing the frequency of connection requests between the vertex and the root), minimize the average hop count ($\mathcal{H}^{w}_{\text{avg}}$).

All these measures have practical implications in different cases: 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 (typical to real time applications), while the average hops 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 [ITU90]), whereas average load measures are relevant for general purpose networks, in which many independent layouts coexist and try to minimize local bottlenecks at any location in the network.

After defining the model and measures (in Section 2), we show (in Section 3) that it is sufficient to consider layouts of a canonic form (Lemma 3.1 and Theorem 3.3), and consider the number of such different layouts (Catalan number C_{n-1} for a chain of n vertices). Next we focus (in Section 4) on the maximal load and hops measures and define a new class $\mathcal{T}(\ell, h)$ of ordered trees which include as subtrees all feasible layouts that satisfy given load and hops constraints (ℓ, h resp.). This tree helps in characterizing tight bounds for both the $\mathcal{L}_{\max}$ and $\mathcal{H}_{\max}$ measures, namely, given a chain with n vertices such that $\binom{\ell+h-1}{\ell} < n \leq \binom{\ell+h}{\ell}$ then $\mathcal{H}_{\max} = h$ and if $\binom{\ell+h-1}{h} < n \leq \binom{\ell+h}{h}$ then $\mathcal{L}_{\max} = \ell$.

In Section 5 we study the average measures. We first obtain an $O(n^2h)$ algorithm for finding the optimal average load layout, based on dynamic programming, and achieve a similar

$O(n^2\ell)$ algorithm for the unweighted average hops measure (i.e., when all weights are 1). Then using the dynamic programming formulation we prove explicit expressions for the values of the optimal layouts (for both measures), and characterize layout constructions achieving the optimum. Finally, we study the more complex weighted average hops measure and present an $O(n^3\ell)$ optimal algorithm for it. We conclude and list a few of the remaining open problems in Section 6.

A preliminary version of this paper can be found in [GWZ95].

2 The Model

We model the underlying communication network as an undirected graph $G = (V, E)$, where V corresponds to the set of switches and E to the set of physical links between them.

Definition 2.1 *A* rooted virtual path layout (RVPL for short) Ψ *is a collection of simple paths in G, termed* virtual paths *(*VP*s for short), and a vertex $r \in V$ termed the* root *of the layout (denoted root(Ψ)).*

Definition 2.2 *The* load $\mathcal{L}(e)$ *of an edge $e \in E$ in an* RVPL Ψ *is the number of* VP*s* $\psi \in \Psi$ *that include e.*

Definition 2.3 *The* maximal load $\mathcal{L}_{\max}(\Psi)$ *of an* RVPL Ψ *is* $\max_{e \in E} \mathcal{L}(e)$. [1]

Definition 2.4 *The* average load *of an* RVPL Ψ *is* $\mathcal{L}_{\text{avg}}(\Psi) \equiv \frac{1}{|E|}\sum_{e\in E}\mathcal{L}(e)$.

Definition 2.5 *The* hop count $\mathcal{H}(v)$ *of a vertex $v \in V$ in an* RVPL Ψ *is the minimum number of* VP*s whose concatenation forms a shortest path in G from v to root(Ψ). If no such* VP*s exist, define* $\mathcal{H}(v) \equiv \infty$.

Definition 2.6 *The* maximal hop count *of an* RVPL Ψ *is* $\mathcal{H}_{\max}(\Psi) \equiv \max_{v\in V}\{\mathcal{H}(v)\}$.

Definition 2.7 *Let $w(v)$, $v \in V$ be non-negative weights assigned to the vertices and let* $W = \sum_{v\in V} w(v)$. *The* weighted total hop count *of an* RVPL Ψ *is* $\mathcal{H}^w_{\text{tot}}(\Psi) \equiv \sum_{v\in V} w(v)\mathcal{H}(v)$, *and the* weighted average hop count *is* $\mathcal{H}^w_{\text{avg}}(\Psi) \equiv \frac{1}{W}\mathcal{H}^w_{\text{tot}}(\Psi)$. *When the weights are all $w(v) = 1$ then we denote the total hop count by $\mathcal{H}_{\text{tot}}(\Psi)$, and the average hop count is* $\mathcal{H}_{\text{avg}}(\Psi) \equiv \frac{1}{n-1}\mathcal{H}_{\text{tot}}(\Psi)$.

The above weighted hop count is of particular practical interest since it measures the expected time for setting up a connection between a pair of users, given the relative frequency of connection requests between network vertices.

In the rest of this paper we assume that the underlying network is a *chain.* Therefore w.l.o.g. we can assume that the root of every RVPL we consider is the left-most vertex of the chain. For simplicity we denote the vertices $1, 2, \ldots, n$ and the root is always vertex 1. In a chain, the simple path between two vertices is unique, hence we can denote a VP $\psi \in \Psi$ between vertices u and v by the names of its endpoints, i.e., $\psi \equiv (u, v)$.

[1]As mentioned above, the load on an edge is identical to its congestion.

Definition 2.8 *Let $\psi = (u, v)$ be a* VP*. Then the* dilation *of ψ, denoted $|\psi|$, is the number of physical links that ψ traverses, $|\psi| = v - u$. Let Ψ be an* RVPL*, then the* total load *of Ψ is $\mathcal{L}_{\text{tot}}(\Psi) \equiv \sum_{\psi \in \Psi} |\psi|$.*

The following lemma is well known in the graph embedding literature: the sum of congestions on all edges of a host graph is equal to the sum of dilations of the edges of the embedded graph.

Lemma 2.9 *For any* RVPL *Ψ on a chain, $\mathcal{L}_{\text{avg}}(\Psi) = \frac{1}{n-1}\mathcal{L}_{\text{tot}}(\Psi)$.*

Proof. Every VP ψ contributes one to the load on each link it uses, hence its total contribution to the sum $\sum_{e \in E} \mathcal{L}(e)$ is $|\psi|$. Summing this up for all VPs we get $\sum_{e \in E} \mathcal{L}(e) = \mathcal{L}_{\text{tot}}(\Psi)$. □

To minimize the load, one can use an RVPL Ψ which has a VP on each physical link, i.e., $\mathcal{L}_{\max}(\Psi) = 1$, however such a layout has a hop count of $n - 1$. The other extreme is connecting a direct VP from the root to each other vertex, yielding $\mathcal{H}_{\max} = 1$ but $\mathcal{L}_{\max} = n - 1$. For the intermediate cases we need the following definitions.

Definition 2.10 *Let $\mathcal{H}_{\text{opt}}(n, \ell)$ denote the* optimal hop count *of any* RVPL *Ψ on a chain of n vertices such that $\mathcal{L}_{\max}(\Psi) \leq \ell$, i.e.,*

$$\mathcal{H}_{\text{opt}}(n, \ell) \equiv \min_{\Psi} \{\mathcal{H}_{\max}(\Psi) : \mathcal{L}_{\max}(\Psi) \leq \ell\}.$$

Definition 2.11 *Let $\mathcal{L}_{\text{opt}}(n, h)$ denote the* optimal load *of any* RVPL *Ψ on a chain of n vertices such that $\mathcal{H}_{\max}(\Psi) \leq h$, i.e.,*

$$\mathcal{L}_{\text{opt}}(n, h) \equiv \min_{\Psi} \{\mathcal{L}_{\max}(\Psi) : \mathcal{H}_{\max}(\Psi) \leq h\}.$$

3 The Structure of an Optimal RVPL

We first establish a canonic form of an RVPL, which will simplify the rest of the discussion.

Lemma 3.1 *Given a chain network, for every optimality measure ($\mathcal{L}_{\max}$, $\mathcal{L}_{\text{avg}}$, $\mathcal{H}_{\max}$, or $\mathcal{H}^{w}_{\text{avg}}$) there exists an optimal* RVPL *in which every vertex $i \geq 2$ is the right-most endpoint of a single* VP*. In other words, this* RVPL *induces a tree rooted at vertex 1 with the* VP*s corresponding to tree edges.*

Proof. By Definition 2.5, every node i is the right endpoint of at least one VP, otherwise the path from 1 to i traverses more than the minimal number of edges. Assume that there exists a node i with two VPs ψ_1, ψ_2 towards node 1. Let P_i denote the sequence of VPs from i to 1 using ψ_i. W.l.o.g. the length of P_1 does not exceed that of P_2 (in terms of the number of VPs it includes). Thus we may remove ψ_2 from the RVPL without increasing the hop count of any node in the chain. The process may be repeated until no more such duplicate paths exist. □

Definition 3.2 *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* RVPL *is called* crossing-free *if no pair of* VP*s constitute a crossing.*

Theorem 3.3 *For each performance measure ($\mathcal{L}_{\max}$, $\mathcal{H}_{\max}$, $\mathcal{L}_{\text{avg}}$, and $\mathcal{H}^{w}_{\text{avg}}$) there exists an optimal* RVPL *which is crossing-free.*

Proof. Assume there is a pair of crossing VPs (l_1, r_1) and (l_2, r_2) in Ψ. We will show transformations that remove the crossing and do not increase any of the above measures.

(1) The $\mathcal{H}_{\max}$ case. Denote by a_i ($i \in \{1, 2\}$) the minimal number of hops from l_i to the root, denote by b_i the maximum number of hops between r_i to a vertex $v_i > l_i$ (see Figure 1(a)). By Lemma 3.1, there is a single VP path from v_i to the root, which must traverse (l_i, r_i), thus $a_i + 1 + b_i \leq h$ to satisfy $\mathcal{H}_{\max}(\Psi) \leq h$. If $b_1 > b_2$ then replace the VP (l_2, r_2) by (r_1, r_2) in Ψ (see Figure 1(b)). The only vertices whose hop count is affected by the change are in the subtree of r_2 in the RVPL "tree", and in the worst case the new hop count of v_2 is

$$\mathcal{H}(v_2) = b_2 + 1 + 1 + a_1 \leq b_1 + 1 + a_1 \leq h.$$

In case $b_1 \leq b_2$, we replace (l_1, r_1) in Ψ by (l_2, r_1) (see Figure 1(c)). The hop counts change only in the subtree of r_1 in the RVPL, where in the worst case

$$\mathcal{H}(v_1) = b_1 + 1 + a_2 \leq b_2 + 1 + a_2 \leq h.$$

Note that both transformations reduce the number of crossings by at least one, and do not increase the maximum hop count. Hence all the crossings may be eliminated by iterating the transformation a finite number of times.

(2) The $\mathcal{H}^{w}_{\text{avg}}$ case. Refer again to Figure 1(a), but define b_i to be the sum of weights of all vertices in the RVPL which are in the subtree of r_i (a_i remains the number of hops from l_i to the root). Let c_i denote the weighted sum of hops of all vertices except those in the subtree of r_i in Ψ, and d_i denote the weighted sum of hops of in r_i's subtree up to r_i (i.e., *not* all the way to vertex 1). Thus initially we have

$$\mathcal{H}^{w}_{\text{tot}}(\Psi) = c_i + d_i + b_i(1 + a_i)$$

where the third component of the sum is the additional cost of vertices in r_i's subtree, from r_i to vertex 1.

If $a_1 < a_2$ then transform Ψ to Ψ' by applying the transformation of Figure 1(b). Clearly $\mathcal{H}^{w}_{\text{tot}}(\Psi') = c_2 + d_2 + b_2(1 + 1 + a_1)$ so we get $\mathcal{H}^{w}_{\text{tot}}(\Psi') \leq \mathcal{H}^{w}_{\text{tot}}(\Psi)$, and therefore $\mathcal{H}^{w}_{\text{avg}}(\Psi') \leq \mathcal{H}^{w}_{\text{avg}}(\Psi)$. On the other hand, if $a_1 \geq a_2$ then we use the transformation of Figure 1(c). In this case $\mathcal{H}^{w}_{\text{tot}}(\Psi) \geq c_1 + d_1 + b_1(1 + a_2) = \mathcal{H}^{w}_{\text{tot}}(\Psi')$. Again, in both cases the number of crossings has decreased by at least one.

(3) The $\mathcal{L}_{\max}$ and $\mathcal{L}_{\text{avg}}$ cases. In case (1) we have proved that it is possible to transform the RVPL without increasing the VP hop count. Note that in both transformations (Figure 1(b,c)), the load is reduced on some links, and remains the same on all other links. It is therefore clear that the maximum load is not increased and the average load is decreased without changing the hop constraint. □

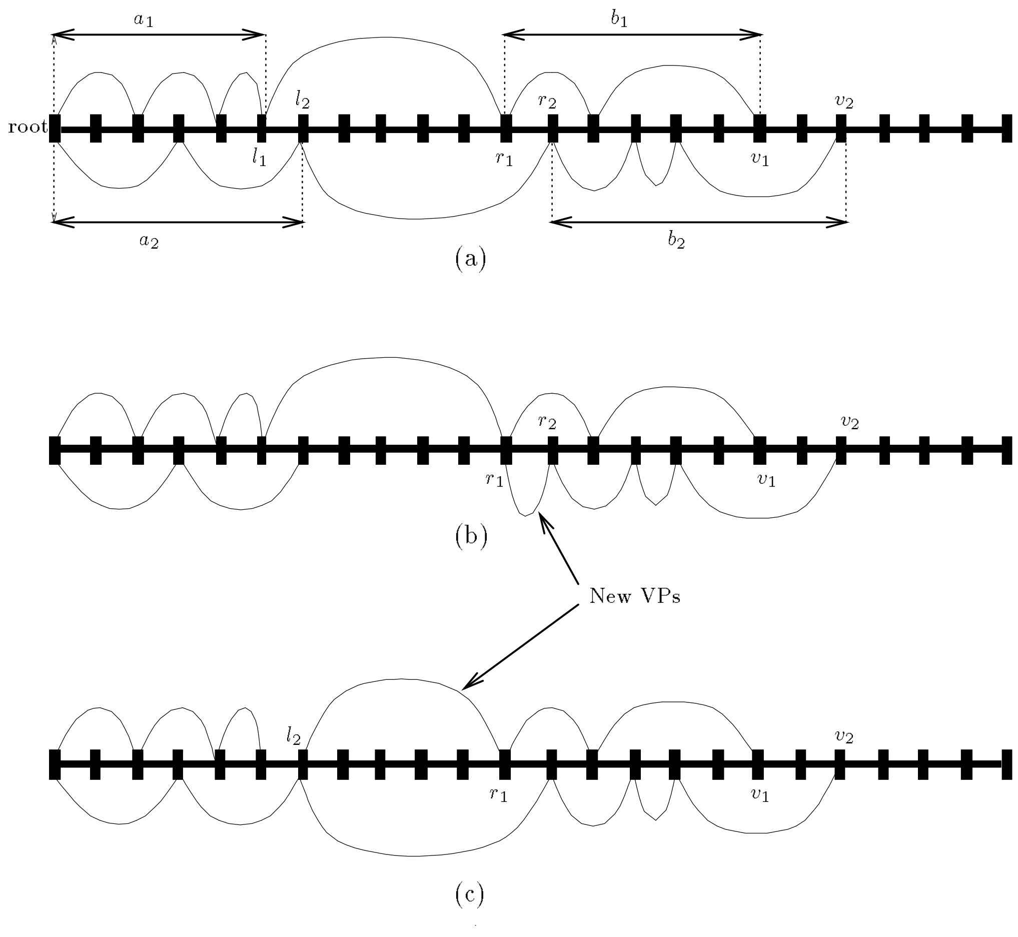


Figure 1: Crossing elimination transformations

```
INDUCEVPL($T$): Induce an RVPL 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 $\Psi_T$, the collection of generated VPs.
```

Figure 2: Procedure INDUCEVPL(T).

In the rest of the paper we restrict ourselves to RVPLs satisfying Lemma 3.1 and Theorem 3.3.
The next Lemma characterizes the number of canonic RVPLs. It shows that even if we restrict ourselves to RVPLs satisfying Theorem 3.3, there still is an exponential number of possible layouts.

Lemma 3.4 *The number $\gamma(n)$ of crossing-free* RVPL *constructions on a chain with n vertices is a Catalan number,*

$$\gamma(n) = C_{n-1} = \frac{1}{n}\binom{2n-2}{n-1} = \Omega\left(\frac{4^n}{n^{3/2}}\right).$$

Proof. Consider the longest VP, $(1, d+1)$, that touches the root of an RVPL Ψ. Since Ψ is crossing-free, no VP of Ψ connects a node $i \leq d$ with a node $j > d+1$. Therefore by removing the VP $(1, d+1)$ we obtain two disjoint RVPLs, one on d nodes (rooted at 1) and the other on $n-d$ nodes (rooted at $d+1$). Therefore summing over all possible values of d we obtain the following recurrence.

$$\gamma(n) = \begin{cases} 1, & n = 1, \\ \sum_{d=1}^{n-1} \gamma(d)\gamma(n-d), & n \geq 2. \end{cases}$$

Therefore $\gamma(n)$ is a Catalan number, (cf. [Knu68], pp. 388–389) and the claim follows. □

Remark: An alternative proof can be obtained by proving a $1-1$ correspondence between crossing-free layouts and legal parenthetic expressions with $n-1$ ('s and $n-1$)'s, since the number of legal parenthetic expressions is known to be a Catalan number (cf. [Knu73], p. 63). Each VP is mapped to a () pair. The parenthesis of a VP ψ_1 which is *under* a VP ψ_2 will appear inside the parenthesis of ψ_2. The fact that the layout is crossing-free ensures that the resultant parenthetic expression is legal. For example, the parenthetic expression for the RVPL in Figure 3 is "((()()())(()())(()))((()())(()))((()))".

In Lemma 3.1 we showed that an RVPL induces a tree. The next lemma shows that the converse holds too, namely, any tree induces an RVPL.

Lemma 3.5 *Let T be an ordered tree. Then procedure* INDUCEVPL(T) *(see Figure 2 for the pseudo-code, Figure 3 for an example) induces a crossing-free* RVPL.

Proof. Let Ψ be the output of procedure InduceVPL(T). To obtain a contradiction, assume that Ψ contains a crossing, i.e., a pair of VPs (l_1, r_1) and (l_2, r_2) such that $l_1 < l_2 < r_1 < r_2$. Each VP in Ψ was created from some edge in the tree T, therefore there exist tree edges $(u_1, v_1), (u_2, v_2) \in T$ such that $\lambda(u_1) = l_1$ and $\lambda(v_1) = r_1$ and similarly for (u_2, v_2). Since $\lambda(u_1) = l_1 < r_1 = \lambda(v_1)$ it follows that v_1 is a child of u_1 and similarly v_2 is a child of u_2. From the property of depth-first order, certainly all the labels in the range $[l_1+1, r_1]$ belong to vertices in u_1's subtree, hence u_2 must be a descendent of u_1. Clearly u_2 is on the left of v_1 in the subtree of u_1. But then v_2 is also on the left of v_1 (as u_2's child), so $r_2 = \lambda(v_2) < \lambda(v_1) = r_1$, a contradiction. $\square$

4 Optimal $\mathcal{L}_{\max}$, $\mathcal{H}_{\max}$ Layouts and the Tree $\mathcal{T}(\ell, h)$

In this section we consider optimal RVPL layouts for the worst-case (maximal) load and hop count measures. Specifically, if the load is required to be $\mathcal{L}_{\max} \le \ell$ we characterize the layout with the minimal worst case hop count, and if the hop count is $\mathcal{H}_{\max} \le h$ we characterize the layout with the minimal worst-case load. This is done using a new class of trees $\mathcal{T}(\ell, h)$ (see next definition). These trees contain all RVPLs on a chain that satisfy the above load and hop constraints.

Definition 4.1 *The ordered tree $\mathcal{T}(\ell, h)$ is defined recursively as follows. The root r has ℓ children. The i^{th} child from the left is the root of a $\mathcal{T}(i, h-1)$ subtree, for $1 \le i \le \ell$. A tree $\mathcal{T}(\ell, 0)$ or $\mathcal{T}(0, h)$ is a single vertex (see Figure 3 for an example).*

Remarks: An internal vertex of $\mathcal{T}(\ell, h)$, which is the i^{th} child (from the left) of its parent, has i children. The tree $\mathcal{T}(1, h)$ is a rooted chain of $h+1$ vertices. Note also that $\mathcal{T}(\ell, h)$ has height h and maximum degree ℓ.

Lemma 4.2 *The tree $\mathcal{T}(\ell, h)$ contains $\binom{\ell+h}{h}$ vertices.*

Proof. Let $N(\ell, h)$ denote the number of vertices in $\mathcal{T}(\ell, h)$. Then $N(\ell, h)$ satisfies the recurrence $N(\ell, h) = 1 + N(1, h-1) + N(2, h-1) + \cdots + N(\ell, h-1)$. Since $N(0, h) = 1$ we can write

$$N(\ell, h) = \begin{cases} 1, & \ell = 0 \text{ or } h = 0, \\ \sum_{j=0}^{\ell} N(j, h-1), & \text{otherwise.} \end{cases}$$

One can prove by induction (cf. [GKP89], p. 174) that $N(\ell, h) = \binom{\ell+h}{h}$. $\square$

Lemma 4.3 *Let $T = \mathcal{T}(\ell, h)$ and let $\Psi_T =$ InduceVPL(T), then $\mathcal{L}_{\max}(\Psi_T) = \ell$ and $\mathcal{H}_{\max}(\Psi_T) = h$.*

Proof. It is straightforward to see that $\mathcal{H}_{\max}$ is bounded by h, since the height of $\mathcal{T}(\ell, h)$ is h. The claim for $\mathcal{L}_{\max}$ is proven by induction on ℓ and h: For $\ell = 1$, recall that $\mathcal{T}(1, h)$ is a chain of $h+1$ nodes, thus its induced RVPL has $\mathcal{L}_{\max} = 1$. For $h = 1$, $\mathcal{T}(\ell, h)$ is a "star" with a root of degree ℓ and ℓ leaves. It follows that its induced RVPL has $\mathcal{L}_{\max} = \ell$. To prove the induction step, let r be T's root, and let T_i denote the subtree of the i^{th} child of r, c_i; Note that the VPs induced by T_i in Ψ_T involve exactly the chain nodes in the segment

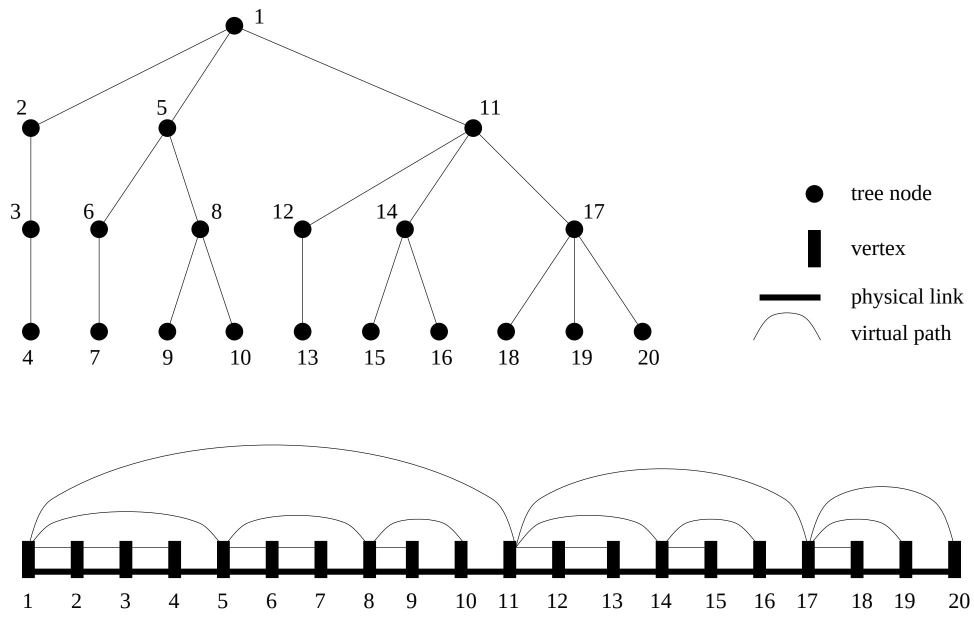


Figure 3: The tree $\mathcal{T}(3,3)$ and its induced RVPL.

$S_i = [\lambda(c_i), \lambda(c_i) + |T_i| - 1]$, and that the only other VPs that traverse nodes in S_i are induced by tree edges from r to children c_j for $j > i$, i.e., $\ell - i$ such VPs. By the induction hypothesis it follows that the maximum load incurred by $T_i = \mathcal{T}(i, h-1)$ on S_i is i, and thus $\mathcal{L}_{\max}$ on S_i does not exceed $\ell - i + i = \ell$. □

Definition 4.4 *An ordered tree T is* subsumed *in $\mathcal{T}(\ell, h)$ if its root is subsumed in the root of $\mathcal{T}(\ell, h)$ 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 $\mathcal{T}(\ell, h)$.*

It is easy to see that the RVPL of a subsumed tree T has lower load and hop counts than that of the tree $\mathcal{T}(\ell, h)$ it is subsumed in, since T may be obtained from $\mathcal{T}(\ell, h)$ by deleting subtrees. Thus, the next corollary follows from Lemma 4.3.

Corollary 4.5 *Let T be an ordered tree that is subsumed in $\mathcal{T}(\ell, h)$, and let Ψ_T be the output $\Psi_T =$ INDUCEVPL(T). Then $\mathcal{L}_{\max}(\Psi_T) \le \ell$ and $\mathcal{H}_{\max}(\Psi_T) \le h$.*

Lemma 4.6 *For every crossing-free* RVPL *Ψ, with $\mathcal{L}_{\max}(\Psi) \le \ell$ and $\mathcal{H}_{\max}(\Psi) \le h$ there exists a tree T which is subsumed in $\mathcal{T}(\ell, h)$ such that $\Psi =$ INDUCEVPL(T).*

Proof. We construct the tree T by induction on h. When $h = 1$ then T is a rooted star with ℓ children, for any ℓ. Assume we can construct such a tree for RVPLs with up to $h - 1$ hops, for any ℓ. Let $a_1, \ldots, a_k$ be the nodes connected directly to the root 1, and let C_j be the set of nodes in Ψ whose path to the root goes through a_j. Note that $k \le \ell$, otherwise the load on the link $(1, 2)$ exceeds ℓ. Since Ψ is crossing-free, no VP connects nodes in different C_j's, and each C_j is a segment of consecutive chain nodes. The VPs touching the nodes in each C_j form a RVPL Ψ_j rooted at a_j, with at most $h-1$ hops. Note that $\mathcal{L}_{\max}(\Psi_{k-i}) \leq \ell - i$ for $0 \leq i \leq k-1$. Therefore by the induction hypothesis we can construct trees $T_1, \ldots, T_k$ corresponding to the RVPLs $\Psi_1, \ldots, \Psi_k$ such that T_{k-i} can be subsumed in $\mathcal{T}(\ell - i, h-1)$ for $0 \leq i \leq k-1$. The root of the requested tree T has k subtrees, which are $T_1, \ldots, T_k$ ordered from left to right. T can be subsumed in $\mathcal{T}(\ell, h)$, since the root has at most ℓ children, and its subtrees can be subsumed in the rightmost k subtrees of $\mathcal{T}(\ell, h)$. $\square$

Theorem 4.7 *Consider a chain of n, and a maximal load requirement ℓ. Let h be such that*

$$\binom{\ell+h-1}{\ell} < n \leq \binom{\ell+h}{\ell}.$$

Then $\mathcal{H}_{\mathrm{opt}}(n, \ell) = h$.

Proof. It is easy to verify that h hops are sufficient (i.e., there exists an RVPL Ψ such that $\mathcal{H}_{\max}(\Psi) = \mathcal{H}_{\mathrm{opt}}(n, \ell)$): use Ψ_T, the output of procedure OptimalMaxH(n, ℓ) of Figure 4. By Corollary 4.5 it follows that $\mathcal{H}_{\max}(\Psi_T) \leq h$ and $\mathcal{L}_{\max}(\Psi_T) \leq \ell$. To show that there exists no RVPL with a lower hop count, assume (to obtain a contradiction) that there exists an RVPL Ψ' with $\mathcal{L}(\Psi') \leq \ell$ and $\mathcal{H}(\Psi') \leq h-1$. Then by Lemma 4.6 there exists a tree T' subsumed in $\mathcal{T}(\ell, h-1)$ which induces Ψ', but

$$|T'| \leq |\mathcal{T}(\ell, h-1)| = N(\ell, h-1) = \binom{\ell+h-1}{h-1} = \binom{\ell+h-1}{\ell} < n ,$$

a contradiction. $\square$

OptimalMaxH(n, ℓ): Construct an RVPL on a chain of n vertices with minimal $\mathcal{H}_{\max}$ such that $\mathcal{L}_{\max} \leq \ell$.

1. Find h such that $\binom{\ell+h-1}{\ell} < n \leq \binom{\ell+h}{\ell}$.
2. Construct the tree $T = \mathcal{T}(\ell, h)$.
3. Repeatedly remove leaves from T until exactly n vertices remain.
4. Return $\Psi_T =$ InduceVPL(T).

Figure 4: Procedure OptimalMaxH(n, ℓ).

A similar result holds for the maximum load measure:

Theorem 4.8 *Consider a chain of n, and a maximal hop requirement h. Let ℓ be such that*

$$\binom{\ell+h-1}{h} < n \leq \binom{\ell+h}{h}.$$

Then $\mathcal{L}_{\mathrm{opt}}(n, h) = \ell$.

Proof. Analogous to the proof of Theorem 4.7

The next simple observation follows from the proof of Theorem 4.7.

Corollary 4.9 *Given a chain with $n = N(\ell, h)$ there exists a unique* RVPL *with $\mathcal{L}_{\max}(\Psi) = \ell$ and $\mathcal{H}_{\max}(\Psi) = h$. As is evident from the* OptimalMaxH(n, ℓ) *procedure (Figure 4) there exist several such* RVPL*s for values of n i.e., with $N(\ell, h-1) < n < N(\ell, h)$).*

Remarks: The above results relate to results in [CGZ94, GZ94] in the following way:

- Theorem 4.7 precisely characterizes the optimal solution on a chain. In addition it shows that if $n = N(\ell, h)$ then $(h!n)^{1/h} - \frac{h+1}{2} \leq \ell \leq (h!n)^{1/h} - 1$. This is an improvement to the upper bound $\ell \leq hn^{1/h}$ of [CGZ94], since $(h!)^{1/h} < h$.

- In [GZ94], a greedy algorithm[2] for finding an RVPL for the more general case of tree networks is presented and proven to be optimal with respect to the $\mathcal{H}_{\max}$ measure. The algorithm does not give insight into the structure of the obtained RVPL, and in particular no upper bound is easily derived from it. When $n = N(\ell, h)$ then Corollary 4.9 and Theorem 4.7 give a characterization of this greedy solution.

5 Minimizing the average case

5.1 Dynamic algorithms to find optimal $\mathcal{L}_{\text{avg}}$, $\mathcal{H}_{\text{avg}}$ layouts

In this section we show a systematic approach to find the optimal layouts for the (unweighted) average load and average hop count measures, which uses dynamic programming algorithms. In Section 5.2 we prove explicit formulas for the optimal values and describe explicit layouts which achieve the optima, according to both measures.

5.1.1 The average load

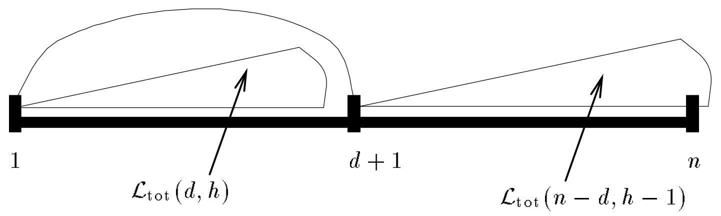
We start with the case where the maximal number of hops is limited to h, and it is required to find the layout with smallest average load. Recalling Lemma 2.9, we observe that a layout Ψ_{opt} that minimizes $\mathcal{L}_{\text{avg}}$ also minimizes its total load $\mathcal{L}_{\text{tot}}(\Psi_{\text{opt}})$. Hence the following definition.

Definition 5.1 *Let $\mathcal{L}_{\text{tot}}(n, h)$ denote the minimal total load of any* RVPL *on n vertices with at most h hops, namely*

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

The rationale behind our dynamic programming algorithm for finding optimal $\mathcal{L}_{\text{tot}}$ ($\mathcal{L}_{\text{avg}}$) layouts is the following. Let Ψ_{opt} be the optimal RVPL (that achieves $\mathcal{L}_{\text{tot}}(\Psi_{\text{opt}}) = \mathcal{L}_{\text{tot}}(n, h)$). Let $(1, d+1)$ be the longest VP connected to the root (see Figure 5). Since by Theorem 3.3 we can assume that Ψ_{opt} is crossing-free, it follows that no VP of Ψ_{opt} connects a vertex $i \leq d$

[2]The model in [GZ94] differs from ours in that the load is measured on the vertices rather on the edges of the network. Therefore the greedy algorithm should be modified to use the edge-load constraint. We refer to this modified algorithm in the comparison.



1
$d+1$
n
$\mathcal{L}_{\text{tot}}(d,h)$
$\mathcal{L}_{\text{tot}}(n-d,h-1)$

Figure 5: The optimal average load layout

with a vertex $j > d+1$, thus Ψ_{opt} can be split into two disjoint optimal layouts, one on vertices $1,\ldots,d$ and the other on $d+1,\ldots,n$. However, the second layout (rooted at $d+1$) may use only $h-1$ hops since one hop is used to traverse the VP $(1,d+1)$. Thus if d is known, then $\mathcal{L}_{\text{tot}}$ satisfies the recurrence

$$\mathcal{L}_{\text{tot}}(n,h) = d + \mathcal{L}_{\text{tot}}(d,h) + \mathcal{L}_{\text{tot}}(n-d,h-1),$$

so clearly

$$\mathcal{L}_{\text{tot}}(n,h) = \min_{1 \le d \le n-1} \{d + \mathcal{L}_{\text{tot}}(d,h) + \mathcal{L}_{\text{tot}}(n-d,h-1)\}. \tag{1}$$

There are two simple "boundary" cases: (i) If $1 \le n \le h+1$ then clearly $\mathcal{L}_{\text{tot}}(n,h) = n-1$, (ii) If $h = 1$ then we must connect a direct VP to each vertex, so $\mathcal{L}_{\text{tot}}(n,1) = n(n-1)/2$. The above argument leads to the dynamic programming algorithm in Figure 6 for finding an optimal RVPL.

Lemma 5.2 *The* OPTIMALAVGL(n,h) *procedure finds an* RVPL *on a chain of n vertices with minimal $\mathcal{L}_{\text{avg}}$ among all* RVPL*s with $\mathcal{H}_{\max} \le h$.*

Proof. We prove by induction on n and h that $A[n,h] = \mathcal{L}_{\text{tot}}(n,h)$: This is clearly true for the boundary cases set in the initialization (Step 2). Consider a crossing-free optimal RVPL Ψ_{opt} for the chain, and assume that $A[i,j] = \mathcal{L}_{\text{tot}}(i,j)$ for all $i \le n$ when $j < h$, and for $i < n$ when $j = h$. When the procedure calculates the value of $A[n,h]$, (in Step 3), it performs the computation of eq. (1), since by the induction hypothesis $A[d,j]$ and $A[n-d,j-1]$ already contain the correct (optimal) values of $\mathcal{L}_{\text{tot}}$. Therefore this computation yields $A[n,h] = \mathcal{L}_{\text{tot}}(n,h)$).

Another simple induction shows that the auxiliary table $D[\cdot,\cdot]$ maintained by the procedure indeed holds the length of the longest VP touching vertex 1 in the optimal RVPL for any values i,j for which $A[i,j]$ has been computed. Therefore the recursive procedure OUTPUTVPL outputs an optimal RVPL. □

Proposition 5.3 *The time complexity of the* OPTIMALAVGL(n,h) *algorithm is $O(n^2h)$.*

Proof. The time complexity is dominated by the calculation step. The algorithm loops nh times in Step 3, and considers at most n values for d to calculate the minimum. □

```
OPTIMALAVGL($n,h$): Construct an RVPL on a chain of $n$ vertices with minimal $\mathcal{L}_{\text{avg}}$ such that $\mathcal{H}_{\max} \le h$.

1. Maintain two tables: $A[i,j]$ holds the value of $\mathcal{L}_{\text{tot}}(i,j)$, and $D[i,j]$ holds
   the length of the longest VP touching vertex 1 in the optimal RVPL, for
   $1 \le i \le n$ and $1 \le j \le h$.

2. For all $j \in \{1,...,h\}$, $i \in \{1,..,j+1\}$ do                (* Initialize *)
       $A[i,j] \leftarrow i-1$; $D[i,j] \leftarrow 1$.
   For all $i \in \{1,...,n\}$ do $A[i,1] \leftarrow \frac{i(i-1)}{2}$; $D[i,1] \leftarrow i-1$.

3. For all $j \in \{2,...,h\}$, $i \in \{j+2,..,n\}$ do                (* Calculate *)
       $A[i,j] \leftarrow \min_{1 \le d \le n-1}\{d + A[d,j] + A[i-d,j-1]\}$.
       $D[i,j] \leftarrow d$ for which the minimum is obtained.

4. Output the RVPL by calling procedure OUTPUTVPL($1,n,h$). The optimal average load is $A[n,h]/(n-1)$.

OUTPUTVPL($k,i,j$): Output the RVPL rooted at node $k$, on $i$ nodes, with at most $j$ hops (using the array $D[\cdot,\cdot]$).

If $i \le 1$ then return.                (* Output nothing *)
$d \leftarrow D[i,j]$; Output the VP $(k,k+d)$.
Recursively call OUTPUTVPL($k,d,j$) and OUTPUTVPL($k+d,i-d,j-1$).
```

Figure 6: Procedure OPTIMALAVGL(n,h).

5.1.2 The average hop count

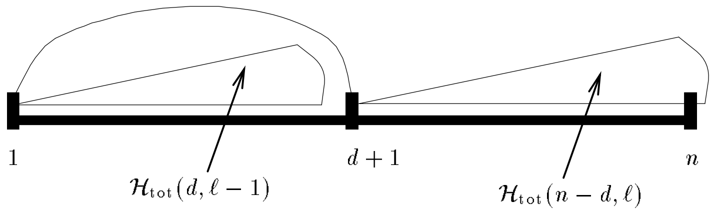
We now turn to the unweighted average hops measure, given a maximum bound ℓ on the load. This problem can be solved by an algorithm similar to that of the average load.

Definition 5.4 *Consider a crossing-free optimal* RVPL Ψ_{opt} *for a chain with n vertices and maximum load ℓ (which achieves the minimum $\mathcal{H}_{\text{tot}}(\Psi)$). Recalling Definition 2.7, define* $\mathcal{H}_{\text{tot}}(n,\ell) \equiv \mathcal{H}_{\text{tot}}(\Psi_{\text{opt}})$.

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 $d+2,\ldots,n$, and thus the layouts in these two segments are disjoint and should both be optimal in Ψ_{opt}. In this case however, the layout on the vertices $1,...,d$ should not exceed the load $\ell-1$ (since together with the VP $(1,d)$ the load should not exceed ℓ). By the above discussion, it is evident that

$$\mathcal{H}_{\text{tot}}(n,\ell) = \min_{1 \le d \le n-1}\{\mathcal{H}_{\text{tot}}(d,\ell-1) + (n-d) + \mathcal{H}_{\text{tot}}(n-d,\ell)\}. \tag{2}$$

The first and third components of the sum are the values of $\mathcal{H}_{\text{tot}}$ in the two separate segments, and the second component is the cost of an additional hop incurred by all vertices in the segment $d+1,...,n$ (see Figure 7). The boundary cases here are $\mathcal{H}_{\text{tot}}(n,1) = n(n-1)/2$ (if



1
$d+1$
n
$\mathcal{H}_{\mathrm{tot}}(d,\ell-1)$
$\mathcal{H}_{\mathrm{tot}}(n-d,\ell)$

Figure 7: The optimal average hops layout

the maximum load is 1 then the only possible VPs are identical to the network edges), and if $n \leq \ell+1$ then $\mathcal{H}_{\mathrm{tot}}(n,\ell) = n-1$ (since we can afford to construct direct VPs from all vertices to the root).

In this case too, a similar dynamic programming algorithm can be devised with time complexity $O(n^2\ell)$. The algorithm is omitted for the sake of brevity.

5.2 Characterizing the Optimal Layouts

5.2.1 The average load

In this section we show an explicit expression for the optimal $\mathcal{L}_{\mathrm{tot}}$ and prove its correctness, based on the recurrence (1) and the dynamic programming algorithm in Figure 6. The following theorem shows that for fixed h, the total load increases as an arithmetic progression between *critical points* of n, which are precisely the values $n = N(\ell,h)$ which we have seen before. The increment of the progression between two consecutive critical points is $\ell+1$.

Theorem 5.5 *Let n and h be given. Let ℓ be the largest integer such that $n \geq \binom{\ell+h}{\ell}$, and let $r = n - \binom{\ell+h}{\ell}$. Then*

$$\mathcal{L}_{\mathrm{tot}}(n,h) = h\binom{\ell+h}{\ell-1} + r(\ell+1).$$

Proof. First note that $0 \leq r < \binom{\ell+h}{\ell+1}$ by the maximality of ℓ. Hence the following definition:

Definition 5.6 $[\ell;h;r]$ *is a* legal representation *of n if $n = \binom{\ell+h}{\ell} + r$ and $0 \leq r < \binom{\ell+h}{\ell+1}$.*

We use induction on n and h. For the induction base we check two cases. If $1 \leq n \leq h$ then to obtain a legal representation we must take $\ell = 0$ and $r = n - \binom{h+0}{0} = n-1$. Therefore $h\binom{h}{-1} + (n-1)(0+1) = (n-1)$. If $n = h+1$ then we need $\ell = 1$ and $r = 0$ for a legal representation, so again $(n-1)\binom{n}{0} + 0\cdot 2 = n-1$. Therefore boundary condition (i) of recurrence (1) holds. If $h = 1$ then $\ell = n-1$ and $r = 0$ for any n. Therefore $h\binom{\ell+h}{\ell-1} + r(\ell+1) = \binom{n}{n-2} = \binom{n}{2}$, as required by boundary condition (ii).

Assume that the theorem holds for n' and h' when $h' < h$ (for all n') and when $n' < n$ for $h' = h$, and we prove for n and h. Let $[\ell;h;r]$ be the legal representation of n. We have two cases to consider.

Case (1): $r < \binom{\ell+h-1}{\ell}$. We take $d = \binom{\ell+h-1}{\ell-1} + r$, and $n - d = \binom{\ell+h-1}{\ell}$. Because of the condition on r it follows that $[\ell-1;h;r]$ is a legal representation of d. Obviously $[\ell;h-1;0]$

is a legal representation of $n-d$. Therefore we can use the induction hypothesis twice, for $\mathcal{L}_{\rm tot}(d,h)$ and for $\mathcal{L}_{\rm tot}(n-d,h-1)$, to obtain

$$\begin{aligned} L(d) &= d+\mathcal{L}_{\rm tot}(d,h)+\mathcal{L}_{\rm tot}(n-d,h-1) \\ &= \binom{\ell+h-1}{\ell-1}+r+h\binom{\ell+h-1}{\ell-2}+r\ell+(h-1)\binom{\ell+h-1}{\ell-1} \\ &= r(\ell+1)+h\binom{\ell+h-1}{\ell-2}+h\binom{\ell+h-1}{\ell-1} \\ &= h\binom{\ell+h}{\ell-1}+r(\ell+1). \end{aligned}$$

Therefore $\mathcal{L}_{\rm tot}(n,h) \le h\binom{\ell+h}{\ell-1}+r(\ell+1)$.

We now show that taking any other d' instead would not yield a smaller value. Recall that the theorem implies that for a fixed h, $\mathcal{L}_{\rm tot}$ increases as an arithmetic progression with increment $\ell+1$ between consecutive critical points, i.e., between values of n for which $[\ell;h;0]$ is a legal representation. If we take $d'=d+1$ then the first term in $d+\mathcal{L}_{\rm tot}(d,h)+\mathcal{L}_{\rm tot}(n-d,h-1)$ increases by 1. Since d has a legal representation of $[\ell-1;h;r]$, the second term increases by ℓ, which is the progression increment for d' and h. Similarly, $n-d$ is a critical point for $h-1$ and ℓ, so moving to smaller values $n-d'$ decreases the third term by ℓ. Therefore $L(d') \ge L(d)+1$. Taking larger values of d' would make the difference even larger. $d'<d$ implies $L(d')=L(d)$ for $d' \ge d-r$, and taking $d'<d-r$ strictly increases L.

Case (2): $\binom{\ell+h-1}{\ell} \le r < \binom{\ell+h}{\ell+1}$. Here we take $d=\binom{\ell+h}{\ell}$, and $n-d=r$. Obviously $[\ell;h;0]$ is a legal representation of d. Let $s=r-\binom{\ell+h-1}{\ell}$. Then $0 \le s < \binom{\ell+h-1}{\ell+1}$ so $[\ell;h-1;s]$ is a legal representation of $n-d$. Again by the induction hypothesis we derive

$$\begin{aligned} L(d) &= d+\mathcal{L}_{\rm tot}(d,h)+\mathcal{L}_{\rm tot}(n-d,h-1) \\ &= \binom{\ell+h}{\ell}+h\binom{\ell+h}{\ell-1}+(h-1)\binom{\ell+h-1}{\ell-1}+s(\ell+1) \\ &= h\binom{\ell+h}{\ell-1}+\binom{\ell+h}{\ell}-\binom{\ell+h-1}{\ell-1}+h\binom{\ell+h-1}{\ell-1}+\left[r-\binom{\ell+h-1}{\ell}\right](\ell+1) \\ &= h\binom{\ell+h}{\ell-1}+r(\ell+1)+h\binom{\ell+h-1}{\ell-1}-\ell\binom{\ell+h-1}{\ell} \\ &= h\binom{\ell+h}{\ell-1}+r(\ell+1), \end{aligned}$$

Hence $\mathcal{L}_{\rm tot}(n,h) \le h\binom{\ell+h}{\ell-1}+r(\ell+1)$. As in case (1) it is easy to check that other values of d never decrease $L(d)$. □

Once we know the optimal value of $\mathcal{L}_{\rm tot}(n,h)$ for all n and h we can fully characterize the optimal layouts. It is easy to verify that when $n=N(\ell,h)$ for some ℓ then the optimal layout is *unique*, and it is precisely the layout which also optimizes the maximum load. In other words, it is the output of procedure INDUCEVPL(T) on the tree $T=\mathcal{T}(\ell,h)$.

For other values of n we suggest the following procedure that outputs one of the possible optimal layouts. Start by finding the maximal ℓ such that $n \ge \binom{\ell+h}{\ell}$, and let $r=n-\binom{\ell+h}{\ell}$.

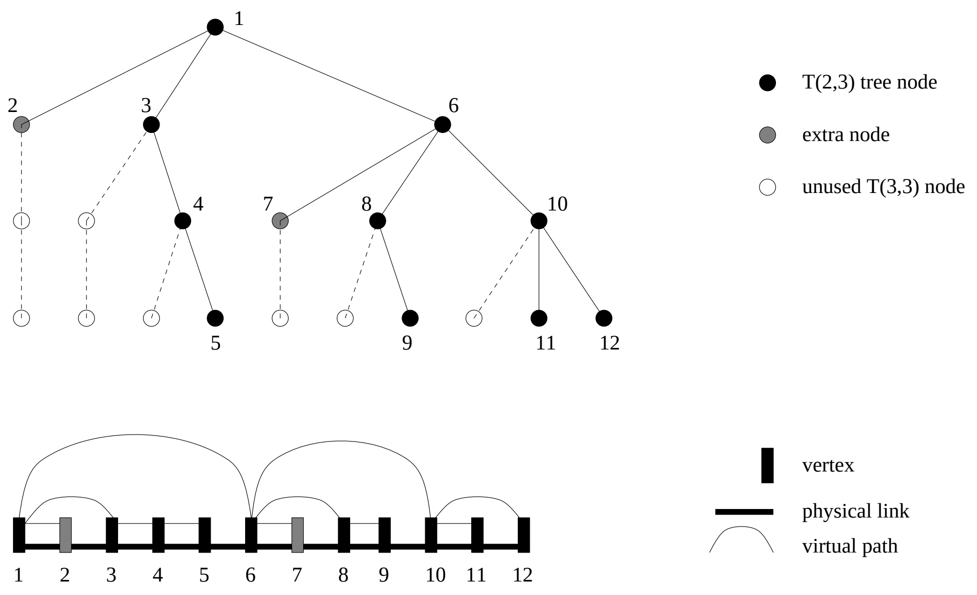


Figure 8: An optimal layout for the $\mathcal{L}_{\text{avg}}$ measure when $n = 12$ and $h = 3$. Shown is the tree $\mathcal{T}(2,3)$ with two extra nodes (vertices 2 and 7) subsumed in $\mathcal{T}(3,3)$, and its induced RVPL.

Then, for this ℓ, embed the tree $\mathcal{T}(\ell,h)$ in $\mathcal{T}(\ell+1,h)$ by recursively "shifting" each node's subtrees to the right. Formally, identify the root of $\mathcal{T}(\ell,h)$ with the root of $\mathcal{T}(\ell+1,h)$, and then recursively embed the subtree $\mathcal{T}(i,h-1)$ rooted at child i from the left into the subtree of child $\mathcal{T}(i+1,h-1)$ rooted at child $i+1$ (see Figure 8). Then arbitrarily connect r extra nodes in the unused portions of $\mathcal{T}(\ell+1,h)$ to obtain a tree T. Finally use procedure InduceVPL(T) on the tree T. As is evident from Figure 8, each of the r extra nodes contributes a load of $\ell+1$ to the load of $h\binom{\ell+h}{\ell-1}$ which is incurred by the original nodes of the tree $\mathcal{T}(\ell,h)$.

5.2.2 The average hop count

The key observation here is that if we take recurrence (2), namely

$$\mathcal{H}_{\text{tot}}(n,\ell) = \min_{1\le d\le n-1}\{\mathcal{H}_{\text{tot}}(d,\ell-1) + (n-d) + \mathcal{H}_{\text{tot}}(n-d,\ell)\},$$

and change variables by $k = n-d$ we get

$$\mathcal{H}_{\text{tot}}(n,\ell) = \min_{1\le k\le n-1}\{\mathcal{H}_{\text{tot}}(n-k,\ell-1) + k + \mathcal{H}_{\text{tot}}(k,\ell)\},$$

which is identical to the average load recurrence (1) when ℓ replaces h. Note that the boundary cases are identical as well, and are unaffected by the variable change. Therefore we can deduce the following corollary by substituting the symbols in Theorem 5.5.

Corollary 5.7 *Let n and ℓ be given. Let h be the maximal such that $n \geq \binom{\ell+h}{h}$, and let $r = n - \binom{\ell+h}{h}$. Then*

$$\mathcal{H}_{\text{tot}}(n,\ell) = \ell\binom{\ell+h}{h-1} + r(h+1). \qquad \Box$$

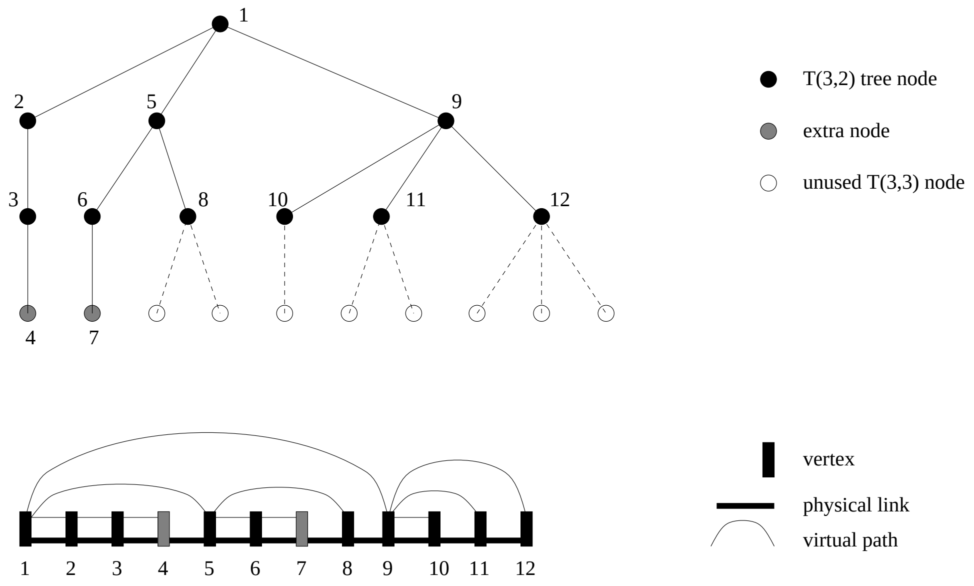


Figure 9: An optimal layout for the $\mathcal{H}_{\text{avg}}$ measure when $n = 12$ and $\ell = 3$. Shown is the tree $\mathcal{T}(3,2)$ with two extra nodes (vertices 4 and 7) subsumed in $\mathcal{T}(3,3)$, and its induced RVPL.

Again when $n = N(\ell,h)$ then the optimal layout is unique, and is defined by activating InduceVPL(T) on the tree $T = \mathcal{T}(\ell,h)$. However for general values of n the procedure to construct optimal layouts is different (and much more intuitive) than the one sketched in the previous section.

Here we start by finding the maximal h such that $n \geq N(\ell,h)$ and constructing the tree $\mathcal{T}(\ell,h)$ for this h. Then we connect the extra $r = n - N(\ell,h)$ nodes as *leaves* at level $h+1$, and activate InduceVPL(T) on the resultant tree T (see Figure 9). Note that the layouts in Figures 8 and 9 are quite different, however the numerical *value* of the total load and total hop count are equal: $\mathcal{L}_{\text{tot}}(12,3) = \mathcal{H}_{\text{tot}}(12,3) = 21$.

5.3 The weighted average hops measure

Finally we handle the weighted average hop count. Here we are able only to show a dynamic programming algorithm based on similar consideration to the unweighted case. The resulting algorithm turns out to have a higher time complexity since the values for $\mathcal{H}^w_{\text{tot}}$ depend not only on the size of the chain but on the specific weights assignment. Finding an explicit construction for the optimal layout seems to be a harder problem here, since the weights destroy the combinatorial structure that the proofs of Section 5.2 rely on.

Definition 5.8 *Given a chain with n vertices and weights assignment $w(v)$, let $W(i,j) = \sum_{v=i}^{j} w(v)$. Define $\mathcal{H}^w_{\rm tot}(i,j,\ell)$ to be the optimal value for $\mathcal{H}^w_{\rm tot}$ for a chain rooted at i and ending at $j \geq i$, with maximum load ℓ.*

Using similar considerations to those in Section 5.1.2, we wish to write a recurrence for $\mathcal{H}^w_{\rm tot}$. However unlike the previous measures, the weighted hop counts of two isomorphic layouts can be different. Therefore in the following recurrence $\mathcal{H}^w_{\rm tot}$ has three indices, with i and j denoting the left- and right-most ends of the current sub-chain. Therefore, given any weight assignment w, the minimal weighted total hop count $\mathcal{H}^w_{\rm tot}$ satisfies

$$\mathcal{H}^w_{\rm tot}(i,j,\ell) = \min_{i \leq d \leq j-1} \{\mathcal{H}^w_{\rm tot}(i,d,\ell-1) + W(d+1,j) + \mathcal{H}^w_{\rm tot}(d+1,j,\ell)\}. \quad (3)$$

The boundary cases here are: (i) When the maximum load is 1 the only possible layout has $\mathcal{H}^w_{\rm tot}(i,j,1) = W(i+1,j)$, and (ii) For a segment which is a single chain link we have $\mathcal{H}^w_{\rm tot}(i,i+1,\ell) = w(i+1)$.

Based on recurrence (3), we can create a dynamic programming algorithm for this problem, similar to procedure OptimalAvgL(n,h) of Figure 6. This algorithm maintains a three dimensional table A with dimensions $n \times n \times \ell$, where $A[i,j,k]$ contains the value of $\mathcal{H}^w_{\rm tot}(i,j,k)$. However, for each value k of the maximum load, the entries of $A[i,j,k]$ are filled in increasing order of the *distance* $j-i$. In other words, entries for immediate neighbors are filled during initialization, then the entries $A[i,i+2,k]$ and so forth. This order ensures that the optimal values for all the sub-chains shorter than $j-i+1$ are available when the algorithm calculates $A[i,j,k]$. Note that roughly half the table is unused since the only entries filled have $i < j$.

Proposition 5.9 *The time complexity of the above algorithm for calculating $\mathcal{H}^w_{\rm tot}(1,n,\ell)$ is $O(n^3\ell)$.*

Proof. The algorithm fills $O(n^2\ell)$ entries in table A. For each entry the algorithm finds the minimum of $O(n)$ sums of eq. (3), each of which contains an evaluation of $W(i,j)$. A naive implementation, which calculates $W(i,j)$ every time, would induce a total time complexity of $O(n^4\ell)$. However the values of W for all $i < j$ can be tabulated in a pre-processing step in time $O(n^2)$, so the main calculation loop performs only $O(n^3\ell)$ operations. □

6 Open Problems

The most immediate open problem is to generalize these results for arbitrary trees, a task which seems non-trivial, as far as the dynamic programming algorithms are concerned, due to the additional structural information that is attached to each subtree (which does not exist in chains).

Another open problem which is of practical interest is to use our RVPLs to construct an optimal many-to-many VPL even for chains. Of particular practical interest is such a construction for the $\mathcal{H}^w_{\rm avg}$ measure.

Acknowledgment

We would like to thank Nicola Santoro for some stimulating discussions. We thank the anonymous referees for their remarks which improved our presentation, and for prompting us to derive the exact formulas of Section 5.2.

References

[ABLP89] B. Awerbuch, A. Bar-Noy, N. Linial, and D. Peleg. Compact distributed data structures for adaptive routing. In *21st Symp. on Theory of Computing*, pages 479–489, 1989.

[AP92] B. Awerbuch and D. Peleg. Routing with polynomial communication-space tradeoff. *SIAM Journal on Discrete Math*, 5(2):151–162, May 1992.

[ATTD94] S. Ahn, R.P. Tsang, S.R. Tong, and D.H.C. Du. Virtual path layout design on ATM networks. In *IEEE Infocom'94*, pages 192–200, 1994.

[BD91] J. Burgin and D. Dorman. Broadband ISDN resource management: The role of virtual paths. *IEEE Communicatons Magazine*, 29, 1991.

[BTS94] H.L. Bodlaender, G. Tel, and N. Santoro. Trade-offs in non-reversing diameter. *Nordic Journal of Computing*, 1:111–134, 1994.

[CCRS87] F. Chung, E. Coffman, M. Reiman and B. Simon. The forwarding index of communication networks. IEEE Trans. on Information Theory, 33, 224-232, 1987.

[CGZ94] I. Cidon, O. Gerstel, and S. Zaks. A scalable approach to routing in ATM networks. In G. Tel and P.M.B. Vitányi, editors, *The 8th International Workshop on Distributed Algorithms (LNCS 857)*, pages 209–222, Terschelling, The Netherlands, October 1994.

[CS94] R. Cohen and A. Segall. Connection management and rerouting in ATM networks. In *IEEE Infocom'94*, pages 184–191, 1994.

[FJ86] G.N. Frederickson and R. Janardan. Separator-based strategies for efficient message routing. In *27th Symp. on Foundations of Computer Science*, pages 428–437, 1986.

[FJ88] G.N. Frederickson and R. Janardan. Designing networks with compact routing tables. *Algorithmica*, 3:171–190, 1988.

[Ger95] O. Gerstel. *Virtual Path Design in ATM Networks.* PhD thesis, Technion, Israel Inst. of Technology, December 1995.

[GKP89] R.L. Graham, D.E. Knuth, and O. Patashnik. *Concrete Mathematics.* Addison-Wesley, 1989.

[GS95] O. Gerstel and A. Segall. Dynamic maintenance of the virtual path layout. In *IEEE Infocom'95*, pages 330–337, April 1995.

[GWZ95] O. Gerstel, A. Wool, and S. Zaks. Optimal layouts on a chain ATM network. In Paul Spirakis, editor, *The 3rd Annual European Symposium on Algorithms (LNCS 979)*, pages 508–522, Corfu, Greece, September 1995. Springer Verlag.

[GZ94] O. Gerstel and S. Zaks. The virtual path layout problem in fast networks. In *The 13th ACM Symp. on Principles of Distributed Computing*, pages 235–243, Los Angeles, USA, August 1994.

[HH91] R. Händler and M.N. Huber. *Integrated Broadband Networks: an introduction to ATM-based networks.* Addison-Wesley, 1991.

[HMS89] M-C. Heydemann, J-C. Meyer and D. Sotteau. On Forwarding Indices of Networks. Discrete Applied Mathematics 23, 103-123, 1989.

[ITU90] ITU recommendation. I series (B-ISDN), Blue Book, November 1990.

[KK77] L. Kleinrock and F. Kamoun. Hierarchical routing for large networks; preformance evaluation and optimization. *Computer Networks*, 1:155–174, 1977.

[KK80] L. Kleinrock and F. Kamoun. Optimal clustering structures for hierarchical topological design of large computer networks. *Networks*, 10:221–248, 1980.

[Knu68] D.E. Knuth. *The Art of Computer Programming, Vol. 1—Fundamental Algorithms*. Addison-Wesley, 1968.

[Knu73] D.E. Knuth. *The Art of Computer Programming, Vol. 3—Sorting and Searching*. Addison-Wesley, 1973.

[LC93] F.Y.S. Lin and K.T. Cheng. Virtual path assignment and virtual circuit routing in ATM networks. In *IEEE Globecom'93*, pages 436–441, 1993.

[Par94] C. Partridge. *Gigabit Networking*. Addison Wesley, 1994.

[PU88] D. Peleg and E. Upfal. A tradeoff between space and efficiency for routing tables. In *20th Symp. on Theory of Computing*, pages 43–52, 1988.

[SOT90] K.I. Sato, S. Ohta, and I. Tokizawa. Broad-band ATM network architecture based on virtual paths. *IEEE Transactions on Communications*, 38(8):1212–1222, August 1990.

[SS91] Y. Sato and K.I. Sato. Virtual path and link capacity design for ATM networks. *IEEE Journal of Selected Areas in Communications*, 9, 1991.