Random Hierarchies that facilitate Self-Organization

Aaron Harwood[†,*], Hong Shen[‡]

† Department of Computer Science and Software Engineering
The University of Melbourne
Victoria, 3010, AUSTRALIA.

‡ Japan Advanced Institute of Science and Technology
Tatsunokuchi, Ishikawa 923-1292, JAPAN

Abstract

Since it is widely accepted that self-organization is difficult to achieve using constructive or centrally run algorithms a random hierarchy is proposed that intrinsically facilitates self-organization. The random hierarchy consists of each node in the network independently choosing a *rank* at random such that a mean $2(\Delta-1)\Delta^{i-1}$ nodes have rank i, where Δ is a network wide hierarchy parameter. Each node of rank i chooses the nearest node of rank $i-1$ as its *leader* which forms the hierarchy. The mean and variance of the relevant properties is derived, for example it is shown that each leader has a mean Δ *followers*. Simulations were used to demonstrate the effectiveness of the proposed hierarchy and a "barebones" set of procedures where provided that may be used to implement the hierarchy over a network of autonomous nodes in a robust way.
keywords: self-organizing, hierarchy, randomness

1 Introduction

It is widely accepted that large systems cannot efficiently operate if a single component of the system is responsible for the global operation. Such operation is called *centrally run* and indeed, centrally run operation is easier to implement since in this case, a non-distributed alogorithm can be adapted to a distributed platform with relative ease. However, it is empirically observed that the Internet is not centrally run and indeed, it is reasoned that this was necessary for its remarkably successful growth. The Internet is said to be *self-organizing* in contrast to being centrally run. If other communication systems are to achieve comparable growth then they must operate using similar principles of self-organization.

But certain kinds of communication services, such as *virtual circuit switching*, cannot provide high QoS if they adhere to protocols such as the Internet Protocol (IP). This is because the IP does not reserve resources in advance and does not distribute information at the Data Link Layer[1]. These communication systems typically make use of distributed algorithms that distribute a relatively large amount of topological information and this fact leads to problem of *scalability*. For a distributed algorithm to be scalable it must make use of a *hierarchy*. Distributed algorithms for self-organizing hierarchies are non-trivial because of coherency and decision deadlocking problems that severely frustrate fast formation. For example, when a hierarchy accepts a new member, it must make sure that the maximum number of members is not exceeded (the hierarchy is only effective if different sub-hierarchies have roughly the same membership size). If a node of a hierarchy makes a decision to accept a new member, the decision must be broadcast to all the other nodes in the hierarchy. If two nodes make such a decision at about the same time, then a coherency problem is highlighted, where the membership may increase above the maximum allowed. Selecting a central or leader node to overcome this imposes a different problem of leader selection. Even when leader selection is achieved, aggregation of hierarchies into higher levels is problematic in the sense that membership to super-hierarchies and leader negotiation can be frustrated by deadlock, where the decision to aggregate is pending a joint decision with a third party which in turn is pending the first decision. It is clear that constructive algorithms are not only non-trivial but perhaps totally infeasible for large networks which cannot make use of a centrally run control.

Dolev et al [6] give a thorough treatment regarding adaptive hierarchies. Unfortunately their method requires that a central authority instigates the initial hierarchy. The method of *Krishnan et al* [11] for clustering arbitrary topologies to form hierarchies also requires a centrally run algorithm.

[*] Some of this work was completed while the author was at Griffith University.

[1] This work uses the OSI model.

Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN'02)
1087-4089/02 $17.00 © 2002 IEEE

Related work involves embedding a *virtual network* over a network which directly supports the construction of virtual circuits and so leads to high performance. *Chlamtac et al* [5] discuss *Lightnets* as topologies for high-speed optical networks. Their work investigates embeddings of the common topologies such as the hypercube and torus. They rigorously show the number of wavelengths required in an arbitrary topology to embed the given topology. Their algorithms operate off-line and the entire network topology must be known for the embedding to be computed. Early work in 1994 by *Chlamtac and Faragó* [4] uses randomly generated paths to optimize a system of virtual paths in terms of capacity. The thesis of *Banerjee* [3] shows extensive use of linear integer programming techniques to provide virtual networks for wavelength routing networks. Later in 1996 *Gerstel et al* [8] show how to decompose networks to embed a cost efficient virtual network over arbitrary topology ATM networks. Again their algorithms execute off-line. Recently, *Afek* and *Bremler-Barr* [1] investigate virtual networks, for MPLS technology, with analogy to a train network. Their method requires modifying the basic technology to include counters and the exact implementation is unclear. *Varela* and *Sinclari* [13] propose an evolutionary computation which is strictly off-line. *Gawlick et al* [7] provide an on-line method for routing permanent virtual circuits but it requires a centrally run authority which is not scalable to large inter-networks. *Gerstel* and *Segall* [9] propose an on-line dynamic maintenance scheme to provide optimal virtual networks given a virtual path layout for ATM networks. Dynamic maintenance can lead to disruptions of existing virtual circuits/virtual paths. Their method is distributed but it requires circulating a token throughout all nodes in the network which makes it unscalable.

Recent work in the field examined the use of randomness to overcome the complexity of self-organizing hierarchies. *Baccelli et al* [2] investigate the use of randomness for self organizing hierarchical multi-cast trees and their optimization. They state that, because so called "core nodes" require knowledge of the entire multi-cast group, their algorithms are not scalable. Although they claim that intermediate solutions can be envisaged, they do not consider them in their work.

In this paper a relatively simple method for constructing a network wide hierarchy is proposed, analyzed and simulated. Section 2 gives the mathematical treatment and simulation results. Section 3 provides a "bare-bones" procedure for a distributed algorithm. Section 4 provides some final comments and future directions.

2 Random Hierarchy

Consider as input an arbitrary network G, with parameters n, $\widehat{d} = \widehat{D}(G)$ and $k = K(G)$. Recall that $\widehat{D}(G) = \frac{2|E(G)|}{n}$ where usually $m = |E(G)| = M(G)$. Assign to each node, $u \in G$, a rank using a set of independent random variables $\mathbf{r}_u$. Let $r_u = \mathbf{r}_u$ be a realization of the random variable for node u. Consider a hierarchy where each node is assigned its rank according to the following probability distribution:

$$\mathbb{P}[\mathbf{r}_u = r] = \frac{\Delta^r}{\sum_{i=1}^{\kappa} \Delta^i} = \frac{\Delta^r (\Delta - 1)}{\Delta^{\kappa+1} - \Delta},$$

where Δ is called the hierarchy parameter, $\kappa = \lfloor \log_\Delta n \rfloor$ with $\Delta \leq n$, and $r = 1, 2, 3, \ldots, \kappa$ over all $u \in G$. The cost product $\kappa\Delta$ is minimized when $\Delta = \exp(1)$. Δ may be constrained by the technology limits or where required for different topological properties to hold. Since $n < \Delta^{\lfloor \log_\Delta n \rfloor + 1} \leq n\Delta$, $\Delta^{\lfloor \log_\Delta n \rfloor + 1} \approx \frac{n\Delta}{2}$ and the number of nodes, $\mathbf{t}_r$, of rank r is a b.r.v., it follows that:

$$\left.\begin{array}{rcl} \mathbb{M}[\mathbf{t}_r] & = & 2(\Delta-1)\Delta^{r-1} \\ \mathbb{S}^2[\mathbf{t}_r] & = & (\Delta-1)\Delta^{r-1}\left(2 - \frac{1}{n}\right) \end{array}\right\} 1 \leq r \leq \kappa. \quad (1)$$

For every $u \in G$ consider the set $L_u = \{ v \mid r_v = r_u - 1 \}$ and then construct the set $L_u^* = \{ v \mid v \in L_u, \delta(u,v) = \delta(u, L_u) \}$. Note that $\delta(x, X) = \min\{ \delta(x,v) \mid v \in X \}$ is the distance from a node to a set of nodes. If $x \in X$ then $\delta(x, X) = 0$. For each node select a single *leader*, $l_u \in L_u^*$, uniformly from L_u^*, ie u chooses the nearest node of rank $r_u - 1$ as a leader. Define $\vec{H}(V, E)$ with $V\left(\vec{H}\right) = V(G)$ and $E\left(\vec{H}\right) = \left\{ (u, l_u) \mid u \in V\left(\vec{H}\right) \right\}$.

It may be that $l_u = \emptyset$. The choice is transmitted to the leader (apart from in the case when it is null) for the leader to maintain a list of *followers*. The list is broadcast in the distributed algorithm to generate a network wide hierarchy as explained in the next chapter. Mechanisms to reduce the broadcast traffic are also discussed. Each node in the set $\{ u \mid r_u = r \}$, is said to have a number of followers of rank $r+1$. All nodes of rank 1 must have the null leader since no node has rank 0. A small example of a random hierarchy is shown in Fig. (1) (without depicting the null leader). The nodes rank from 1 to 3 and each node has selected a leader as indicated by the arrows. The connections of G are not shown but some information about them can be inferred from the leader selections since a leader is always selected on the basis that it is closer than the other leaders. The random hierarchy, $\vec{H}$, is thus a set of upward directed trees with the root of each tree having rank 1 and in general the nodes on level $i > 0$, rank i. The height of a tree is equal to the number of connections traversed from a leaf node to the root.

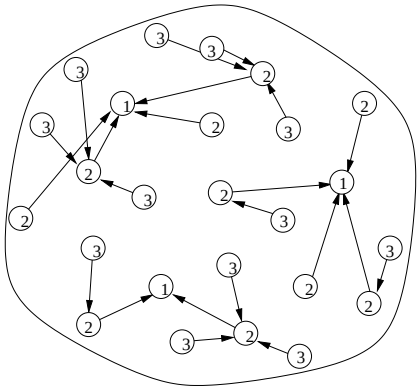

Figure 1: A randomly constructed hierarchy $\vec{H}$.

Lemma 1 *The height of any component tree from* $\vec{H}$ *is at most* $\kappa - 1$.

Proof A node of rank $r_u > 1$ is at most $r_u - 1$ traversals from a rank 1 node. No node has rank greater than κ, so the height is no greater than $\kappa - 1$.□

Let $d^+(u) = |\{\,(v, l_v) \mid u = l_v\,\}|$ be the in degree of node $u \in \vec{H}$ and $d^-(u) = 1$ the out degree so that $d(u) = d^+(u) + d^-(u)$. Node u is a descendant of v if either $u = v$ or u can be reached from v via a directed path. Here, descendant is used to mean closer to the root. Let H_u be the set of nodes for which u is a descendant.

Lemma 2 $\mathbb{M}[d^+(u)] = \Delta$ *with* $\mathbb{S}^2[d^+(u)] = \Delta - \frac{1}{(\Delta-1)\Delta^{r-2}}$.

Proof Node u of rank r, will have probability $p = \frac{1}{\mathbb{M}[\mathbf{t}_r]}$ for each node of rank $r+1$ to select node u as its leader. Each selection is independent of all others. Thus $d^+(u)$ is a b.r.v. with mean $\mathbb{M}[\mathbf{t}_{r+1}]\,p$ and variance $\mathbb{M}[\mathbf{t}_{r+1}]\,p(1-p)$. Hence from Eq. (1) for u of rank r,

$$\mathbb{M}[d^+(u)] = \mathbb{M}\left[\frac{\mathbf{t}_{r+1}}{\mathbf{t}_r}\right] = \Delta,$$

with

$$\mathbb{S}^2[d^+(u)] = \mathbb{M}\left[\frac{\mathbf{t}_{r+1}}{\mathbf{t}_r} - \frac{\mathbf{t}_{r+1}}{\mathbf{t}_r^2}\right] = \Delta - \frac{1}{(\Delta-1)\Delta^{r-2}},$$

□

Lemma 3 *For* u *of rank* 1 *(*$t_1 \neq 0$*),* $\mathbb{M}[|H_u|] \approx \frac{n}{2\Delta}$ *with* $\mathbb{S}^2[|H_u|] \approx \frac{(\Delta-1)n}{2\Delta^2}$.

Proof Using $\Delta^\kappa \approx \frac{(\Delta+1)n}{2\Delta}$ and Eq. (1),

$$\mathbb{M}[|H_u|] = \mathbb{M}\left[\sum_{i=1}^{\kappa} \frac{\mathbf{t}_i}{\mathbf{t}_1}\right] \approx \frac{n}{2\Delta},$$

and

$$\mathbb{S}^2[|H_u|] = \mathbb{M}\left[\sum_{i=1}^{k} \frac{\mathbf{t}_i}{\mathbf{t}_1} - \frac{\mathbf{t}_i}{\mathbf{t}_1^2}\right] \approx \frac{(\Delta-1)n}{2\Delta^2}.$$

□

Remark If $t_1 = 0$ then there is no node of rank 1 and as a consequence the hierarchy components of $\vec{H}$ will be smaller but more plentiful. The probability of a rank r node not occurring among the n nodes is $(1 - \mathbb{P}[\mathbf{r}_u = r])^n$. Since

$$\lim_{n\to\infty} \left(1 - \frac{\Delta^r\,(\Delta-1)}{\Delta^{\kappa+1} - \Delta}\right)^n \approx \exp\left(-2\Delta^r\right),$$

the probability decreases exponentially with a linear increase in Δ^r which is itself increasing exponentially. However, for values of say $\Delta^r = 3$ the probability is about 0.002 which may be considered significant. The use of the null leaders ensures that such cases do not result in an invalid hierarchy. Additional mechanisms are introduced (as shown later) to avoid the use of null leaders in practice but these mechanisms complicate the analysis. In what follows it is assumed that a null leader is not required.

2.1 Rank Spread

Constructive methods for constructing similar hierarchies may attempt to distribute an exact number of ranks evenly throughout the network, ie, to spread them out. Also, it may be stipulated that the nodes of rank i be direct neighbors of their leaders of rank $i-1$. The random hierarchy does not provide such a definitive result but is is possible to estimate the "spread" of a set of nodes chosen at random from G. To be specific, consider x nodes, $X = \{u_1, u_2, \ldots, u_x\} \subset V(G)$ chosen at random. For any $u \in X$ the average distance to the nearest node $v \in (X - u)$ over G is

$$\begin{aligned}\phi(x) &= \sum_{j=1}^{k} j \binom{n - n_{j-1}}{x-1}\binom{n-1}{x-1}^{-1} \\ &\quad \left(1 - \binom{n-n_j}{x-1}\binom{n-1}{x-1}^{-1}\right) \\ &\approx \frac{k^2}{24n^2}\,(n-2x)^2, \qquad (4)\end{aligned}$$

where recall that $n_j = \sum_{i=0}^{j} b_i$ is the expected neighborhood growth of the network (for arbitrary G it would need to be given, computed on-line, or estimated). The approximation should only be used as a rough guide. It was computed by calculating $\sum_{j=1}^{a} j y_1 y_2$ where $a = k(1/2 - x/n)$, $y_1 = (1 - c_0)(j-1)/(a-1) + c_0$, $y_2 = (a-j)(a-1)$ and $c_0 = 1 - \binom{n-\hat{d}-1}{x-1}\binom{n-1}{x-1}^{-1}$. Fig. (2) shows the

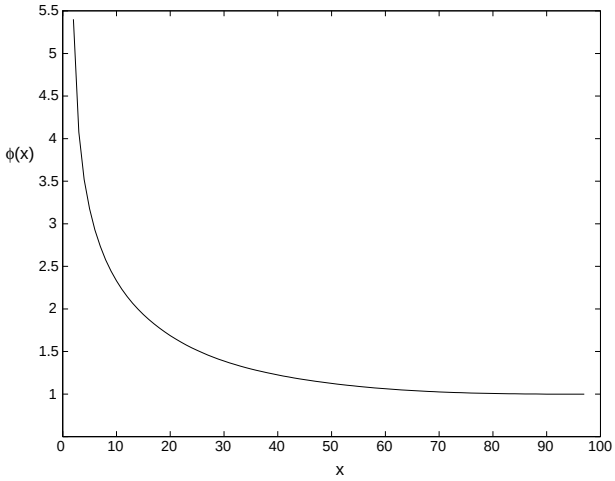

Figure 2: The expected separation $\phi(x)$ of x nodes chosen at random from some $G \in \{G_{n,m}\}$ with $n = 100$ and $\widehat{D}(G) = 3$.

exact $\phi(x)$ for some $G \in \{G_{n,m}\}$ when $n = 100$, $\widehat{D}(G) = 3$, and x ranges from 2 to 100. Intuitively, $\lim_{x \to n} \phi(x) = 1$. Indeed, we should find these nodes on average to be the greatest possible distance apart, since they are uniformly distributed over the network.

2.2 Test Networks

Two specific networks were generated and substituted for G. The first was a random network with 283 nodes, called $R283$, average degree 3, and thus diameter $2 \log_3 283 \approx 10$ (the measured diameter is 12). The second, sample network, called $S275$, shown in Fig. (3), was generated by uniformly distributing nodes in the unit square and choosing edges at random such that no edge had a length greater than 0.2. The $S275$ network has 275 nodes, average degree of 3 and diameter 26.

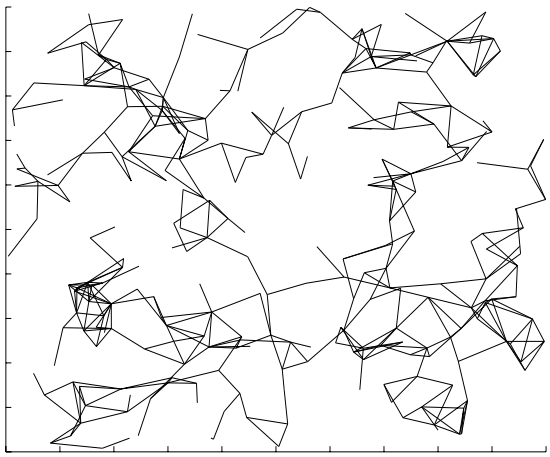

Figure 3: A sample network S275.

Fig. (4) shows an example where nodes have been plotted with shape indicating to which hierarchy they belong to. In this case the hierarchy was computed with $\Delta = 3$. Clearly the network is divided into regions dominated by a particular hierarchy. The example also shows that for instance a rank 1 node may in fact have no followers as exemplified by the lone node at position $(0.45, 0.65)$. Also see that some hierarchies do overlap as in the top left and top right.

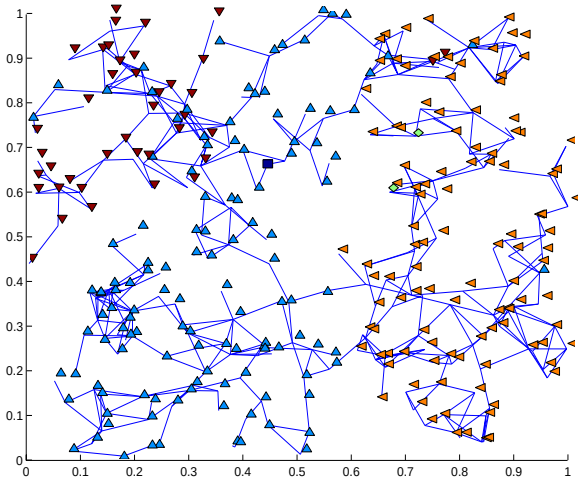

Figure 4: An example random hierarchy in $S275$ with shape indicating to which rank 1 hierarchy each node belongs.

3 Procedures for construction

Due to the simple randomized construction of $\vec{H}$ the hierarchy can be implemented in both a distributed and robust way. The following procedures must be implemented at every node: **Node In**, **Receive Node In Message**, **Receive Leader Response Message**, **Receive Follower Joining Message**, **Receive Follower Leaving Message**,**Node Exits** and **Receive Leader Leaving Message**. It is assumed that error free transmission is available by the underlying communication service.

Node In

1. Node u begins or resumes service.
2. Determines its rank r_u either randomly (according to the probability distribution $\mathbf{r}_u$) or as input from a network administrator.
3. Initialize a table of leaders and a table of followers.
4. Set the current leader to null, $l_u = \emptyset$.
5. If $r_u = 1$ initialize a table of peers.

6. Broadcast r_u to all other nodes using a **Node In** message so that these nodes might select it as a leader in the future and so that potential leaders of u may respond. The extent of this broadcast need only be proportional to the rank, for example a rank 1 node would need entire network coverage but a rank κ node may need only to broadcast to its closest neighbors. The next section explains how to limit the broadcast distance appropriately.

7. Set timer for the case when no leader replies. If the timer expires before a **Leader Response** message is received, the node initiates a **Node Exit** followed by *Node In*.

Receive Node In Message

1. Node u receives a **Node In** message from node v of rank r_v.

2. If $r_v = r_u - 1$ then treat the message as a **Leader Response**.

3. If $r_v = r_u = 1$ then v becomes a peer of u.

4. If $r_v = r_u + 1$ then u sends a **Leader Response** message to v.

Receive Leader Response Message

1. Node u receives **Leader Response** message from node v.

2. Add v to the table of leaders.

3. If v is closer to u than all other leaders in the table, then set $l_u = v$.

4. If the leader has changed, send a **Follower Leaving** message to the old leader and a **Follower Joining** to the new leader.

Receive Follower Joining Message

1. Node u receives a **Follower Joining** message from node v .

2. Add v to the table of followers.

Receive Follower Leaving Message

1. Node u receives a **Follower Leaving** message from node v.

2. Remove v from the table of followers or from the table of peers (depending on r_v).

Node Exits

1. The node u is leaving the hierarchy.

2. Send a *Follower Leaving* message to either the leader, l_u, or each of the peer nodes.

3. Send a *Leader Leaving* message to each of the followers.

Receive Leader Leaving Message

1. Node u receives a **Leader Leaving** message from node v.

2. Remove v from the table of leaders.

3. Choose the closest leader from the table of leaders and send a *Follower Joining* message.

4. If the table of leaders is empty, go to the broadcast step in the **Node In** procedure.

Note that the existing hierarchy structure may be reconfigured when a new node enters due to nodes having to switch leaders to the closer leader. Also see that the use of a null leader is avoided using a peer table for rank 1 nodes and that nodes failing to find a leader will exit and enter until a leader is found. The timer period may be randomly chosen over an exponential distribution.

These procedures give a "bare-bones" approach to establishing $\vec{H}$. In the next section is given a brief discussion on just how the broadcast traffic can be reduced.

3.1 Constraining the Broadcast Traffic

For the random hierarchy to be generated within a network, each node, u, of rank $r+1$ is required to choose the nearest leader of rank r. The $\phi(x)$ function gives an estimate of just how far away in the network this leader is expected to be. For instance $\phi\left(2(\Delta-1)\Delta^{r-1}\right)$ is an estimate for the maximum distance through a network that a node of rank r will be from node u. Conversely, it is expected that a node of rank r need only broadcast its rank to those nodes within a distance $\phi\left(2(\Delta-1)\Delta^{r-2}\right)$ since those nodes at a distance greater than this will be expected to choose some other node of rank r as their leader. This observation can be used to significantly reduce the broadcast traffic required for rank dissemination. This is because nodes of low rank that must broadcast widely are few and nodes of high rank that may broadcast locally are many. The broadcast traffic is not studied in this work and is left as a result of future research.

4 Conclusion

The Internet is said to be self-organizing in contrast to being centrally run. If other communication systems are to achieve comparable growth

then they must operate using similar principles of self-organization. Distributed algorithms for self-organizing hierarchies are non-trivial because of coherency and decision deadlocking problems that severely frustrate fast formation. This paper proposed a relatively simple method for constructing a network wide hierarchy based on randomness, directly facilitating self-organization. The mean and variance of the hierarchy properties were computed. Simulations were used to demonstrate the effectiveness of the proposed hierarchy and a "bare-bones" set of procedures where provided that may be used to implement the hierarchy over a network of autonomous nodes in a robust way.

It is seen that the variance of the hierarchy properties can be relatively large and so some work could be continued to address this issue. As yet, the procedures have not been simulated to reveal such things as network traffic etc, however it was shown that (necessary) network broadcasts can be limited using some knowledge of the networks neighborhood growth structure. An extended study could be undertaken to model and simulate these aspects.

A Random Graph Functions

Consider a network $G \in \{G_{n,m}\}$ of average degree $\widehat{d} = \frac{2m}{n}$ and the function $b_\epsilon(u)$ for any $u \in V(G)$ being the number of nodes of distance ϵ from u. Clearly $b_0 = 1$. The set of graphs $\{G_{n,m}\}$ contains all graphs of n nodes with exactly m edges. If G is chosen randomly from the set then it is known that for every $v \in G$,

$$\begin{aligned} b_{i+1} &= (n - n_i)\left(1 - \left(1 - \frac{\widehat{d}}{(n - n_{i-2} - 1)}\right)^{b_i}\right) \\ n_{i+1} &= n_i + b_{i+1} \end{aligned}$$

A derivation is given by *Rose* [12] which relies on the existence of a Hamilton cycle. Another derivation is given by *Harwood* and *Shen* [10] whereby such a cycle is not required.

References

[1] Y. Afek and A. Bremler-Barr. Trainet: A new label switching scheme. In *INFOCOM: The Conference on Computer Communications, joint conference of the IEEE Computer and Communications Societies*, 2000.

[2] F. Baccelli, D. Kofman, and J. L. Rougier. Self organizing hierarchical multicast trees and their optimization. In *INFOCOM: The Conference on Computer Communications, joint conference of the IEEE Computer and Communications Societies*, pages 1081–1089, 1999.

[3] D. Banerjee. *Design and analysis of wavelength-routed optical networks*. PhD thesis, Univ. Calif., 1996.

[4] I. Chlamtac, A. Farag, and T. Zhang. Optimizing the system of virtual paths. *IEEE/ACM Transactions on Networking*, 2, 6:581–587, 1994.

[5] I. Chlamtac, A. Ganz, and G. Karmi. Lightnets: Topologies for high-speed optical networks. *Journal of Lightwave Technology*, 11(5/6):951–961, 1993.

[6] Shlomi Dolev, Evangelos Kranakis, Danny Krizanc, and David Peleg. BUBBLES: Adaptive routing scheme for high-speed dynamic networks. *SIAM J. Comput.*, 29(3):804–833, 1999.

[7] Rainer Gawlick, Charles Kalmanek, and K. G. Ramakrishnan. On-line routing for permanent virtual circuits. In *IEEE INFOCOM'95*, pages 330–337. IEEE Computer Society Press, April 1995.

[8] Gerstel, Cidon, and Zaks. The layout of virtual paths in ATM networks. *IEEETNWKG: IEEE/ACM Transactions on NetworkingIEEE Communications Society, IEEE Computer Society and the ACM with its Special Interest Group on Data Communication (SIGCOMM), ACM Press*, 4, 1996.

[9] Ornan Gerstel and Adrian Segall. Dynamic maintenance of the virtual path layout. In *IEEE INFOCOM'95*, pages 330–337. IEEE Computer Society Press, April 1995.

[10] Aaron Harwood and Hong Shen. Implementation and performance analysis of stochastic flow generation for IP/ATM-LSR over a random network. In *Proc. International Parallel and Distributed Processing Systems (IPDPS2001)*, 2001.

[11] Rajesh Krishnan, Ram Ramanathan, and Martha Steenstrup. Optimization algorithms for large self-structuring networks. In *INFOCOM: The Conference on Computer Communications, joint conference of the IEEE Computer and Communications Societies*, pages 71–78, 1999.

[12] Christopher Rose. Mean internodal distance in regular and random multihop networks. *IEEE Trans. on Communications*, 40(8):1310–1318, August 1992.

[13] Griselda Navarra Varela and Mark C. Sinclair. Ant colony optimization for virtual-wavelength-path routing. In *1999 Congress on Evolutionary Computation*, pages 1809–1816, Piscataway, NJ, 1999. IEEE Service Center.