Laboratoire Bordelais de Recherche en Informatique
UMR 5800 - Université Bordeaux I, 351, cours de la Libération,
33405 Talence CEDEX, France

Research Report RR-1261-01

Compact and Localized Distributed Data Structures

by Cyril Gavoille and David Peleg

August, 2001

Compact and Localized Distributed Data Structures

Cyril Gavoille* David Peleg†

Abstract

This survey concerns the role of data structures for compactly storing and representing various types of information in a localized and distributed fashion. Traditional approaches to data representation are based on global data structures, which require access to the entire structure even if the sought information involves only a small and local set of entities. In contrast, localized data representation schemes are based on breaking the information into small local pieces, or *labels*, selected in a way that allows one to infer information regarding a small set of entities directly from their labels, without using any additional (global) information. The survey focuses on combinatorial and algorithmic techniques, and covers complexity results on various applications, including compact localized schemes for message routing in communication networks, and adjacency and distance labeling schemes.

Keywords: Compact routing tables, distance labeling, informative labeling schemes

*LaBRI, Université Bordeaux I, 351, cours de la Libération, 33405 Talence Cedex, France. *gavoille@labri.fr*

†Department of Computer Science and Applied Mathematics, The Weizmann Institute of Science, Rehovot, 76100 Israel. *peleg@wisdom.weizmann.ac.il*

1 Introduction

Efficient data structures are at the heart of any data-manipulating computer system, and this fundamental fact is just as valid for distributed systems. Some distributed programming languages support distributed data structures explicitly (cf. [CGL86]), and many distributed algorithms use such structures (explicitly or implicitly) for their data-management purposes. But more importantly, such structures are in many cases constructed not as part of any particular algorithm but for direct use as the building blocks of various storage and retrieval mechanisms, such as distributed dictionaries, name servers in communication networks, bulletin boards, resource allocation managers and the like [Ree78, OD81, FWB85, LEH85, Ter87, MV88, GS89]. The function common to all of these mechanisms is supplying facilities for storing accumulated information in the system and making it available to potential users throughout the system.

The topic of distributed data structures is rather wide, and the distributed setting raises several issues that are not encountered in the usual, shared-memory sequential setting (cf. [Pel90]). In particular, this topic touches upon several large research areas, such as ordinary data structures and data types, distributed database, and concurrency control theory (cf. [LM79, Her84, Wei84, BHG86, Pap86, Her87]). We shall make no attempt to review the relevant literature here, nor shall we address the area of *concurrent data structures*, which refers to data structures stored in common (shared) memory but accessible by many processes concurrently, cf. [BS79, Ell80a, Ell80b, KL80, ML84, Man86, BB87, PK87, RK88], or the area of designing special purpose VLSI machines for implementing data structures, cf. [Lei79, ORS82, DS85, SS85, CCIR86, DS87, OB87, SL87, AK93].

Rather, in this paper we shall restrict ourselves to dealing with one specific aspect of distributed data structures, namely, the relationships between the topology of the underlying communication network (and its graph-theoretic properties) and efficient schemes for organizing and distributing data in the various sites. When the communication network underlying the system is based on a network of arbitrary topology, various graph-theoretic parameters become significant in determining the appropriate way for distributing the data and the resulting complexity.

Our primary concern is to maintain the data structure using reasonable overall space requirements. However, no less important is the need to *balance* the memory loads over the sites of the system. Future systems are expected to carry enormous loads of data, and a single site can hardly be expected to function as the sole storing site for a large data structure. It is therefore desirable to be able to distribute the data in several sites and balance the memory requirements of the data structure between all sites in the network. Such a geographic distribution of data among the different network sites is also dictated by considerations of access speed and reliability.

This survey discusses combinatorial and algorithmic techniques related to these issues, and covers complexity results on various applications. In particular, we will focus on two specific problems which will be used for illustrating the main issues and ideas involved. The first of these problems concerns the design of compact localized schemes for message routing in communication networks, and the second deals with adjacency and distance labeling schemes, and more generally, informative localized labeling schemes. For more detailed presentation of various aspects of these problems see [Gav00, Gav01, Pel00a].

2 Compact Localized Structures for Routing

Delivering messages between pairs of processors is a basic activity of any distributed communication network. This task is performed using a routing scheme, which is a mechanism for routing messages in the network. The routing mechanism can be invoked at any origin vertex and be required to deliver a message to some destination vertex.

Using edge lengths to reflect transmission costs and delays, it is naturally desirable to route messages along paths that are as short as possible. A straightforward approach to achieving this goal is to store a complete routing table in each vertex v in the network, specifying for each destination u the first edge (or an identifier of that edge, indicating the output port) along some shortest path from v to u. This approach clearly guarantees optimal routes, but may be too expensive for large systems since it requires a total of $O(n^2 \log n)$ memory bits in an n-processor network. Thus, an important problem in large scale communication networks is the design of routing schemes that produce efficient routes and have relatively low memory requirements. The efficiency of a routing scheme is measured in terms of its *stretch*, namely, the maximum ratio between the length of a route produces by the scheme for some pair of processors, and their distance.

As a basic example, let us describe an efficient method, known as *interval routing scheme (IRS)* for storing shortest path routing information in a tree network. Start by traversing

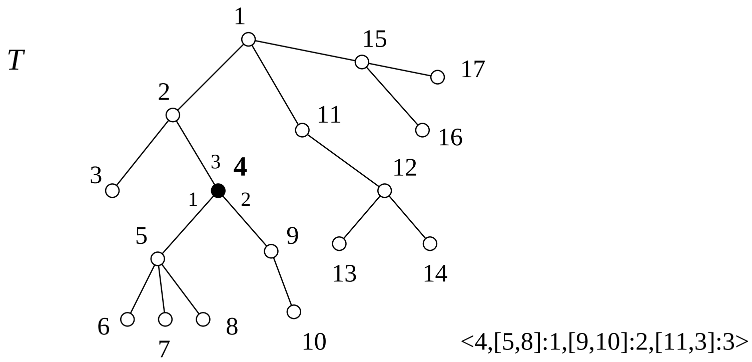

Figure 1: An interval routing for T, and the data structure for the vertex 4.

the n-node tree T in depth first search fashion [Tar72], associating with each vertex v an address label $\ell(v) \in [1, n]$ cf. Fig. 1. Then, associate with each outgoing edge (v, u) of v the set $I(v, u)$ of labels of all the vertices w with the property that the route from v to w starts with the edge (v, u). By construction, the labels contained in $I(v, u)$ are consecutive (modulo n). Hence, to represent $I(v, u)$ in v's memory it suffices to store its *interval* boundaries, at a cost of $O(\log n)$ bits per set. Overall, the routing table of v is a data structure of the form $\langle \ell(v), I(v, u_1) : p_1, I(v, u_2) : p_2, \ldots, I(v, u_d) : p_d \rangle$ where $p_1, p_2, \ldots, p_d$ are respectively the output port numbers leading to the neighbors $u_1, u_2, \ldots, u_d$ (see Fig. 1 for the vertex labeled

4). Therefore the size of the routing table for v is $O(d \log n)$ bits, where d is the degree of v. This should be compared with the $O(n \log d)$ bit bound for a standard routing table. An important feature of this method is the assignment of both vertex and edge labels. Routing a message from a source v to a destination w is done by searching for the interval $I(v, u_i)$ in v's table such that $\ell(w) \in I(v, u_i)$ (implying that the message has to include the destination label $\ell(w)$ in its header), and then forwarding the message through output port p_i to the neighbor u_i. This local computation is then repeated at any intermediate node along the route. This search can be performed using $\log n$ comparisons by sorting the intervals or in $O(\log \log n)$ time using more sophisticated data structures [vBoas77]. Observe that the local memory requirement increases with the degree of the vertex (in Section 4 we discuss other labeling schemes aiming at overcoming the problem of large degree vertices), but the total memory requirement is $O(n \log n)$ bits only.

The interval routing scheme for trees is due to [SK85]. This scheme can be applied rather efficiently to certain restricted graph families, and it has also been extended later to a wider variety of networks, cf. [vLT87, FJ89, Fre93, FGS96, FG98]. When considering interval routing schemes for classes of graphs other than trees, a natural question is to identify which graphs admit interval routing along shortest paths [FG94, NS96, Fla97, EMZ97]. Another natural extension is to allow using more than one interval on each edge, raising the question of how many intervals are necessary to ensure shortest path routing, and how such a scheme can be implemented [FvLS98, GG98, GP98, GP99]. For surveys of the many recent developments in this area see [vLT94, Gav00].

The problem of efficiency-memory tradeoffs for routing schemes was first raised in [KK77], which proposed the general approach of hierarchically clustering a network into κ levels and using the resulting structure for routing. The total memory used by the scheme is $O(n^{1+1/\kappa} \cdot \log n)$. However, in order to apply the method of [KK77] one needs to make some fairly strong assumptions regarding the existence of a certain partition of the network. Several variations and/or improvements were studied later, cf. [KK80, Per82, Sun82].

Most subsequent work on the problem has focused on solutions for special classes of network topologies. Shortest path (i.e., stretch factor 1) routing schemes with total memory requirement $O(n \log n)$ were designed for simple topologies like trees [SK85], unit-cost rings, complete networks and grids [vLT86, vLT87] and networks at the lower end of a hierarchy (beginning with the outerplanar networks) identified in [FJ88]. The problem of designing memory-efficient near-optimal routing schemes was cast in a theoretical formulation in [FJ86, FJ88, FJ89], where it was also given precise solutions for various graph classes up to and including planar graphs. Near-optimal stretched routing schemes were constructed in [FJ89, FJ90] for c-decomposable networks, for constant c, and for planar networks. The schemes for c-decomposable networks guarantee stretch factor ranging between 2 and 3 (specifically, $1 + 2/a$ where $a > 1$ is the positive root of the equation $a^{\lceil (c+1)/2 \rceil} - a - 2 = 0$) and have total memory requirement $O(c^2 \log c \cdot n \log^2 n)$. The schemes for planar networks guarantee stretch factor 7 and have total memory requirement $O(\frac{1}{\epsilon} n^{1+\epsilon} \cdot \log n)$ bits for any constant $0 < \epsilon < \frac{1}{3}$. A crucial step in constructing these routing schemes is assigning names to the vertices as part of the routing scheme. The above optimal schemes use $O(\log n)$ bit labels, and the routing mechanism needs to modify the header during the message propagation.

The schemes of [FJ90] for c-decomposable networks use $O(c \log c \cdot \log n)$ bit names and the schemes of [FJ89, FJ90] for planar networks use $O(\frac{1}{\epsilon} \log n)$ bit names and rewritable headers. Recently, [GH99] have constructed new routing schemes for planar graphs with optimal stretch 1. Based on book embeddings, these schemes use label names $\in [1, n]$ and work with non rewritable headers. The memory bound is $8n + o(n)$ bits per vertex. The schemes can be extended to g genus graphs with $n \log g + O(n)$ memory bits per vertex. As another example, the construction of 3-spanners for the family of chordal graphs described in [PS89] can be used to construct routing schemes for these graphs with stretch factor 3 and $O(n \log^2 n)$ bits of memory in total. For Euclidean networks, namely, networks whose sites are embedded in the 2-dimensional plane with Euclidean distances, recent papers have dealt with proposing efficient designs for compact routing schemes, based on *compass routing* methods (cf. [BCSW98, KSU99]) or efficient spanner constructions [HP00].

The problem of constructing compact routing schemes for arbitrary unweighted networks was studied in [PU89], which presents a family of hierarchical routing schemes (for every fixed integer $\kappa \geq 1$) that guarantee stretch $O(\kappa)$ and require storing a total of $O\left(\kappa^3 \cdot n^{1+1/\kappa} \cdot \log n\right)$ bits of routing information in the network. Just as for the IRS approach and the routing schemes of [FJ89, FJ90], the schemes of [PU89] require assigning suitable names to the vertices. these names are of size $O(\log^2 n)$ bit. However, the headers attached to the messages require only $O(\log n)$ bits. Furthermore, these schemes (as well as most earlier approaches, such as IRS and the schemes of [FJ89, FJ90]) have the disadvantage that the routing information is not balanced on the set of vertices. In the worst-case, some vertices may require $\Omega(n \log n)$ bits of memory.

The solution of [PU89] was later generalized in a number of ways, and various qualities of the resulting schemes were improved in [PU87, ABLP89, ABLP90, AP92, Pel93]. For instance, the schemes were extended to weighted graphs, they were modified to work in a setting where vertices can freely select their own names or routing labels, they were provided with efficient and distributed preprocessing procedures and so on. These developments parallel a chain of successive improvements in the corresponding cluster-based representations used by the schemes. Recent developments concerning compact routing schemes with low stretch (mainly integral stretches ≤ 5) are presented in [KKU95, FG97, GG97, NO97, EGP98, CG00, Cow01], and also in [TZ01] for higher stretches. See [Gav01, Pel00a] for more detailed overviews.

Lower bounds for the space-efficiency tradeoff of routing schemes were studied in [PU89, FG95, FG96, GP96, BHV96, KK96, GG97]. More precisely, in [PU89] it is shown that every routing strategy that guarantees an s stretched routing scheme for every n-vertex graph must provide at least a total of $2^{\Omega(n^{1+1/(2s+4)})}$ different routing schemes. Thus no routing strategy can guarantee for every graph a routing scheme with a stretch factor $O(\kappa)$ and $o(n^{1+1/\kappa})$ bits of total memory. For the case of optimal stretch 1, it is shown in [GP96] that for every shortest path routing strategy and for every d such that $3 \leq d \leq (1-\epsilon)n$, there exists a worst-case graph of degree bounded by d on which the total memory requirement is $\Omega(n^2 \log d)$, matching with the memory requirements of standard routing tables. Both lower bounds assume that routes and $O(\log n)$ bit label names can be computed and optimized by the routing strategy in order to decrease the memory requirement.

The issues of name independence and balancing the memory requirements were first raised in [ABLP89]. The schemes proposed in [ABLP89, Pel93] are name-independent and apply to arbitrary weighted networks. However, they have an inferior efficiency-space tradeoff. For instance, the schemes of [ABLP89], for $\kappa \geq 1$, use $O(\kappa \cdot n^{1/\kappa} \cdot \log n)$ bits of memory per vertex and guarantee a stretch of $O(\kappa^2 \cdot 9^\kappa)$. The tradeoff was finally improved by the schemes of [AP92], which are simpler, and possess the additional attractive features discussed above. The stretch is $O(\kappa^2)$ and the memory requirement is $O(\kappa \cdot n^{1/\kappa} \log^2 n \cdot \log D)$ bits per vertex, where D is the weighted diameter of the network. In fact, the tradeoff obtained in the schemes of [AP92] is still not optimal, and it is conceivably possible to reduce the stretch factor of the routing schemes from $O(\kappa^2)$ to $O(\kappa)$. Several other types of routing schemes for general networks are presented in [PU87, ABLP90, TZ01].

It is worth noting that the scheme of [ABLP89] has one additional advantage, namely, its memory complexity is independent of the range of the edge costs, or the network's diameter (or put another way, the routing algorithm is "purely combinatorial"). Finally, it has been proven in [EGP98] that, whatever the stretch bound is, every name-independent routing strategy that guarantees less than $O(\sqrt{n})$ bits per vertex needs rewritable headers. (Actually it is not difficult to see that the memory bound can be pushed to $\Theta(n \log n)$ bits.) So, the routing protocols of [ABLP89] and [AP92] that, in essence, proceed in $O(\kappa)$ routing phases (which need to be memorized in the headers) cannot be simplified below a certain point.

Other related work deals with routing with succinct routing tables. The case of dynamic networks is dealt with in [AGR89] in the limited setting of networks whose topology is a tree, and the topological changes are restricted to growing (i.e., new edges and vertices are occasionally added to the network). And recently, the routing problem was dealt with in the context of the new generation of ATM and optical networks [DKKP95].

3 Adjacency and Distance Labeling Schemes

Most traditional *centralized* approaches to the problem of network representation are based on storing adjacency information using some kind of a data structure, e.g., an adjacency matrix. Such representation enables one to decide, given the indices of two vertices, whether or not they are adjacent in the network, simply by looking at the appropriate entry in the table. However, note that (a) this decision cannot be made in the absence of the table, and (b) the indices themselves contain no useful information, and they serve only as "place holders", or pointers to entries in the table, which forms a *global* representation of the network.

In contrast, for a distributed computing setting we are interested in more *informative* and *localized* schemes for representing the network. In particular, *labeling schemes* are based on the idea of associating with each vertex a label selected in a such way, that will allow us to infer the adjacency of two vertices *directly* from their labels, without using *any* additional information sources. Hence in essence, this rather extreme approach to the network representation problem *discards* all other components, and bases the entire representation on the set of labels *alone*.

Obviously, labels of unrestricted size can be used to encode any desired information. Specifically, it is possible to encode the entire row i in the adjacency matrix of the graph in the label chosen for vertex i. As another concrete example, adjacency labeling systems of general graphs based on Hamming distances were studied in [Bre66, BF67]. Specifically, in [BF67] it is shown that it is possible to label the vertices of every n-vertex graph with $2n\Delta$ bit labels such that two vertices are adjacent if and only if their labels are at Hamming distance $4\Delta - 4$ or less of each other, where Δ is the maximum vertex degree in the graph.

However, efficiency considerations, similar to those discussed in the previous section regarding routing schemes, dictate the use of relatively *short* labels (say, of length polylogarithmic in n), which nevertheless allow us to deduce adjacencies efficiently (say, within polylogarithmic time).

Efficient *adjacency labeling schemes* were introduced in [KNR88]. In particular, a labeling scheme using $2\log n$ bit labels was proposed for the class of trees. Given a tree T with n vertices, choose a root and associate a distinct integer $\ell(v) \in [1, n]$ with each vertex v of T, and then assign a vertex v with parent w the label $\langle \ell(v), \ell(w) \rangle$. Given two labels $\langle \ell(v), \ell(w) \rangle$ and $\langle \ell(v'), \ell(w') \rangle$, one can check if the vertices v and v' are neighbors, as this happens if and only if one is the parent of the other, i.e., if either $\ell(v) = \ell(w')$ or $\ell(v') = \ell(w)$. This scheme was extended in [KNR88] to $O(\log n)$ adjacency labeling schemes for a number of other graph families, such as bounded arboricity graphs (including, in particular, graphs of bounded degree or bounded genus, e.g., planar graphs), various intersection-based graphs (including interval graphs), and c-decomposable graphs.

This natural idea lay dormant for over a decade, until interest in this direction was revived by the observation that the ability to decide adjacency is only *one* of *a number* of basic properties a representation may be required to possess. In particular, another natural property of interest may be the ability to determine the *distance* between two vertices efficiently (say, in polylogarithmic time again) given their labels. This has led to the notion of *distance labeling schemes*, which are schemes possessing this ability [Pel99]. It is clear that distance labeling schemes with short labels are easily derivable for highly regular graph classes, such as rings, meshes, tori, hypercubes, and the like. It is less clear whether more general graph classes can be labeled in this fashion. It was shown in [Pel99] that the class of n-vertex weighted trees with m bit edge weights enjoys an $O(m \log n + \log^2 n)$ distance labeling scheme. This scheme is complemented by a matching lower bound [GPPR01], showing that $\Omega(m \log n + \log^2 n)$ bit labels are necessary for this class. In [GPPR01] the scheme is extended to n-vertex graphs with an $r(n)$-separator. It is shown that this class supports a scheme with labels of size $O(R(n) \cdot \log n)$, where $R(n) = \sum_{i=1}^{\log n} r(n/2^i)$. We have $R(n) \leq r(n) \log n$, and for monotone $r(n) \geq n^\epsilon$ with constant $\epsilon > 0$, $R(n) = O(r(n))$. For bounded treewidth graphs (including trees, outerplanar graphs, series-parallel graphs, k-outerplanar graphs and c-decomposable graphs for constant k and c), $R(n) = O(\log n)$ since $r(n) = O(1)$, and for bounded genus graphs (including planar graphs), $R(n) = O(r(n)) = O(\sqrt{n}\,)$.

Roughly speaking, the scheme is based on building a tree-decomposition T of the n-vertex graph G (cf. Fig. 2). Each node of T corresponds to a separator of G. In particular, the root of T corresponds to a subset S of vertices of G such that $|S| \leq r(n)$ and such that $G \setminus S$ consists of connected components of size at most $n/2$. (If G is itself a tree then $r(n) = 1$, and the singleton S is a center of the tree.) Each connected component of $G \setminus S$ corresponds to a subtree of T, so that any shortest path from u to v in G taken from different subtrees has to cross some vertices of S. The label of u, $L(u)$, consists of the concatenation of all the distances in G between u and the vertices of G contained in all the ancestor nodes of u in T (the node containing u has at most $\log n$ ancestors). To compute the distance between u and v, it suffices to compute their least common ancestor in T, say S, and then to compute $d(u,v) = \min_{z \in S} \{d(u,z) + d(v,z)\}$. Note that to compute this minimum, the labels of u and of v must encode the vertices of S.

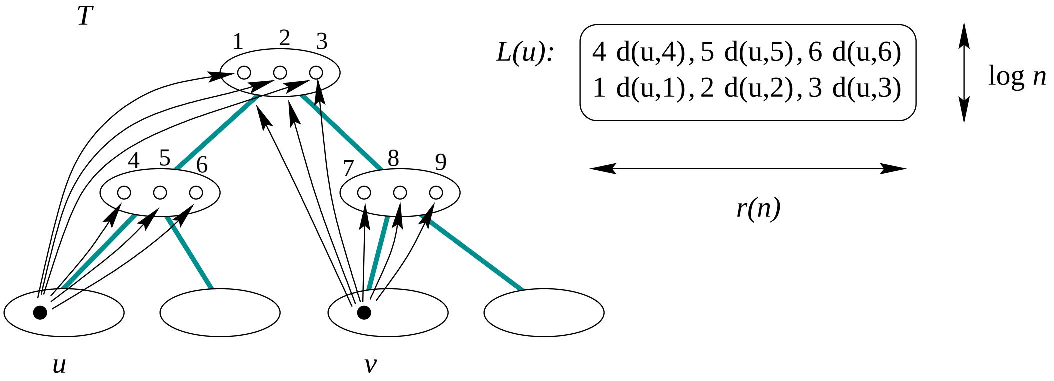

Figure 2: The separator technique for distance labeling.

This scheme is near-optimal since there is a lower bound of $\Omega(r(n))$ on the label size for the class of all the graphs having an $r(n)$-separator [GPPR01]. However, for the class of planar graphs (which is a proper subclass of the class of graphs with $O(\sqrt{n})$-separator) there is a specific lower bound of $\Omega(n^{1/3})$, leaving an intriguing (polynomial) gap. More recently, schemes with $O(\log^2 n)$ bit labels that do not make use of the separator technique were presented for n-vertex interval and permutation graphs [KKP00] and for distance hereditary graphs [GP01a].

As observed in [KNR88], a class of $2^{\Omega(n^{1+\epsilon})}$ n-vertex graphs, must use adjacency labels (and thus distance labels) whose total combined length is $\Omega(n^{1+\epsilon})$, hence at least one label must be of $\Omega(n^{\epsilon})$ bits. Specifically, for the class of all unweighted graphs, any distance labeling scheme must label some n-vertex graphs with labels of size $\Omega(n)$. Conversely, there exists a scheme for the class of arbitrary unweighted n-vertex graphs with $O(n)$ bit labels, which requires $O(\log\log n)$ time to decode the distance from the labels [GPPR01]. Hence $\Theta(n)$ bits is the optimal distance label length for general unweighted graphs.

This raises the natural question of whether more efficient labeling schemes can be constructed if we abandon the ambitious goal of capturing *exact* information, and settle for obtaining *approximate* estimates. An (s,r)-*approximate* distance labeling scheme is a distance labeling scheme such that for u, v coming from the same graph, the estimated distance $\tilde{d}(u,v)$ computed by the scheme from the labels $L(u)$ and $L(v)$ satisfies $d(u,v) \le \tilde{d}(u,v) \le s \cdot d(u,v) + r$. In particular, distance labeling schemes coincide with $(1,0)$-approximate distance labeling schemes.

General weighted graphs were given an $(8\kappa, 0)$-approximate distance labeling scheme, for every integer $\kappa \geq 1$, with $O(\kappa \cdot n^{1/\kappa} \log n \cdot \log D)$ bit labels [Pel99], where D is the weighted diameter of the graph, and later an improved $(2\kappa - 1, 0)$-approximate scheme with $O(n^{1/\kappa} \log^{1-1/\kappa} n \cdot \log(nD))$ bit labels [TZ01a]. The time to decode the estimated distance is $O(\kappa)$. This implies a $(2 \log n, 0)$-approximate scheme with $O(\log^2 n)$ bit labels for general unweighted graphs. These results are complemented by a lower bound in $\Omega(n^{1/\kappa})$ on the label size of $(\Omega(\kappa), 0)$-approximate schemes, presented independently in [TZ01a] and in [GKK+01].

It is interesting to notice that a small variation on the quality of the estimators, say moving from $(1, 0)$-approximate to $(1 + o(1), 0)$-approximate or to $(1, O(1))$-approximate schemes, results in a significant impact on the label size. Trees, and more generally graphs with $r(n)$-separator, support a $(1 + 1/\log n, 0)$-approximate scheme with $O(R(n) \cdot \log \log n)$ bit labels [GKK+01]. In particular, trees enjoy $O(\log n \cdot \log \log n)$ bit label $(1 + 1/\log n, 0)$-approximate distance labeling scheme. A lower bound of $\Omega(\log n \cdot \log \log n)$ is also shown in [GKK+01] for any $(1 + 1/\log n, 0)$-approximate distance labeling scheme on the class of trees. Recently, [KM01] proposed a $\log n + O(\sqrt{\log n})$ bit labeling scheme that allows computing the exact distance between two vertices of a tree at distance $d < \sqrt{\log n}$, thus improving on the $2 \log n$ bit solution of [KNR88] for adjacency labeling schemes in trees.

A number of additional approximate distance labeling schemes are presented in [GKK+01], including a $(3, 0)$-approximate scheme with $O(n^{1/3} \log n)$ bit labels for planar graphs, a $(1, 1)$-approximate scheme with $O(\log n)$ bit labels for interval graphs, a $(1, 2)$-approximate scheme with $O(\log n)$ bit labels for permutation and AT-free graphs, and a $(1, \lfloor c/2 \rfloor)$-approximate scheme with $O(\log^2 n)$ bit labels for c-chordal graphs (namely, all graphs whose longest induced cycle is no greater than c). In particular, it yields a $(1, 1)$-approximate labeling scheme for chordal graphs, to be contrasted with the fact that every exact ($(1, 0)$-approximate) scheme requires $\Omega(n)$ bit labels on some chordal graphs. The question of the exact label size complexity of distance labeling scheme of interval and permutation graphs is left open, with the bounds ranging from $\Omega(\log n)$ to $O(\log^2 n)$ [KKP00].

Finally, a quality measure of interest is the time required for decoding the labels and deducing the information stored in them. All the approximate schemes presented in [GKK+01] (for trees, planar, c-chordal and interval graphs and so on) require a constant time complexity for decoding the distance estimator on a word-RAM computer. However, an intriguing result established in [GPPR01] is that there exist n-vertex graphs G_n which enjoy a distance labeling with labels of size $O(\log n)$ on the one hand, but on the other hand, if one uses on G_n labels with fewer than $n/2$ bits, then time exponential in n may be required for decoding the distance. A similar result is obtained therein for planar graphs.

4 Informative Localized Labeling Schemes

Routing, adjacency and distance labeling schemes have several common features. Most importantly, they all address the general question of developing label-based network representations that allow retrieving useful information about arbitrary functions or substructures in a graph in a *localized* manner, i.e., using only the local pieces of information available to, or associated with, the vertices under inspection, and not having to search for additional *global* information. We refer to such representations as *informative labeling schemes*, formally described in [Pel00b].

To illustrate this concept with respect to additional functions, let us concentrate on the class of *rooted trees*. In addition to finding out whether two given vertices v and w are adjacent, or what is the distance between them, one may be interested in many other pieces of information concerning these vertices. For example, in some cases it may be useful to know if v is an *ancestor* (or a descendant) of w. It is rather easy to encode the ancestry relation in a tree with $2\log n$ bit labels using interval-based schemes (cf. [SK85]). It turns out that ancestor queries can be handled by a scheme using $\log n + O(\sqrt{\log n})$ bit labels [TZ01, KM01a]. More sophisticated labeling schemes allow us to combine parent and ancestor queries with $2\log n + O(\log\log n)$ bit labels [KM01a]. Moreover, queries can be answered in constant time on a word-RAM computer.

Another example for a piece of non-numeric information that may be required is the *least common ancestor* of v and w. Standard solutions [HT84, SV88] can answer such queries in constant time with suitable preprocessing of the tree, but cannot be applied in a localized computation setting., as they require some accesses to a global table of $O(n)$ items. In [Pel00b], it is shown that the identifier of the least common ancestor can be found using a labeling scheme with $O(\log^2 n)$ bit labels. This scheme is asymptotically optimal if vertices have freely chosen their own identifier. However, if it is only required to return the *label* of the least common ancestor (that is, all the vertex identifiers consist of the labels issued by the labeling scheme), then it can be done with $O(\log n)$ bit labels [AGKR01]. Another related function is the *separation level* of two vertices of a rooted tree, defined as the depth of their common ancestor. This function is given in [Pel00b] a labeling scheme similar to the one for distance labeling, with (asymptotically optimal) $O(\log^2 n)$ bit labels. As an additional example, labeling schemes for flow and connectivity were studied in [KKKP01]. An (asymptotically optimal) flow labeling scheme using $O(\log n \cdot \log \omega)$ bit labels is presented for general n-vertex graphs with maximum (integral) capacity ω. For edge-connectivity, this yields a tight bound of $\Theta(\log^2 n)$ bits. Also, a k-vertex connectivity labeling scheme is given for general n-vertex graphs using $O(\log n)$ bit labels for fixed k. Finally, a lower bound of $\Omega(k \log n)$ is established for k-vertex connectivity on n-vertex graphs where k is polylogarithmic in n.

The types of localized information to be encoded by an informative labeling scheme are not limited to *binary* relations. An example for information involving *three* vertices v, w and u is finding their *center*, namely, the unique vertex z such that the paths connecting it to v, w and u are edge disjoint. More generally, for any subset of vertices S in a weighted graph, one may be interested in inferring $w(S)$, the weight of their *Steiner tree* (namely, the lightest tree spanning them), based on their labels. It is easy to verify that an exact Steiner labeling scheme for the the class of n-vertex graph requires $\Omega(n)$ bit labels. However, the class of arbitrary n-vertex graphs with m bit edge weights admit a $O(\log n)$-multiplicative approximate Steiner labeling scheme using $O(m \log^2 n + \log^3 n)$ bit labels [Pel00b]. For n-vertex trees with m bit edge weights, there exists an exact scheme with $O(m \log n + \log^2 n)$ bit labels, which is asymptotically optimal [Pel00b].

Revisiting compact routing schemes as informative labeling schemes recently yielded some improvements for routing problem on trees. Informally speaking, the routing problem can be presented as requiring us to assign two kinds of labels to every vertex of a graph. The first is the *address* of the vertex, whereas the second label is a data structure called the *local routing table*. The labels are assigned such a way that at every source vertex v and given the address of any destination vertex u, one can decide the first edge (or an identifier of that edge) outgoing of v that leads to u (say, through a shortest path). The decision must be taken locally in v, based solely on the two labels of v and with the address label of u. For instance, a labeling for trees that uses $3 \log n$ bits for the addresses and $O(\min \{d \log n, \sqrt{n} \log n\})$ bits for the local routing table, where d is the degree of the vertex, is constructed in [Cow01]. (This improves on the labeling presented in Section 2 for large degree vertices). In [EGP98] it is shown that every routing scheme that selects addresses in the range $[1, n]$ has an $\Omega(\sqrt{n}\,)$ bit local routing table for some n-vertex trees. Hence by increasing the address size, a variant of this problem would be to consider routing labels such that the message can be routed between v to u relying solely on their label (and possibly the labels of the intermediate vertices along the route) without any routing tables. This leads to the notion of *routing labeling schemes*. Obviously such a labeling can be obtained from a standard routing scheme by concatenating in a single label, for every vertex v, the address of v and its local routing table. Surprisingly, n-vertex trees have routing labeling schemes with only $c \log n$ bit labels [FG01], for a small constant c. It is even proved in [TZ01] that the constant c can be reduced to $c = 1 + O(1/\log\log n)$. Combined with the $\Omega(\sqrt{n}\,)$ lower bound of [EGP98], this emphasizes that a variation of an additive term of $O(\log n/\log\log n)$ bits on the size of the addresses plays an important role on the size of the routing table.

At this stage, let us discuss potential *applications* for informative labeling schemes. It seems likely that labeling schemes may prove useful for various applications in the contexts of communication networks and distributed protocols. The relevance of distance labeling schemes in the context of communication networks has been pointed out in [Pel99], and illustrated by discussing the potential application of such labeling schemes to distributed connection setup procedures in circuit-switched networks. Some other problems where it seems that distance labeling schemes may be useful include memory-free routing schemes, bounded ("time-to-live") broadcast protocols, topology update mechanisms etc. For specific classes of graphs, like rooted trees, it is shown in [AKM01] how to use ancestor labeling schemes to optimize queries on large database with XML search engines. It is also plausible that other types of informative labeling schemes may prove useful for other applications. For instance, one can envision using Steiner labeling schemes as a tool for optimizing multicast schedules and selection of subtrees for group communication, and potentially even for certain information representation problems on the Web. Moreover, one may expect that suitable informative labeling schemes will be applicable in entirely different application domains as well, including for instance computational geometry (in the context of Euclidean graphs) and combinatorial optimization in general, by viewing a vertex labeling as a "nice," i.e., easily managable, representation of the graph

Let us conclude with a brief discussion of future prospects. As observed along this paper, both the quality and the cost of an informative labeling scheme depend on two central factors: the *type of information* handled by the scheme, and the *class of networks* for which the scheme is designed. Nevertheless, there is hope that general and uniform algorithmic and data-structuring techniques will emerge that will facilitate the design of informative labeling schemes for many types of information, or even the design of general schemes capable of encoding a group of information types together, for instance, routing and distance.

Finally, the information types handled by the labeling scheme may not necessarily be directly related to the topology of *the graph itself*. Rather, it may be derived from various other types of (external) data, stored in its vertices of the network. The idea is to eventually be able to come up with data structures that will allow "local" deductions on the basis of small parts of the data, without having to inspect the entire data structure. Conceivably, this may lead to the development of abstract types of "fragmented" (or "localized") data structures, whose dependencies on the topology are only partial, giving rise to many interesting and new problems.

References

[AKM01] S. Abiteboul, H. Kaplan, and T. Milo. Compact labeling schemes for ancestor queries. In 12^{th} *ACM Symp. on Discrete Algorithms*, pages 547-556, January 2001.

[AGR89] Y. Afek, E. Gafni, and M. Ricklin. Upper and lower bounds for routing schemes in dynamic networks. In 30^{th} *IEEE Symp. on Foundations of Computer Science*, pages 370–375, 1989.

[AK93] S. Aggarwal and S. Kutten. Time-optimal self stabilizing spanning tree algorithms. In 13^{th} *Conference on the Foundations of Software Technology and Theoretical Computer Science*, pages 400–410, Bombay, India, December 1993.

[AGKR01] S. Alstrup, C. Gavoille, H. Kaplan, and T. Rauhe. Identifying nearest common ancestors in a distributed environment. Submitted manuscript. July 2001.

[ABLP89] B. Awerbuch, A. Bar-Noy, N. Linial, and D. Peleg. Compact distributed data structures for adaptive network routing. In 21^{st} *ACM Symp. on Theory of Computing*, pages 230–240, May 1989.

[ABLP90] B. Awerbuch, A. Bar-Noy, N. Linial, and D. Peleg. Improved routing strategies with succinct tables. *Journal of Algorithms*, pages 307–341, 1990.

[AP92] B. Awerbuch and D. Peleg. Routing with polynomial communication-space tradeoff. *SIAM Journal on Discr. Math.*, pages 151–162, 1992.

[BCSW98] S. Basagni, I. Chlamtac, V.R. Syrotiuk and B.A. Woodward. A distance routing effect algorithm for mobility (dream). In *Proc. MOBICOM*, pages 76–84, 1998.

[BS79] R. Bayer and M. Schkolnick. Concurrency of operations on B-trees. *Acta Informatica*, 9:1–21, 1979.

[BHG86] P. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems.* Addison-Wesley, 1986.

[BB87] J. Biswas and J.C. Browne. Simultaneous update of priority structures. In *IEEE Int. Conf. on Parallel Process.*, pages 124–131, 1987.

[Bre66] M.A. Breuer. Coding the vertexes of a graph. *IEEE Trans. on Information Theory*, IT-12:148–153, 1966.

[BF67] M.A. Breuer and J. Folkman. An unexpected result on coding the vertices of a graph. *Journal of Mathematical Analysis and Applications*, 20:583–600, 1967.

[BHV96] H. Buhrman, J.-H. Hoepman, and P. Vitányi. Optimal routing tables. In 15^{th} *ACM Symp. on Principles of Distributed Computing*, pages 134–142, May 1996.

[CGL86] N. Carriero, D. Gelernter, and J. Leichter. Distributed data structures in Linda. In 13^{th} *ACM Symp. on Principles of Prog. Lang*, pages 236–242, 1986.

[CCIR86] J.H. Chang, M.J. Chung, O.H. Ibarra, and K.K. Rao. Systolic tree implementation of data structures. In *IEEE Int. Conf. on Parallel Process.*, pages 669–671, 1986.

[Cow01] L.J. Cowen. Compact routing with minimum stretch. *Journal of Algorithms*, 38:170–183, 2001.

[CG00] L.J. Cowen and C.G. Wagner. Compact roundtrip routing in directed networks. In 15^{th} *ACM Symp. on Principles of Distributed Computing*, pages 51–59, July 2000.

[DS85] W.J. Dally and C.L. Seitz. The balanced cube: a concurrent data structure. Technical Report 5174:TR:85, California Institute of Technology, 1985.

[DS87] F. Dehne and N. Santoro. Optimal VLSI dictionary machines on meshes. In *IEEE Int. Conf. on Parallel Process.*, pages 832–840, 1987.

[DKKP95] S. Dolev, E. Kranakis, D. Krizanc, and D. Peleg. Bubbles: adaptive routing scheme for high-speed dynamic networks. *SIAM Journal on Computing*, 29:804–833, 1999. Preliminary version appeared in STOC 1995.

[EGP98] T. Eilam, C. Gavoille, and D. Peleg. Compact routing schemes with low stretch factor. In 17^{th} *Annual ACM Symp. on Principles of Distributed Computing*, pages 11–20, August 1998.

[EMZ97] T. Eilam, S. Moran, and S. Zaks. The complexity of the characterization of networks supporting shortest-path interval routing. In 4^{th} *International Colloq. on Structural Information & Communication Complexity*, pages 99–111. Carleton Scientific, July 1997.

[Ell80a] C.S. Ellis. Concurrent search and insertion in 2-3 trees. *Acta Informatica*, 14:63–86, 1980.

[Ell80b] C.S. Ellis. Concurrent search and insertion in AVL trees. *IEEE Trans. on Computers*, C-29:811–817, 1980.

[Ell85] C.S. Ellis. Distributed data structures, a case study. *IEEE Trans. on Computers*, C-34:1178–1185, 1985.

[Fla97] M. Flammini. On the hardness of devising interval routing schemes. *Parallel Processing Letters*, 7(1):39–47, March 1997.

[FGS96] M. Flammini, G. Gambosi, and S. Salomone. Interval routing schemes. *Algorithmica*, 16(6):549–568, December 1996.

[FvLS98] M. Flammini, J. van Leeuwen, and A. Marchetti-Spaccamela. The complexity of interval routing on random graphs. *The Computer Journal*, 41(1):16–25, 1998.

[FG94] P. Fraigniaud, and C. Gavoille. A characterization of networks supporting linear interval routing. In 13^{th} *Annual ACM Symp. on Principles of Distributed Computing*, pages 216–224, August 1994.

[FG95] P. Fraigniaud and C. Gavoille. Memory requirement for universal routing schemes. In 14^{th} *ACM Symp. on Principles of Distributed Computing*, pages 223–230, August 1995.

[FG96] P. Fraigniaud and C. Gavoille. Local memory requirement of universal routing schemes. In 8^{th} *ACM Symp. on Parallel Algorithms and Architecture*, pages 183–188, June 1996.

[FG97] P. Fraigniaud and C. Gavoille. Universal routing schemes. *Journal of Distributed Computing*, 10:65–78, 1997.

[FG98] P. Fraigniaud and C. Gavoille. Interval routing schemes. *Algorithmica*, 21:155–182, 1998.

[FG01] P. Fraigniaud and C. Gavoille. Routing in trees. In 28^{th} *Int. Colloq. on Automata, Languages & Prog.*, vol. 2076 of LNCS, pages 757–772, July 2001.

[FWB85] A.J. Frank, L.D. Wittie, and A.J. Bernstein. Maintaining weakly-consistent replicated data on dynamic groups of computers. In *IEEE Int. Conf. on Parallel Process.*, pages 155–162, 1985.

[Fre93] G.N. Frederickson. Searching among intervals and compact routing tables. In Andrzej Lingas, Rolf Karisson, and Svante Carlsson, editors, 20^{th} *Int. Colloq. on Automata, Languages & Prog.*, vol. 700 of LNCS, pages 28–39. Springer-Verlag, July 1993.

[FJ86] G.N. Frederickson and R. Janardan. Separator-Based Strategies for Efficient Message Routing. In 27^{th} *IEEE Symp. on Foundations of Computer Science*, pages 428-437, October 1986.

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

[FJ89] G.N. Frederickson and R. Janardan. Efficient message routing in planar networks. *SIAM Journal on Computing*, 18:843–857, August 1989.

[FJ90] G.N. Frederickson and R. Janardan. Space-efficient message routing in c-decomposable networks. *SIAM Journal on Computing*, pages 164–181, 1990.

[Gav00] C. Gavoille. A survey on interval routing schemes. *Theoretical Computer Science*, 245(2):217–253, 2000.

[Gav01] C. Gavoille. Routing in distributed networks: overview and open problems. *ACM SIGACT News - Distributed Computing Column*, 32(1):36–52, March 2001.

[GG97] C. Gavoille and M. Gengler. Space-efficiency of routing schemes of stretch factor three. In 4^{th} *International Colloq. on Structural Information & Communication Complexity*, pages 162–175. Carleton Scientific, July 1997.

[GG98] C. Gavoille and E. Guévremont. Worst case bounds for shortest path interval routing. *Journal of Algorithms*, 27:1–25, 1998.

[GH99] C. Gavoille, and N. Hanusse. Compact routing tables for graphs of bounded genus. In 26^{th} *International Colloq. on Automata, Languages and Programming*, vol. 1644 of LNCS, pages 351–360, July 1999.

[GKK$^+$01] C. Gavoille, M. Katz, N.A. Katz, C. Paul, and D. Peleg. Approximate distance labeling schemes. In 9^{th} *European Symp. on Algorithms*, August 2001.

[GP01a] C. Gavoille, and C. Paul. Split decomposition and distance labelling: an optimal scheme for distance hereditary graphs. In *Euro. Conf. on Combinatorics, Graph Theory and Applications*, September 2001.

[GP98] C. Gavoille and D. Peleg. The compactness of interval routing for almost all graphs. Research Report RR-1202-98, LaBRI, University of Bordeaux, 351, cours de la Libération, 33405 Talence Cedex, France, April 1998. To appear in *SIAM Journal on Computing.*

[GP99] C. Gavoille and D. Peleg. The compactness of interval routing. *SIAM Journal on Discr. Math.*, 12(4):459–473, October 1999.

[GPPR01] C. Gavoille, D. Peleg, S. Pérennes, and R. Raz. Distance labeling in graphs. In 12^{th} *ACM Symp. on Discrete Algorithms*, pages 210–219, January 2001.

[GP96] C. Gavoille and S. Pérennès. Memory requirement for routing in distributed networks. In 15^{th} *ACM Symp. on Principles of Distributed Computing*, pages 125–133, May 1996.

[GS89] D. Ginat and A.U. Shankar. Decentralized ordering of contending nodes in a distributed system. Unpublished manuscript, 1989.

[HT84] D. Harel and R.E. Tarjan. Fast algorithms for finding nearest common ancestors. *SIAM Journal on Computing*, 13(2):338–355, May 1984.

[HP00] Y. Hassin and D. Peleg. Sparse Communication Networks and Efficient Routing in the Plane. In 19^{th} *ACM Symp. on Principles of Distributed Computing*, pages 41–50, August 2000.

[Her84] M. Herlihy. Replication methods for abstract data types. Technical Report TR-319, MIT, Lab. for Computer Science, May 1984.

[Her87] M. Herlihy. Concurrency versus availability: atomicity mechanisms for replicated data. *ACM Trans. on Comput. Syst.*, 5:249–274, 1987.

[KNR88] S. Kannan, M. Naor, and S. Rudich. Implicit representation of graphs. *SIAM Journal on Discr. Math.*, 5(4):596–603, 1992. Preliminary version appeared in STOC 1988.

[KM01] H. Kaplan and T. Milo. Short and simple labels for small distances and other functions. In *Workshop on Algorithms and Data Structures*, August 2001.

[KM01a] H. Kaplan and T. Milo. Parent and ancestor queries using a compact index. In 20^{th} *ACM Symp. on Principles of Database Systems*, May 2001.

[KKKP01] M. Katz, N.A. Katz, A. Korman and D. Peleg. Labeling schemes for flow and connectivity. Submitted manuscript. July 2001.

[KKP00] M. Katz, N.A. Katz, and D. Peleg. Distance labeling schemes for well-separated graph classes. In 17^{th} *Symp. on Theoretical Aspects of Computer Science*, vol. 1770 of LNCS, pages 516–528, February 2000.

[KK77] L. Kleinrock and F. Kamoun. Hierarchical routing for large networks; performance 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. *Computer Networks*, 10:221–248, 1980.

[KK96] E. Kranakis and D. Krizanc. Lower bounds for compact routing. In 13^{th} *Symp. on Theoretical Aspects of Computer Science*, vol. 1046 of LNCS, pages 529–540, February 1996.

[KKU95] E. Kranakis, D. Krizanc, and J. Urrutia. Compact routing and shortest path information. 2^{nd} *Colloq. on Structural Information & Communication Complexity*, pages 101–112. Carleton University Press, June 1995.

[KSU99] E. Kranakis, H. Singh and J. Urrutia. Compass routing on geometric networks. In *Proc. 11th Canadian Conference on Computational Geometry (CCCG'99)*, pages 51–54, 1999.

[KL80] H.T. Kung and P.L. Lehman. Concurrent manipulation of binary search trees. *ACM Trans. on Programming Lang. and Syst.*, 5:339–353, 1980.

[LEH85] K. A. Lantz, J. L. Edighoffer, and B. L. Histon. Towards a universal directory service. In 4^{th} *ACM Symp. on Principles of Distributed Computing*, pages 261–271, August 1985.

[Lei79] C.E. Leiserson. Systolic priority queues. Technical Report CMU-CS-79-115, Carnegie-Mellon University, 1979.

[LM79] N. Lynch and M. Merritt. Introduction to the theory of nested transactions. In *Int. Conf. on Database Theory*, pages 278–305, Rome, Italy, 1979.

[Man86] U. Manber. On maintaining dynamic information in a concurrent environment. *SIAM Journal on Computing*, 15:1130–1142, November 1986.

[ML84] U. Manber and R.E. Ladner. Concurrency control in a dynamic search structure. *ACM Trans. on Database Syst.*, 9:439–455, 1984.

[MV88] S.J. Mullender and P. Vitányi. Distributed match-making. *Algorithmica*, 3:367–391, 1988.

[NO97] L. Narayanan and J. Opatrny. Compact routing on chordal rings. In 4^{th} *Colloq. on Structural Information & Communication Complexity*, pages 125–137. Carleton Scientific, July 1997.

[NS96] L. Narayanan and S. Shende. Characterizations of networks supporting shortest-path interval labeling schemes. In 3^{rd} *International Colloq. on Structural Information & Communication Complexity*, pages 73–87. Carleton University Press, June 1996.

[OB87] A.R. Omondi and J.D. Brock. Implementing a dictionary on hypercube machines. In *IEEE Int. Conf. on Parallel Process.*, pages 707–709, 1987.

[OD81] D. Oppen and Y.K. Dalal. The clearinghouse: a decentralized agent for locating named objects in a distributed environment. Technical Report OPD-T8103, Xerox Corp., October 1981.

[ORS82] T.A. Ottman, A.L. Rosenberg, and L.J. Stockmeyer. A dictionary machine for VLSI. *IEEE Trans. on Computers*, C-31:892–897, 1982.

[Pap86] C.H. Papadimitriou. *The Theory of Concurrency Control.* Computer Science Press, Rockville, MD, 1986.

[Pel90] D. Peleg. Distributed data structures: A complexity oriented view. In *Proc. 4th Workshop on Distributed Algorithms*, September 1990.

[Pel93] D. Peleg. Distance-dependent distributed directories. *Info. and Computation*, pages 270–298, 1993.

[Pel99] D. Peleg. Proximity-preserving labeling schemes and their applications. In 25^{th} *Int. Workshop on Graph-Theoretic Concepts in Computer Science*, vol. 1665 of LNCS, pages 30–41, June 1999.

[Pel00a] D. Peleg. *Distributed Computing: A Locality-Sensitive Approach.* SIAM, 2000.

[Pel00b] D. Peleg. Informative labeling schemes for graphs. In 25^{th} *Symp. on Mathematical Foundations of Computer Science*, vol. 1893 of LNCS, pages 579–588. Springer-Verlag, August 2000.

[PS89] D. Peleg and A.A. Schäffer. Graph spanners. *Journal of Graph Theory*, 13:99–116, 1989.

[PU87] D. Peleg and E. Upfal. Efficient message passing using succinct routing tables. Research Report RJ5768, IBM, August 1987.

[PU89] D. Peleg and E. Upfal. A tradeoff between size and efficiency for routing tables. *Journal of the ACM*, 36:510–530, 1989.

[Per82] R. Perlman. Hierarchical networks and the subnetwork partition problem. In 5^{th} *Conference on System Sciences*, 1982.

[PK87] S. Pramanik and M.H. Kim. HCB_tree: A B_tree structure for parallel processing. In *IEEE Int. Conf. on Parallel Process.*, pages 140–146, 1987.

[RK88] V.N. Rao and V. Kumar. Concurrent insertions and deletions in a priority queue. In *IEEE Int. Conf. on Parallel Process.*, pages 207–211, 1988.

[Ree78] D.P. Reed. *Naming and Synchronization in a Decentralized Computer System.* PhD thesis, MIT, Dept. of Electrical Engineering, 1978.

[SK85] N. Santoro and R. Khatib. Labelling and implicit routing in networks. *The Computer Journal*, 28:5–8, 1985.

[SV88] B. Schieber, U. and Vishkin. On finding lowest common ancestors: simplification and parallelization. *SIAM Journal on Computing*, 17(6):1253–1262, December 1988.

[SS85] H. Schmeck and H. Schröder. Dictionary machines for different models of VLSI. *IEEE Trans. on Computers*, C-34:472–475, 1985.

[SL87] A.M. Schwartz and M. Loui. Dictionary machines on cube-class networks. *IEEE Trans. on Computers*, C-36:100–105, 1987.

[Sun82] C.A. Sunshine. Addressing problems in multi-network systems. In *IEEE INFOCOM*, pages 12–18, March 1982.

[Tar72] R.E. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[Ter87] D.B. Terry. Cashing hints in distributed systems. *IEEE Trans. on Software Eng.*, SE-13:48–54, 1987.

[TZ01] M. Thorup and U. Zwick. Compact routing schemes. In 13^{th} *ACM Symp. on Parallel Algorithms and Architecture*, pages 1–10, Hersonissos, Crete, Greece, July 2001.

[TZ01a] M. Thorup and U. Zwick. Approximate distance oracles. In 33^{rd} *ACM Symp. on Theory of Computing*, pages 183–192, Hersonissos, Crete, Greece, July 2001.

[vBoas77] P. van Emde Boas. Preserving order in a forest in less than logarithmic time and linear space. *Information Processing Letters*, 6(3):80–82, 1977.

[vLT86] J. van Leeuwen and R.B. Tan. Routing with compact routing tables. In G. Rozenberg and A. Salomaa, editors, *The Book of L*, pages 259–273. Springer-Verlag, New York, New York, 1986.

[vLT87] J. van Leeuwen and R.B. Tan. Interval routing. *The Computer Journal*, 30:298–307, 1987.

[vLT94] J. van Leeuwen and R.B. Tan. Compact routing methods: a survey. 1^{st} *Colloq. on Structural Information & Communication Complexity*, pages 99–110. Carleton University Press, Ottawa, May 1994.

[Wei84] W.E. Weihl. Specification and implementation of atomic data types. Technical Memo MIT/LCS/TM-314, MIT, Lab. for Computer Science, March 1984.