Distrib. Comput. (2003) 16: 111–120
Digital Object Identifier (DOI) 10.1007/s00446-002-0073-5

DISTRIBUTED COMPUTING

Compact and localized distributed data structures

Cyril Gavoille[1], **David Peleg**[2]

[1] LaBRI, Université Bordeaux I, 351, cours de la Libération, 33405 Talence Cedex, France (e-mail: gavoille@labri.fr)
[2] Department of Computer Science and Applied Mathematics, The Weizmann Institute of Science, Rehovot, 76100 Israel (e-mail: peleg@wisdom.weizmann.ac.il)

Received: August 2001 / Accepted: May 2002

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 concentrates mainly on combinatorial and algorithmic techniques, such as adjacency and distance labeling schemes and interval schemes for routing, and covers complexity results on various applications, focusing on compact localized schemes for message routing in communication networks.

Key words: Informative labeling schemes – Distance labeling – Compact routing tables

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. Certain 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 [Re78, 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.

Supported in part by a grant from the Israel Science Foundation.

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. In particular, this topic touches upon several large research areas, such as ordinary data structures and data types, distributed databases, and concurrency control theory (cf. [LM79, H84, Wei84, BHG86, Pap86, H87]). 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, El80, El80b, KL80, ML84, Man86, BB87, PK87, RK88], or the area of designing special purpose VLSI machines for implementing data structures, cf. [Le79, ORS82, DS85, SS85, CCIR86, DS87, OB87, SL87, AK].

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 amounts 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 adjacency and distance labeling schemes, and more generally,

informative localized labeling schemes, and the second deals with the design of compact localized schemes for message routing in communication networks. For more detailed presentation of various aspects of these problems see [Gav00, Gav01,Pel00a].

2 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 following idea. Given a graph, the scheme associates with each of its vertices a special label. These labels are selected in a such way that will allow us, later on, to infer the adjacency of two vertices *directly* from their labels, without using *any* additional information sources. 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 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 an n-vertex tree T, the scheme assigns labels to its vertices as follows. Choose a root and associate a distinct integer $L(v) \in [1,n]$ with each vertex v of T, and then assign a vertex v with parent w the label $\langle L(v), L(w)\rangle$. (For the root, replace $L(w)$ by 0.) Given two labels $\langle L(v), L(w)\rangle$ and $\langle L(v'), L(w')\rangle$, 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 $L(v) = L(w')$ or $L(v') = L(w)$. 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 (namely, graphs whose edge set can be partitioned into k forests, 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 for constant c (namely, graphs having a recursive separator of size at most c). Additional $O(\log n)$ adjacency labeling schemes have been designed in [CV01] for bounded *clique-width* graphs (see [CO02] for more details about these class), a class generalizing and including bounded tree-width graphs [Bod98], c-decomposable graphs for constant c, and other families like hereditary graphs (namely, graphs such that every induced path is a shortest path, cf. [BLS99]).

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 for monotone function $r(n)$. 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 $R(n) \le r(n)\log n$, and $R(n) = O(r(n))$ if $r(n) \ge n^\epsilon$ for constant $\epsilon > 0$. For bounded tree-width graphs, including trees, outerplanar graphs, series-parallel graphs, k-outerplanar graphs (defined for $k = 1$ as outerplanar graphs and for $k > 1$ as planar graphs in which removing the outer face yields a $(k-1)$-outerplanar graph) and c-decomposable graphs for constant k and c, $R(n) = O(\log n)$ since $r(n) = O(1)$. 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. 1). Each vertex 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 $|S| \le r(n)$ 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 vertices of u in T (the vertex 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.

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

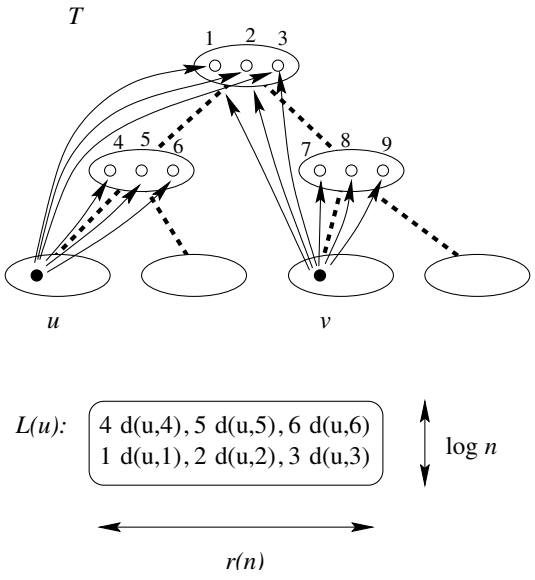

Fig. 1. The separator technique for distance labeling

an $r(n)$-separator [GPPR01]. However, for the class of planar graphs (which is a proper subclass of the class of graphs with $O(\sqrt{n}\,)$-separators) there is a specific lower bound of $\Omega(n^{1/3})$ on the label size, leaving an intriguing (polynomial) gap. More recently, schemes with $O(\log^2 n)$ bit labels that do not make use of the "small 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 $d(u, v) \leq \tilde{d}(u, v) \leq s \cdot d(u, v) + r$. In particular, (exact) 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 [GKKPP01].

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 $r(n)$-separators, support a $(1 + 1/\log n, 0)$-approximate scheme with $O(R(n) \cdot \log\log n)$ bit labels [GKKPP01]. 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 [GKKPP01] for any $(1 + 1/\log n, 0)$-approximate distance labeling scheme on the class of trees. However, distances in a tree between vertices at distance at most d can be computed with labels of $\log n + O(d\sqrt{\log n}\,)$ bits [KM01]. A similar phenomenon between the label length and the quality of the estimators holds for planar graphs. [Tho01] has presented for planar digraphs a $(1 + \epsilon, 0)$-approximate scheme with $O(\frac{1}{\epsilon}\log^2 n)$ bit labels, for every $\epsilon > 0$, to be contrasted with the fact that for $\epsilon = 0$ labels must have $\Omega(n^{1/3})$ bits [GPPR01]. Additional results appear in [CHKZ02,Tho01].

A number of additional approximate distance labeling schemes are presented in [GKKPP01], including a $(1, 2)$-approximate scheme with $O(\log n)$ bit labels for permutation graphs (namely, graphs constructed from a permutation σ on the vertices such that i and j are adjacent iff $i < j$ and $\sigma(i) > \sigma(j)$) and AT-free graphs (namely, graphs without any triple of pairwise non-adjacent vertices such that there is a path connecting any two of them that avoids the neighborhood of the third), and a $(1, \lfloor c/2 \rfloor)$-approximate scheme with $O(\log^2 n)$ bit labels for c-chordal graphs (namely, graphs whose longest induced cycle is no greater than c). In particular, it yields a $(1, 1)$- approximate labeling scheme for chordal (i.e., 3-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 label size complexity of distance labeling schemes for interval and permutation graphs is left open, with the bounds ranging from $\Omega(\log n)$ to $O(\log^2 n)$ [KKP00].

Another 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 [GKKPP01] (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.

Finally, an interesting direction involves localized labeling schemes for the *dynamic distributed* setting, namely, distributed online schemes on dynamically changing networks. Such schemes are clearly necessary for handling applications such as distributed systems and communication networks. A first step in this direction is made in [KPR02], which studies labeling schemes on dynamic tree networks.

3 Informative localized labeling schemes

Adjacency and distance labeling schemes motivate 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: associate with each vertex u an interval $I(u) = [i, i+w]$ where $i \in [1, n]$ is a label obtained by traversing the n-vertex tree in a depth-first search fashion [Tar72], and w is the number of descendants of u. It is easy to verify that u is an ancestor of v iff $I(v) \subseteq I(u)$. It turns out that ancestor queries can be handled by a more compact labeling scheme, using $\log n + O(\sqrt{\log n})$ bit labels, independently discovered by [AR02, KM01a, TZ01]. (Actually, [TZ01] proposed a $\log n + O(\log n/\log\log n)$ bit labeling scheme but answering more general routing queries, see Sect. 4). 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. A comparative study of ancestor labeling schemes appears in [KMS02]. Additional results on labeling schemes (like reachability in planar digraphs) appear in [CHKZ02, Tho01].

Another example for a piece of non-numeric information that may be required in rooted trees is the *least common ancestor* of v and w. Standard solutions [HT84, SV88] can answer such queries in constant time with a 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 least 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 [KKKP02]. In a weighted undirected graph, where the weights represent edge capacities, a flow labeling scheme should provide, given the two labels $L(u)$ and $L(w)$, the maximum flow that can be pushed from u to w. Similarly, a k-connectivity labeling scheme should tell us, given the two labels $L(u)$ and $L(w)$, whether or not u and w are k-connected, namely, there are k disjoint paths connecting them. An (asymptotically optimal) flow labeling scheme using $O(\log^2 n + \log n \cdot \log\omega)$ bit labels is presented therein for general n-vertex graphs with maximum (integral) edge 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 when 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 class of n-vertex graphs 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].

Actually, labeling schemes for answering a graph property $P(v_1, \ldots, v_k)$ defined on k vertices can be constructed for the class of bounded clique-width graphs. More precisely, [CV01] have proposed a $O(\log n)$ bit labeling scheme for every fixed graph property P expressible in *monadic second-order logic*, that is, first-order logic over the vertices augmented with variables denoting subsets of vertices and with membership atomic formulas (adjacency and reachability belong to this class). Similarly, a labeling scheme for every *monadic second-order optimization function* P, that is, a function that returns the minimum or maximum cardinality of a set, can be constructed for the class of bounded clique-width graphs with $O(\log^2 n)$ bit labels [CV01]. (For instance, the unweighted distance between two vertices can be expressed as a minimum cardinality set of edges connecting them.)

4 Routing and other applications

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. The efficiency of a routing scheme can be measured in terms of its *stretch*, namely, the maximum ratio between the length of a route produced by the scheme for some pair of processors, and their distance. A straightforward approach for achieving the goal of guaranteeing optimal routes 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

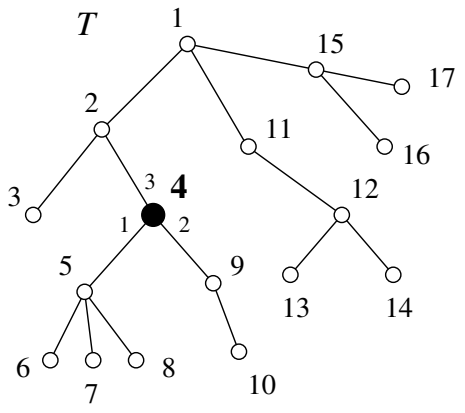

Fig. 2. An interval routing for T, and the data structure for the vertex 4

some shortest path from v to u. However, this approach may be too expensive for large systems since it requires a total of $O(n^2 \log d)$ memory bits in an n-processor network with maximum degree d. 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.

This problem can be approached via localized techniques based on labeling schemes. 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 in such a way that at every source vertex v and given the address of any destination vertex u, one can decide the output port of an outgoing edge of v that leads to u. The decision must be taken locally in v, based solely on the two labels of v and with the address label of u.

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 a depth-first search traversal of the n-vertex tree T, associating with each vertex v an address label $L(v) \in [1, n]$, see Fig. 2. 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 $2 \log n$ bits per set. Overall, the local routing table of v is a data structure of the form

$$\langle I(v, u_1) : p_1, I(v, u_2) : p_2, \ldots, I(v, u_d) : p_d \rangle$$

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. 2 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 $O(n \log d)$ bits bound for a standard routing table. 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 $L(w) \in I(v, u_i)$ (implying that the message has to include the destination label $L(w)$ in its header), and then forwarding the message through the output port p_i to the neighbor u_i. This local computation is then repeated at intermediate vertices 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 (other labeling schemes, discussed later, aim at overcoming the problem of large degree vertices), but the total memory requirement over the entire tree network is only $O(n \log n)$ bits.

The interval routing scheme for trees is due to [SK]. 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,EMZ97b]. 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 [FG98,NS96,Fla97,EMZ97a], or to minimize the length of the longest route [EMZ99,1,KRS00]. 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 [KKR96,FvLS98,NN98,GG98,GP01]. It is worth noting that an interval routing scheme, once computed for a graph, can be used to perform other tasks than routing. This idea was investigated by [vLT87] and then by [TNP98], which proposed an approach for efficiently performing *broadcast* on graphs supporting an interval routing scheme. Subsequently, [FGM01] have proposed for graphs supporting a shortest path interval routing scheme with one interval per edge a $\Theta(n)$ messages broadcast algorithm that uses only interval routing labels. A vast array of other routing scheme types has been developed, including prefix routing schemes [BLT90], boolean routing schemes [FG97a], multi-dimensional interval routing schemes [FGNT98,RS00] and more. For surveys of the many developments in this area see [vLT94f,Gav00].

The problem of efficiency-memory tradeoffs for routing schemes was first raised in [KK1], 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, the method of [KK1] is based on making some fairly strong assumptions regarding the existence of a certain partition of the network, and moreover, the method does not provide an algorithm for computing such a partition if it exists. Several variations and/or improvements were studied later, cf. [KK2,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 [SK], unit-cost rings, complete networks and grids [vLT86,vLT87], and outerplanar networks [FJ88]. The problem of designing memory-efficient near-optimal routing schemes was cast in a theoretical formulation in [FJ86, FJ88,FJ89], which also gave precise solutions for various graph classes up to and including planar graphs. Near-optimal stretched routing schemes were constructed in [FJ89,**?**] for c-decomposable networks and for planar networks. The schemes for c-decomposable networks guarantee a 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 $O(c^2 \log c \cdot n \log^2 n)$. A crucial step in constructing these routing schemes is assigning names to the vertices as part of the routing scheme, namely $O(c \log c \cdot \log n)$ bit labels. (For stretch factor 3 a simpler scheme was proposed in [FJ90] that uses $O(\log n)$ bit names.) Although planar networks can be considered as c-decomposable networks for $c = \Theta(\sqrt{n})$, better schemes can be achieved. A first scheme in [FJ89] has a total memory requirement $O(n^{4/3} \log n)$, stretch factor 3, and uses $O(\log n)$ bit names. The second one achieves for every constant $d > 3$, a total memory requirement $O(dn^{1+1/d} \log n)$, stretch factor 7, and uses $O(d \log n)$ bit names. Recently, [GH99] have constructed new routing schemes for planar graphs with optimal stretch 1. Based on book embedding, these schemes use label names $\in [1, n]$. (A *k-page embedding* of a graph consists of a linear ordering of its nodes drawn on a line and of a partition of its edges into k "pages" so that edges residing on the same page do not intersect; cf. [B92].) 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 coordinates of the vertices and *compass routing* methods (cf. [BCSW98, BM99, KSU99, BM01]) 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 $O\left(\kappa^3 n^{1+1/\kappa} \log n\right)$ bits of routing information in the network. Similar to [KK1], these schemes are also based on suitable partitions of the network, but here the partitions are shown to exist and an algorithm is given for computing them. Just as for the IRS approach and the routing schemes of [FJ89, FJ90, GH99], the schemes of [PU89] require assigning suitable names to the vertices. These names are of size $O(\log^2 n)$ bit. However, 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, AP92]. 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 factor (mainly integral stretch factors ≤ 5) are presented in [KKU95, NO97, EGP98, CW00, Cow01, TZ01], and also in [TZ01] for higher stretches. The currently best memory-stretch tradeoff for arbitrary weighted graphs is achieved by a scheme in which the stretch factor is $O(\kappa)$ and the memory requirements are $O(n^{1/\kappa} \log^{O(1)} n)$ bits per vertex [TZ01a].

Lower bounds for the space-efficiency tradeoff of routing schemes were studied in [PU89, GP96, KK96, FG97b, EGP98, BHV99, GG01]. 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 for $\kappa \geq 5$, no routing strategy can guarantee for every graph a routing scheme with a stretch factor $O(\kappa)$ (specifically $\kappa/2 - 2$) 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 all d and fixed $\epsilon < 1$ such that $3 \leq d \leq \epsilon n$, there exists a worst-case graph of degree bounded by d on which the total memory requirement is $\Omega(n^2 \log d)$, matching 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] are name-independent (i.e., the original name of the destination vertex is used for the routing, and cannot be relabeled) 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 $O(\kappa\, n^{1/\kappa} \log n)$ bits of memory per vertex and guarantee a stretch of $O(\kappa^2\, 9^\kappa)$. The tradeoff was 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 $O(\kappa\, n^{1/\kappa} \log^2 n \cdot \log D)$ bits per vertex, where D is the weighted diameter of the network. The scheme of [TZ01a] has better memory-stretch tradeoff but it does not enjoy name-independence (the scheme needs $O(\kappa \log^2 n / \log \log n)$ bit names). On the other hand, it is worth noting that the schemes of [ABLP89] and of [TZ01] have one additional advantage over [AP92], namely, their memory complexity is independent of the range of the edge costs, or the network diameter (or put another way, the routing algorithm is "purely combinatorial").

All the schemes presented in [ABLP89, AP92, EGP98, Cow01, TZ01] have a cluster-based representation as a common feature. Inside each cluster, the routing is performed along some spanning trees. Hence the basic problem of efficient routing in n-vertex trees is of major significance when building efficient routing schemes in general networks. The basic IRS scheme of [SK] for trees presented at the beginning of the section can be improved in several respects. For instance, by extending the address range from $\log n$ to $3 \log n$ bits, [Cow01] has shown that $O(\sqrt{n} \log n)$ bits suffice for the local routing table, an improvement for high degree trees. In [EGP98] it is shown that every routing scheme that selects addresses in the range $[1, n]$ must use an $\Omega(\sqrt{n})$ bit local routing table for some 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 *memory-free routing*, or using the terminology of the previous sections, *routing labeling schemes*. Obviously such a routing labeling scheme 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, if output port numbers can be permuted in advance, 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 c can be reduced to $c = 1 + O(1/\log\log n)$. If the output port number cannot be permuted (as required in a general graph where several tree routing schemes overlap), then trees have routing labeling schemes with $O(\log^2 n/\log\log n)$ bit labels [FG01, TZ01], and this bound was shown to be tight [FG02]. This emphasizes that a small variation on the size of the addresses has a significant effect on the size of the routing table.

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). The routing problem was also dealt with in the context of the new generation of ATM and optical networks [DKKP95]. See [Gav01,Pel00a] for more detailed overviews.

At this stage, let us discuss some other 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 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 possibly 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, say, in the context of Euclidean graphs (cf. [RU99]), and combinatorial optimization in general, by viewing a vertex labeling as a "nice," i.e., easily manageable, representation of the graph.

Let us conclude with a brief discussion of future prospects. As observed in 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 the 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, T. Milo. Compact labeling schemes for ancestor queries. In *Proc.* 12^{th} *ACM Symp. on Discrete Algorithms*, 2001 pp. 547–556

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

[AK] S. Aggarwal, S. Kutten. Time-optimal self stabilizing spanning tree algorithms. In *Proc.* 13^{th} *Conf. on the Foundations of Software Technology and Theoretical Computer Science*, 1993 pp. 400–410

[AGKR01] S. Alstrup, C. Gavoille, H. Kaplan, T. Rauhe. Nearest Common Ancestors: A Survey and a new Distributed Algorithm. In 14^{th} *Annual ACM Symposium on Parallel Algorithms and Architecture (SPAA)*, ACM Press, 2002, pp. 258–264

[AR02] S. Alstrup, T. Rauhe. Improved Labeling Scheme for Ancestor Queries. In *Proc.* 13^{th} *Symposium on Discrete Algorithms (SODA)*, ACM-SIAM, 2002 pp. 947–953

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

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

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

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

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

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

[BLS99] A. Brandstädt, V.B. Le, J.P. Spinrad. *Graph Classes – A survey*. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, 1999

[BLT90] E.M. Bakker, J. van Leeuwen, R.B. Tan. Prefix routing schemes in dynamics networks. Technical Report RUU-CS-90-10, Utrecht University, Mar. 1990

[B92] T. Bilski. Embedding graphs in books: a survey. *IEEE Proceedings* 139: 134–138, 1992

[BM99] P. Bose, P. Morin. Online routing in triangulations. In *Proc.* 10^{th} *Int. Symp. on Algorithms and Computation*, pp. 113–122, SV LNCS 1741, 1999

[BM01] P. Bose, P. Morin. Competitive online routing in geometric graphs. In *Proc.* 8^{th} *Colloq. on Structural Information & Communication Complexity*, pp. 35–44, Carleton University Press, Jun. 2001

[Bod98] H.L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science* 209(1-2): 1–45, 1998

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

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

[BHV99] H. Buhrman, J.-H. Hoepman, P. Vitányi. Space-efficient routing tables for almost all networks and the incompressibility method. *SIAM Journal on Computing*, 28(4): 1414–1432, 1999

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

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

[CHKZ02] E. Cohen, E. Halperin, H. Kaplan, U. Zwick. Reachability and Distance Queries via 2-hop Labels. In *Proc.* 19^{th} *ACM-SIAM Symp. on Discrete Algorithms*, Jan. 2002

[CO02] B. Courcelle, S. Olariu. Upper bounds to the clique-width of graphs. *Discrete Applied Mathematics* 101: 77–114, 2000

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

[CW00] L.J. Cowen, C.G. Wagner. Compact roundtrip routing in directed networks. In *Proc.* 19^{th} *ACM Symp. on Principles of Distributed Computing*, 2000 pp. 51–59

[CV01] B. Courcelle, R. Vanicat. Query efficient implementation of graphs of bounded clique width. *Discrete Applied Mathematics*, to appear

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

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

[DKKP95] S. Dolev, E. Kranakis, D. Krizanc, 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, D. Peleg. Compact routing schemes with low stretch factor. In *Proc.* 17^{th} *Annual ACM Symp. on Principles of Distributed Computing*, 1998 pp. 11–20

[EMZ97a] T. Eilam, S. Moran, S. Zaks. The complexity of the characterization of networks supporting shortest-path interval routing. In *Proc.* 4^{th} *Int. Colloq. on Structural Information & Communication Complexity*, pages 99–111. Carleton Scientific 1997

[EMZ97b] T. Eilam, S. Moran, S. Zaks. A simple DFS-based algorithm for linear interval routing. In *Proc.* 11^{th} *Int. Workshop on Distributed Algorithms*, vol. 1320 of LNCS, pp. 37–51, Berlin Heidelberg New York: Springer 1997

[EMZ99] T. Eilam, S. Moran, S. Zaks. Lower bounds for linear interval routing. *Networks* 34(1): 37–46, 1999

[El80] C.S. Ellis. Concurrent search and insertion in 2-3 trees. *Acta Informatica* 14: 63–86, 1980

[El80b] C.S. Ellis. Concurrent search and insertion in AVL trees. *IEEE Trans. on Computers* C-29: 811–817, 1980

[El85] 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, 1997

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

[FG97a] M. Flammini, G. Gambosi. On devising boolean routing schemes. *Theoretical Computer Science* 186: 171–198, 1997

[FGNT98] M. Flammini, G. Gambosi, U. Nanni, R.B. Tan. Multidimensional interval routing schemes. *Theoretical Computer Science* 205: 115–133, 1998

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

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

[FG97b] P. Fraigniaud, C. Gavoille. Universal routing schemes. *Journal of Distributed Computing* 10: 65–78, 1997

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

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

[FG02] P. Fraigniaud, C. Gavoille. A space lower bound for routing in trees. In *Proc.* 19^{th} *Symp. on Theoretical Aspects of Computer Science*, Mar. 2002

[FGM01] P. Fraigniaud, C. Gavoille, B. Mans. Interval routing schemes allow broadcasting with linear message-complexity. *Distributed Computing* 14: 217–229, 2001

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

[Fre93] G.N. Frederickson. Searching among intervals and compact routing tables. In *Proc.* 20^{th} *Int. Colloq. on Automata, Languages & Prog.*, vol. 700 of LNCS, pp. 28–39. Berlin Heidelberg New York: Springer 1993

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

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

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

[FJ90] G.N. Frederickson, R. Janardan. Space-efficient message routing in c-decomposable networks. *SIAM Journal on Computing* 19: 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, 2001

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

[GG01] C. Gavoille, M. Gengler. Space-efficiency of routing schemes of stretch factor three. *Journal of Parallel and Distributed Computing* 61: 679–687, 2001

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

[GKKPP01] C. Gavoille, M. Katz, N.A. Katz, C. Paul, D. Peleg. Approximate distance labeling schemes. In *Proc.* 9^{th} *European Symp. on Algorithms*, Aug. 2001

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

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

[GP01] C. Gavoille, D. Peleg. The compactness of interval routing for almost all graphs. *SIAM Journal on Computing* 31(3): 706–721, 2001

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

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

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

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

[H84] M. Herlihy. Replication methods for abstract data types. Technical Report TR-319, MIT, Lab. for Computer Science, 1984

[H87] M. Herlihy. Concurrency versus availability: atomicity mechanisms for replicated data. *ACM Trans. on Comput. Syst.* 5: 249–274, 1987

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

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

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

[KMS02] H. Kaplan, T. Milom, R. Shabo. A Comparison of Labeling Schemes for Ancestor Queries. In *Proc.* 19^{th} *ACM-SIAM Symp. on Discrete Algorithms*, Jan. 2002

[KKKP02] M. Katz, N.A. Katz, A. Korman, D. Peleg. Labeling schemes for flow and connectivity. In *Proc.* 19^{th} *ACM-SIAM Symp. on Discrete Algorithms*, Jan. 2002

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

[KK1] L. Kleinrock, F. Kamoun. Hierarchical routing for large networks; performance evaluation and optimization. *Computer Networks* 1: 155–174, 1977

[KK2] L. Kleinrock, F. Kamoun. Optimal clustering structures for hierarchical topological design of large computer networks. *Computer Networks* 10: 221–248, 1980

[KPR02] A. Korman, D. Peleg, Y. Rodeh. Labeling Schemes for Dynamic Tree Networks. In *Proc.* 19^{th} *Symp. on Theoretical Aspects of Computer Science*, Mar. 2002

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

[KKR96] E. Kranakis, D. Krizanc, S.S. Ravi. On multi-label linear interval routing schemes. *The Computer Journal* 39: 133–139, 1996

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

[KRS00] R. Kráľovič, P. Ružička, D. Štefankovič. The complexity of shortest path and dilation bounded interval routing. *Theoretical Computer Science* 234(1-2): 85–107, 2000

[KSU99] E. Kranakis, H. Singh, J. Urrutia. Compass routing on geometric networks. In *Proc.* 11^{th} *Canadian Conf. on Computational Geometry*, 1999 pp. 51–54

[KL80] H.T. Kung, 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, B. L. Histon. Towards a universal directory service. In *Proc.* 4^{th} *ACM Symp. on Principles of Distributed Computing*, 1985 pp. 261–271

[Le79] C.E. Leiserson. Systolic priority queues. Technical Report CMU-CS-79-115, Carnegie-Mellon University, 1979

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

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

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

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

[NN98] L. Narayanan, N. Nishimura. Interval routing on k-trees. *Journal of Algorithms* 26(2): 325–369, 1998

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

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

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

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

[ORS82] T.A. Ottman, A.L. Rosenberg, 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*. Rockville, MD: Computer Science Press 1986

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

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

[Pel00b] D. Peleg. Informative labeling schemes for graphs. In *Proc.* 25^{th} *Symp. on Mathematical Foundations of Computer Science*, vol. 1893 of LNCS, pp. 579–588. Berlin Heidelerg New York: Springer 2000

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

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

[PU89] D. Peleg, 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 *Proc.* 5^{th} *Conf. on System Sciences*, 1982

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

[RU99] S. Rajsbaum, J. Urrutia. Some problems in distributed computational geometry. In *Proc.* 6^{th} *Int. Colloq. on Structural Information & Communication Complexity*, Carleton University Press 1999 pp. 233–248

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

[Re78] D.P. Reed. *Naming and Synchronization in a Decentralized Computer System*. PhD thesis, MIT, Dept. of Electrical Engineering, 1978

[RS00] P. Ružička, D. Štefankovič. On the complexity of multi-dimensional interval routing schemes. *Theoretical Computer Science* 245(2): 255–280, 2000

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

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

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

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

[Sun82] C.A. Sunshine. Addressing problems in multinetwork systems. In *Proc. IEEE INFOCOM*, 1982 pp. 12–18

[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

[TNP98] P. de la Torre, L. Narayanan, D. Peleg. Thy neighbor's interval is greener: A proposal for exploiting interval routing schemes. In *Proc.* 5^{th} *Int. Colloq. on Structural Information & Communication Complexity*, Carleton Scientific, Jun. 1998 pp. 214–228

[Tho01] M. Thorup. Compact oracles for reachability and approximate distances in planar digraphs. In *Proc.* 42^{nd} *IEEE Symp. on Foundations of Computer Science*, Oct. 2001

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

[TZ01a] M. Thorup, U. Zwick. Approximate distance oracles. In *Proc.* 33^{rd} *ACM Symp. on Theory of Computing*, 2001 pp. 183–192

1. [TL97] S.S.H. Tse, F.C.M. Lau. A lower bound for interval routing in general networks. *Networks* 29(1): 49–53, 1997

[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, R.B. Tan. Routing with compact routing tables. In: G. Rozenberg, A. Salomaa (eds.) *The Book of L*, pages 259–273. New York: Springer 1986

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

[vLT94f] J. van Leeuwen, R.B. Tan. Compact routing methods: a survey. In *Proc.* 1^{st} *Int. Colloq. on Structural Information & Communication Complexity*, pp. 99–110. Carleton University Press 1994

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