Multi-dimensional Interval Routing Schemes

by

Yashar Ganjali

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2001

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be electronically available to public.

Abstract

Routing messages between pairs of nodes is one of the most fundamental tasks in any distributed computing system. An Interval Routing Scheme (IRS) is a well-known, space-efficient routing strategy for routing messages in a network. In this scheme, each node of the network is assigned an integer label and each link at each node is labeled with an interval. The interval assigned to a link l at a node v indicates the set of destination addresses of the messages which should be forwarded through l at v. When studying interval routing schemes, there are two main problems to be considered: a) Which classes of networks do support a specific routing scheme? b) Assuming that a given network supports IRS, how good are the paths traversed by messages? The first problem is known as the characterization problem and has been studied for several types of IRS. In this thesis, we study the characterization problem for various schemes in which the labels assigned to the vertices are d-ary integer tuples (d-dimensional IRS) and the label assigned to each link of the network is a list of d 1-dimensional intervals. This is known as Multi-dimensional IRS (MIRS) and is an extension of the the original IRS. We completely characterize the class of network which support MIRS for linear (which has no cyclic intervals) and strict (which has no intervals assigned to a link at a node v containing the label of v) MIRS. In real networks usually the costs of links may vary over time (dynamic cost links). We also give a complete characterization for the class of networks which support a certain type of MIRS which routes all messages on shortest paths in a network with dynamic cost links. The main criterion used to measure the quality of routing (the second problem) is the length of routing paths. In this thesis we also investigate this problem for MIRS and prove two lower bounds on the length of the longest routing path. These are the only known general results for MIRS. Finally, we study the relationship between various types of MIRS and the problem of drawing a hypergraph. Using some of our results we prove a tight bound on the number of dimensions of the space needed to draw a hypergraph.

Key words: Computer networks, interval routing schemes, graph theory, multi-dimensional, characterization, shortest path, dynamic, bounds.

Acknowledgements

First and foremost I would like to express my sincere gratitude to my supervisor, Prof. Naomi Nishimura, for all her support, encouragement and also for being an enthusiastic and genius supervisor. I cannot thank her adequately. I really enjoyed doing research in the past two years, mostly because of her thoughtful supervision.

I also wish to express how grateful I am to my friends in Iran which supported me during this research, specially Mehdi Niamanesh, Afshin Khezerloo, Behzad Mesgarzadeh, Ali Zafari, Hamid Hooshiari-far, Okhtai Azarmanesh, Foad Dabiri, and many others. I have been far away from them but their kind e-mails kept me in touch with the life I used to have. I must also express my gratitude to my friends here in Canada and U.S., namely, Afshin Mantegh, Saeed Behzadipoor, Mohammad-Taghi Hajiaghayi, Mohammad Mahdian, Salem Derisavi, Amir-Hossein Nejad-malayeri, Peyman Servati and specially Kamran Sartipi, Zarrin Langari, and their families. Just being with these kind people made me feel like home, here in Canada.

Many thanks to my English tutor, Holly B. Gauthier, for being such a great teacher and also for all her help and encouragement during the times when I really needed them.

It is a great pleasure to acknowledge the work of the readers of this thesis, Prof. Ian Munro, and Prof. Therese Biedl. This work has benefited from their thoughtful comments and suggestions. I also would like to thank the staff and faculty members of the Department of Computer Science, University of Waterloo. Doing research in such a friendly and nice environment seemed to be very easy and enjoyable.

Last but definitely not least, I would like to thank my dear parents and my marvelous brother, Afshar. I cannot even express how special they are. Even though we have been living far away for the last few years, they have always been the first and greatest source of love and inspiration in my life. This thesis is dedicated to them.

Contents

List of Figures

Chapter 1

Introduction

One of the most fundamental tasks in any network of parallel or distributed systems is routing messages between pairs of nodes [Tan95, Tan96]. When sending a message from a source to a destination, a decision has to be made as to which neighbor (*i.e.* through which incident link) the message must be sent. The *routing problem* is the problem of choosing such a neighbor.

A *routing scheme* is a strategy that determines which path a message, originating from a known source and going to a known destination, should take in the network. Routing schemes can be classified into explicit and implicit ones [SK85]. In *explicit* routing schemes each node of the network has an arbitrary label (name) and some detailed routing information for all destinations is maintained at each node of the network. The classic method for routing messages in a network is to store a *routing table* at each node of the network; this is an explicit routing scheme. A routing table has one entry for each destination node that indicates which outgoing link should be used to forward a message going to that destination (Figure 1.1).

In *implicit* routing schemes, no detailed information is maintained; instead, labels are assigned according to a scheme so that the information implicit in the labeling can be used to choose the neighbor to which a message should be sent. It is usually easy to develop implicit routing schemes when the network has a regular topology. For example, if the underlying graph of the network is ring, we can easily label the nodes of the network clockwise with consecutive integers. The node labeled i will send a message with destination address j clockwise if and only if $[j-i] (\mathrm{mod}\ n) < [i-j] (\mathrm{mod}\ n)$, where

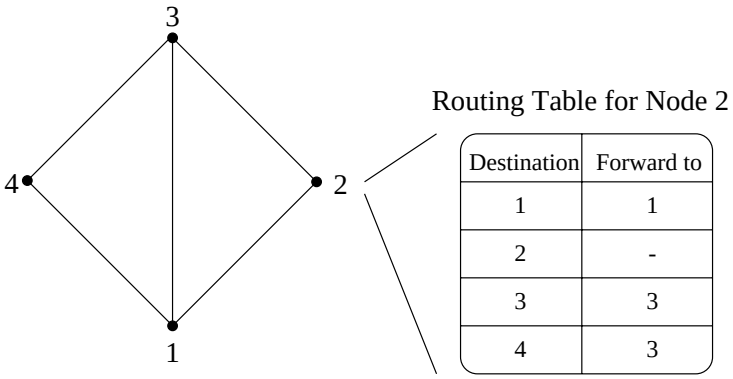

Figure 1.1: A routing table.

n is the number of nodes in the network. Clearly, this routing scheme always routes messages on the shortest path around the ring.

An explicit routing scheme requires $\Theta(n)$ space at each node of an n-node network, which is not efficient (or maybe even feasible) for large networks of computers. In other words, using explicit routing schemes we cannot *scale* the communication network because the amount of space available at each node of the network is limited.

Techniques to decrease the amount of space needed at each node of the network have been studied intensively [FJ86, ABNLP90, FGS93]. The general idea is to group the destination addresses that correspond to the same outgoing link (at a node), and to encode the group so that it is easy to verify if a given destination address is in the group or not. In these routing schemes routing information is succinctly stored at each node of the network in a *preprocessing* phase. Later, when a node needs to route a message, it uses this preprocessed information to determine the link through which the message should be forwarded. These routing schemes are called *compact routing schemes* in general and usually are dynamically adjustable with the expansion of the network.

In an *interval labeling scheme*, which was originally introduced by Santoro and Khatib [SK85], each node of the network is assigned an integer label and integer intervals are used to group destination addresses. Each link of the network at each node is assigned an interval which encodes the information to route the messages in the network [vLT87].

Routing messages is completed in a distributed way. At each intermediate node v, if the label of the node equals the destination address, *dest*, the routing process ends.

Otherwise, the message is forwarded through a link labeled by an interval I, such that $dest \in I$.

An interval labeling scheme is said to be *valid* if for any pair of nodes s and t, a message originating from s eventually reaches t. A valid interval labeling scheme is also called an *Interval Routing Scheme (IRS)*. Clearly, this method requires $O(l)$ space at each node of the network (where l is the number of links at the node), which is an efficient allocation of memory compared to explicit routing schemes. Throughout this thesis, we consider only valid interval labeling schemes.

It has been shown that an IRS can route messages on shortest paths on particular network topologies, such as trees, rings, hyper-cubes, and others [SK85, vLT87, FJ89]. Unfortunately, this is not true in general networks; there are classes of networks which do not have any IRS which routes messages on shortest paths. Many schemes have been introduced in order to overcome this problem and to expand the classes of networks which support IRS with the desired properties [BvLT91, FGNT98]. In designing such schemes, the aim is to keep the memory efficiency property and to gain other properties (*e.g.* routing on shortest paths, bounds on the length of the routing paths, and so on).

A very interesting IRS is a *Multi-dimensional Interval Routing Scheme (MIRS)* in which the labels assigned to the nodes are elements from $\mathbb{N}^d$ (in the d-dimensional case) and each link is labeled with a d-tuple $([a_1, b_1], [a_2, b_2], ..., [a_d, b_d])$ of intervals, $a_i, b_i \in \mathbb{N}$, for $1 \leqslant i \leqslant d$ [FGNT98]. Messages are routed in a manner similar to that in IRS.

The only known results about MIRS are for specific classes of networks like hypercubes, grids, tori or for the one-dimensional case. In this thesis, we investigate different aspects of MIRS. More precisely, we characterize the class of networks which support various types of MIRS. We also prove some lower and upper bounds on the length of routing paths using any MIRS. These bounds can be used as a criteria for comparing this routing scheme with other known routing schemes.

1.1 Overview

In this thesis we investigate different characteristics of MIRS. Chapter 2 is devoted to providing the concepts and definitions that the reader will require. In this chapter,

we introduce some graph theoretic definitions and concepts and also give formal definitions of routing schemes, IRS, and some variants of IRS. We conclude this chapter by reviewing some known results about IRS and briefly sketching the results in the rest of this thesis. In Chapter 3 we focus on the problem of characterizing the class of networks which support MIRS. We give complete characterization of the networks which support three variants of MIRS and show that in all these cases, increasing the number of dimensions causes the class of networks supporting that specific variant of MIRS to be strictly expanded.

Assuming that a given network supports a specific variant of MIRS, the most important question is the quality of routing. We may have different criteria to evaluate a routing scheme such as the length of the routing paths, the time needed to route messages in the intermediate nodes, and so on. In Chapter 4 we consider the quality of routing problem and give some lower and upper bounds on the length of routing paths in some variants of MIRS.

There is a strong relationship between the problem of finding a MIRS for a given network and the problem of drawing hypergraphs in multi-dimensional spaces. This relationship is studied in Chapter 5. We use some of the results from the previous sections to prove a tight bound on the number of dimensions of the space needed to draw a hypergraph. Finally, we conclude this thesis in Chapter 6 which also contains a list of open problems and some directions for future research.

Chapter 2

Preliminaries

2.1 Graph theoretic preliminaries

In this section we introduce graph theoretic definitions and mention several known results which will be used later. Throughout this thesis, a network is modeled by a graph $G = (V, E)$. The set V of vertices of the graph represents nodes in the network and the set E of edges represents the links between nodes in the network. For basic graph theoretic definitions the reader is referred to standard texts [BM76, Wes96].

In any communication network each link connects two distinct nodes of the network and there is usually at most one link connecting a pair of nodes. In graph theory, if both endpoints of an edge e are the same the edge is said to be a *self-loop*. If there is at most one edge between each pair of vertices in the graph, the graph is *simple*. The underlying graph of any network in this thesis is assumed to be simple and without any self-loops. If the edges of a graph are unordered pairs of vertices the graph is *undirected*. If the underlying graph of a network is undirected the nodes incident to a link of the network can exchange messages in both directions. We usually deal with networks in which any node of the network can send (receive) a message to (from) any other node in the network. If there is a path connecting each pair of vertices in the graph G, G is *connected* and so is any network for which G is an underlying graph.

A vertex v of the graph G is called a *cut-vertex* if removing v disconnects G. A cut-vertex in the graph G is sometimes called an *articulation point* of G. A graph having

no cut-vertex is called a *block*. A *block of a graph* is a maximal subgraph that is a block. Any graph is the union of its blocks. It is easy to verify that any two blocks of the graph can share at most one vertex which is an articulation point. A vertex which is not an articulation point is a *non-articulation point*. In any communication network, articulation points are of higher importance than other nodes of the network. They connect different parts of the network and route messages between those parts.

Example 1. *The graph depicted in Figure 2.1 (a) has a self-loop, is not connected and since there are two edges connecting the same pair of vertices, is not simple. Graph G, depicted in Figure 2.1 (b), is a simple, connected graph with no self-loops, the vertices v_3 and v_4 are cut-vertices. The subgraph induced on vertices v_1, v_2 and v_3 is a block of G. Figure 2.1 (c) illustrates the blocks of G. In this figure the vertices v_1, v_2, v_5, v_6, and v_7 are non-articulation points of G.*

Similarly, we call an edge e of a graph G a *cut-edge* if removing e disconnects G. In a communication network, a cut-edge corresponds to a link which connects two disjoint parts of the network. A network with a cut-edge has a very low tolerance for any fault at that cut-edge; therefore, networks are usually designed so that for any two disjoint parts of the network there are at least two links connecting those parts. In graph theoretic language, the underlying graph of such a network is called *edge-biconnected*; a graph that has no cut-edges. If a maximal induced subgraph G' of a graph G is edge-biconnected, G' is an *edge-biconnected component of G*. An edge connecting two edge-biconnected components of a graph is called a *bridge*.

Example 2. *In the graph G depicted in Figure 2.2 the subgraph G_1 induced on vertices $v_1, v_2, ..., v_6$ is an edge-biconnected component of the graph. The subgraph G_1 is*

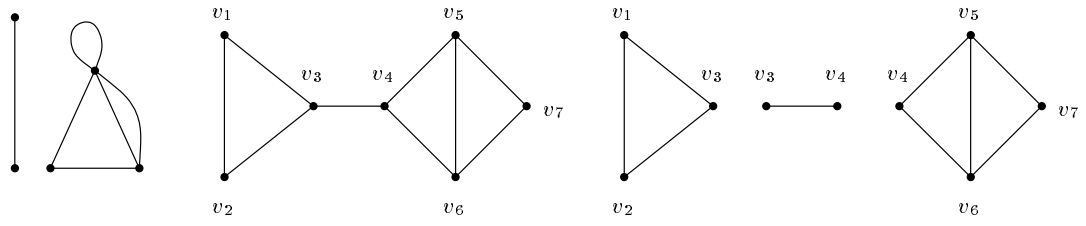

Figure 2.1: a) A graph with a self-loop which is neither connected or simple b) A simple connected graph G c) Blocks of G.

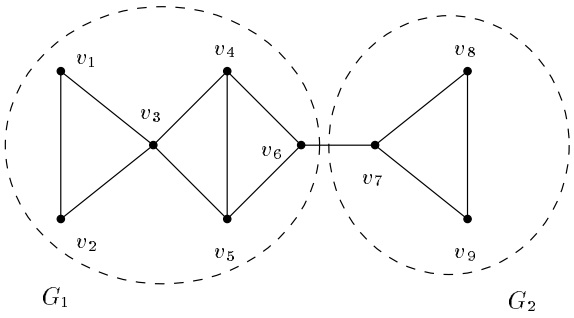

Figure 2.2: Edge-biconnected components of a graph.

composed of two blocks (one consisting of the vertices v_1, v_2, and v_3 and the other consisting of the vertices v_3, v_4, v_5 and v_6). The subgraph G_2 induced on vertices v_7, v_8 and v_9 is also an edge-biconnected component. The edge (v_6, v_7) which connects the two edge-biconnected components G_1 and G_2 is a bridge.

Observation 1. *If G_1 and G_2 are two edge-biconnected components of a graph G, then any path P connecting G_1 and G_2 goes through a unique bridge connected to G_1.*

In real communication networks, we usually have two type of nodes. One group of nodes are those which operate as routers in the network. Each of these nodes is usually connected to more than one other router in the network. Another type of node is connected to just one node in the network and has to route any message through that unique node. These are usually user terminals. Each router in the network may be connected to zero or more such terminals. As we will see later, if we do not consider terminals as parts of the network we may have different properties than the case in which terminals are considered as parts of the network.

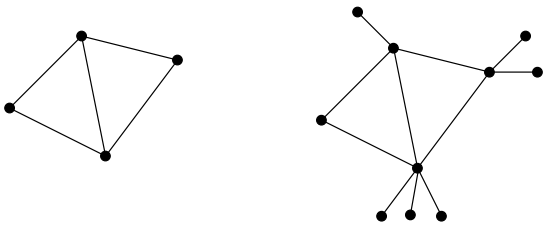

Figure 2.3: a) A graph G b) G-star.

Definition 1. *Given a graph G, a* G-star graph *is the graph G with zero or more*

leaves (nodes of degree 1 or terminals in a communication network) attached to each of the nodes of G (see Figure 2.3).

Definition 2. *[Wes96] The* product *of graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$, written $G \times H$, is the graph with vertex set $V_G \times V_H$ specified by putting the vertex (u, v) adjacent to the vertex (u', v') if and only if (1) $u = u'$ and the edge $(v, v') \in E_H$, or (2) $v = v'$ and the edge $(u, u') \in E_G$.*

Figure 2.4 depicts an example of a product graph.

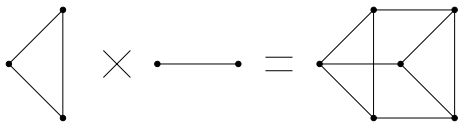

Figure 2.4: An example of a product graph.

2.2 Routing schemes

As mentioned earlier, throughout this thesis we assume that the underlying graph of any network is simple, connected, and does not have any self-loops. For any edge $(u, v) \in E$ we will use both (u, v) and (v, u) in order to assign two unidirectional labels to the edge (as we will see later in this chapter), but the graph is assumed to be undirected. It is also reasonable to assume that the networks and their underlying graph are finite. From now on, we will use a network and its underlying graph interchangeably, wherever there is no ambiguity.

In a network with underlying graph $G = (V, E)$, for any vertex $x \in V$, we denote by $N(x)$ the set of vertices which are adjacent to x (the set of *neighbors* of x), that is $N(x) = \{y | y \in V \text{ and } (x, y) \in E\}$. The number of vertices in this set is denoted by $deg(x)$. At each node of the network we can define a *routing function* that determines how to route messages at that node. In other words, if there is a message heading towards a prespecified destination address, the function specifies through which of the neighbors the message should be forwarded. A routing function for the whole network is the union of routing functions at each node of the network. More precisely, a routing function is defined as follows.

Definition 3. A routing function R *of a graph* $G = (V, E)$ *is a set of functions*

$$R = \{R_x | x \in V, R_x : V \rightarrow (\{x\} \cup N(x))\}$$

such that for any pair of vertices $x, y \in V$ *there exists a sequence of vertices* $x = x_0, x_1, ..., x_k = y$ *such that* $\forall i, 0 \leqslant i < k, R_{x_i}(y) = x_{i+1}$ *and* $R_y(y) = y$.

Definition 4. *An* optimum routing function *is a routing function which routes messages on shortest paths.*

A *routing scheme* in general is a strategy which determines how to route messages using a specified routing function in a network. It determines the information needed to be stored at each node of the network and the preprocessing needed to generate that information. The routing scheme also indicates what happens at each node when it needs to route a message towards a specific destination.

The routing scheme and the routing function are two separate concepts. We may have two different routing schemes which use the same routing function on a network and therefore the paths a message traverses using each of these routing schemes are the same. On the other hand, for a given routing scheme, we may be able to implement different routing functions. For example, using a routing table at each node of a network is a routing scheme. In this routing scheme the information needed to route messages, or the routing function, is stored at each node of the network in the form of a table. Therefore, we can have many different routing functions by assigning different routing tables to the nodes of the network, although the routing scheme is the same.

2.2.1 Properties of routing schemes

When routing messages in a network, there are several properties which are desirable *e.g.* correctness, simplicity, low delay at intermediate nodes, short routing paths, high throughput and low memory requirements. In this thesis we consider routing schemes in which any message eventually reaches its destination. In other words, we are interested only in the routing schemes which route messages correctly.

In any routing scheme, we assume that the delay at each intermediate node is proportional to the running time of the algorithm which determines through which link a message should be forwarded. When using a routing table, for example, the running time of this algorithm is $O(\log n)$ using a simple binary search. Obviously, we are interested in algorithms which are fast.

The routing scheme may also have different properties based on the characteristics of the underlying network. For example, the cost of all links in the network can be the same (*uniform cost links*) or may have different costs (*weighted links*). The cost of the links may be fixed, or may vary over time (*dynamic cost links*). A routing scheme which works on a network with fixed cost links is called *static* and a routing scheme used on a network with dynamic cost links is called a *dynamic* routing scheme.

Some of the characteristics of a routing scheme is based on the properties of the routing function which is implemented in that routing scheme. For example, a routing scheme which implements an optimum routing function is called an *optimum routing scheme.*

In the next section we introduce some routing schemes which have an efficient memory usage and study their properties.

2.3 Compact routing schemes

Each routing scheme consists of some preprocessed information which is stored in the nodes of the network and an algorithm which determines how to use this information to route the messages in the network. There is a trade-off between the complexity of the algorithm and the amount of space needed to store the preprocessed information.

At one extreme, we can have a routing scheme which implements a very simple algorithm but requires a lot of space. For example, we can have a routing table at each node of the network and an algorithm which finds an entry of the routing table which corresponds to a specified destination address and forwards the message to the node mentioned in that entry of the routing table. This scheme uses a simple algorithm and, assuming that we are given the routing function, it does not require a lot of preprocessing. This method requires $\Theta(n)$ space at each node of the network, which is not feasible for large networks of computers.

At the other extreme, for some graphs like grids and hypercubes we can have an algorithm which computes the routing path by using the address of the current node and the destination address. Ideally, for general networks, we would like to have algorithms which are simple and which use a small amount of space to store the preprocessed information [TvL95, FGS93, NO99, Fre96]. Such routing schemes are called *compact routing schemes* and have been studied extensively [FJ86, FJ88, FJ89, Cow99, KK96].

One example of such a routing scheme is a *prefix routing scheme* [TvL95]. In this scheme, we label each node of the network with a string, over some alphabet Σ, which serves as a name. We also label each link at a node with a unique string, possibly by ϵ, the *empty* string. When a message arrives at a node u it is forwarded through the link e such that the label of e is the *maximum length prefix* of the destination address. For example, if the destination address is abs and the link labels available are ϵ, a, ab and cb, then the message will be forwarded through the link with label ab.

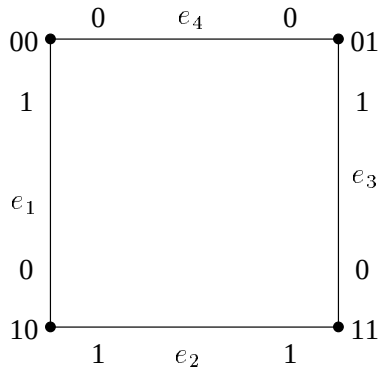

Figure 2.5: An example of prefix routing.

Example 3. *Figure 2.5 illustrates an example of prefix routing. In this example* $\Sigma = \{0, 1\}$. *The label of each node and each outgoing link at each node is shown in the figure. If there is a message at the node labeled* 00 *with destination address* 11*, it is forwarded through* e_1*, which is labeled by* 1*, since it is the link leaving* 00 *with the maximum length prefix of* 11*. From node* 10 *the message is then forwarded through the link* e_2.

Clearly, each link must be properly labeled so that for any pair of nodes s and t the message originating from s eventually reaches t. Bakker *et al.* have shown the feasibility of such a scheme in a *dynamically growing network*; a network which results from a single node by adding new nodes and inserting new links. In this result, the *adaption cost* is the time needed to change the labels of nodes and links after each insertion.

Theorem 1. *[BvLT93] There is a prefix routing scheme for any dynamically growing network. Insertions of the links and nodes require an adaption cost of* $O(1)$.

It has been proved that trees, rings of size less than four, complete graphs, complete bipartite graphs, hypercubes and d-dimensional grids have optimum prefix routing schemes [BvLT93]. Unfortunately, the complete class of networks which have optimum prefix routing schemes has not yet been characterized.

Another example of a compact routing scheme is the *boolean routing scheme* which was originally introduced by Flammini *et al.* [FGS93]. In this method, destinations in the network are grouped together to share the same link at a node if they satisfy a certain boolean predicate on their name labels. Each node of the network is assigned a string of bits, and boolean predicates are assigned to the links based on the labels of the nodes. Elementary boolean functions such as $\neg, \vee$ and $\wedge$ are used to form the predicates.

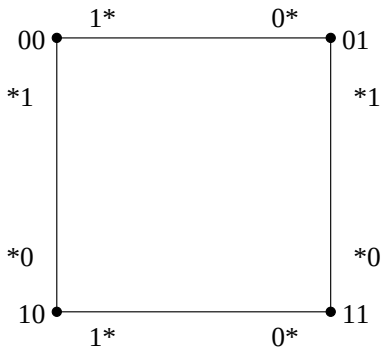

Figure 2.6: An example of boolean routing.

Example 4. *Figure 2.6 shows an example of a boolean routing scheme, where* $0*$ *is the predicate* $bit_1(v) = 0$, $*1$ *is the predicate* $bit_2(v) = 1$, *and so on. Here,* $bit_i(v)$ *denotes the* i*th bit of* v *counting from left to right. If a destination label satisfies more than one predicate, any satisfied predicate can be used to forward the message.*

It has been shown that with no more than $(2 \log n)$-bit strings as labels of the nodes, one can design predicates such that there are optimum boolean routing schemes for rings, trees, hypercubes, d-dimensional grids, complete graphs and complete bipartite graphs [FGS93].

In the following section we will continue the study of the trade-off between the amount of information needed at each node of the network and the complexity of the routing algorithm. We will introduce another compact routing scheme and investigate its properties.

2.4 Interval routing scheme

If the labels of the nodes in a network are integers, a natural method for encoding the routing information at each node of the network is to use intervals as groups of destination addresses. This method, which is known as Interval Routing Scheme (IRS), is a well-known compact routing scheme and was originally introduced by Khatib and Santoro [SK85]. It has been implemented in the C104 Router Chip which is used in the INMOS T9000 Transputer design [INM91, WMT93].

Definition 5. *An* interval $I = [a, b]$ *of* $\{1, 2, ..., n\}$, *where* $a, b \in \{1, 2, ..., n\}$, *is the set of integers* i *such that:*

$$\begin{cases} a \leqslant i \leqslant b & \text{if } a \leqslant b \text{ (linear interval); or} \\ a \leqslant i \leqslant n \text{ or } 1 \leqslant i \leqslant b & \text{if } a > b \text{ (cyclic interval)} \end{cases}$$

Definition 6. *[SK85, FG98] We let* $G = (V, E)$ *be a graph, such that* $|V| = n$. *An* interval routing function *on* G *is a routing function* $R = \{R_x | x \in V\}$ *on* G *defined by:*

i) a one-to-one function $\mathcal{L} : V \to \{1, 2, ..., n\}$ *which labels the vertices of* G;

ii) a set of intervals $\mathcal{I} = \{I_{x,e} | e = (x, y) \in E \text{ and } x, y \in V\}$ *such that the following properties are satisfied:*

- *Union property:*
  $(\cup_{e=(x,y)} I_{x,e}) \cup \{\mathcal{L}(x)\} = \{1, 2, ..., n\}$;
- *Disjunction property:*
  $\forall e = (x, y), e' = (x, y'), y \neq y' \Rightarrow I_{x,e} \cap I_{x,e'} = \emptyset$;
- $R_x(y) = z \Leftrightarrow \mathcal{L}(y) \in I_{x,e}$, *where* $e = (x, z)$.

We denote an interval routing function with a pair $(\mathcal{L}, \mathcal{I})$ that satisfies the conditions of Definition 6. A routing scheme which uses an interval routing function is called an *Interval Routing Scheme* or an IRS for short. In this routing scheme when a node v has a message with destination address *dest*, if *dest* is not the same as the label of v the message is forwarded through the link e whose label contains *dest*. If *dest* equals the label of v, the routing process ends.

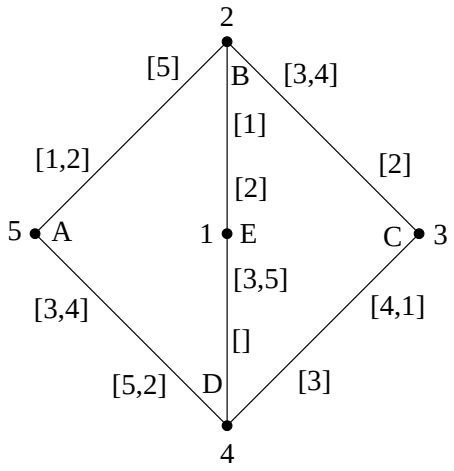

Figure 2.7: An example of interval routing.

Example 5. *In the network depicted in Figure 2.7 if there is a message originating from node B which is labeled by 2 and with destination address 4, since the label of the destination is in the interval* $[3..4]$ *the message will first be forwarded to node C. At node C, since 4 belongs to the interval* $[4, 1]$ *the message is forwarded to node D, which is the destination address, and routing is completed.*

An IRS requires $\Theta(deg(v) \log n)$ space at a node v, where n is the number of nodes in the network ($2 \times \log n$ bits for each of the $\deg(v)$ intervals). This is more efficient memory allocation than required by explicit routing methods *e.g.* using routing tables.

2.5 Variants of IRS

When studying characteristics of IRS, a natural question is: can we slightly change IRS to improve various properties, *e.g.* the length of the routing paths? There are also

cases in which due to a practical restriction we need to restrict the definition of IRS. If we are able to find a scheme of type $\mathcal{R}$ on a specific network, the network is said to *support* $\mathcal{R}$, *have* $\mathcal{R}$, or *belong to the class of networks supporting* $\mathcal{R}$.

2.5.1 Linear, strict and optimum schemes

In designing very small and fast routing chips, testing whether an integer is in a linear (not cyclic) interval is much easier than the same test for cyclic intervals. Therefore, it might be interesting to consider an IRS in which the intervals assigned to the links of the network are linear. This variation of IRS is called a *Linear IRS* or LIRS. The IRS illustrated in Figure 2.7 is not an LIRS since some edges have cyclic intervals, *e.g* the interval assigned to the link (C, D) is $[4, 1]$, which is cyclic.

A *Strict IRS* or SIRS is an IRS in which the label assigned to a link l at a node v does not contain the label of v. For example, the IRS illustrated in Figure 2.7 is strict.

Like any other routing scheme, an IRS is said to be an *optimum* IRS if it routes messages on shortest paths.

2.5.2 Multi-label schemes

One way to make schemes more flexible and the routing more efficient is to assign more than one label to each link of the network. Instead of assigning just one interval to each outgoing link we may assign k intervals to each link. This scheme is known as k-IRS. Obviously, the scheme introduced in Definition 6 is a 1-IRS. Since we can replace each cyclic interval with at most two linear intervals, any network which supports a cyclic IRS has a 2-LIRS. Bakker, Leeuwen and Tan have proved that the class of graphs supporting LIRS with k ($k \geqslant 1$) intervals assigned to each link is a strict subset of the class of graphs supporting LIRS with $k + 1$ intervals at each link [vLT87, BvLT91].

Clearly, the amount of information needed in a multi-label IRS is more than that of a regular IRS. The following theorem states this fact more precisely.

Theorem 2. *[Gav00] Every k-interval scheme on an n-node graph can be implemented in each node x with $\log\binom{n}{K} + \log\binom{K}{d} + (K - d)\log d + O(\log n)$ bits, which*

is in $O(dk \log(n/k))$ bits, where K is the total number of intervals for the node x, and d the number of links incident to x that have non-empty labels.

This theorem implies that every 1-interval scheme can be encoded with $n + O(\log n)$ bits per node. It has been shown that for specific graphs we can reduce the number of bits needed to route messages.

Theorem 3. *[Gav00] Every n-node tree has a 1-SIRS which can be implemented with $O(\sqrt{n})$ bits in each node.*

2.5.3 Multi-dimensional schemes

A very interesting extension of an IRS is a *Multi-dimensional Interval Routing Scheme (MIRS)* in which we assign multi-dimensional labels to the nodes and multi-dimensional interval labels to the links of the network. This scheme was originally proposed by Flammini *et al.* [FGNT98]. Before giving the formal definition of a MIRS let us define a multi-dimensional interval.

Definition 7. *A* d-dimensional interval $I = [a_1..b_1, a_2..b_2, ..., a_d..b_d]$ *($a_i, b_i \in 1, 2, ..., n$ for $1 \leqslant i \leqslant d$) is the set of all d-ary tuples, $p = (p_1, p_2, ..., p_d)$, such that $a_i \leqslant p_i \leqslant b_i$, for every i, $1 \leqslant i \leqslant d$.*

Definition 8. *A* d-dimensional interval routing scheme *in an n-node network is an IRS in which the labels assigned to the links are d-ary tuples of the form $(p_1, p_2, ..., p_d)$, $1 \leqslant p_i \leqslant n$, for $1 \leqslant i \leqslant d$. The labels assigned to links (at each node) are also d-dimensional intervals. If k intervals are assigned to each link of the network (at each node) we have a $\langle k, d \rangle$-MIRS.*

By this definition, a 1-IRS is the same as a $\langle 1, 1 \rangle$-MIRS. Any d-dimensional label associated with a node of a network denotes a point in d-dimensional Cartesian space with integer coordinates. We will use this point and the label interchangeably throughout this thesis.

If we omit the disjunction property from the definition of an IRS, we are able to represent more than one (and maybe *all*) shortest paths between any pair of nodes with an MIRS, which is called a *multi-path* MIRS. A multi-path MIRS is useful for fault-tolerance and traffic distribution in a network. Ružička and Štefankovič have studied the trade-off between the congestion and the space complexity of a multi-path MIRS [RŠ00].

It is easy to verify that the amount of space needed at each node of the network in a d-dimensional MIRS is d times the amount of space needed in a 1-dimensional IRS. Since usually d is much smaller than n, the number of nodes in the network, this amount of space is still considered efficient compared to explicit routing schemes (*e.g.* routing tables). In Chapter 3 we will show that by increasing d the class of networks which support various multi-dimensional schemes is strictly expanded.

2.5.4 Dynamic versus static schemes

If the costs of the links are fixed over time we have an IRS with static cost links. In real communication networks, the cost of the link may vary over time due to different reasons such as congestion in the network and overloaded links. Hence, it is natural to assume that the costs of the links may vary over time but the labels of the nodes are fixed. Like any other routing scheme, an IRS defined on such a network is said to be an IRS with dynamic cost links. After each change in the costs of links we may or may not be allowed to recompute the labels of the links. When the costs of the links changes in the network, we can assume either that the labels of nodes remain the same (*static node names*) or that nodes can be relabeled (*dynamic node names*).

2.5.5 Hybrid schemes

Any set of properties mentioned above can be combined into an IRS to form a *hybrid IRS*. For example, we may have a multi-label, multi-dimensional and linear IRS with dynamic cost links which is denoted by $\langle k, d\rangle$-MLIRS with dynamic cost links; here, k is the number of intervals at each link and d is the number of dimensions.

2.6 Evaluation of schemes

Considering a specific routing scheme, $\mathcal{R}$, there are two important problems which solving them determines if $\mathcal{R}$ is a suitable routing scheme for a given network or not. The first problem is determining the class of networks which support $\mathcal{R}$. This problem is known as the *characterization problem*. Naturally, if a type of scheme is supported by a large class of networks, it might be quite useful.

The second problem arises after finding out the class of networks which support $\mathcal{R}$. Assuming that a given network belongs to the class of networks supporting $\mathcal{R}$, the important problem is figuring out how well the messages are routed by $\mathcal{R}$. We may have different measures such as the length of the routing paths, the congestion of the network (which is proportional to the number of messages which are routed through each link), the delay of each message, and so on. This is known as the *quality of routing* problem.

In this section we briefly review some of the known results related to these problems.

2.6.1 Characterization Problem

The first important question in studying a scheme is: which class of networks support this routing scheme? Santoro and Khatib have shown that every acyclic digraph has a 1-SIRS [SK85]. For specific cases we have much better results. For example, it has been shown that every graph which is a tree or a ring has an optimum 1-IRS [SK85]. For general graphs, van Leeuwen and Tan have proved the following theorem.

Theorem 4. *[vLT87] All graphs support 1-SIRS and hence 1-IRS.*

Proof. For a given graph G, we let T be a spanning tree of G, rooted at an arbitrary node r. We let $\mathcal{L}$ be a depth first labeling from the root, where $\mathcal{L}(r) = 1$ is the smallest label.

For each vertex u of T we let M_u be the maximum value of $\mathcal{L}(v)$ for all the vertices v that belong to the subtree of T which is rooted at u. Any edge of G which is not in T is assigned an empty interval in both directions. For an edge $e = (u, v)$ in T,

assuming that u is the parent of v, we assign the label $[\mathcal{L}(v), M_v]$ to e at node u. We assign the interval $]M_v + 1, \mathcal{L}(v)[$ to the edge e at node v. It is a trivial task to verify the correctness of this SIRS. ■

Any path constructed by the routing function defined as in the proof of Theorem 4 is embedded in a tree. Therefore, the length of any routing path is at most two times the depth of the tree. If we choose T as a spanning tree of G which consists of the shortest paths from the root to all other nodes in the network, it is easy to verify that the length of the longest routing path is less than two times the diameter of the graph G. This is not a shortest path between a pair of source and destination nodes, unless G is a tree.

It is usually desirable to have an IRS which uses all links of the network. This may reduce congestion in the links. It has been proved that every graph has a 1-SIRS such that all links have non-empty labels [vLT87].

Ružička has showed that not every network has an optimum IRS [Ruž88]). On the bright side, we know some specific classes of networks support optimum schemes. For example, it has been shown that any graph which is a path, $2D$-grid, or a complete graph supports an optimum 1-SLIRS. Any graph which is a $2D$-grid with column-wrap-around, or a complete bipartite graph belongs to the class of networks supporting an optimum 1-SIRS [vLT87].

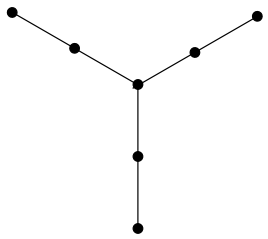

Figure 2.8: The Y-graph.

As mentioned before, an LIRS is a scheme which does not have any cyclic intervals. The following classes of graphs are known to support an optimum LIRS: complete graphs, hypercubes, n-dimensional grids, rings of size at most four, n-dimensional tori $\Pi_{i=1}^{n} d_i$ with $d_i \leqslant 4$ for each i, trees which do not contain the Y-graph (Figure 2.8) as a subgraph [BvLT91], complete r-partite graphs $K_{n_1,n_2,\ldots,n_r}$ with $r \geqslant 2$, $n_i \geqslant 1$, the product $\Pi_{i=1}^{n} G_i$ if the graph G_i has an optimum LIRS for each i [KKR94], unit interval graphs [FG98].

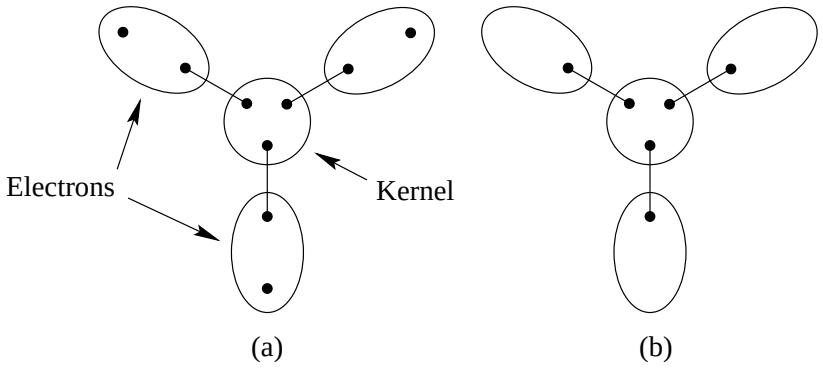

Figure 2.9: a) A lithium graph b) A weak lithium graph

Definition 9. *[FG94] A* lithium graph *is a connected graph with four connected subgraphs* E_1, E_2, E_3 *and* K *such that*

(i) each component E_i, $i = 1, 2, 3$ *has at least 2 vertices;*

(ii) there is no edge connecting E_i *with* E_j *for* $i, j = 1, 2, 3$ *and* $i \neq j$*;*

(iii) each component E_i, $i = 1, 2, 3$ *is connected with* K *by exactly one bridge.*

Fraigniaud and Gavoille have completely characterized the class of networks which support an LIRS.

Theorem 5. *[FG94] A graph* G *supports an LIRS if and only if it is not a lithium graph.*

We can verify that an interval graph cannot be a lithium graph. Therefore, based on the previous theorem, every interval graph supports an LIRS. The class of networks supporting an SLIRS has also been characterized by Fraigniaud and Gavoille.

Definition 10. *[FG94] A* weak lithium graph *is a graph with at least three bridges that connect a connected component (the* kernel*) with three other distinct connected components (the* electrons*).*

In contrast to a lithium graph, in a weak lithium graph the cardinality of the electrons does not matter.

Theorem 6. *[FG94] A graph G supports an SLIRS if and only if G is not a weak lithium graph.*

It is easy to show that if G and H are two graphs each with at least 2 vertices, then the graph $G \times H$ cannot be a weak lithium graph. Therefore, $G \times H$ supports an SLIRS.

In Chapter 3 we generalize Theorem 5 to multi-dimensional schemes. We define k-windmill graphs as a generalization of lithium graphs (where a lithium graph is a 3-windmill graph) and show that a similar result holds for higher dimensions. More precisely, a graph supports a $\langle 1, d \rangle$-MLIRS if and only if it is not a $(2d+1)$-windmill graph. We also generalize the definition of weak lithium graphs to weak windmill graphs and show that a graph supports a $\langle 1, d \rangle$-MSLIRS if and only if it is not a weak $(2d+1)$-windmill graph. The only characterization results for MIRS which were already known are for specific regular graphs.

Theorem 7. *[FGNT98] Any graph which is a d-grid, d-tori or a d-hypercube has an optimum $\langle 1, d \rangle$-MSIRS.*

Fredrickson and Janardan have proved that a graph G with dynamic cost links has an optimum SIRS if and only if G is an outer-planar graph [FJ86]. Tan and van Leeuwen have shown that a graph G with dynamic cost links supports an optimum IRS if and only if G is an outer-planar graph or a K_4 [TvL95]. They also show that a graph G with dynamic cost links has an optimum SIRS with dynamic node names if and only if its biconnected components are either outer-planar or K_4. Bakker *et al.* have shown that a graph with dynamic cost links and dynamic node names has an optimum SLIRS if and only if it is a line or a ring of size three or four [TvL95].

In a very restricted scheme we assume that the costs of the links are dynamic, but the edge labels must be the same for any set of link costs.

Definition 11. *[BvLT91] A* segment *is either a C_3-star graph with leaves attached to only two of the cycle nodes or a C_2-star graph (Figure 2.10). The two nodes of the segment with leaves attached to them are called the* head *and the* tail *of the segment respectively.*

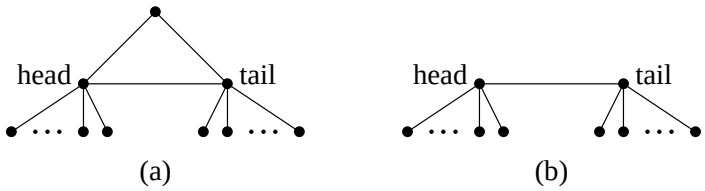

Figure 2.10: a) C_3-star graph b) C_2-star graph.

Definition 12. *[BvLT91] A* centipede *is either a segment or a centipede joined with a segment. By joining we mean that the head of the centipede is identified with the tail of the segment that is "attached" to it* i.e. *all the neighbors of these two nodes now become neighbors of the one new node. The head of the joined segment becomes the head of the new centipede (Figure 2.11).*

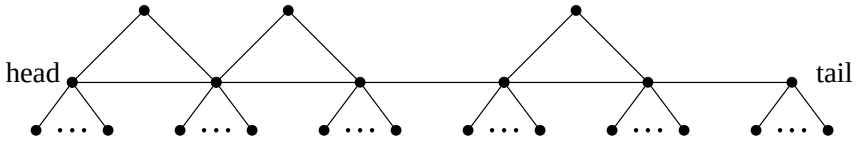

Figure 2.11: A centipede graph.

Bakker *et al.* have also shown that a graph with dynamic cost links and fixed link labels has an optimum LIRS if and only if it is a centipede [BvLT91]. They also have proved that a graph with dynamic cost links and dynamic node names has an optimum LIRS if and only if it is a centipede, a K_3-star or a K_4-star.

In Chapter 3 we also consider the problem of characterizing the class of networks which support an optimum $\langle 1, d\rangle$-MSLIRS. We give a complete characterization of this class of networks there and show that by increasing the number of dimensions, d, the class of networks which support this scheme is also strictly expanded.

2.6.2 Quality of routing

The main quantity used to measure the quality of a routing scheme in this thesis is the length of the routing paths. We usually consider the length of the longest routing paths in the network, since it is a critical value for many applications.

Definition 13. *[Gav00] Let R be an IRS on a graph G. The* dilation *of R, denoted by dilation(R), is the length of the longest routing path which a message traverses. The* k-dilation of G, *denoted by k-dilation(G), is the minimum over all the k-IRS R on G, of the dilation of R.*

For 1-IRS, Tse and Lau has shown that for every even D, there is a graph G of diameter D and girth $2D$ such that 1-dilation(G) $\geqslant 2D-3$ [TL97b]. Therefore, the 1-IRS proposed in the proof of Theorem 4 is close to the optimal. Unfortunately, no upper bound is known for this value for LIRS. Eilam *et al.* have proved that for every fixed D, there exists a graph G of diameter at least D such that every 1-LIRS has a dilation at least $D^2/16$. Moreover, G is planar and of maximum degree four [EMZ96, EMZ99].

Clearly, if we allow more than one interval at each edge, it is possible to decrease the dilation. Tse and Lau have proved a series of lower bounds for different numbers of intervals at each link. They have shown that there is a graph G with diameter D such that for any 2-IRS the longest routing path is at least $5D/4-1$ [TL95]. More generally, they show that there exists a graph G such that for any k-IRS, $k = 2, ..., \Omega(\sqrt[3]{n})$, the longest routing path is not shorter than $\frac{2k+1}{2k}D-1$ and for $k = \Omega(\sqrt[3]{n}), ..., \Omega(\sqrt{n})$, the longest routing path is not shorter than $\frac{6k+1}{6k}D-1$. Kráľovič *et al.* have proved an upper bound and shown that for every n-node graph G of diameter D, there exists a $k \leqslant \lceil\sqrt{n \ln n}\rceil + 1$ such that k-dilation(G) $\leqslant \lceil 3D/2\rceil$ [KRŠ00].

In Chapter 4 we investigate the quality of routing in MIRS. Similar to the characterization problem, the only previously known results in this case were for regular graphs or for the 1-dimensional case. We show that for any integer values k and d, there is a graph G which for any $\langle k, d\rangle$-MLIRS the length of the longest routing path is at least $\frac{3}{2}D$, where D is the diameter of the graph G. This bound is better than the $\frac{5}{4}D-1$ lower bound of Tse and Lau, even though they just consider the 1-dimensional case and here we consider the d-dimensional case. We also prove a lower bound of $\Omega(D^2/d)$ for $\langle 1, d\rangle$-MIRS and an upper bound for interval graphs.

2.7 Interval routing and hypergraph drawing

A hypergraph can be viewed as a generalization of classical notion of a graph in which each hyperedge represents a relationship between two or more vertices. In other words, a hypergraph is an ordered pair (V, E), where $V = \{v_1, v_2, ..., v_n\}$ is a set of vertices and $E = \{e_1, e_2, ..., e_m\}$ is a set of hyperedges, where each hyperedge e_i, $1 \leqslant i \leqslant m$, is a subset of V. We usually assume that each hyperedge has at least two vertices. Figure 2.12 represents a possible graphical representation of a hypergraph in which closed curves indicate hyperedges.

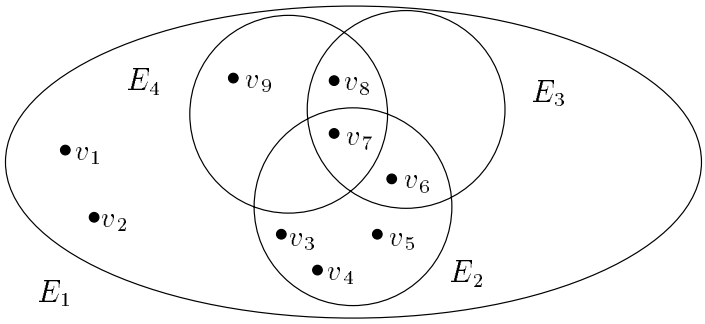

Figure 2.12: A hypergraph H with 9 vertices and 4 hyperedges.

The problem of drawing graphs in the plane and in spaces with higher dimensions has been studied for several years. In Chapter 5 we will introduce a specific drawing for hypergraphs which we call a *box representation of hypergraphs*. We will show a strong relationship between the problem of finding a box representation for a hypergraph and the problem of finding a certain type of MIRS for graphs. We also prove a tight bound on the number of dimensions of the space needed to draw a hypergraph, based on the results we have for MIRS.

Chapter 3

Characterization results

When studying the characteristics of a scheme $\mathcal{R}$ one of the main problems is characterizing the class of networks which support $\mathcal{R}$. The solution to this problem is used to determine if $\mathcal{R}$ can be used to route messages in a given network. The class of networks supporting IRS have been characterized [SK85]. The class of networks which support an LIRS or a SLIRS, which excludes a large class of networks, have been characterized by Fraigniaud and Gavoille [FG94]. Fraigniaud and Gavoille define a class of graphs called *lithium graphs* (Section 2.6.1) and show that a network supports an LIRS if and only if its underlying graph is not a lithium graph. They also define a class of networks called *weak lithium graphs* and show that a network supports an SLIRS if and only if its underlying graph is not a *weak lithium graph*.

The only known classes of networks which support different types of multi-dimensional schemes are specific interconnection networks such as rings, grids, tori, hypercubes and chordal rings [FGNT98]. In this chapter we investigate the problem of characterizing classes of networks which support MIRS. We give a complete characterization of the class of networks supporting $\langle 1, d\rangle$-MLIRS (Section 3.1) and $\langle 1, d\rangle$-MSLIRS (Section 3.2). We also consider networks with dynamic cost link and completely characterize the class of networks which support an optimum $\langle 1, d\rangle$-MSLIRS with dynamic cost links(Section 3.3).

3.1 Characterization of networks supporting $\langle 1, d \rangle$-MLIRS

In order to give a complete characterization for the class of networks which support an $\langle 1, d \rangle$-MLIRS we start with examples of graphs which do not support $\langle 1, d \rangle$-MLIRS. Then, using the idea behind these examples, we introduce a class of graphs which do not support$\langle 1, d \rangle$-MLIRS. Finally, we show that for any graph that is not in this class, one can always construct a $\langle 1, d \rangle$-MLIRS.

Bakker et al. [BvLT91] have shown that the graph shown in Figure 3.1 (a) (known as the Y graph) does not have an LIRS (which is a $\langle 1, 1 \rangle$-MLIRS). Here, we prove a similar result in the d-dimensional case. First, let us start by generalizing the definition of a Y graph.

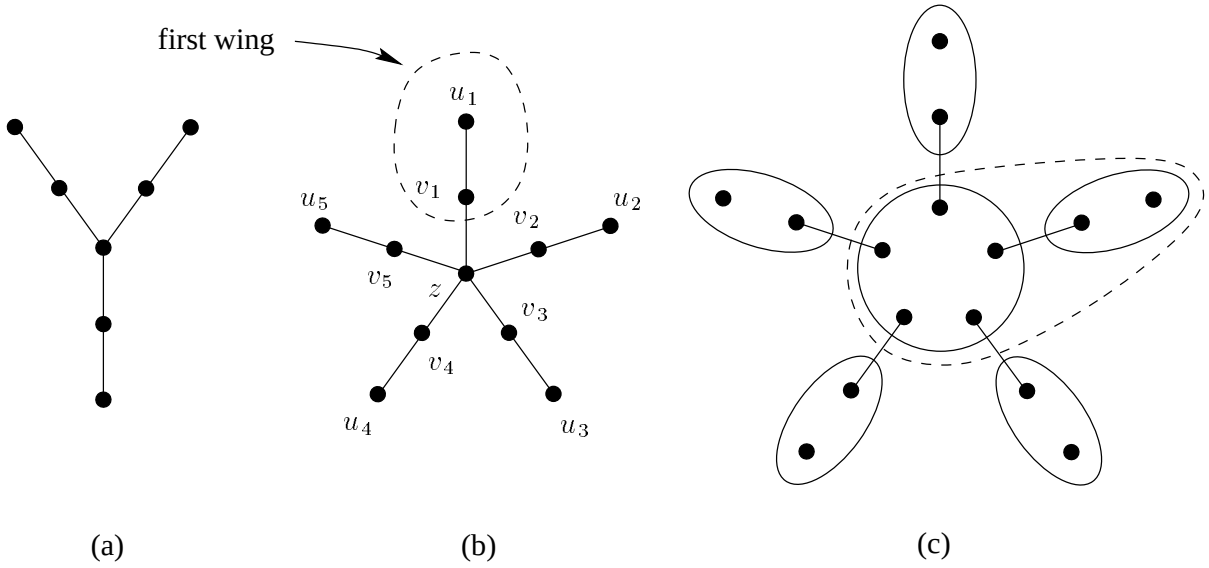

Figure 3.1: (a) The Y graph (b) The Y_5 graph (c) A 5-windmill graph.

Definition 14. *The* Y_k graph *is a graph having* $2k+1$ *vertices* $u_1, u_2, ..., u_k$, $v_1, v_2, ..., v_k$ *and* z. *There is an edge connecting* u_i *to* v_i, *for every* i, $1 \leqslant i \leqslant k$, *and another edge connecting each* v_i *to* z, $1 \leqslant i \leqslant k$ *(Figure 3.1 (b)). We call the subgraph consisting of* u_i *and* v_i *the* ith wing *of the graph.*

The Y graph of Figure 3.1 (a) is a Y_3 graph by our new definition. To prove that the Y_3 graph does not have an LIRS let us assume it has an LIRS and the vertices of the

graph are assigned integer labels taken from $\{1, 2, ..., 7\}$. Since we have three wings, there is a wing, say the ith wing, which does not contain 1 or 7 (the minimum or the maximum label). Now, the interval assigned to the edge (v_i, z) at v_i must contain both 1 and 7. Therefore, this interval contains the label of u_i which is not possible.

We can prove a similar result for d-dimensional LIRS and for the Y_{2d+1} graph. In fact, we can immediately observe that if each wing of the Y_{2d+1} graph had more than just two vertices, as long as those vertices where not directly connected to the vertex z or to the vertices in other wings, the graph cannot support a d-dimensional MLIRS. In order to prove this more general statement, we define a k-windmill graph as follows.

Definition 15. *A* k-windmill *graph is a connected graph with* $k + 1$ *connected components* $A_1, A_2, ..., A_k$ *(*arms of the k-windmill graph*) and* R *(*center of the k-windmill graph*) such that:*

(i) each component $A_i, 1 \leqslant i \leqslant k$, *has at least two vertices;*

(ii) there is no edge connecting A_i *to* A_j *for* $1 \leqslant i, j \leqslant k$ *and* $i \neq j$; *and*

(iii) each component $A_i, 1 \leqslant i \leqslant k$, *is connected with* R *by exactly one bridge.*

Figure 3.1 (c) illustrates a 5-windmill graph. Obviously, by this definition, a Y_k graph is also a k-windmill graph. Also, as Figure 3.1 (c) indicates, a k-windmill graph is an i-windmill graph for any i, $1 \leqslant i \leqslant k - 1$. This can easily be shown by expanding R to include $A_{i+1}, ..., A_k$.

Lemma 1. *Any* $(2d + 1)$*-windmill graph does not support a* $\langle 1, d \rangle$*-MLIRS.*

Before giving the proof of this lemma, let us start with a new definition which will be used in the proof and in other sections. Let us consider a set of points P in d-dimensional space. If for any dimension i, $1 \leqslant i \leqslant d$, the ith coordinate of a point b in P is less than or equal to the ith coordinate of every other point in P, b is called a *minimum point for the* i*th dimension*. A maximum point is defined similarly. A *boundary set* B *of* P is a minimal set of points in P containing a minimum and a maximum point for each dimension $i, 1 \leqslant i \leqslant d$, where one point can be both the minimum and the maximum point for the same or different dimensions.

Example 6. *Figure 3.2 illustrates an example of a boundary set in 2-dimensional space. Here,* $P = \{1, \ldots, 7\}$ *and* $\{1, 5, 7\}$ *is a boundary set of* P*. The set* $\{2, 5, 7\}$ *is also a boundary set of* P*. We note that point* 7 *is the maximum point for one dimension and the minimum point for another dimension.*

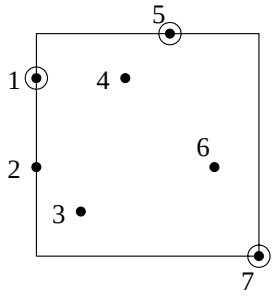

Figure 3.2: An example of a boundary set in 2-dimensional space.

For any set of points in d-dimensional space, the number of points in any boundary set is at most $2d$. It is easy to show that if an interval contains the points in the boundary set B of a set of points P, it contains all points in P. Now we can easily prove Lemma 1.

Proof. (Lemma 1) Let us assume, by way of contradiction, that there is a $\langle 1, d \rangle$-MLIRS for a given $(2d + 1)$-windmill graph ($d \geqslant 1$) and consider the boundary set B of the vertices of the graph. We have at most $2d$ vertices in the boundary set B. Since a $(2d + 1)$-windmill graph has $2d + 1$ arms, there is an arm, say the jth arm, that does not contain any vertex in the boundary set B. Every d-dimensional interval containing all of the vertices in B contains all vertices of the $(2d + 1)$-windmill graph as well. Thus, the interval assigned to the bridge connecting the jth arm to the center of the $(2d + 1)$-windmill graph, say (u, v) (u is in the jth arm and v is a vertex in the center of the graph) contains all vertices in the $(2d + 1)$-windmill graph. The jth wing has at least another vertex other than u, say u'. Thus, the interval assigned to the edge (u, v) includes u'. Obviously, there is no path going through (u, v) to reach u', which is a contradiction. ■

Lemma 1 introduces a class of graphs which do not support $\langle 1, d \rangle$-MLIRS. In other words, it states a necessary condition for a graph to support a $\langle 1, d \rangle$-MLIRS. In the following sections we will show that this is also a sufficient condition.

Fraigniaud and Gavoille have proved that a graph supports LIRS if and only if it is not a lithium graph [FG94] (which is exactly the 3-windmill graph). We will use this result as the basis for an inductive construction of a $\langle 1, d\rangle$-MLIRS for a given graph G. We start with new definitions.

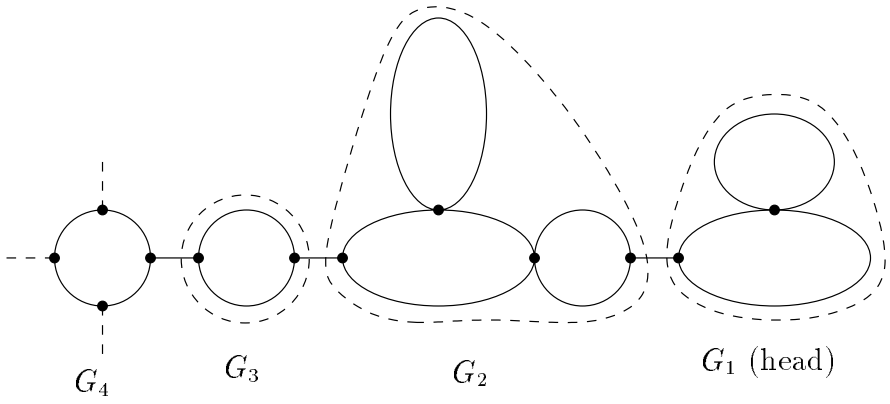

Figure 3.3: The dashed curves indicate edge-biconnected components in this figure. The edge-biconnected components $G_1, G_2, ..., G_4$ form a chain. The edge-biconnected components G_1, G_2 and G_3 form a perfect chain.

Definition 16. *In a graph G, a* chain of edge-biconnected components*, or a* chain *for short, is a set of edge-biconnected components of G with a special ordering of these edge-biconnected components, say $G_1, G_2, ..., G_k$, such that for each $i, 1 \leqslant i \leqslant k-1$, there is a bridge connecting G_i to G_{i+1}. A chain is said to be* perfect *if:*

(i) G_1 is connected to exactly one bridge in G.

(ii) Each edge-biconnected component G_i, $2 \leqslant i \leqslant k$ is connected to exactly two bridges in G.

(iii) If G' is the edge-biconnected component in the graph G which is connected to G_k and $G' \neq G_{k-1}$, then G' is connected to at least three bridges.

We call G_1 the head *and G_k the* tail *of the chain. Trivially if $k = 1$ then G_1 is both the head and the tail of the chain.*

3.1.1 Properties of chains and k-windmill graphs

In this section we review properties of chains and k-windmill graphs. The first observation follows directly from the definition of a chain.

Observation 2. *A perfect chain in a graph G is a proper induced subgraph of G, and the tail of a perfect chain (which is an edge-biconnected component) is connected to the rest of the graph by a bridge.*

The edge-biconnected components $G_1, G_2, ..., G_4$ in the graph depicted in Figure 3.3 and the bridges connecting them form a chain. G_1 and G_4 are the head and the tail of this chain, respectively. In this graph, if we consider the subgraph containing the edge-biconnected components G_1, G_2 and G_3 and the bridges connecting them, then we have a perfect chain. The head of this perfect chain is G_1 and the tail is G_3. As mentioned in Observation 2, G_3 (which is the tail of the perfect chain) is connected to G_4 by a bridge and G_4 is connected to more than two bridges. Since, G_3 is connected to exactly two bridges, the edge-biconnected components G_1 and G_2 does not form a perfect chain.

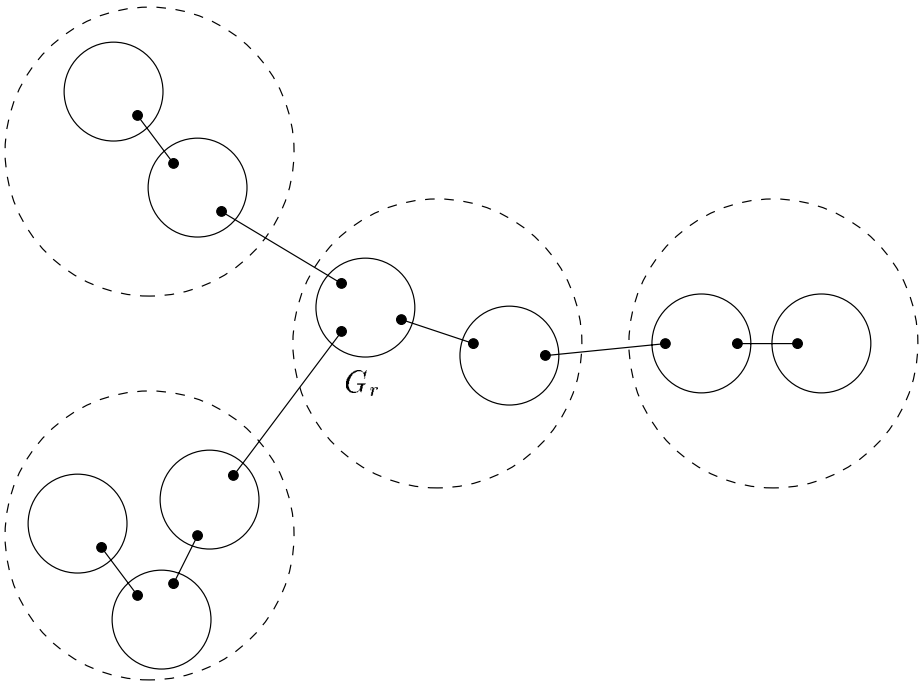

Figure 3.4: Edge-biconnected components in a 3-windmill graph.

Lemma 2. *If a graph G is a k-windmill graph for $k \geqslant 3$, then it is not a chain.*

Proof. We consider each edge-biconnected component of G as a super-node. Clearly, the resulting graph is a tree. Since G is a k-windmill graph ($k \geqslant 3$), there is a node v in this tree such that the degree of v is at least 3 (the super-node G_r in Figure 3.4).

In any chain, each edge-biconnected component is connected to at most two other edge-biconnected components. Therefore, G is not a chain. ■

Lemma 3. *Any non-trivial (having at least one vertex) graph G which is not a chain contains a perfect chain as a proper induced subgraph.*

Proof. Since G is not a chain and is non-trivial, it has more than one edge-biconnected component. It also has an edge-biconnected component which is connected to exactly one bridge (otherwise the edge-biconnected components would form a cycle and this would force the bridges connecting edge-biconnected components to be contained in a cycle, a contradiction to the definition of a bridge). We denote this component by G_1, which is a chain consisting of one edge-biconnected component. Let us denote this chain by C_1. G_1 is both the head and the tail of C_1. In the ith iteration, we expand C_i by adding to it a new edge-biconnected component. If G_i denotes the tail of C_i, since $C_i \neq G$ (G is not a chain) and G is connected, G_i is connected to an edge-biconnected component, say G_{i+1}, which is not in C_i. If G_{i+1} is connected to more than two bridges, then C_i is a perfect chain and is a proper induced subgraph of G. This completes the proof. Otherwise, we expand C_i to C_{i+1} by adding G_{i+1} and the bridge connecting G_i to G_{i+1}. The tail of the new chain C_{i+1} is now G_{i+1}. If we repeat this step, the algorithm will eventually terminate since G is finite and not a chain. ■

For example, in the graph depicted in Figure 3.3 we start with the edge-biconnected component G_1, which is connected to exactly one bridge ($C_1 = G_1$). G_1 is connected to the edge-biconnected component G_2 which has exactly two bridges, so we let C_2 be the chain consisting of G_1 and G_2. Similarly, we add G_3 to C_2 to obtain C_3. Now, G_3 is connected to the edge-biconnected component G_4, which is connected to more than two bridges. This terminates our algorithm and the chain C_3, which is the subgraph consisting of G_1, G_2 and G_3 and the bridges connecting them, is a perfect chain.

In constructing a $\langle 1, d\rangle$-MLIRS, we will use this lemma in the induction step to reduce the size of the graph. This reduction has a very nice property that is the heart of the main proof, which is stated in the following lemma.

Lemma 4. *If a graph G is not a chain and is not a k-windmill graph ($k > 3$), we can remove any perfect chain from G and the resulting graph is not a $(k-1)$-windmill graph.*

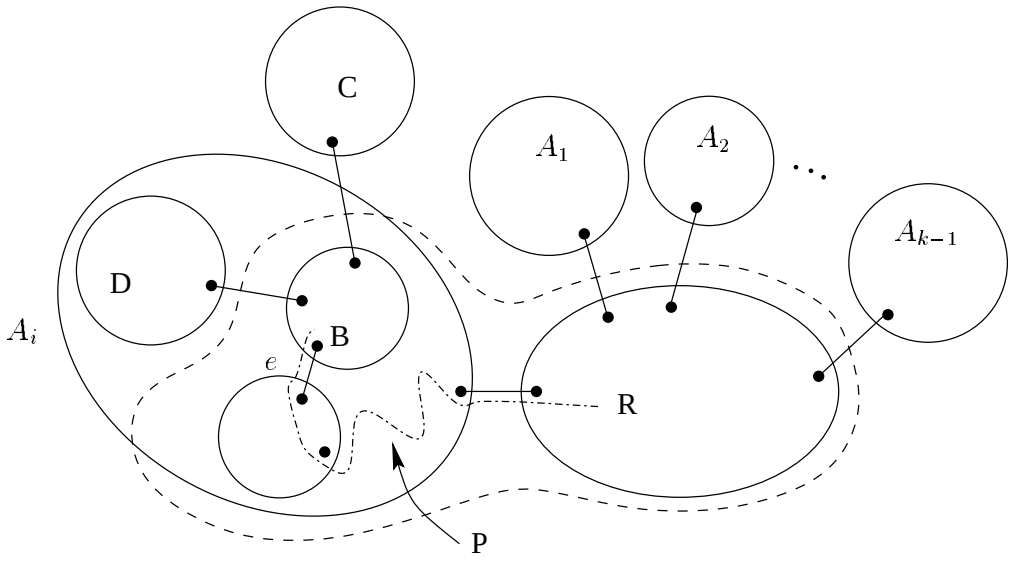

Figure 3.5: C and D will become arms in the k-windmill graph.

Proof. Since G is not a chain, by Lemma 3, there is a perfect chain C which is a proper induced subgraph of G. We let G' denote the graph $G - C$. We assume, to the contrary, that G' is a $(k-1)$-windmill graph. By the definition of a $(k-1)$-windmill graph, G' has k disjoint sets of vertices $A_1, A_2, ..., A_{k-1}$ and R. Since C is a perfect chain, by Observation 2 its tail is connected to G' by a bridge. C cannot be connected to R, otherwise G would be a k-windmill graph. Let us assume that C is connected to an edge-biconnected component, B, which is in the arm A_i for some i, $1 \leqslant i \leqslant k-1$ (Figure 3.5).

By part *(iii)* of the definition of a perfect chain, the edge-biconnected component B is connected to at least three bridges, one connecting B to C and at least two other bridges connecting B to some other edge-biconnected components in G'. By Observation 1 all the paths connecting B and R go through one of the bridges connected to B, say e. We let D be the edge-biconnected component which is connected to B and is not connected to e.

Now, we expand R to contain B and all the edge-biconnected components in the arm A_i except D (and any edge-biconnected component which is attached to D). Since G is a $(k-1)$-windmill graph it has $k-2$ arms other than A_i. We can also consider C and D as two new arms. Hence, G has k arms and is a k-windmill graph, a contradiction. ■

3.1.2 Assigning Labels

In this section we will prove one of the main results of this chapter. First, we need to show how to convert a d-dimensional IRS into a $(d+1)$-dimensional IRS.

If a graph G supports a $\langle 1, d\rangle$-MLIRS ($\langle 1, d\rangle$-MSLIRS), we can convert the d-dimensional scheme to a $(d+1)$-dimensional one by adding a new coordinate to the labels of vertices. The label of this coordinate is set to zero for all vertices. We also set the newly added coordinate of each interval to be $[0..0]$. It is a trivial task to verify that this IRS routes the messages exactly like the d-dimensional IRS. In other words, we can expand a d-dimensional IRS to a $(d+1)$-dimensional IRS.

Lemma 5. *If a graph G supports a $\langle 1, d\rangle$-MLIRS ($\langle 1, d\rangle$-MSLIRS) it also supports a $\langle 1, d+1\rangle$-MLIRS ($\langle 1, d+1\rangle$-MSLIRS).*

Now, we have all the tools we need to prove the main theorem of this section.

Theorem 8. *A graph G has a $\langle 1, d\rangle$-MLIRS if and only if it is not a $(2d+1)$-windmill graph.*

Proof. First, we show that if a graph is not in the class of $(2d+1)$-windmill graphs, then it has a $\langle 1, d\rangle$-MLIRS. We use induction on d, the number of dimensions. Fraigniaud and Gavoille [FG94] have proved that if a graph G is not a lithium graph, which is exactly a 3-windmill graph, then there is a 1-LIRS for G (a $\langle 1, 1\rangle$-MLIRS). This is the basis of our induction.

Let us suppose that for any $i \leqslant d-1$, if a graph is not a $(2i+1)$-windmill graph, it has a $\langle 1, i\rangle$-MLIRS. Now, we want to show that if a graph G is not a $(2d+1)$-windmill graph, $d > 1$, then it has a $\langle 1, d\rangle$-MLIRS. We first show how to label the vertices of G. Then, we describe how we can update intervals in each step of the induction. Finally, we prove the correctness of this vertex and link labeling.

Labeling vertices:
Although G is not a $(2d+1)$-windmill graph it can be a $(2d-1)$-windmill graph. If G is not a $(2d-1)$-windmill graph, by the induction hypothesis it has a $\langle 1, d-1\rangle$-MLIRS

and by Lemma 5, G also has a $\langle 1, d\rangle$-MLIRS, completing the proof. Hence, we can assume that G is a $(2d-1)$-windmill graph and by recalling Lemma 2, we can assume that G is not a chain. Therefore, by Lemma 3, G has a perfect chain, say C_1, as a proper induced subgraph. Since G is not a $(2d+1)$-windmill graph and $d > 1$, by applying Lemma 4 we can remove C_1 and the resulting graph will not be a $2d$-windmill graph. Since $2d > 3$, we can repeat these steps and remove another perfect chain, C_2, so that the resulting graph, G', is not a $(2d-1)$-windmill graph.

By the induction hypothesis, G' has a $\langle 1, d-1\rangle$-MLIRS. We need to expand this labeling to a $\langle 1, d\rangle$-MLIRS for G.

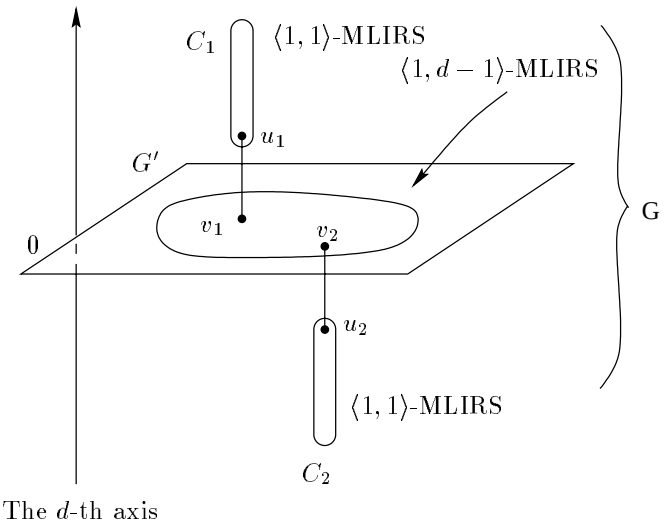

Figure 3.6: Expanding the labels of vertices in G' to labels for vertices in G.

C_1 and C_2 are chains and therefore, by Lemma 2, they are not 3-windmill graphs. Thus, by the induction hypothesis, there is a $\langle 1,1\rangle$-MLIRS for each of them. In fact, Fraigniaud and Gavoille have proved that if a given graph is not a 3-windmill (lithium) graph, we can specify a vertex and find a labeling for the vertices such that the label of the specified vertex is 1 [FG94]. We find such a $\langle 1,1\rangle$-MLIRS for C_1 (C_2) such that the label for the vertex in C_1 (C_2) joining C_1 (C_2) to the rest of the graph G, say u_1 (u_2), is 1 (Figure 3.6).

To construct the new labeling for G, each vertex in G' is assigned a d-dimensional label in which the first $d-1$ coordinates are the same as the labels in the linear $\langle 1, d-1\rangle$-MIRS corresponding to G' and the dth coordinate is 0. Figure 3.6 illustrates an example in which $d = 3$. The third coordinates of the labels assigned to the vertices of G' are all 0, so G' lies in the plane passing through the first and the second axes.

For now, we assume that the labels assigned to the vertices can have any integer values (including 0 and negative integers) as their dth coordinates. We can later shift all the labels such that the dth coordinates of all labels becomes positive.

Let (v_1, u_1) and (v_2, u_2) respectively denote the bridges connecting G' to C_1 and C_2 and let v_1 and v_2 be vertices of G'. We will set the first $d-1$ coordinates of each vertex in C_1 to be equal to the first $d-1$ coordinates of v_1. The dth coordinates of vertex labels in C_1 are the labels assigned to vertices in the previously mentioned $\langle 1, 1\rangle$-MLIRS. In Figure 3.6 the vertices in C_1 all lie on the line passing through v_1 and parallel to the dth axis.

For the vertices in C_2, we will similarly set the first $d-1$ coordinates of each vertex to be equal to the first $d-1$ coordinates of v. If the label of a vertex v in the previously mentioned $\langle 1, 1\rangle$-MLIRS is $l(v)$, we assign $-l(v)$ as the dth coordinates of the new labeling (Figure 3.6). Now as mentioned before, we can shift the dth coordinate of all the labels such that the dth coordinate of the vertex with minimum value becomes 1. We let s denote the amount of this shifting and M denote the maximum value in the dth coordinate of all new labels.

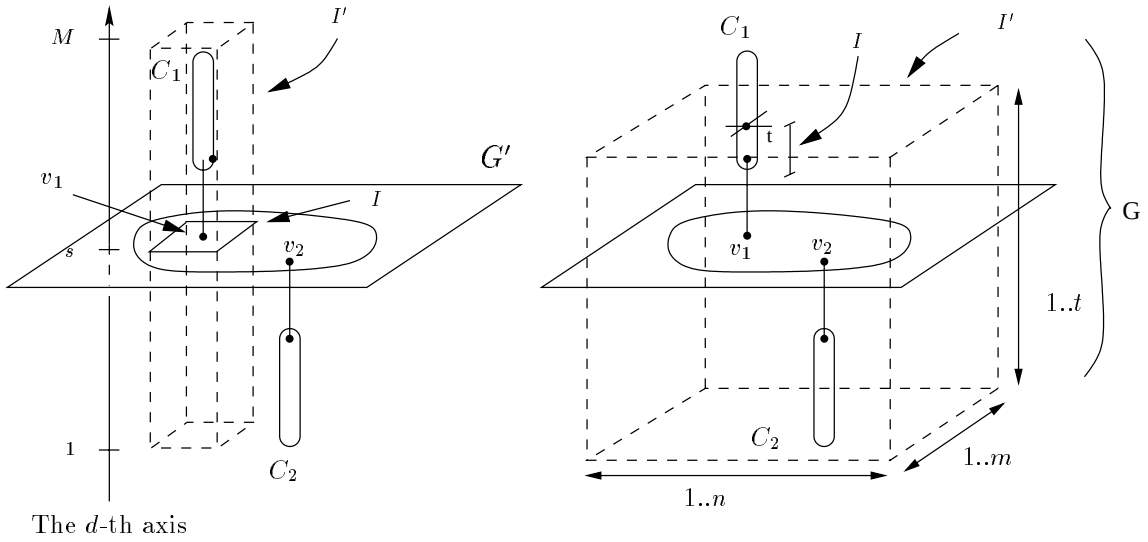

Figure 3.7: (a) Updating an interval in G' (b) Updating an interval, which includes u_1, in C_1 (I is the old interval, I' is the new one in both (a) and (b))

Updating Intervals:

We update intervals as follows: the first $d-1$ coordinates of each interval assigned to a link in G' are the same as the $(d-1)$-dimensional interval associated with that edge in the $\langle 1, d-1\rangle$-MLIRS defined on G'. The dth coordinates of all intervals are

set to be $[1..M]$. Any $(d-1)$-dimensional interval in G' that does not contain v_1 or v_2 will still contain the same set of vertices and any interval containing v_1 (respectively v_2) will also contain all the vertices in C_1 (C_2). For example, the two dimensional interval I, shown in Figure 3.7 (a), contains v_1, so the new three-dimensional interval I' contains all the vertices in C_1. Since I does not contain v_2, I' does not contain any of the vertices in C_2.

For the intervals associated with the links in C_1 or C_2, the first $d-1$ coordinates are set to $[1..n]$. To set the dth coordinate of each interval we will use the previously mentioned $\langle 1,1\rangle$-MLIRS. Let us assume that in the $\langle 1,1\rangle$-MLIRS defined on C_1 the interval assigned to a link e is $I_e = [a..b]$. If I_e does not contain u_1, the dth coordinate of the newly assigned d-dimensional interval will be $[a+s..b+s]$ (we shift the dth coordinate by s units because we have already shifted the vertices in this dimension). If I_e contains u_1, *i.e.* $I_e = [1..b]$ for some b, the dth coordinate of the newly assigned interval will be $I_e = [1..b+s]$. This means that any 1-dimensional interval defined in C_1 will be transformed into a d-dimensional interval containing the same set of vertices in C_1 and if it contains u_1, it will also contain all the vertices in G' and C_2. The interval I depicted in Figure 3.7 (b) contains u_1, so the new interval I' contains the set of vertices in C_1 that where in I and also all the vertices in C_2 and G'. We will analogously assign intervals to the links in C_2.

The only remaining labels to update are labels of the links $(v_1,u_1),(u_1,v_1),(v_2,u_2)$ and (u_2,v_2). The first $d-1$ coordinates of intervals associated with $(v_1,u_1),(u_1,v_1),(v_2,u_2)$ and (u_2,v_2) are set to $[1..n]$ and the dth coordinates will respectively be $[s+1..n],[1..s]$, $[1..s-1]$ and $[s..n]$.

Correctness:
Now, let us consider a message originating from vertex w_s and with destination w_t. If both w_s and w_t are in C_1 (similarly C_2 or G') one can easily check that the newly defined $\langle 1,d\rangle$-MLIRS will route the messages on the same path as the $\langle 1,1\rangle$-MLIRS defined on C_1 (C_2 or the $\langle 1,d-1\rangle$-MLIRS defined on G'). This is because if we consider the set of vertices in C_1 (C_2 or G') each interval assigned to a link contains the same set of vertices as it contained before expanding the labels to d dimensions. If w_s is in C_1 and w_t in G', the message must go through the link (u_1,v_1) because this is the only link connecting C_1 to G'. The intervals in C_1 which contain w_t are exactly the intervals containing u_1. Therefore, this message will be forwarded through the same links as the links through which a message towards u_1 would be forwarded. When the

message reaches u_1, the bridge (u_1, v_1) forwards the message to v_1, because the interval assigned to (u_1, v_1) contains all the vertices in G' and C_2. The rest of the routing will be the same as in the $\langle 1, d-1\rangle$-MLIRS defined on G'.

We can show that if there is a message at node x ($x = u_2, v_1$ or v_2) which is supposed to be forwarded through the bridge connected to x, say e_x ($e_x = (u_2, v_2), (v_1, u_1)$ or (v_2, u_2) respectively), it will be sent to the other end of e_x. Verifying the cases in which w_s is in C_2 or G' is similar. Hence, a message originating at any vertex and going to an arbitrary destination will eventually reach the destination, and the $\langle 1, d\rangle$-MLIRS routes messages on G properly.

We now have shown that if a graph is not in the class of $(2d+1)$-windmill graphs it has a $\langle 1, d\rangle$-MLIRS. Lemma 1 shows that no graph in this class can support a $\langle 1, d\rangle$-MLIRS. Combining these two results completes the proof of the theorem. ■

Since for each $d > 1$, we have a $(2d+1)$-windmill graph which is not a $(2d+3)$-windmill graph (for example the Y_{2d+1} graph), we can state the following corollary:

Corollary 1. *The class of graphs supporting a $\langle 1, d\rangle$-MLIRS is a strict subset of the class of graphs supporting a $\langle 1, d+1\rangle$-MLIRS.*

In other words, increasing the number of dimensions increases the power of the routing scheme.

3.2 Characterization of networks supporting $\langle 1, d\rangle$-MSLIRS

In this section we will give a characterization of the class of graphs supporting $\langle 1, d\rangle$-MSLIRS. We present new definitions and show that with slight changes in some steps in proofs, we can use the same ideas used to characterize the class of graphs supporting $\langle 1, d\rangle$-MLIRS.

In proving Lemma 1, we needed to have at least two vertices in each arm of a $(2d+1)$-windmill graph. Otherwise, if the arm which did not have any vertex in the boundary

set, say A_i, had just one vertex, say x, the interval assigned to the edge connecting A_i to R could contain x and this was not a contradiction. If instead the intervals assigned to the links are supposed to be strict, we could prove a similar lemma, even if we had an arm having just one vertex. This is the main difference between the proofs of this section and the previous one. More formally, let us start with a new definition.

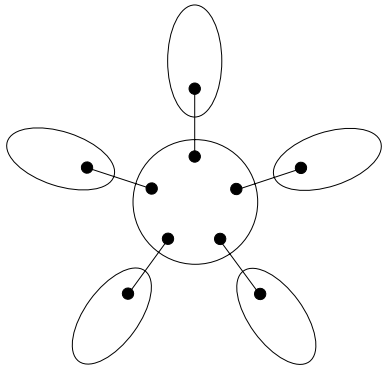

Figure 3.8: A weak 5-windmill graph.

Definition 17. *A* weak k-windmill *graph is a connected graph G with $k+1$ connected components $A_1, A_2, ..., A_k$ (arms) and R such that:*

(i) there is no edge in G connecting A_i to A_j for $1 \leqslant i, j \leqslant k$ and $i \neq j$;

(ii) each component $A_i, 1 \leqslant i \leqslant k$ is connected to R by exactly one bridge (Figure 3.8).

Lemma 6. *Any weak $(2d+1)$-windmill graph does not have a $\langle 1, d\rangle$-MLIRS.*

Proof. The proof is similar to the proof of Lemma 1. We assume there is a $\langle 1, d\rangle$-MSLIRS for a given weak $(2d+1)$-windmill graph ($d \geqslant 1$) and define the boundary set as in the proof of Lemma 1. Since a weak $(2d+1)$-windmill graph has $2d+1$ arms, there is an arm, say the jth arm, that does not contain any vertex in the boundary set B. Every d-dimensional interval containing all of the vertices in B contains all vertices of the weak $(2d+1)$-windmill graph as well. Thus, the interval assigned to the bridge connecting the jth arm to the center of the $(2d+1)$-windmill graph, say (u, v) (u is in the jth arm and v is a vertex in the center of the graph) contains all vertices in the weak $(2d+1)$-windmill graph. In other words, the interval assigned to the link (u, v) contains the label of u which is a contradiction to the fact that the routing scheme is strict. ■

We can also verify, with the same argument as the proof of Lemma 4, that removing any perfect chain from a graph G which is not a weak k-windmill graph will produce a graph which is not a weak $(k-1)$-windmill graph.

The only remaining step is to show that the induction basis and step are also valid in constructing a $\langle 1, d\rangle$-MSLIRS for any graph that is not a weak $(2d+1)$-windmill graph. We already know that any graph which is not weak 3-windmill graph (a weak lithium graph as defined in [FG94]) has a $\langle 1, 1\rangle$-MSLIRS, so the induction basis is true. Since Lemmas 3 and 4 also work for weak windmill graphs and strict MIRS, the induction step holds as well. This gives us the complete characterization of graph supporting $\langle 1, d\rangle$-MSLIRS as follows:

Theorem 9. *A graph G has a $\langle 1, d\rangle$-MSLIRS if and only if it is not a weak $(2d+1)$-windmill graph.*

Corollary 2. *The class of graphs supporting a $\langle 1, d\rangle$-MSLIRS is a strict subset of the class of graphs supporting a $\langle 1, d+1\rangle$-MSLIRS.*

3.3 Optimum multi-dimensional schemes with dynamic cost links

The characterization problem for graphs supporting SIRS has been studied by Fredrickson and Janardan [FJ86] who characterized the class of graphs supporting optimum SIRS with dynamic cost links. Bakker *et al.* give a complete characterization for the class of networks supporting optimum LIRS [BvLT91]. They assume that the labels assigned to the links of the graph remain fixed, even if the costs of the links change. This makes the class of graphs supporting optimum LIRS very restricted. Tan and Leeuwen have also studied the problem of characterizing networks supporting optimum IRS with dynamic cost links and have a characterization for this class of networks [TvL95].

In this section, we completely characterize the class of networks supporting an optimum $\langle 1, d\rangle$-MSLIRS with dynamic cost links. This is a natural generalization of the characterization results (for the 1-dimensional case) mentioned above.

3.3.1 Dividing d-dimensional space

In the following sections we will need a way to divide d-dimensional space and represent the resulting subspaces. In this section, we introduce concepts and notation which simplify this task.

The axes in d-dimensional space are denoted by $x_1, x_2, ..., x_d$. Let us consider a point $p = (p_1, p_2, ..., p_d)$ in d-dimensional space. A *region in d-dimensional space having p as the origin* is a set of points in that space, such that for every point $q = (q_1, q_2, ..., q_d)$ in the region the constraint $q_i C_i p_i$ holds for each dimension i, where C_i is one of $\leqslant, =, \geqslant$ or a null constraint meaning that there is no constraint on the ith coordinate of the points in the region. We will use $\leftarrow, -, \rightarrow, \leftrightarrow$ to denote each of the four constraints $\leqslant, =, \geqslant$ and the null constraint, respectively. To denote a region we use the coordinates of the origin and add these symbols on top of each coordinate to show the type of constraint in that dimension. If there is no constraint for the ith dimension of the region, the ith coordinate of the origin can have any value. We use 0 for this coordinate for simplicity.

Example 7. *The region R containing all the points in the second quadrant in the plane, such that $x_1 \leq -1$ and $x_2 \geq 1$ is denoted by $(\overleftarrow{-1}, \overrightarrow{1})$ (Figure 3.9).*

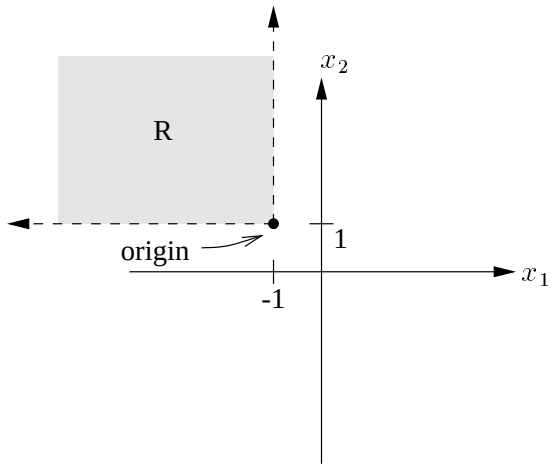

Figure 3.9: The region R with two open directions in 2-dimensional space.

For a region R, if R contains points with infinitely large positive (negative) values in the ith dimension, the region is said to be *open in the positive (negative) side of the ith axis* and the positive (negative) direction of the ith axis is said to be an *open direction for R* and will be denoted by $\overrightarrow{x_i}$ ($\overleftarrow{x_i}$). It is worth mentioning that a region is defined

by the origin and the set of open directions. The negative direction of the first axis and the positive direction for the second axis are open in the region shown in example 7, so this region has two open directions. We can consider the d-dimensional space as a region with origin $(0, 0, ..., 0)$ and call it *the universal region*. This region has $2d$ open directions (one positive and one negative direction for each of the d axes) and can be denoted by $(\overleftrightarrow{0}, \overleftrightarrow{0}, ..., \overleftrightarrow{0})$.

A region S is said to be *a subregion of a region* R if the origin of S is in R and the set of open directions of S is a subset of the open directions of R. We also say that two regions R and S are *disjoint* if they have disjoint sets of open directions and neither origin is inside the other region. The generalization to more than two regions is analogous. The complement of a region R is a region, denoted by $\overline{R}$, such that the origin of $\overline{R}$ is the same as the origin of R and the set of of open directions of $\overline{R}$ is the complement of the set of open directions of R relative to the set of open directions of the universal region.

Example 8. *The complement of the region* $R = (\overleftarrow{-1}, \overrightarrow{1})$ *is the region* $(\overrightarrow{-1}, \overleftarrow{1})$.

There are points in the universal region that belong to neither R nor $\overline{R}$. For example, the point $(0, 2)$ is not in R or $\overline{R}$ in the previous example.

For a point $p = (p_1, p_2, ..., p_d)$ and a subset U of the set of open directions of the universal region, we define a function *move(P, U)* which generates a new point $p' = (p'_1, p'_2, ..., p'_d)$ such that for the ith dimension, $1 \leqslant i \leqslant d$, $p'_i = p_i$ if U does not contain either the positive direction or the negative direction of the ith axis or if U contains both of them. If S contains only the positive direction then $p'_i = p_i + 1$ and if it contains only the negative direction of the ith axis, $p'_i = p_i - 1$.

3.3.2 Constructing an optimum MIRS

In this section we characterize graphs supporting an optimum $\langle 1, d \rangle$-MSLIRS with dynamic cost links. We can consider the assignment of d-dimensional labels to the vertices of a graph as assigning corresponding points in d-dimensional space to each vertex. We will use a vertex and its corresponding point interchangeably.

We start with an observation about boundary sets.

Observation 3. *For a set of points Q in d-dimensional space and a boundary set B of Q, any d-dimensional interval I containing all points in B contains all points of Q. This is true because if $p = (p_1, p_2, ..., p_d)$ is an arbitrary point in Q, then for each dimension i, $1 \leqslant i \leqslant d$, there is a minimum point m_i and a maximum point M_i in Q (m_i and M_i can be the same) such that $m_i \leqslant p_i \leqslant M_i$. Since I contains these minimum and maximum points, the ith dimension of I covers the ith dimension of p and so I contains p. Therefore, I contains all of the points in Q.*

In the following lemma, we use this observation to prove a restriction on the number of non-articulation points in a graph supporting an optimum $\langle 1, d\rangle$-MSLIRS with dynamic cost links.

Lemma 7. *Any connected non-trivial graph G with more than $2d$ non-articulation points cannot support an optimum $\langle 1, d\rangle$-MSLIRS with dynamic cost links.*

Proof. If G has an optimum $\langle 1, d\rangle$-MSLIRS with dynamic cost links, the points corresponding to the labels of the vertices in G will have a boundary set B of at most $2d$ points. Since G has more than $2d$ non-articulation points, we have at least one non-articulation point, say v, such that the point corresponding to v is not in B.

G is a connected and non-trivial graph, so v has at least two adjacent vertices. We let u be an arbitrarily chosen neighbor of v. Recalling that the links have dynamic costs, and there must be a labeling of the links of G for any assignment of costs. We can consider a case in which the cost of the link (v, u) is 1 and the cost of any other link adjacent to v is arbitrarily large, say M (where M is at least n^2). The cost of any other link of the graph is set to be 1 (Figure 3.10).

Since v is a non-articulation point, the shortest path from v to any other vertex in G must go through the link (v, u). To prove this, let us assume that the shortest path from v to some other vertex t in G goes through a neighbor z of v such that $z \neq u$. Since v is not an articulation point, if we remove v there exists a path connecting u and z. The cost of this path is fewer than M because we have fewer than n^2 links of cost 1, and we know $M \geqslant n^2$. Therefore, the path going from v to u, to z, and then z to t, has a smaller cost than the path going from v to t through the edge (v, z). This is a contradiction because the path from v to t passing through z is a shortest path. If v instead were an articulation point, this argument would not work, because by removing v the graph becomes disconnected.

This argument together with Observation 3 shows that the interval I assigned to (v, u) contains all other points including the points in the boundary set B. Therefore, I contains v, which contradicts the fact that the IRS is strict. ■

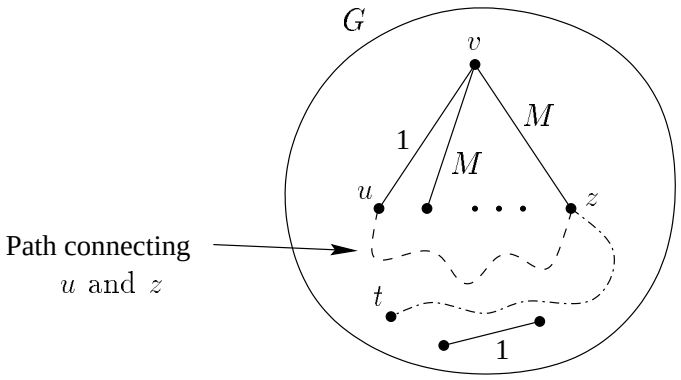

Figure 3.10: Costs assigned to the links of the graph G. Here, we consider a case in which M is at least n^2.

In the following sections, we will show that the necessary condition stated in Lemma 7 is also a sufficient condition for a graph to support an optimum $\langle 1, d\rangle$-MSLIRS with dynamic cost links. We will first give an algorithm to assign labels to the vertices of the graph. Then, we will show that with those labels assigned to the vertices, and for assignment of any costs to the links, one can always find a suitable set of labels for the links so that the graph supports an optimum $\langle 1, d\rangle$-MSLIRS.

Labels of Vertices

We consider a graph G supporting an optimum $\langle 1, d\rangle$-MSLIRS with dynamic cost links. There is a labeling of the vertices of G such that for any set of costs assigned to the links of the graph, one can always find a suitable set of labels for the links. By Lemma 7, G has at most $2d$ non-articulation points. In this section we show how to find such a labeling for the vertices of any graph having at most $2d$ non-articulation points.

We will use a structure, which we call *the block tree of a graph*, in order to find such a labeling of vertices. This structure defines an ordering of the vertices of the graph, based on which we will assign the labels to the vertices. By using this ordering we will assign labels of vertices such that all the non-articulation points will be in a boundary set and articulation points are placed so that they are not contained in any boundary set. In other words, when we assign a point in d-dimensional space, this assignment

is done in a way that in some direction (one of the $2d$ directions of the d-dimensional space) this point is a minimum or a maximum point and we will not place any other point beyond this one in that specific direction.

The *block tree of a graph* G, which is denoted by $BT(G)$, is a structure in which each block of the graph G is represented by a vertex. $BT(G)$ also has one vertex for each articulation point in G. Whenever there is no ambiguity, we will use the same name for a block in G and its corresponding vertex in $BT(G)$ and also for any articulation point in G and its corresponding vertex in $BT(G)$. If and only if an articulation point v is in a block B of G, the corresponding vertices in $BT(G)$ will be joined by an edge.

Figure 3.11 depicts an example of a block tree. The graph indicated in this example, has four blocks B_0, B_1, B_2, B_3 and two articulation points u_1 and u_2. The vertex u_1 is connects B_0, B_1 and B_2 in G, so in the block tree $BT(G)$ the vertex representing u is connected to the vertices representing B_0, B_1 and B_2.

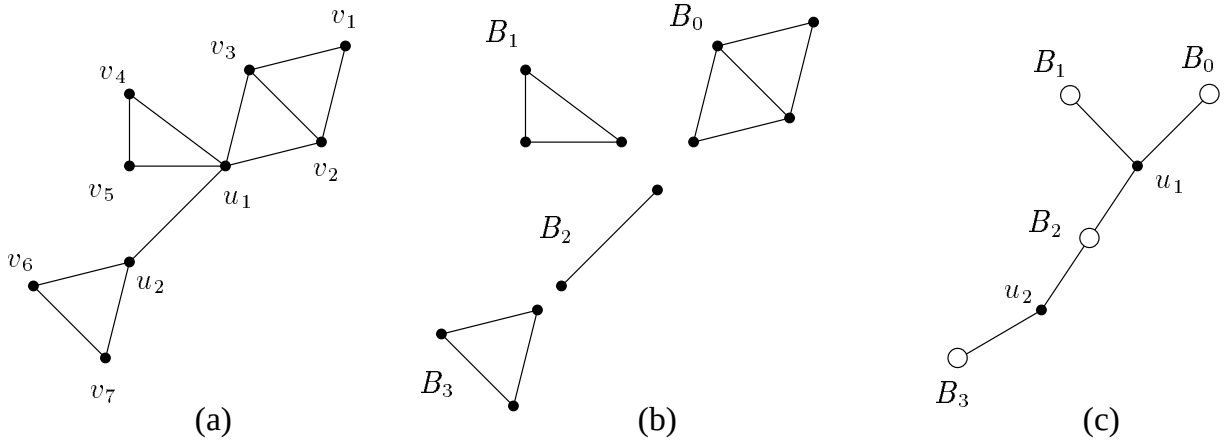

Figure 3.11: (a) A graph G (b) blocks of G and (c) the block tree of G.

It is a trivial task to verify that the block tree of a graph G is a tree (otherwise the blocks of the graph form a cycle, which is impossible).

As mentioned earlier, we will use this tree (the block tree $BT(G)$) to assign labels to the vertices of the graph G. We can consider $BT(G)$ as a tree rooted at an arbitrary block B_0. To assign labels in d-dimensional space to the vertices of a graph G, we will assign each vertex v a region in d-dimensional space. The label of a vertex v will then be the origin of the region assigned to v.

Intuitively, we will initially assign the whole d-dimensional space to the root B_0 of the block tree. Then we subdivide the regions assigned to the root of each subtree among the vertices in that subtree. For a node v which is the root of more than one subtrees the regions assigned to different subtrees will be disjoint. This property will allow us to assign intervals to the links of the graph without any conflicts, as we will see later.

Formally, starting at the root B_0 (an arbitrary block) of the block tree, we let B_0, B_1, B_2, ... be a topological sort of the blocks in $BT(G)$. We denote by $v_1, v_2, ..., v_\alpha$ ($\alpha \leqslant 2d$) the list of all non-articulation points in B_0 followed by the set of non-articulation points in B_1 and so on. For each non-articulation vertex, v_i, ($1 \leqslant i \leqslant \min\{\alpha, d\}$) we assign an open direction $OD(v_i)$ which is the positive direction of the ith axis. If $\alpha > d$, for each v_i, ($d < i \leqslant \alpha$) we also assign an open direction $OD(v_i)$ which is the negative direction of the $(i-d)$-th axis.

Example 9. *The graph depicted in Figure 3.11 has 7 non-articulation points, so if we want this graph to have an optimum $\langle 1, d\rangle$-MSLIRS with dynamic cost links, d must be at least 4. Recalling that the axes are denoted by x_1, x_2, x_3, x_4 then $OD(v_1), OD(v_2), ...,$ $OD(v_7)$ will be $\overrightarrow{x_1}, \overrightarrow{x_2}, \overrightarrow{x_3}, \overrightarrow{x_4}, \overleftarrow{x_1}, \overleftarrow{x_2}$ and $\overleftarrow{x_3}$, respectively.*

Now that each non-articulation point has an open direction, we assign a set of open directions to each articulation point and each block in $BT(G)$ as follows: the set of open directions assigned to an articulation point v is the union of the open directions of all non-articulation points in a subtree of $BT(G)$ rooted at v. We denote this set by $OD(v)$.

Similarly, the set of open directions assigned to a vertex B in $BT(G)$ representing a block, which is denoted by $OD(B)$, is the union of the open directions of all non-articulation points in a subtree of $BT(G)$ rooted at B. Obviously, this subtree includes all non-articulation points in the block B.

Example 10. *For the graph G denoted in Figure 3.11 (a), the block tree is depicted in Figure 3.11 (b). Here, B_0 is the root of the block tree. In this graph, $OD(u_1) = OD(B_2) = \{\overleftarrow{x_2}, \overleftarrow{x_3}\}$ which is the same as $OD(v_6) \cup OD(v_7)$ (the non-articulation points in the subtree rooted at B_2 or u_1).*

The next step is to assign an origin to each set of open directions associated with a vertex in $BT(G)$. The origin of the region assigned to v (any vertex in $BT(G)$) will be

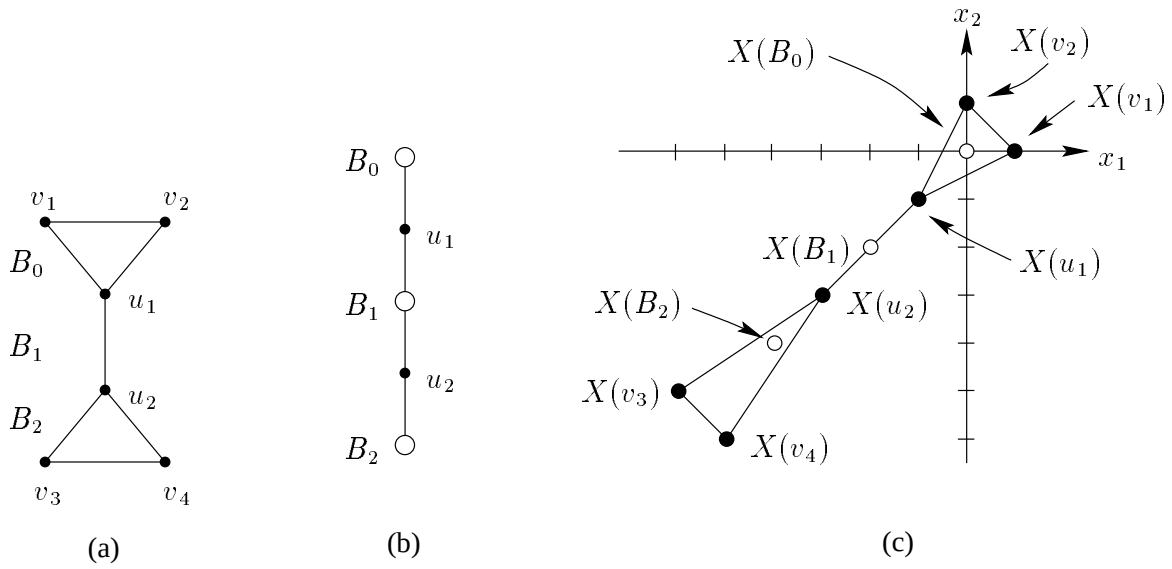

Figure 3.12: (a) A graph G (b) The block tree of G rooted at B_0 (c) The origin of each region assigned to each block and each vertex in G.

denoted by $X(v)$. This will be used for calculating the origin of each non-articulation point later.

We start with block B_0 and let $X(B_0) = (0, 0, ..., 0)$. If G has $2d$ non-articulation points, recalling that we have already assigned all $2d$ open directions to B_0, the region assigned to B_0 would be the universal region, $(\overset{\leftrightarrow}{0}, \overset{\leftrightarrow}{0}, ..., \overset{\leftrightarrow}{0})$. To compute the origin of the region assigned to u, a child of a vertex v with a known origin, we let $X'(v, u) =$ move$(X(v), OD(v))$. Then we let $X(u) =$ move$(X'(v, u), OD(u))$. This is the origin of the region associated with u. Since the root of the tree $BT(G)$ has a known origin, by repeating this step every vertex in $BT(G)$ will eventually have an origin.

If v is a non-articulation point in a block B of G, the origin of the region assigned to v (which has exactly one open direction), $X(v)$, is computed as follows: we first let $X'(B, v) =$ move$(X(B), OD(B))$. Then we let $X(v) =$ move$(X'(B, v), OD(v))$ which is the origin of the region assigned to v.

Example 11. *In the graph G depicted in Figure 3.12 the region associated with B_0 is $(\overset{\leftrightarrow}{0}, \overset{\leftrightarrow}{0})$. To find $X(v_1)$ we move $X(B_0) = (0, 0)$ in the direction of $OD(B_0)$ (which includes all four directions) and obtain $(0, 0)$. Then this point is moved in the direction of $OD(v_1)$ which is $\overrightarrow{x_1}$. Therefore, $X(v_1) = (1, 0)$.*

In the optimum $\langle 1, d\rangle$-MLIRS defined on G, we let the label assigned to each vertex v, denoted by $L(v)$, be the same as the origin of the region assigned to that vertex ($L(v) = X(v)$). Figure 3.13 is the pseudo code for labeling the vertices of a graph G (the Vertex Labeling algorithm or VL algorithm for short). We can verify that this algorithm can be executed in $O(n + m)$ time (where m is the number of edges in the graph).

Before showing how to find the labels of links for a given set of link costs, we review properties of the labels assigned to the vertices.

Observation 4. *For a block B in G, we let $X'(B) = (a_1, a_2, ..., a_d)$ be the point resulting from moving $X(B)$ in the direction of $OD(B)$. If v is a non-articulation point in B, then $L(v) = (b_1, b_2, ..., b_d)$ where $b_i = a_i$ for all i, $1 \leqslant i \leqslant d$ except for one dimension j such that $OD(v) = \overleftarrow{x_j}$ or $OD(v) = \overrightarrow{x_j}$. In this dimension $b_j = a_j + 1$ or $b_j = a_j - 1$ based on the direction of $OD(v)$.*

Lemma 8. *If v is a vertex in a block B or in the subtree of $BT(G)$ rooted at B, the region R_v assigned to v by the VL algorithm, is a subregion of the region R_B assigned to B.*

Proof. First, let us consider a point p in a region R. We let S be a subset of open directions of R. The point $p' =$ move(p, S) is a point in R. We can repeat this with another subset of open directions of R as many times as we want. The final point will still be in R. This is exactly what happens to the origin of R_B in the VL algorithm, so $X(v)$ is in R_B.

The set of open directions assigned to any vertex in the subtree rooted at B (a non-articulation vertex as well as an articulation vertex or a block) is a subset of open directions of B. The origin of the region R_v is in R_B and the set of open directions of R_v is a subset of the set of open directions of R_B. Therefore, R_v is a subregion of R_B. ■

If R_v is the region assigned to a vertex v in $BT(G)$, the previous lemma shows that all vertices in the subtree of $BT(G)$ rooted at v are in R_v. With an argument similar to that of the proof of Lemma 8 we can verify the following lemma.

Lemma 9. *Any vertex not in the subtree of $BT(G)$ rooted at v is in the region $\overline{R_v}$.*

Lemma 10. *For $e = (u, v)$ an edge in block B and z a vertex contained in a block $B' \neq B$ which is in the subtree of $BT(G)$ rooted at B, if the shortest path from u to z goes through e, then there is a shortest path from u to any other vertex t in the subtree of $BT(G)$ rooted at B which goes through e.*

Proof. To verify this lemma, we notice that any shortest path going from u to any vertex in the subtree of $BT(G)$ rooted at B not including the node representing B must go through the articulation point w connecting B to the rest of that subtree. Since the shortest path from u to z (which is one of those shortest paths) goes through e, there is a shortest path from u to w going through e. This path can be expanded to a shortest path for any other vertex t by just adding the shortest path from w to t. ■

In the following section, we show how to assign intervals to the links for any set of link costs.

Labels of Links

In this section we show that for a given graph G and the labels assigned to the vertices of G using the VL algorithm, introduced in Section 3.3.2, we can always find labels for the links of G for any set of link costs. By this labeling of links, a message from any source vertex to any destination vertex in G will be routed on a shortest path.

First, we show that if we consider the non-articulation vertices in one block, we can always find labels for the links for any set of costs assigned to the links, so that the messages are routed on shortest paths.

Lemma 11. *With the labels assigned by the VL algorithm, for any subset C of the non-articulation vertices in a block B, we can always find a d-dimensional interval containing the vertices in C and no other non-articulation vertex in B.*

Proof. We assume that $X'(B)$ is the point resulting from moving $X(B)$ in the direction of $OD(B)$. Without loss of generality, we can assume that $X'(B) = (0, 0, ..., 0)$

(otherwise we can shift every label by $-X'(B)$). By Observation 4 one can verify that the label of each non-articulation vertex v in B is of the form $(0,0,...,1,...,0)$ or $(0,0,...,-1,...,0)$ (exactly one coordinate is 1 or -1 and the rest of coordinates are all 0).

We let m_i be -1 if there is a vertex in C having -1 as its ith coordinate and 0 otherwise. Similarly, M_i will be set to 1 if there is a vertex in C having 1 as the ith coordinate and 0 otherwise. Obviously $m_i \leqslant 0 \leqslant M_i$.

For any non-articulation point v in C (with $L(v) = (b_1, b_2, ..., b_d)$) and for any dimension i, $1 \leqslant i \leqslant d$, we have $m_i \leqslant b_i \leqslant M_i$. As a consequence, the d-dimensional interval $I = [m_1..M_1, m_2..M_2, ..., m_d..M_d]$ contains all the vertices in C.

We define $OD_C = \cup OD(v)$ for all $v \in C$. For any non-articulation vertex u in $B - C$, $OD(u) \notin OD_C$. Therefore, if $L(u) = (b_1, b_2, ..., b_d)$, there is a dimension j such that $b_j < m_j$ or $b_j > M_j$ (this is the direction which belongs to $OD(u)$ but not to OD_C). Hence, u is not in I and thus I contains exactly the vertices of B which are in C. ■

The next step is to generalize this argument to the case in which C contains articulation points of B, not including the parent of B in $BT(G)$ (if it has any).

Lemma 12. *With the labels assigned by the VL algorithm, for any subset C of the vertices in a block B which does not include the parent of B in BT(G), we can always find a d-dimensional interval containing exactly the vertices in C and no other vertex.*

Proof. For any articulation point z, we let $t(z)$ denote the number of non-articulation points in the subtree(s) of $BT(G)$ rooted at z. We also let B' be the set resulting from replacing each articulation point z in B with the a set of $t(z)$ new non-articulation points $z_1, z_2, ..., z_t$ ($B' = B - \{z\} \cup \{z_1, z_2, ..., z_t\}$ for any articulation point z in B). These new vertices ($\{z_1, z_2, ..., z_t\}$) all together represent the articulation point z in B. We run the VL algorithm one more time on the vertices in B' assuming that the vertices z_1, z_2, ..., z_t are considered consecutively in the order of non-articulation points and they get the same open directions as the open directions assigned to the non-articulation points in the subtree(s) of $BT(G)$ rooted at z.

Lemma 11 shows that for any subset C' of B' there is an interval containing exactly the vertices in C'. If C contains the articulation point z, we let $C' = C - \{z\} \cup \{z_1, z_2, ..., z_t\}$.

By Lemma 11 there is an interval I containing exactly the vertices in C'. If I contains all of the points $z_1, z_2, ..., z_t$, it is easy to show that it will also contain z. Hence, this interval contains all the vertices in C. On the other hand, any vertex v in $B - C$ is in $B' - C'$. It means if I contains a vertex v in $B - C$, it also contains a point from $B' - C'$ which is impossible by Lemma 11. ■

Example 12. *In graph G shown in Figure 3.12, the origin of the region associated with B_2 is $(-4, -4)$ and moving this point in the direction of $OD(B_2)$ results in the point $(-5, -5)$, because $OD(B_2)$ contains the negative direction of both axes. If we consider this point as the origin, the coordinates of $X(v_3)$ and $X(v_4)$ (v_3 and v_4 are non-articulation points in B_2) are $(-1, 0)$ and $(0, -1)$ respectively. If $C = \{v_3, v_4\}$ then the interval covering C would be $I = [-1..0, -1..0]$.*

In the VL algorithm, each block has at most one articulation point as its parent in $BT(G)$. For a block B and v the vertex in $BT(G)$ which is the parent of B, Lemma 8 shows that all the vertices in the subtree of $BT(G)$ rooted at B are contained in the region R_B. Lemma 9 states that any other vertex is in $\overline{R_B}$.

Lemma 13. *If I is an interval in the region $\overline{R_B}$ containing the articulation point v, we can find another interval I' such that I' contains all the vertices in the subtree rooted at v and the same set of points in $\overline{R_B}$ as I. Also, if I is an interval in R_B containing v, we can find another interval I' such that I' contains all the vertices which are not in the subtree rooted at v and the same set of points in R_B as I.*

Proof. If we repeatedly move $L(v)$ (which is $X(v)$) in the direction of $OD(B)$ and if I' denotes the interval that contains the resulting point, we can verify that I' contains the same set of points in $\overline{R_B}$ as I. By moving $L(v)$ sufficiently often in the direction of $OD(B)$, the new interval I' will also contain all points in R_B (any point with finite coordinates which is in R_B), which completes the proof. The other claim can be proved similarly. ■

Now, let us assume that we are given the costs of the links in a graph G and want to find the labels for the links based on the labels given by the VL algorithm to the vertices of G. The following lemma illustrates how to do this.

Lemma 14. *For any assignment of costs to the edges of a graph G, and with labels assigned to the vertices of the graph by the VL algorithm, we can always find suitable intervals for the links so that the result is an optimum $\langle 1, d\rangle$-MSLIRS.*

Proof. First, we will consider a link $e = (u, v)$ and the set of vertices S_e reachable (by shortest paths) through e. The link e is in a block, say B, of G. We let $Q_1 = B \cap S_e$, that is, Q_1 is the subset of vertices in B that are contained in S_e. Let us consider a vertex z which is a vertex in the subtree of $BT(G)$ rooted at B. If z is not in B but is contained in S_e, by Lemma 10, S_e also contains all the vertices in the subtree of $BT(G)$ containing z (we denote the vertices in this subtree by the set Q_2). Finally, if z is not in a child block of B, S_e must contain all the vertices that are not children of B (we denote the set containing all these vertices by Q_3).

Lemma 12 shows that we can always find an interval covering exactly the vertices in Q_1. If there is any point in Q_2 (or Q_3) then the articulation point joining B to the vertices in Q_2 (Q_3 respectively) must also be in Q_1. This is because this articulation point is the only vertex connecting B to the child subtree (or the vertices of G that are not contained in the subtree rooted at B) and hence the only way to reach those vertices. Lemma 13 shows that we can always find an interval containing the same set of points in Q_1 at the previously assigned interval, and covering all the points in Q_2 (Q_3). This completes the proof. ■

Example 13. *In the graph G of Figure 3.12, let us assume that the cost of the edge (v_1, v_2) is extremely large and the cost of any other edge is 1. The interval assigned to the edge $e = (v_1, u_1)$ should contain all the vertices, except v_1. The interval $[-1..0, -1..1]$ contains $Q_1 = \{v_2, u_1\}$. Therefore we can find another interval which contains all vertices in the subtree of $BT(G)$ rooted at u_1 (Lemma 13). This interval is $[-6..0, -6..1]$.*

Now we can easily prove the main result of this section.

Theorem 10. *A graph G has an optimum $\langle 1, d\rangle$-MSLIRS with dynamic cost links, if and only if G has at most $2d$ non-articulation vertices.*

Proof. Lemma 7 states that any graph having more than $2d$ non-articulation points can not support an optimum $\langle 1, d\rangle$-MSLIRS with dynamic cost links. By Lemma 14 if

a graph G has at most $2d$ non-articulation points we can always find a fixed labeling for the vertices such that for any costs assigned to the links, we can find intervals for each link to support an optimum $\langle 1, d\rangle$-MSLIRS. ■

Corollary 3. *The class of graphs supporting an optimum $\langle 1, d\rangle$-MSLIRS with dynamic cost links is a strict subset of the class of graphs supporting an optimum $\langle 1, d+1\rangle$-MSLIRS with dynamic cost links.*

In this chapter we completely characterized the class of networks supporting three variants of MIRS: $\langle 1, d\rangle$-MLIRS, $\langle 1, d\rangle$-MSLIRS and optimum $\langle 1, d\rangle$-MSLIRS with dynamic cost links. We showed that increasing the number of dimensions in all these routing schemes makes the routing scheme more powerful. These are the only known general results for MIRS (for more than one dimension). In the following chapter, we will study the quality of routing problem for MIRS.

Algorithm VertexLabeling(G, BT);
Input: G (a simple connected and undirected graph).
BT (the block tree of G).
Output: L (an array containing a d-dimensional label for each vertex of G).

begin
let $k \leftarrow$ number of non-articulation points in G;
let $d \leftarrow \lceil k/2 \rceil$;
let $B_0, B_1, ...$ be the DFS order of blocks in BT;
let v_1, v_2, ..., v_k be the order of non-articulation vertices
in $B_0, B_1, ...$ respectively;
for each v_i, $1 \leqslant i \leqslant d$
let $OD(v_i) \leftarrow \overrightarrow{x_i}$;
for each v_i, $d < i \leqslant k$
let $OD(v_i) \leftarrow \overleftarrow{x_{i-d}}$;
for each vertex v of BT
let $OD(v) \leftarrow$ empty set;
for each non-articulation vertex u in the subtree of BT rooted at v
let $OD(v) \leftarrow OD(v) \cup \{OD(u)\}$;
let $X(B_0) \leftarrow (0, 0, ..., 0)$;
for each vertex v in BT such that $X(v)$ is already known
for each child c of v
let $Y \leftarrow \text{move}(X(v), OD(v))$;
let $X(c) \leftarrow \text{move}(Y, OD(c))$;
for each block B in G
for each non-articulation point v in B
let $Y \leftarrow \text{move}(X(B), OD(B))$;
let $X(v) \leftarrow \text{move}(Y, OD(v))$;
for each vertex v in G
let $L(v) \leftarrow X(v)$;
end;

Figure 3.13: Algorithm for labeling the vertices of a given graph G.

Chapter 4

Bounds on the length of routing paths

A common way to measure the efficiency of a routing scheme is in terms of the maximum length of a path which a message traverses. We will say that a routing scheme has upper bound B_u if for every graph the maximum path length is at most B_u, and lower bound B_l if there exists a graph G for which the maximum path length is at least B_l. For interval routing, there is a trivial upper bound of $2D$ on the maximum length of the routing path, where D is the diameter of the underlying graph [SK82]. The problem of finding a lower bound for routing schemes has been studied for several years and there have been many improvements for this bound. Tse and Lau have proved a $2D-3$ lower bound which is very close to the best known upper bound [TL97a].

This quantity has also been studied for LIRS by Eilam *et al.* [EMZ99]. They construct a graph and prove that using any LIRS the length of the longest path traversed by a message in this graph is in $\Omega(D^2)$. They also present a family of graphs for which this lower bound is tight, but the problem of finding an upper bound for this quantity remains open even for most known classes of graphs.

In this chapter, we introduce some upper and lower bounds for different multi-dimensional schemes. In Section 4.1 we prove a lower bound of $\frac{3}{2}D$ on the length of the longest routing path under any $\langle k, d\rangle$-MLIRS. Then, in Section 4.2 we illustrate a lower bound of $\Omega(D^2/d)$ for $\langle 1, d\rangle$-MSLIRS and $\langle 1, d\rangle$-MLIRS. Finally, in Section 4.3 we prove an upper bound of $O(D)$ for $\langle 1, d\rangle$-MLIRS on interval graphs.

4.1 A lower bound for $\langle k, d \rangle$-MLIRS

Tse and Lau have proved a $\frac{3}{2}D$ lower bound on the length of the longest routing path for $\langle 2, 1 \rangle$-MLIRS where D is the diameter of the network [TL95]. We generalize their result to the d-dimensional case in this section and prove a lower bound of $\frac{3}{2}D$ for $\langle k, d \rangle$-MLIRS.

4.1.1 Preliminary results

The labels assigned to the nodes of a network in a d-dimensional MIRS can be considered as points in d-dimensional space. Recalling this fact, in this section we show that for any $k \in \mathbb{N}$, there is a set of points P in d-dimensional space, such that a subset S of P cannot be covered with k intervals without covering some point in $P - S$. In the next section, we will use this result to construct a graph G and show that, no matter how we assign d-dimensional labels to the vertices of this graph, if the longest routing path is shorter than $\frac{3}{2}D$ in G, there is an edge which requires more than k intervals. In other words, for any $\langle k, d \rangle$-MLIRS on G the length of the longest routing path is at least $\frac{3}{2}D$.

Let us consider a set of points $x_1, x_2, ..., x_{2k+1}$ in one-dimensional space such that $x_i < x_{i+1}$, for $1 \leqslant i \leqslant 2k$. Clearly, in order to cover exactly the points $x_1, x_3, ..., x_{2k+1}$ in this set we need to have at least $k + 1$ intervals. The following definition illustrates how we can generalize this technique to two dimensions.

Definition 18. *A set M of disjoint points in the plane is called* nonincreasing *(*nondecreasing*), if for any two points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ in M, $x_1 < x_2$ implies $y_1 \geqslant y_2$ ($y_1 \leqslant y_2$). If we sort these points in increasing order of their first coordinates, we will have a* nonincreasing (nondecreasing) path. *The* length *of a path is the size of the corresponding set of points. The point p with the largest first coordinate is called the* endpoint *of a path W and we say that the path W* ends at the point p.

Definition 19. *A set M of points in the plane is called* monotone, *if M is nonincreasing or nondecreasing. Similarly, a nonincreasing or nondecreasing path is called a* monotone path.

We are looking for a set of points in the plane which have a subset that cannot be covered by k intervals. More precisely, we define a k-coverable subset S of a set of points P as follows.

Definition 20. *A subset S of a set of points P in the d-dimensional space is called* k-coverable *if there exist k intervals covering the points in S without containing any other point in $P - S$. Let us consider a graph $G = (V, E)$, a d-dimensional MIRS defined on G and, a subset U of the vertices of G. We denote by $P(V)$ the set of points corresponding to the vertices in V and by $P(U)$ the set of points which corresponds to the vertices in U. The set of vertices U is called* k-coverable *if $P(U)$, which is a subset of $P(V)$, is k-coverable.*

Let us consider a monotone path Q defined by the points $p_1, p_2, \ldots, p_{2k+1}$. Clearly, any 2-dimensional interval containing two points p_i and p_j $(1 \leqslant i < j \leqslant 2k+1)$ will contain all points p_m, $i \leqslant m \leqslant j$). Therefore, in order to cover the points $p_1, p_3, \ldots, p_{2k+1}$ without covering $p_2, p_4, \ldots, p_{2k}$, we will need at least $k+1$ intervals (one interval for each point). Thus, we have proved the following lemma.

Lemma 15. *A monotone path of length $2k+1$ has a subset which is not k-coverable.*

The next step is to show that for any $k \in \mathbb{N}$ we can find a set of points P in the plane which contains a monotone path of length at least k. The following lemma shows that such a set of points always exists (for significantly large $|P|$). This lemma was originally proved by Erös and Szekeres [ES35] but since we are going to generalize it, we give the proof.

Lemma 16. *For any set P consisting of at least $(k-1)^2 + 1$ disjoint points in the plane, there exists a subset M of P such that $|M| \geqslant k$ and M is monotone.*

Proof. Each monotone path in the plane is either nondecreasing or nonincreasing or both. For any point $p \in P$, we let $L_+(p)$, $(L_-(p))$ denote the length of the longest nondecreasing (nonincreasing) path ending at point p. We consider two points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ and without loss of generality assume that $x_1 \leqslant x_2$. Obviously we have either $L_+(p_2) > L_+(p_1)$ or $L_-(p_2) > L_-(p_1)$. This means that if we assign

each point p an ordered pair $(L_+(p), L_-(p))$, then for any two distinct points p_1 and p_2 we cannot have $(L_+(p_1), L_-(p_1)) = (L_+(p_2), L_-(p_2))$.

On the other hand, there exist at most $(k-1)^2$ pairs (a, b) such $a, b \in \mathbb{N}$, $1 \leqslant a < k$ and $1 \leqslant b < k$. If we have $(k-1)^2 + 1$ or more disjoint points in the plane, since the pair $(L_+(p), L_-(p))$ assigned to each point p is unique, we will have at least one point p with $L_+(p) \geqslant k$ or $L_-(p) \geqslant k$. This means p is the endpoint of a monotone path of length at least k. ■

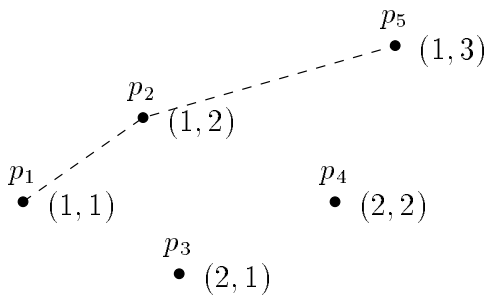

Figure 4.1: A set of five points in the plane and a nondecreasing path of length three.

For example in the set of points illustrated in Figure 4.1 the pair incident to each point p indicates $(L_+(p), L_-(p))$. This example is in two-dimensional space, so if we have five points, we will have a monotone path of length three. The point p_5 is the endpoint of a nondecreasing path of length three in this example.

We have now shown that for any $k \in \mathbb{N}$, there is a set of points P in the plane with a subset S which is not k-coverable. To extend this result to higher dimensions we first need to define monotonicity in higher dimensions.

Definition 21. *In d-dimensional space, we let Δ_i be the plane passing through the first axis and the ith axis ($2 \leqslant i \leqslant d$). A set M is called* monotone in d-dimensional space *if the projection of all points in M on any $\Delta_i, 2 \leqslant i \leqslant d$, produces a monotone set in that plane. If we sort these points in M in increasing order of first coordinates we will have a* monotone path in d-dimensional space.

Observation 5. *It is easy to verify that if M is a monotone set, then projecting points in M to the plane passing through any two axes produces a monotone set in that plane.*

Now we can extend Lemma 16 to higher dimensions (this result has been independently proved by Heinrich-Litan [HL00]).

Lemma 17. *For any set P consisting of $\mathcal{F}(k,d) = (k-1)^{2^{d-1}} + 1$ disjoint points in d-dimensional space, there exists a subset M of P such that $|M| \geqslant k$ and M is monotone.*

Proof. The proof is a trivial extension of the idea used in the proof of Lemma 16. Let us consider a monotone path in d-dimensional space. By the definition of a monotone path, the first coordinates of the points in M are in increasing order. For each other coordinate $i, 2 \leqslant i \leqslant d$, the order of points can be either nonincreasing or nondecreasing. Since we have $d-1$ coordinates other than the first one, and the points can be in a nonincreasing or nondecreasing order in each of these $d-1$ coordinates, we can have 2^{d-1} types of monotone paths.

Now let us consider a point p in P. For each of the 2^{d-1} different monotone path types, we can assign a number to p indicating the length of the longest monotone path of that type ending at p. We let $L_0(p), L_1(p), \ldots, L_{2^{(d-1)}-1}(p)$ denote these numbers and assign a tuple $(L_0(p), L_1(p), \ldots, L_{2^{(d-1)}-1}(p))$ to each point p in P. For any two points p_1 and p_2 in P we cannot have $(L_0(p_1), L_1(p_1), \ldots, L_{2^{(d-1)}-1}(p_1)) = (L_0(p_2), L_1(p_2), \ldots, L_{2^{(d-1)}-1}(p_2))$.

There are at most $(k-1)^{2^{d-1}}$ tuples of the form $(a_0, a_1, \ldots, a_{2^{d-1}-1})$ such that $a_i < k$ $(0 \leqslant i \leqslant 2^{d-1}-1)$. Since each point in P has a unique tuple, any set consisting of at least $(k-1)^{2^{d-1}} + 1$ points has a point p such that $L_i(p) \geqslant k$ for some $i, 0 \leqslant i \leqslant 2^{d-1}$. This point is the endpoint of a monotone path of length at least k. ■

4.1.2 Constructing the graph

In this section we construct a graph G which for any $\langle k,d\rangle$-MLIRS has a longest routing path of length at least $\frac{3}{2}D$, where D is the diameter of G. The graph we are going to construct is such that a subset U of its vertices (using the result of previous section) is not k-coverable. We will show that there is an edge in the graph which must cover all the vertices in U so that the length of the routing path is less than $\frac{3}{2}D$, which is not feasible.

We construct graphs $G(r, w, t)$ for integer values r and w, $1 \leqslant r < w$, and for a non-negative integer t which determines the diameter of the graph. The vertices of $G(r, w, t)$ include a set of vertices $X = \{u_1, u_2, ..., u_w\}$. We can partition the set X into two parts such that the first part has exactly r elements in $\alpha = \binom{w}{r}$ different ways. We let Y_l, $1 \leqslant l \leqslant \alpha$ be the set of elements in the first part and Z_l be the set of elements in the second part in the lth way of partitioning the set X.

The graph $G(r, w, t)$ also has a vertex a_l, $1 \leqslant l \leqslant \alpha$, for each way of partitioning X into two parts as described above. Each vertex a_l has exactly two neighbors b_l and c_l. The vertex b_l, $1 \leqslant l \leqslant \alpha$, is connected to each vertex in Y_l by a path of length $t + 1$. Also, the vertex c_l, $1 \leqslant l \leqslant \alpha$, is connected to each vertex in Z_l through a path of length $t + 1$.

We consider the messages originating from a_l, $1 \leqslant l \leqslant \alpha$, and going to the vertices u_i, $1 \leqslant i \leqslant w$. Each such message must be routed through b_l or c_l since these are the only neighbors of a_l. If a message originating from a_l and going towards a node in Y_l is not routed through b_l it is easy to verify that the length of the routing path is greater than $\frac{3}{2}D$. Now, if we can find a subset Y_i of X which is not k-coverable (we will see later in proof of Theorem 11 that by choosing suitable values for r and w based on the values of k and d, such a subset Y_i always exists), then the intervals assigned to the edge (a_i, b_i) cannot cover all the vertices in Y_i. Therefore, for some node in Y_i the message must be routed through c_i, which results in a path of length at least $\frac{3}{2}D$.

Intuitively, the graph $G(r, w, t)$ consists of three parts: (i) A set of w disjoint subgraphs, called *forks*, where each fork G_f, $1 \leqslant f \leqslant w$ consisting of α parallel paths of length $t - 1$ and a vertex u_f which is connected to one endpoint of all these paths; (ii) a set of α disjoint paths, where the ith path is b_i, a_i, c_i; and (iii) a set of edges connecting forks to the vertices in part (ii). We can consider partitioning X as partitioning forks since each G_f has exactly one element from X. For each way of partitioning X, b_l is connected to any fork which contains u_f, an element from the first partition, and c_l is connected to all other forks (Figure 4.2).

More formally, we define $G(r, w, t) = (V, E)$, $1 \leqslant r < w$ and $0 \leqslant t$, where V and E are defined as follows.

$$\begin{aligned} V &= \{u_f |\ 1 \leqslant f \leqslant w\} \\ &\cup \{a_l, b_l, c_l |\ 1 \leqslant l \leqslant \alpha\} \\ &\cup \{v_{f,c,l} |\ 1 \leqslant f \leqslant w, 1 \leqslant c \leqslant t, 1 \leqslant l \leqslant \alpha\} \end{aligned}$$

The edges of $G(r, w, t)$ are composed of

i) edges connecting a_l to b_l and c_l, $1 \leqslant l \leqslant \alpha$;

ii) edges of the path (of length $t+1$) connecting b_l to the vertices in Y_l $(1 \leqslant l \leqslant \alpha)$;

iii) edges of the path (of length $t+1$) connecting c_l to the vertices in Z_l $(1 \leqslant l \leqslant \alpha)$.

More precisely:

$$\begin{aligned} E &= \{(a_l, b_l), (a_l, c_l) |\ 1 \leqslant l \leqslant \alpha\} \\ &\cup \{(u_f, v_{f,1,l}) |\ 1 \leqslant f \leqslant w, 1 \leqslant l \leqslant \alpha\} \\ &\cup \{(v_{f,c,l}, v_{f,c+1,l}) |\ 1 \leqslant f \leqslant w, 1 \leqslant c \leqslant t-1, 1 \leqslant l \leqslant \alpha\} \\ &\cup \{(b_l, v_{i,t,l}) |\ i \in Y_l, 1 \leqslant l \leqslant \alpha\} \\ &\cup \{(c_l, v_{i,t,l}) |\ i \in Z_l, 1 \leqslant l \leqslant \alpha\} \end{aligned}$$

Figure 4.2 illustrates parts of the graph $G(2, 5, 2)$ as an example.

Proposition 1. *The diameter D of the graph $G(r, w, t)$ is $2t + 4$.*

Proof. Let us consider the fork G_f. There is a path of length at most $t+2$ from any vertex in G_f to a vertex a_l for some l, $1 \leqslant l \leqslant \alpha$. Therefore, the distance between any pair of vertices in G_i and G_j, $1 \leqslant i, j \leqslant w$ is at most $2t + 4$. This is also true for the vertices in $\{b_l |\ 1 \leqslant l \leqslant \alpha\}$ or $\{c_l |\ 1 \leqslant l \leqslant \alpha\}$. Hence, D is at most $2t+4$. We can verify that for any i and j, $1 \leqslant i, j \leqslant \alpha$, the distance between a_i and a_j is $2t + 4$ (simply going from a_i to any vertex u_k, $1 \leqslant k \leqslant \alpha$, and then going to a_j), so the diameter of the graph $G(r, w, t)$ is $2t + 4$. ■

Now we can state the main theorem of this section.

Theorem 11. *There is a graph G such that for any $\langle k, d \rangle$-MLIRS the length of the longest routing path is at least $\frac{3}{2}D$, where D is the diameter of the graph G.*

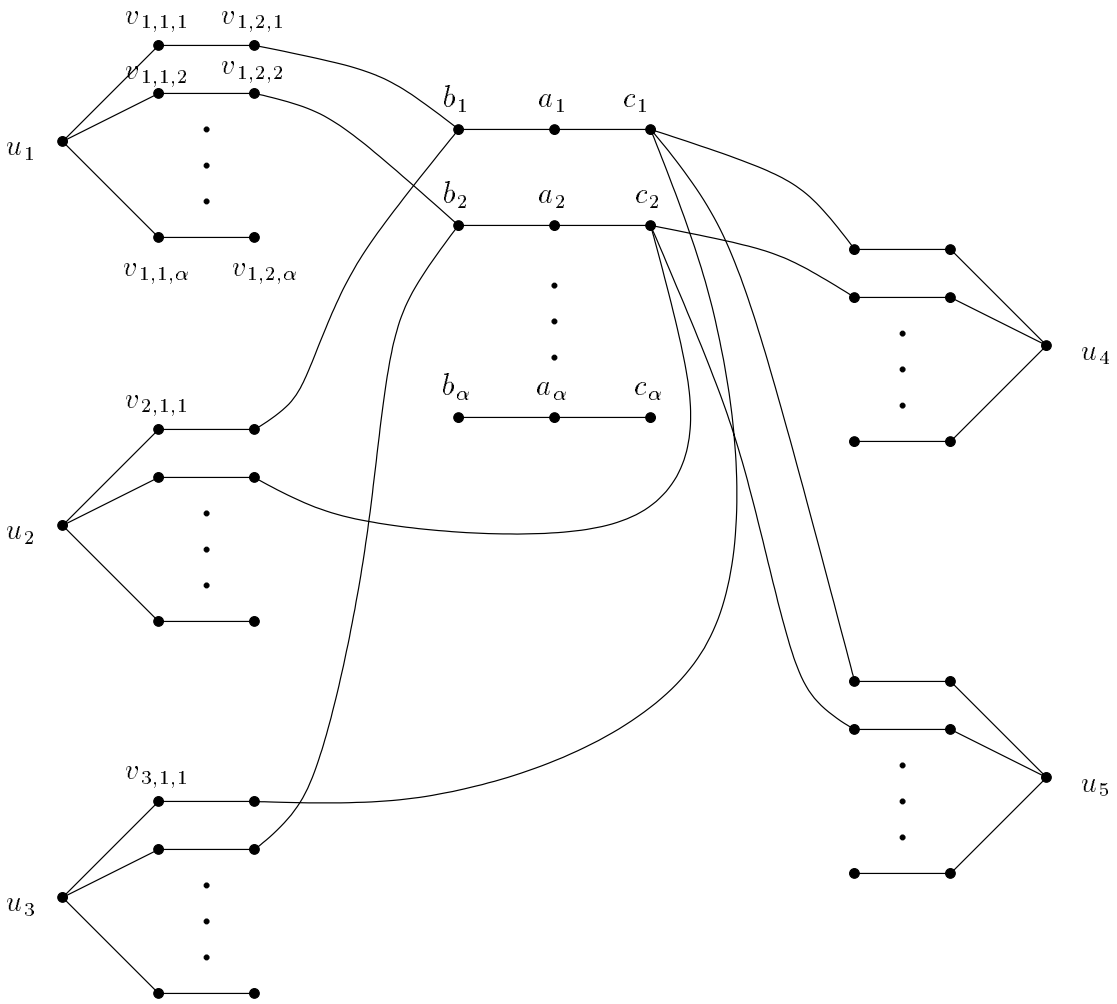

Figure 4.2: Parts of the graph $G(2, 5, 2)$.

Proof. Let us consider the graph $G(r, w, t)$ where $r = k + 1$, $w = \mathcal{F}(2k + 1, d)$ ($\mathcal{F}$ is the function defined in Lemma 17). For a given diameter D, we can choose the value of t based on Proposition 1 to construct a graph with diameter D. We will show that there is a vertex a_l, $1 \leqslant l \leqslant \alpha$ such that the k intervals assigned to the link (a_l, b_l) cannot cover all the vertices in Y_l, and hence the message for some vertex in Y_l is forwarded through the edge (a_l, c_l) which resulting in a path of length at least $3t + 6$.

Suppose that there is a $\langle k, d\rangle$-MLIRS for this graph such that the length of the longest routing path is less than $\frac{3}{2}D$. The d-dimensional label assigned to each vertex can be considered as a point in d-dimensional space (with integer coordinates). We will use these two representations interchangeably.

The set $U = \{u_f |\ 1 \leqslant f \leqslant w\}$ has $\mathcal{F}(2k + 1, d)$ points. By Lemma 17, there is a

subset M of these points which forms a monotone path and consists of at least $2k+1$ points. Lemma 15 indicates that M has a subset S containing $k+1$ vertices which is not k-coverable.

Since $r = k+1$, there is an integer l, $1 \leqslant l \leqslant \alpha$, such that $Y_l = S$ and therefore, Y_l is not k-coverable. There are at most k intervals assigned to the edge (a_l, b_l). These intervals cannot cover exactly the vertices in Y_l, so either there is a vertex in Z_l that is routed through the edge (a_l, b_l) or there is a vertex in Y_l which is routed through the edge (a_l, c_l). In either case, the length of the routing path is at least $3t+6$ or $\frac{3}{2}D$ and the proof is complete. ■

In the next section we will prove a better bound for $\langle 1, d\rangle$-MLIRS and $\langle 1, d\rangle$-MSLIRS.

4.2 Lower bounds for $\langle 1, d\rangle$-MLIRS and $\langle 1, d\rangle$-MSLIRS

Eilam *et al.* have proved a lower bound of $\Omega(D^2)$ on the length of the longest path traversed by a message using one-dimensional LIRS and SLIRS [EMZ99]. In this section we generalize their result and show that for any $d \geqslant 1$, there is a graph which for any $\langle 1, d\rangle$-MSLIRS or $\langle 1, d\rangle$-MLIRS the length of the longest routing path is $\Omega(D^2/d)$.

4.2.1 Bound for $\langle 1, d\rangle$-MSLIRS

To prove this bound, first we construct a graph $G_{l,r}$ for a positive even integer l and an odd integer $r \geqslant 3$. The graph $G_{l,r}$ consists of a r isomorphic subgraphs, called *wings*, each having l^2+1 vertices, say $v_0^i, v_1^i, ..., v_{l^2}^i$ for the ith wing. In the ith wing, each vertex v_j^i is connected to the vertex v_{j+1}^i, for $0 \leqslant j < l^2$, and the vertex v_{cl}^i is connected to the vertex $v_{(c+1)l}^i$, for $0 \leqslant c < l$. We also have edges between v_0^i and $v_0^{(i \bmod r)+1}$ for $1 \leqslant i \leqslant r$ (Figure 4.3).

We will show that there is a wing W of the graph $G_{l,r}$ such that d paths originating from W, cover all edges in the W. Since each wing has $\Theta(D^2)$ edges, there is a path of length $\Omega(D^2/d)$.

Proposition 2. *The diameter D of the graph $G_{l,r}$ is $3l - 2 + (r-1)/2$.*

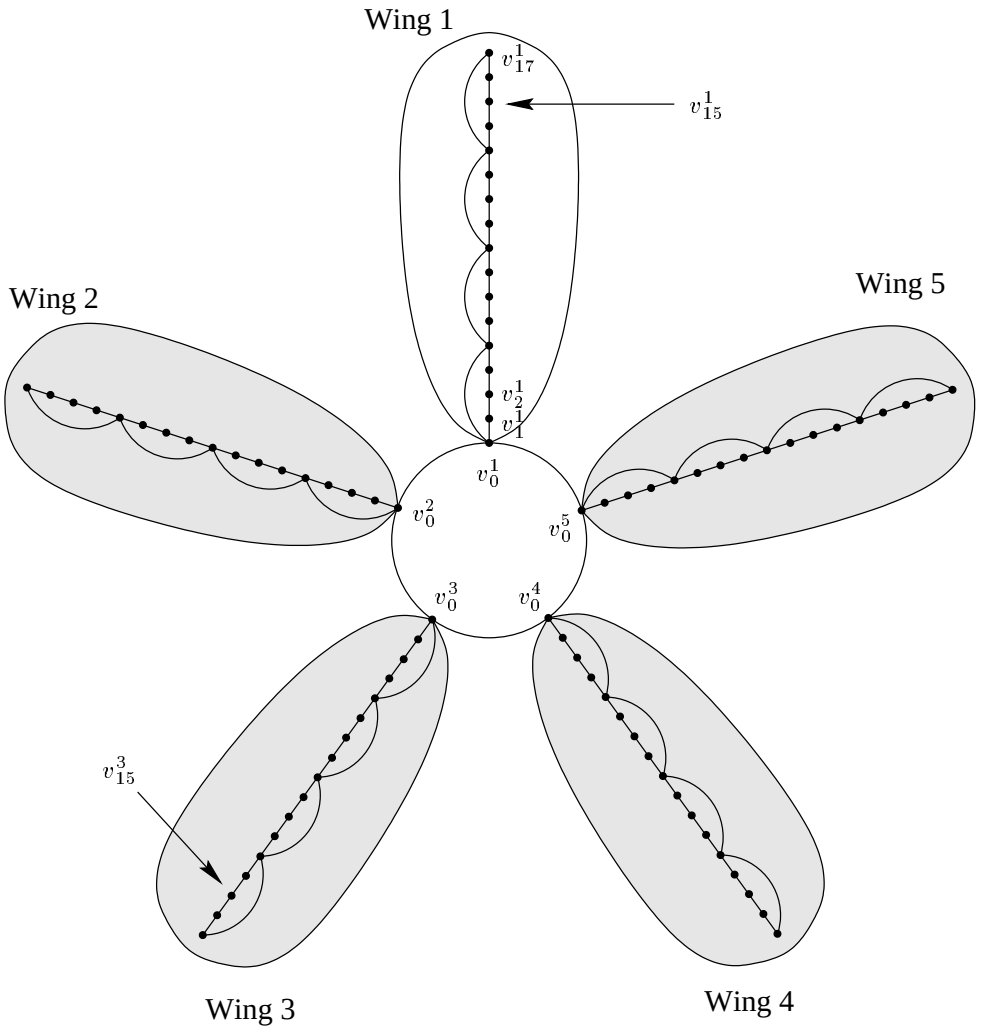

Figure 4.3: The graph $G_{4,5}$ (the distance between v^1_{15} and v^3_{15} is 12 in this graph).

Proof. In each wing, the length of the shortest path from any vertex v^i_j to v^i_0 is at most $3l/2 - 1$. The vertices v^i_0, $1 \leqslant i \leqslant r$, form a cycle of length r, so the longest distance between two vertices v^i_0 and v^j_0, $1 \leqslant i, j \leqslant r$, is at most $(r-1)/2$. Thus, the distance between any two vertices is at most $3l - 2 + (r-1)/2$. It is easy to verify that the shortest path between $v^1_{l^2-l/2+1}$ and $v^{(r+1)/2}_{l^2-l/2+1}$ is $3l - 2 + (r-1)/2$, and the proposition follows (Figure 4.3). ■

Each wing of the $G_{l,r}$ graph has $\Omega(l^2)$ edges. We will show that there is a path which goes through at least $\frac{1}{d}$ of these edges. Clearly, the length of this path is $\Omega(l^2/d)$. Proposition 2 shows that the diameter of the graph is in $\Theta(l)$, so the length of this path is in $\Omega(D^2/d)$.

Theorem 12. *For any $d \geqslant 1$ there is a graph G such that in any $\langle 1, k\rangle$-MSLIRS the length of the longest routing path is in $\Omega(D^2/d)$.*

Proof. For a given diameter, $D \geqslant 1$, let us consider the graph $G_{l,r}$ where $l = \frac{1}{3}(D + d + 2)$ and $r = 2d + 1$. These values are chosen so that the diameter of the graph is D. This graph is not a weak $(2d + 1)$-windmill graph, thus by Theorem 9 in Section 3.2 there is a $\langle 1, d\rangle$-MSLIRS defined on this graph. The labels assigned to the vertices of the graph can be considered as points in d-dimensional space. We denote this set of points by P.

We let B be an arbitrary boundary set for P (Section 3.1). Clearly, B has at most $2d$ points. Since $G_{l,2d+1}$ has $2d + 1$ wings, there is a wing j which does not contain any of the points in B. We claim that the length of the routing path from $v^j_{l^2}$ to one of the points in B is at least $\Omega(D^2/d)$.

Any interval containing all of the points in B will contain every other vertex in the graph. Since the routing scheme is strict, any interval assigned to the edge (u, v) cannot contain u. Therefore, the interval assigned an edge (u, v) cannot contain all the points in B.

Removing any edges in the first wing creates a bridge in the graph. Thus, if there exists an edge e whose label does not contain any points in B, there must be another edge (a bridge resulting from removing e) having all those points which, as we just mentioned, is not possible. Therefore, any edge in the first wing is part of at least one path going from $v^1_{l^2}$ to some point in B. Since we have $l^2 + l$ edges in the first wing and at most $2d$ paths, there is a path, say W, which has at least $(l^2 + l)/2d$ edges. Recalling that $l = \frac{1}{3}(D + d + 2)$, we can immediately check the following.

$$\text{The length of the path } W \geqslant \frac{l^2 + l}{2d} = \frac{(D + d + 2)^2/9 + D + d + 2}{2d}$$

In other words, the length of the path is in $\Omega(D^2/d)$. ∎

In this proof, we use the fact that the routing scheme is strict. In the next section, we will show that even if the routing scheme is not strict, we have the same lower bound of $\Omega(D^2/d)$.

4.2.2 Bound for $\langle 1, d\rangle$-MLIRS

It might seem that not requiring the interval routing scheme to be strict would give us more flexibility. In this section, we will show that even in this case, there is a $\Omega(D^2/d)$ lower bound on the length of the longest routing path.

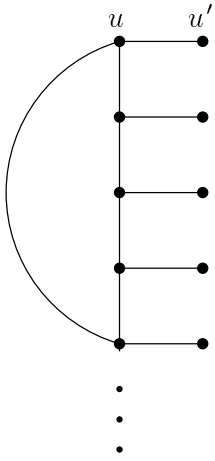

Figure 4.4: A piece of the graph $G'_{l,r}$.

Here, we use the graph $G'_{l,r}$ which is constructed from the graph $G_{l,r}$ in the following way: For each node u in $G_{l,r}$ we add a neighbor u' which is connected only to u via an edge (u, u') (Figure 4.4).

Theorem 13. *For any $d \geqslant 1$ there is a graph G such that in any $\langle 1, k\rangle$-MLIRS defined on G, the length of the longest routing path is in $\Omega(D^2/d)$.*

Proof. The graph $G'_{l,r}$ is not a $(2d + 1)$-windmill graph, hence, by Theorem 8 in Section 3.1.2 it has a $\langle 1, d\rangle$-MLIRS.

The diameter of $G'_{l,r}$ is two greater than the diameter of $G_{l,r}$. Hence, the diameter of $G'_{l,r}$ is $3l + (r - 1)/2$. If (u, v) is an edge in the underlying $G_{l,r}$ graph, the interval assigned to (u, v) cannot contain u' because the only way to reach u' from u is through the edge (u, u'). Therefore, with an argument similar to the proof of Theorem 12 we can prove a lower bound of $\Omega(D^2/d)$ in this case. ■

4.3 An upper bound for interval graphs

Narayanan and Shende have shown that for any interval graph we can always find an optimum IRS [NS98]. An interval graph is not a 3-windmill graph and therefore supports LIRS. There is not a known upper bound for the length of the routing paths induced by LIRS (one dimensional case) in interval graphs. In this section we will prove an upper bound of $2D-2$ for this quantity where D is the diameter of the graph.

We consider an interval graph G. Each vertex in G has a corresponding interval, say (s,t), in the interval representation of the graph. We let u_l be the vertex with the smallest s value, and u_r be the vertex with the largest t value. We also denote the shortest path from u_l to u_r by $P = (u_l = v_1, v_2, ..., v_k = u_r)$ and the interval corresponding to the vertex v_i by $I_i = (s_i, t_i)$, $1 \leqslant i \leqslant k$. The label of a vertex u is denoted by $l(u)$.

Proposition 3. *For any i and j between 1 and k, if $s_i < s_j$ then $t_i < t_j$.*

Proof. If we have $s_i < s_j$ but $t_i > t_j$, the interval I_j is completely contained in the interval I_i (this implies $v_j \neq u_l$ or u_r). By the definition of the interval graphs, any neighbor of v_j is also a neighbor of v_i. Now it is easy to see that by removing v_j the length of the shortest path is decreased, a contradiction to the minimality of the shortest path. ■

Theorem 14. *For any interval graph G there is a 1-LIRS such that the length of any routing path is in $O(D)$.*

Proof. We will proceed inductively: at step i ($1 \leqslant i \leqslant k$) we consider an induced subgraph of G, G_i, whose vertex set consists of v_i and any vertex u in G whose corresponding interval lies completely to the left of the point t_i.

Proposition 3 shows that the intervals corresponding to $v_1, v_2, ..., v_k$ have an ordering that makes this iteration feasible. At each step i, the aim is to show that there is an LIRS for G_i such that the length of any routing path is at most $2(i-1)$. We also show that in this LIRS the label of v_i is the maximum label among the vertices which have been labeled so far.

The statement is obvious for G_1 and the length of the longest routing path is 0 in this graph. Suppose that for $i-1$ $(2 \leqslant i < k+1)$ we have found an LIRS with the stated properties. Any vertex u in G_i that is not a vertex in G_{i-1} is a neighbor of v_i (otherwise the graph would be disconnected). If the number of such vertices is n_i, we label these vertices as follows: we arbitrarily label the neighbors of v_i with $l(v_{i-1})+1$ to $l(v_{i-1})+n_i$ and set $l(v_i) = l(v_{i-1})+n_i+1$ (Figure 4.5).

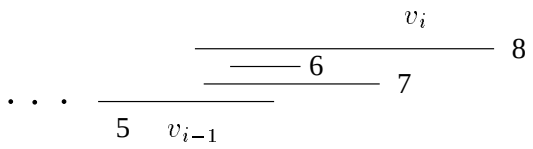

Figure 4.5: Labeling the vertices and edges in an interval graph (Numbers indicate sample labels).

We update the labels assigned to the edges in G_{i-1} so that any message with both origin and destination in G_{i-1} will traverse the same path as it did before updating the labels. The vertex v_i routes messages between any vertex in G_{i-1} and the vertices in $G_i \setminus G_{i-1}$, i.e., any message originating from a vertex in G_{i-1} towards a vertex in $G_i \setminus G_{i-1}$ will be forwarded to v_i via v_{i-1} and then will be forwarded to the destination (if v_i is not the destination). Also, any message from a neighbor of v_i (which is a vertex in $G_i \setminus G_{i-1}$) will be forwarded to G_i by going through v_i and then v_{i-1} (Figure 4.6).

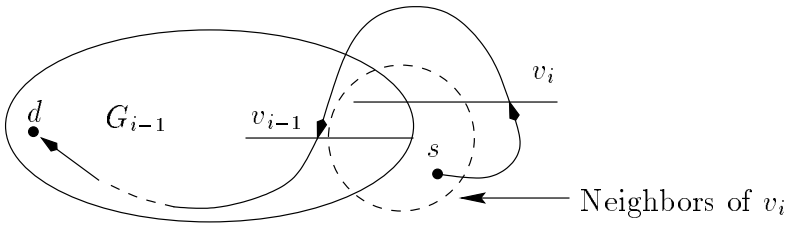

Figure 4.6: Routing messages between the vertices in G_{i-1} and the neighbors of v_i in $G_i - G_{i-1}$.

More formally we update the intervals as follows. First, for the intervals in G_{i-1}, if any interval $I = [s,t]$ containing $l(v_{i-1})$ (we know each vertex in G_{i-1} is connected to an edge with such an interval) we change I to $[s, l(v_i)]$. This is feasible because the label of v_{i-1} is the largest label in G_{i-1}. This causes any message from a node in G_{i-1} toward a node in $G_i \setminus G_{i-1}$ be forwarded to v_{i-1} (messages take the same route as a message directed to v_{i-1}). Any label in G_{i-1} not containing $l(v_{i-1})$ is not changed. Therefore routing within the vertices in G_{i-1} does not change. Now we assign intervals to the edges in $G_i \setminus G_{i-1}$. For each neighbor z of v_i the label of the edge (z, v_i) is set

to $[1..l(v_i)]$ and the label of the edge (v_i, z) is set to $[l(z)]$ (z sends every message to v_i and v_i sends messages with destination z to z). Finally, the edge (v_{i-1}, v_i) is labeled with $[l(v_{i-1})+1, l(v_i)]$ and the edge (v_i, v_{i-1}) is labeled with $[1, l(v_{i-1})]$. Therefore v_{i-1} will forward any message to v_i or one of its neighbors to v_i and vice versa.

In the new routing scheme defined on G_i, the length of routing paths in G_{i-1} is not increased and the length of the longest routing path in G_i is at most two units more than the length of the longest routing path in G_{i-1}. Using the induction hypothesis, we can easily verify that the length of the longest routing path in G_i is at most $2(i-1)$. Obviously, v_i has the maximum label in G_i. Thus, G_i satisfies the desired properties.

Since the length of the shortest path between u_l and u_r is at most D ($k \leqslant D$), the length of any routing path in G is at most $2D-2$. ■

Any upper bound for a $\langle 1,1\rangle$-MLIRS is also an upper bound for any $\langle 1,d\rangle$-MLIRS ($d \geqslant 2$). Thus, the $O(D)$ upper bound stated in this section hold for any $\langle 1,d\rangle$-MLIRS.

Chapter 5

Interval routing and hypergraph drawing

It is always nice to find out the relationship between two (or more) different problems and to extend the results in one area to the other. In this chapter we illustrate a strong relationship between different interval routing schemes and the problem of hypergraph drawing. We use results from the previous chapters to prove a tight bound on the number of dimensions of the space needed to draw a hypergraph.

In Section 5.1 we briefly review known results and definitions of hypergraph drawings. Then, in Section 5.1.1 we consider a specific graph drawing problem which is known as the *rectangle of influence drawing*. We generalize this idea to hypergraphs in Section 5.1.2 and introduce a representation for hypergraphs in d-dimensional space which is called *box representation of hypergraphs*. In Section 5.2 we study the relationship between the problem of finding a box representation for a hypergraph and results from previous chapters. We show that any hypergraph with at most $2d$ vertices can be drawn in d-dimensional space. We also show that there is a hypergraph with $2d + 1$ vertices which cannot be drawn in d-dimensional space, and therefore $2d$ is an optimal bound.

5.1 Hypergraph drawing

The problem of drawing graphs in the plane and even in spaces of higher dimension has been extensively studied [DBETT99]. The objective is to find a representation for the graph in a specific space such that a pre-specified measure is optimized. Here, the pre-specified measure can be the number of edge-crossings, the number of bends in the edges, the area needed, the degree of symmetry, and so on. The existing algorithms use one or more of these measures and try to optimize the drawing with respect to the criterion or criteria selected. For example, in the well-known problem of finding a planar drawing of a (planar) graph, the measure to optimize is the number of edge-crossings. Here, each vertex of the graph is represented by a point in the plane, each edge is represented by an arc, and the goal is to find a representation of the graph such that the number of edge-crossings is zero. The graph drawing problem has been completely solved only for some special cases of graphs, and most algorithmic problems in this area are NP-complete or even more difficult [DBETT99].

Since hypergraphs are generalization of graphs, it is natural to generalize the graph drawing concepts into similar ones for hypergraphs. Johnson and Pollad introduced two notions of planarity of hypergraphs, which was inspired by the Venn diagram representations of sets, and presented some NP-completeness results [JP87].

Mäkinen introduced two distinct kinds of hypergraph drawing [M̈90]. In both cases, vertices are represented by points in the plane. In the *edge standard representation of a hypergraph* a hyperedge e is represented by connecting the points that represent the vertices which are in e by smooth curved lines. Two vertices belong to the same hyperedge if there is a smooth curve between the points that represent these vertices (Figure 5.1 (a)). In the *subset standard representation of a hypergraph*, a hyperedge is represented by a closed curve that contains exclusively the points that represent the vertices in that hyperedge (Figure 5.1 (b)).

Hypergraph drawing has various applications. For example, in the theory of relational databases there is a correspondence between database schemes and hypergraphs. A database scheme consists of a set of relation schemes. Each relation scheme is a collection of attributes. We can consider database schemes as hypergraphs, attributes as the vertices, and relation schemes as hyperedges in those hypergraphs. Drawing of these hypergraphs is used in the study of the cyclicity of database schemes. There are other applications of hypergraph drawings *e.g.* representing functional dependency in

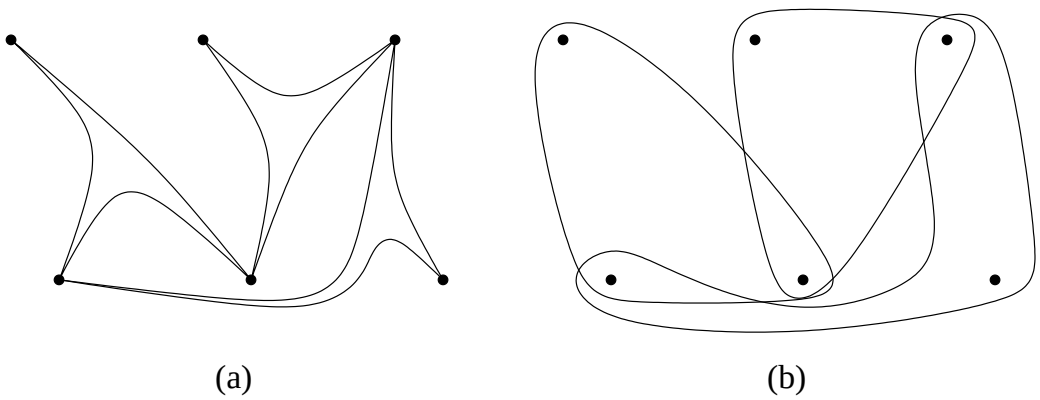

Figure 5.1: a) Edge standard representation of a hypergraph H b) Subset standard representation of H.

database systems, and-or graphs in problem solving, Horn clauses in logic programming and also specification of concurrent systems.

5.1.1 Box representation of a graph

Let us consider a representation of a graph G in which each vertex is represented by a point in the plane, and each edge $e = (u, v)$ is represented by an axis-aligned rectangle such that the points representing u and v are opposite corners of the rectangle and there is no other point corresponding to a vertex of G inside the rectangle. This representation of a graph is known as the *rectangle of influence drawing* of the graph [LLMW98]. If the rectangles representing edges of the graph are closed (open) sets this representation is known as a *closed (open) rectangle of influence drawing.* Figure 5.2 (a) illustrates an example of a rectangle of influence drawing. In this example, there is no other point inside the rectangle defined by any two points. Hence, in the open model we have a complete graph. On the other hand, in Figure 5.2 (b) the point w is on the boundary of rectangle defined by u and v and therefore we cannot have an edge connecting u and v in the closed model.

Liotta *et al.* have studied the problem of characterizing the class of graphs that admit rectangle of influence drawings for several classes of graphs like cycles, trees, and wheels [LLMW98]. They show that for these classes testing whether a graph with n nodes has a rectangle of influence drawing can be done in $O(n)$ time and that it is possible to construct such a drawing if one exists.

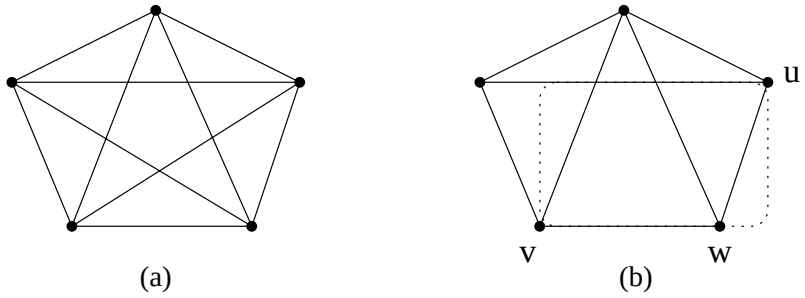

Figure 5.2: Example of rectangle of influence drawing for (a) the open model and (b) the closed model for the same set of vertices as in part (a).

With only a few slight changes in the definition of the rectangle of influence drawing we can generalize the idea of rectangle of influence representation to d dimensions ($d > 2$). Vertices of the graph are still represented by points in d-dimensional space. We can have an edge between two vertices u and v if there is no other point in the d-dimensional (for now let us assume that no coordinates of the points representing u and v have the same value) axis-aligned box defined by the points representing u and v. We call this a *d-dimensional box representation of a graph.* In the next section we show how to generalize this idea to hypergraphs.

5.1.2 Box representation of a hypergraph

In the previous section we considered a representation for a graph in which each edge was represented by a box in d-dimensional space. We can easily use this idea to define a box representation for hypergraphs.

Definition 22. *A* box representation *of a hypergraph $H = (V, E)$ in d-dimensional space is a representation of the hypergraph such that:*

i) Each vertex in V is represented by a unique point in d-dimensional space.

ii) For each hyperedge $e \in E$, there is an axis-aligned box $[a_1..b_1, a_2..b_2, ..., a_d..b_d]$ such that

a) $a_i \leqslant b_i$ for any i, $1 \leqslant i \leqslant d$; and

b) for any point $p = (p_1, p_2, ..., p_d)$ *corresponding to a vertex* v *in* V, $a_i \leqslant p_i \leqslant b_i$, *for* $1 \leqslant i \leqslant d$, *if and only if* $v \in e$.

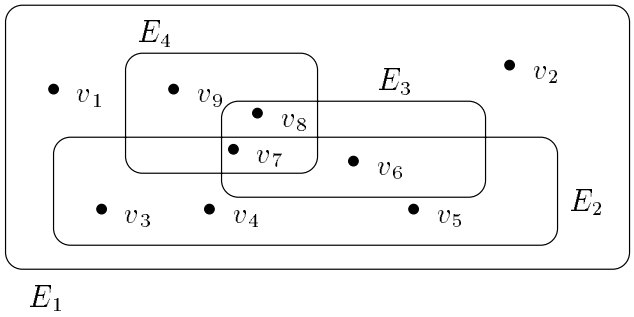

Figure 5.3: Box representation for a hypergraph H.

In this definition, the points in a hyperedge e of the hypergraph do not necessarily lie on the boundaries of the box representing e, but it is easy to verify that if H is a graph the box representation of H is equivalent to the d-dimensional box representation of graphs in the closed model.

Example 14. *Let us consider the hypergraph* $H = (V, E)$, *where* $V = \{v_1, v_2, ..., v_9\}$ *and* $E = \{e_1, e_2, e_3, e_4\}$ *such that* $e_1 = \{v_1, v_2, ..., v_9\}$, $e_2 = \{v_3, v_4, ..., v_7\}$, $e_3 = \{v_6, v_7, v_8\}$ *and* $e_4 = \{v_7, v_8, v_9\}$ *(the hypergraph depicted in Figure 2.12). Figure 5.3 depicts a box representation for this hypergraph. For increased readability the rectangles representing the hyperedges are replaced by rounded rectilinear shapes.*

In the following section we will show the relationship between the problem of finding a box representation of a hypergraph and routing schemes.

5.2 Hypergraph drawing and IRS

Let us consider the problem of finding an MIRS for a graph $G = (V, E)$. We denote by $F_{(v,e)}$ the set of destination addresses of the messages which should be forwarded through the link e at node v. Now, let us consider a hypergraph $H = (V', E')$ such that $V' = V$ and the set of hyperedges of H is $\{F_{(v,e)} | v \in V, e \in E\}$. To be consistent with the definition of a hypergraph, we consider hyperedges which contain at least two vertices. We will call H the *routing hypergraph of* G.

The problem of finding a d-dimensional MIRS for G is equivalent to the problem of finding a d-dimensional box representation for H. To verify this claim, let us assume that we have a d-dimensional MIRS for G. We can consider the labels of vertices in G as coordinates of the vertices in d-dimensional space. For each hyperedge e in E', the interval assigned to the corresponding edge in E contains exactly the vertices in e. Therefore, we have a box representation for H. On the other hand, we can similarly show that if we can find a box representation for H, we will have an MIRS for G. In the rest of this section, we will show some examples of the relation between the concept of hypergraph drawing and finding an MIRS on a graph.

5.2.1 Every hypergraph has a box representation

Lemma 11 in Section 3.3.2 shows how to find a labeling for the vertices of a graph such that in any edge-biconnected component of the graph and for any subset of the vertices there is a interval containing exactly those vertices. Similarly, if we have a hypergraph with at most $2d$ vertices we can assign the vertices to the points $(1, 0, ..., 0)$, $(0, 1, ..., 0)$, ..., $(0, 0, ..., 1)$, $(-1, 0, ..., 0)$, $(0, -1, ..., 0)$, ..., $(0, 0, ..., -1)$ in d-dimensional space. We can use a proof similar to that of Lemma 11 in Section 3.3.2 to show that for any subset of vertices there is a box that contains exactly those vertices. Therefore, we have the following theorem.

Theorem 15. *For any hypergraph H with at most $2d$ vertices, there is a box representation in d-dimensional space.*

Is it possible to find a box representation in a space with fewer dimensions? Clearly, if we have a box representation for a hypergraph H we also have a box representation for any hypergraph H' resulting from removing a number of hyperedges in H. We will use this fact to show how we might be able to reduce the number of dimensions needed.

Let us consider a hypergraph $H = (V, E)$ such that $V = \{v_1, v_2, ..., v_n\}$ and $E = \{e_1, e_2, ..., e_m\}$. We construct a graph G as follows. For each vertex v_i, $1 \leqslant i \leqslant n$, G has a corresponding vertex w_i. For each hyperedge e_i in E, $1 \leqslant i \leqslant m$, G also has a vertex called u_i. There is an edge (u_i, w_j) in G if v_j is a vertex of e_i in H. Also, all the vertices u_i, $1 \leqslant i \leqslant m$, are connected to a vertex in G called z (Figure 5.4). We assume that G is the underlying graph of a network in which all links have arbitrary costs.

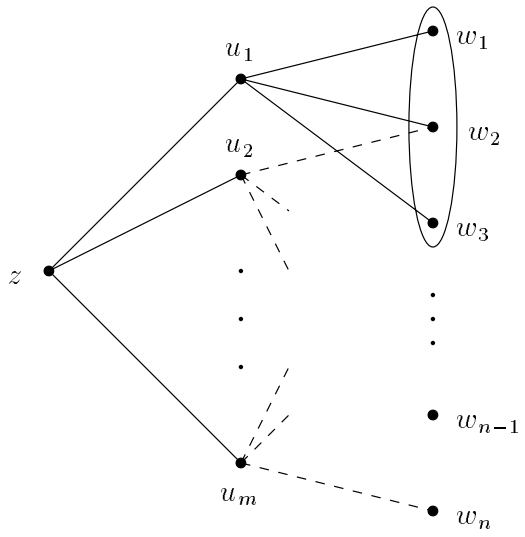

Figure 5.4: Constructing a graph based on a hypergraph.

Theorem 10 in Section 3.3.2 shows that if G has k non-articulation points, then there is an optimum $\lceil k/2 \rceil$-dimensional MLIRS for G. The shortest path going from z to the vertices in e_i goes through the edge (z, u_i). Hence, the interval assigned to the link (z, u_i) in G corresponds to the box which represents the edge e_i in H and therefore we have box representation for H. Since $k \leqslant n$, we need at most $\lceil n/2 \rceil$ dimensions which might be better than the previous result for some cases.

5.2.2 Not every hypergraph has a d-dimensional box representation

In this section we show that for every $d \geqslant 2$ there is a hypergraph which does not have a box representation in d-dimensional space. To prove this we will use the notion of boundary sets introduced in Section 3.1. Let us consider a hypergraph $H = (V, E)$ where $V = \{v_1, v_2, ..., v_{2d+1}\}$ and E is the set of all subset of V which have exactly $2d$ elements. Let us assume that H has a box representation in d-dimensional space and denote the boundary set of the vertices of H by B. In Section 3.1 we showed that B has at most $2d$ vertices. Since H has $2d+1$ vertices there is a vertex in H, say v, such that v is not in B. Since H has a hyperedge for every subset of V with $2d$ elements, there is a hyperedge, say e, which contains exactly $V \setminus \{v\}$. We already know that any box containing all the vertices in B contains all vertices in V and therefore the box assigned to e covers v, which is not allowed.

Theorem 16. *For every $d \geqslant 2$ there is a hypergraph H with $2d+1$ vertices which does not have a box representation.*

Comparing this result with the result of the previous section, we can see that d is a tight bound for the number of dimensions of the space needed to draw a general hypergraph with $2d$ vertices.

Chapter 6

Conclusions and future work

Characterizing the class of networks which support MIRS and measuring the quality of routing in such networks were the two main problems studied in this thesis. We proved that a network supports a $\langle 1, d\rangle$-MLIRS if and only if it is not a $(2d + 1)$-windmill graph. This is a generalization of a result by Fraigniaud and Gavoille in which they characterize the class of networks supporting 1-LIRS [FG94]. We also considered the class of networks which support a $\langle 1, d\rangle$-MSLIRS and showed that a network supports such a routing scheme if and only if its underlying network is not a weak $(2d + 1)$-windmill graph. Based on the structure of windmill graphs and weak windmill graphs, it is intuitively obvious that if a graph is highly connected (has a few bridges) then we can easily find a $\langle 1, d\rangle$-MLIRS or a $\langle 1, d\rangle$-MSLIRS for the graph, for small values of d, which can be consider a constant. Since the amount of space needed to store a d-dimensional MLIRS (MSLIRS) is d times the amount of space needed to store a 1-dimensional LIRS (SLIRS) and since the value of d can be considered a constant for highly connected graphs, this shows multi-dimensional schemes are still better than explicit routing schemes.

Another characterization problem which we considered in this thesis was the problem of characterizing the class of networks which support optimum $\langle 1, d\rangle$-MSLIRS with dynamic cost links. Since in real networks the cost of links may vary over time, this multi-dimensional scheme might be useful for real applications. We proved that a network supports such a routing scheme if and only if its underlying graph has at most $2d$ non-articulation points. Therefore, a network in which most of the nodes are articulation points and has just a few non-articulation points may be more suitable

for this scheme. So far there are no other known characterization results for MIRS except for specific graphs like hypercubes, rings, and grids. This also emphasizes the importance of these results and is a very interesting area for future research.

The second problem we considered in this thesis is the quality of routing problem. To quantify the quality of routing we used the length of routing paths as the main measure. We proved a $\frac{3}{2}D$ lower bound for this quantity for any $\langle k, d\rangle$-MLIRS (where D is the diameter of the graph). If the number of intervals at each link is one we have a better bound of $\Omega(D^2/d)$ for $\langle 1, d\rangle$-MSLIRS and $\langle 1, d\rangle$-MLIRS. These lower bounds suggest that by having more than one interval we might be able to dramatically decrease the length of routing paths. There is not a known upper bound for these routing schemes except for specific networks like interval graphs. For these graphs we have proved an upper bound of $2D - 2$ on the length of the longest routing path.

When studying IRS, there is a problem which is of a higher importance compared to the characterization and the quality of routing problems for a specific scheme. This is the problem of designing a scheme. There has been a lot of effort in designing variants which are more compact (need less space) and have good routing qualities (*i.e.* are supported by a large class of networks and the routing paths are short). Here is a new idea which seems to be very useful.

When implementing an IRS, there is usually a table in each node of the network which has one entry (or k entries for k-IRS) for each incident link. The processor in the node reads these entries one by one, checks if the destination address belongs to the interval in that entry and if so routes the message through the link indicated in that entry. A very interesting idea is to use this intrinsic order of entries in the table. In other words, we define an ordering for the intervals assigned to links and always check the intervals in this pre-specified ordering. Here, even if two intervals intersect, since the order in which the intervals are checked is specified there is no ambiguity in routing the message. We call this IRS an *Ordered Interval Routing Scheme* or OIRS for short.

The OIRS seems to be more useful in higher dimensions. In fact, in a d-dimensional MIRS it is easy to show that if we have two intervals A and B such that A is completely contained in B then in the original MIRS covering the nodes which are in B but not in A requires $2d$ intervals. In an OIRS this can be achieved using one interval, assuming that in the specified order B is placed after A. This idea can be extended to more than two intervals. For example in Figure 6.1 covering the vertices in $B - A$ requires four

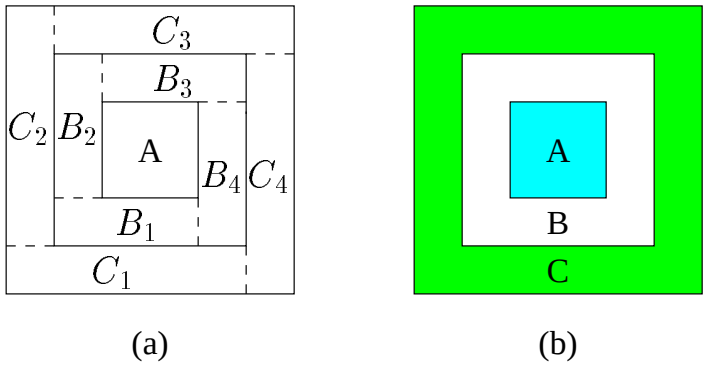

Figure 6.1: An example of an OIRS. The order in which interval should be processed is A, B and C.

intervals B_1, B_2, B_3 and B_4 in an ordinary 2-dimensional IRS (part (a)) but just one interval is sufficient in an OIRS (part (b)). The same statement is true for covering the vertices in $C - B$.

Finally, in this thesis we showed a strong relationship between different variants of MIRS and hypergraph drawing. The idea of hypergraph drawing and the specific variant introduced in Chapter 5 (called the box representation of hypergraphs) seems to be very natural. We have proved that for any hypergraph with at most $2d$ vertices there is a box representation in d-dimensional space. We also have shown that there is a hypergraph with $2d + 1$ vertices which does not have a box representation in d-dimensional space. There are still interesting open problems in this area such as finding the class of graphs or hypergraphs which have box representations in a d-dimensional space. Finding this class of hypergraphs can help in solving some of the interesting MIRS characterization problems.

Bibliography

[ABNLP90] Baruch Awerbuch, Amotz Bar-Noy, Nathan Linial, and David Peleg. Improved routing strategies with succinct tables. *J. Algorithms*, 11(3):307–341, 1990.

[BM76] John A. Bondy and U. S. R. Murty. *Graph Theory with Applications.* American Elsevier Publishing Co., Inc., New York, 1976.

[BvLT91] Erwin M. Bakker, Jan van Leeuwen, and Richard Tan. Linear interval routing. *ALCOM: Algorithms Review, Newsletter of the ESPRIT II Basic Research Actions Program Project no. 3075 (ALCOM)*, 2, 1991.

[BvLT93] Erwin M. Bakker, Jan van Leeuwen, and Richard B. Tan. Prefix routing schemes in dynamic networks. *Computer Networks and ISDN Systems*, 26(4):403–421, December 1993.

[Cow99] Lenore J. Cowen. Compact routing with minimum stretch. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms (Baltimore, MD, 1999)*, pages 255–260, New York, 1999. ACM.

[DBETT99] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing.* Prentice Hall Inc., Upper Saddle River, NJ, 1999.

[EMZ96] Tamar Eilam, Shlomo Moran, and Shmuel Zaks. A lower bound for linear interval routing. In K. Marullo Ö. Babaoğlu, editor, *10th International Workshop on Distributed Algorithms (WDAG)*, volume 1151 of Lecture Notes in Computer Science, pages 191–205, Berlin, October 1996. Springer-Verlag.

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

[ES35] Paul Erdös and George Szekeres. A combinatorial problem in geometry. *Compositio Mathematica*, 2:463–470, 1935.

[FG94] Pierre Fraigniaud and Cyril Gavoille. A characterization of networks supporting linear interval routing. In 13^{th} *Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 216–224. ACM PRESS, August 1994.

[FG98] Pierre Fraigniaud and Cyril Gavoille. Interval routing schemes. *Algorithmica*, 21(2):155–182, 1998.

[FGNT98] Michele Flammini, Giorgio Gambosi, Umberto Nanni, and Richard B. Tan. Multidimensional interval routing schemes. *Theoret. Comput. Sci.*, 205(1-2):115–133, 1998.

[FGS93] Michele Flammini, Giorgio Gambosi, and Sandro Salomone. Boolean routing. In *Distributed algorithms (Lausanne, 1993)*, pages 219–233. Springer, Berlin, 1993.

[FJ86] Greg N. Frederickson and Ravi Janardan. Optimal message routing without complete routing tables. In Joseph Halpern, editor, *Proceedings of the 5th Annual ACM Symposium on Principles of Distributed Computing*, pages 88–97, Calgary, AB, Canada, August 1986. ACM Press.

[FJ88] Greg N. Frederickson and Ravi Janardan. Designing networks with compact routing tables. *Algorithmica*, 3(1):171–190, 1988.

[FJ89] Greg N. Frederickson and Ravi Janardan. Efficient message routing in planar networks. *SIAM J. Comput.*, 18(4):843–857, 1989.

[Fre96] Greg N. Frederickson. Searching among intervals and compact routing tables. *Algorithmica*, 15(5):448–466, 1996.

[Gav00] Cyril Gavoille. A survey on interval routing. *Theoret. Comput. Sci.*, 245(2):217–253, 2000. Algorithms for future technologies (Saarbrücken, 1997).

[HL00] Laura Heinrich-Litan. Monotone subsequences in r^d. Technical Report B 00-19, Freie Universität Berlin, Fachbereich Mathematik und Informatik, 2000.

[INM91] INMOS. The T9000 Transputer Overview Manual. 1991.

[JP87] David S. Johnson and Henry O. Pollak. Hypergraph planarity and the complexity of drawing Venn diagrams. *J. Graph Theory*, 11(3):309–325, 1987.

[KK96] Evangelos Kranakis and Danny Krizanc. Lower bounds for compact routing (extended abstract). In *17th International Symposium on Theoretical Aspects of Computer Science (Grenoble, 1996)*, pages 529–540. Springer, Berlin, 1996.

[KKR94] Evangelos Kranakis, Danny Krizanc, and S. S. Ravi. On multi-label linear interval routing schemes (extended abstract). In *Graph-theoretic concepts in computer science (Utrecht, 1993)*, pages 338–349. Springer, Berlin, 1994.

[KRŠ00] Rastislav Kráľovič, Peter Ružička, and Daniel Štefankovič. The complexity of shortest path and dilation bounded interval routing. *Theoret. Comput. Sci.*, 234(1-2):85–107, 2000.

[LLMW98] Giuseppe Liotta, Anna Lubiw, Henk Meijer, and Sue H. Whitesides. The rectangle of influence drawability problem. *Comput. Geom.*, 10(1):1–22, 1998.

[MÖ90] Erkki Mäkinen. How to draw a hypergraph. *International Journal of Computer Mathematics*, 34:177–185, 1990.

[NO99] Lata Narayanan and Jaroslav Opatrny. Compact routing on chordal rings of degree 4. *Algorithmica*, 23(1):72–96, 1999.

[NS98] Lata Narayanan and Sunil Shende. Partial characterizations of networks supporting shortest path interval labeling schemes. *Networks*, 32(2):103–113, 1998.

[RŠ00] Peter Ružička and Daniel Štefankovič. On the complexity of multi-dimensional interval routing schemes. *Theoret. Comput. Sci.*, 245(2):255–280, 2000. Algorithms for future technologies (Saarbrücken, 1997).

[Ruž88] Peter Ružička. On efficiency of interval routing algorithms. In V. Koubek M. P. Chytil, L. Janiga, editor, *Mathematical Foundations of Computer Science*, volume 324 of Lecture Notes in Computer Science, pages 492–500. Springer-Verlag, 1988.

[SK82] Nicola Santoro and Ramez Khatib. Routing without routing tables. *Technical Report SCS-TR-6 School of Computer Science, Carleton University, Ottawa*, 1982.

[SK85] Nicola Santoro and Ramez Khatib. Labelling and implicit routing in networks. *The Comput. J.*, 28(1):5–8, 1985.

[Tan95] Andrew S. Tanenbaum. *Distributed Operating Systems.* Prentice Hall Inc., Upper Saddle River, NJ, 1995.

[Tan96] Andrew S. Tanenbaum. *Computer Networks.* Prentice Hall Inc., Upper Saddle River, NJ, 3rd edition, 1996.

[TL95] Savio S. H. Tse and Francis C. M. Lau. Lower bounds for multi-label interval routing. In *Proceedings of 2nd Colloquium on Structural Information & Communication Complexity (SIRROCO'95)*, pages 123–134, 1995.

[TL97a] Savio S. H. Tse and Francis C. M. Lau. A lower bound for interval routing in general networks. *Networks*, 29(1):49–53, 1997.

[TL97b] Savio S. H. Tse and Francis C. M. Lau. An optimal lower bound for interval routing in general networks. In *Proceedings of 4th International Colloquium on Structural Information & Communication Complexity (SIROCO'95)*, pages 112–124. Carleton Scientific, July 1997.

[TvL95] Richard B. Tan and Jan van Leeuwen. Compact routing methods: A survey. In *Proceedings of Colloquium on Structural Information and Communication Complexity (SICC'94), SCS, Carleton University, Ottawa*, pages 99–109, 1995.

[vLT87] Jan van Leeuwen and Richard B. Tan. Interval routing. *Comput. J.*, 30(4):298–307, 1987.

[Wes96] Douglas B. West. *Introduction to Graph Theory.* Prentice Hall Inc., Upper Saddle River, NJ, 1996.

[WMT93] Peter H. Welch, David May, and Peter W. Thompson. *Networks, Routers and Transputers: Function, Performance and Application.* IOS Press, Netherlands, February 1993. ISBN 90-5199-129-0.