Ordered Interval Routing Schemes

by

Mustaq Ahmed

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, 2004

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A THESIS

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

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

Abstract

An *Interval Routing Scheme (IRS)* represents the routing tables in a network in a space-efficient way by labeling each vertex with an unique integer address and the outgoing edges at each vertex with disjoint subintervals of these addresses. An IRS that has at most k intervals per edge label is called a *k-IRS*. In this thesis, we propose a new type of interval routing scheme, called an *Ordered Interval Routing Scheme (OIRS)*, that uses an ordering of the outgoing edges at each vertex and allows non-disjoint intervals in the labels of those edges. Our results on a number of graphs show that using an OIRS instead of an IRS reduces the size of the routing tables in the case of *optimal* routing, i.e., routing along shortest paths. We show that optimal routing in any k-tree is possible using an OIRS with at most 2^{k-1} intervals per edge label, although the best known result for an IRS is 2^{k+1} intervals per edge label. Any torus has an optimal 1-OIRS, although it may not have an optimal 1-IRS. We present similar results for the Petersen graph, k-garland graphs and a few other graphs.

Acknowledgements

All praise is due to Allah, the Lord of the Worlds, the Beneficent, the Merciful.

I would like to thank my supervisor Professor Naomi Nishimura for her encouragement and support for the work. Without her guidance, advice and criticism, it would have been impossible to accomplish this immense task. I am also grateful to my readers, Professor Therese Biedl and Professor A. Lopez-Ortiz, for their helpful comments and suggestions.

I am indebted to all members of my family, specially my parents and wife, for their support and encouragement over the years. I also express my deep appreciation to all who have contributed to this work in any way.

Contents

List of Figures

Chapter 1

Introduction

In a network where processors communicate with one another by sending and receiving messages, a *routing scheme* is used to determine the path a message follows to reach its destination. A classical method used for routing is to store a *routing table* at each node of the network that contains, for each destination address, an entry specifying the link a message should follow to reach that particular destination. Thus, the path followed by a message is determined by the routing tables at all the nodes on the path. Using this method, it is easy to ensure that every message follows the shortest path to its destination. However, this method requires an $O(n)$ sized routing table at every node of a n-node network, which makes it unsuitable for large networks. Researchers have investigated several approaches to reduce the space requirements for the routing tables. *Interval Routing* is one of the most popular approaches to achieve compact routing tables [EMZ02].

Interval routing was first proposed by Santoro and Khatib [SK85], and has been implemented in the C104 router chip of the INMOS T9000 transputer design [INM91, WMT93]. The basic idea is to label the nodes of a network with unique integer *addresses* and to use an interval to concisely represent the group of destination addresses to be covered by each link in the network. In this case, the routing table at a vertex

contains one interval for each of the outgoing links. We can model this scheme, called an *Interval Routing Scheme (IRS)*, using a labeled directed graph in which the addresses of the nodes in the network are represented by the labels of the corresponding vertices, and each interval in the routing table at a node is represented by a label on the corresponding outgoing edge. At each vertex, the algorithm used during routing is as follows [Gav00]. When a vertex v gets a packet with destination address L, the routing process ends if L is the label of v; otherwise, the processor at v forwards the packet through the edge e that contains L in the corresponding interval. The idea of interval routing was generalized by van Leeuwen and Tan [vLT87] to allow more than one interval in each edge label. An interval routing scheme that has at most k intervals per edge label is called a *k-interval routing scheme (k-IRS).*

Most of the recent research work on interval routing emphasizes on *optimal IRS's* where the goal is to route each packet along a shortest path to its destination. One reason for the emphasis on optimal IRS's is obvious: optimal routing is crucial for fast and efficient communication in a network. Another reason is that devising a 1-IRS is easy for *all* graphs [vLT87], although this is not true for the case of optimal IRS's. For example, the Petersen graph [Gav00] and 2-trees [NN98] have no optimal 2-IRS. There are graphs in certain subclasses of planar graphs [FJ88, GP96, KRŠ00] that require $\Omega(\sqrt{n})$ intervals per edge label for optimal routing.

One important characteristic of an IRS is that the labels of all the edges adjacent to a vertex v must be disjoint so that the processor at v can uniquely identify the edge e through which a packet should be forwarded. Motivated by the fact that there are many graph classes that need more than one interval per edge label in any optimal IRS, and by the idea that in such cases, dropping the constraint of disjoint edge labels can reduce the number of intervals required by an optimal IRS, we propose a new variant of interval routing scheme in this thesis. We call it an *Ordered Interval Routing Scheme (OIRS).* We observe that if there is a predetermined order of the outgoing edges at vertex v, the labels of the edges at v need not be mutually disjoint

to enable the processor at v to uniquely identify the edge e. One advantage of such an edge order is that it may allow us to label an edge with fewer intervals. For example, suppose two edges e_1 and e_2 are adjacent to a vertex v, and the label of e_1 contains only the interval $[2, 3]$ and the label of e_2 contains two intervals $[0, 1]$ and $[4, 5]$. If e_2 is considered after e_1 during routing, and we change the label of e_2 to $[0, 5]$, then the edge e_2 will still be used to forward only the packets with destination addresses in $[0, 1] \cup [4, 5]$. An OIRS has an ordering of the outgoing edges at each vertex and allows overlapping edge labels to reduce the number of intervals required for optimal routing.

The goal of any interval routing scheme is to reduce the space requirement of the routing tables. Fewer intervals in the edge labels in an OIRS means smaller size of the routing tables than in an IRS. We also note the point that we do not need extra space to store an edge order in the routing table: *any* such order can be achieved by just rearranging the entries in a routing table. As a result, the space requirement of an OIRS is either less than or equal to that of an IRS. The objective of this thesis is to show that for a number of graph classes, an optimal OIRS requires fewer intervals per edge label than an optimal IRS. The graphs examined here include the Petersen graph, k-trees, D-dimensional tori and k-garland graphs (defined in Section 3.4).

The organization of the thesis is as follows. In Chapter 2, we give the formal definitions of IRS, OIRS and their variants and introduce the graphs we investigate later on. This chapter also presents a survey of results regarding the interval routing schemes of these and a few other graphs. Chapter 3 describes optimal OIRS's of selected graphs and shows that the space requirement of an optimal OIRS is less than that of an optimal IRS in each case. In the conclusion in Chapter 4, we summarize our results and propose some open problems.

Chapter 2

Preliminaries and Background

This chapter introduces the terms used in the thesis and provides a list of results in the field of interval routing that are relevant to our work. The chapter starts with the formal definitions of different types of interval routing schemes. Then in the following section, we define the graph classes for which we have investigated our routing scheme. The last section presents a survey of relevant results.

Throughout this thesis, we assume that $G = (V, E)$ is a connected undirected unweighted simple graph that represents a communication network. Let $n = |V|$ and let $\delta(v)$ denote the degree of vertex v. Because each edge of a network is assumed to be bidirectional, we consider it as a pair of directed edges (v, w) and (w, v) (where the direction of the edge (v, w) is from v to w). In our discussion of optimal routing, the *length of a path* means the number of edges in that path. The *distance between vertices v and w*, denoted by $dist(v, w)$, is used to mean the minimum of the lengths of all paths from v to w. Two vertices w_1 and w_2 are said to be *equidistant* from v if $dist(v, w_1) = dist(v, w_2)$.

2.1 Interval Routing Schemes

We elaborate on different types of interval routing schemes in this section. We start with the definition of two relevant terms: cyclic and linear intervals.

For integers i and j, a *cyclic interval* $[i, j]$ with respect to n is the set of consecutive integers between i and j inclusive, considering n and 1 as consecutive. More precisely, the cyclic interval $[i, j]$ is the set $\{l : i \leq l \leq j\}$ if $i \leq j$, and the set $\{l : j \leq l \leq n\} \cup \{l : 1 \leq l \leq i\}$ otherwise. A *linear interval* $[i, j]$ with respect to n is the set of consecutive integers between i and j inclusive, considering n and 1 as non-consecutive. In other words, the linear interval $[i, j]$ is the set $\{l : i \leq l \leq j\}$ if $i \leq j$, and an empty set otherwise.

An interval routing scheme labels each vertex of G with an integer address and each edge with a set of these addresses in a way that, if the processor at vertex v has a packet P to be sent to another vertex w, the processor knows which edge should be used to send P so that P will ultimately reach vertex w. Formally:

Definition 1 (Interval Routing Scheme [Gav00]) *An* Interval Routing Scheme (IRS) on G *consists of a* vertex labeling L *and an* edge labeling I *such that:*

(i) each vertex $v \in V$ is assigned a unique label $L(v)$ from the set $\{0, 1, \ldots, n-1\}$;

(ii) each edge $e \in E$ is assigned a set $I(e)$ of disjoint subintervals $\mathcal{I}$ of the cyclic interval $\{0, 1, \ldots, n-1\}$ in a way that, for each v, the intervals associated with the outgoing edges of v form a partition of the set $\{0, 1, \ldots, n-1\}$ (possibly excluding $L(v)$); and

(iii) for every pair (v, w) of distinct vertices of V, if there is an outgoing edge e adjacent to v such that $L(w) \in \mathcal{I}$ for some interval $\mathcal{I} \in I(e)$, then there exists a path from v to w having e as the first edge.

In the rest of the thesis, we use "$L(v) \in I(e)$" to denote "$L(v) \in \mathcal{I}$ for some interval $\mathcal{I} \in I(e)$". Also, we use $I(v, w)$ instead of $I((v, w))$ to denote the label of the edge (v, w).

The *compactness* of an IRS of a graph G is the maximum, over all edges (v, w) of G, of the number of intervals in $I(v, w)$. An interval routing scheme of compactness k is called a k-interval routing scheme (k-IRS).

A *routing function* [Gav00] maps every pair (v, w) of distinct vertices of V to a path in G that starts at v and ends at w. For a pair (v, w) of distinct vertices of V, we can apply condition (iii) above to vertex v to get a vertex v_1 that is next to v on a path from v to w. We can apply the condition on v_1 to get the vertex v_2 next to v_1 on that path. The path $(v, v_1, v_2, \ldots, w)$ thus formed by repeated application of condition (iii) is called the *path from v to w induced by the IRS*. The set of paths between all pairs of distinct vertices of G defines a *routing function induced by the IRS*. An IRS is *optimal* if the path induced by the IRS from any vertex v to any other vertex w is a shortest path.

There are several variants of IRS's [EMZ02]. An IRS is called a *Linear Interval Routing Scheme* (LIRS) if every interval in the edge labels is a linear interval. An IRS is called a *Strict Interval Routing Scheme* (SIRS) if, for every edge $(v, w) \in E$, $I(v, w)$ does not contain $L(v)$. A *Strict Linear Interval Routing Scheme* (SLIRS) is an IRS that is both linear and strict. The terms k-LIRS, k-SIRS and k-SLIRS, as well as their optimality, are defined similarly.

We note one relation between an IRS and a LIRS of a graph G [Gav00]. If G has a k-IRS, then it may not have a k-LIRS since one cyclic interval is equal to at most two linear intervals. However, G has a $(k + 1)$-LIRS in this case. This is because in any edge label of an IRS, at most one of the k intervals is equal to two linear intervals, and each of the other intervals is equal to one linear interval. On the other hand, a k-LIRS of G is also a k-IRS of G.

In this thesis, we deal with a new type of routing scheme in which the edge labels are considered in a predetermined order. It is natural that the processor at a vertex v considers the edge labels in a particular order. In an IRS, because the labels of outgoing edges at v are mutually disjoint, this order is not important at all; the routing function induced by a given IRS remains the same regardless of the order used at any vertex. Our scheme, as defined below, allows overlap in edge labels, which makes the edge ordering significant. Let $\pi_v = (\pi_v(1), \pi_v(2), \ldots, \pi_v(\delta(v)))$ denote an ordered set where, for each $1 \leq i \leq \delta(v)$, $\pi_v(i)$ denotes a distinct outgoing edge at vertex v.

Definition 2 (Ordered Interval Routing Scheme) *An* Ordered Interval Routing Scheme (OIRS) on G *consists of a vertex labeling L, an edge labeling I and an edge ordering π_v for each $v \in V$, such that:*

(i) each vertex $v \in V$ is assigned a unique label $L(v)$ from the set $\{0, 1, \ldots, n-1\}$;

(ii) each edge $e \in E$ is assigned a set $I(e)$ of subintervals of the cyclic interval $\{0, 1, \ldots, n-1\}$ in a way that, for each v, the union of the intervals associated with the outgoing edges of v forms the set $\{0, 1, \ldots, n-1\}$; and

(iii) for every pair (v, w) of distinct vertices of V, if there is an integer i, $1 \leq i \leq \delta(v)$, such that $L(w) \in I(\pi_v(i))$ and $L(w) \notin I(\pi_v(j))$ for all $1 \leq j < i$, then there exists a path from v to w having $\pi_v(i)$ as the first edge.

The terms OLIRS, k-OIRS and k-OLIRS, as well as their optimality, are defined in the same way as their IRS counterparts. In an OIRS, as we allow overlaps in the intervals on the edges adjacent to any vertex, Ordered *Strict* IRS and its variants make no sense.

As in the case of IRS and LIRS, if a graph G has a k-OIRS, then it may not have a k-OLIRS. Unlike IRS and LIRS, G may not even have a $(k+1)$-LIRS in this case, because an edge label of an OIRS can contain more than one cyclic interval each of

which is equal to two linear intervals. On the other hand, if G has a k-OLIRS, then G has a k-OIRS.

We note one point regarding the relative compactness of IRS's and OIRS's. Given a k-IRS (k-LIRS) of a graph G, we can readily devise a k-OIRS (k-OLIRS) of G by just assigning an arbitrary order to the outgoing edges at each vertex of G. Therefore, for any given graph, the smallest compactness that can be achieved by an OIRS (OLIRS) cannot be greater that the one achieved by an IRS (LIRS).

In the rest of our discussion about OIRS's and OLIRS's, we use the sentence "edge (v, w) covers the set S of vertices" to mean that for all the destinations in S and no other vertices, the processor at v routes a packet through the edge (v, w).

2.2 Definitions of Selected Graphs

In this section, we present the graph classes that we study in the next chapter. The graphs we introduce here are k-trees, D-dimensional tori, k-garland graphs and the Petersen graph. Our results in Chapter 3 show that for each of these graphs, the compactness of an OIRS is less than that of an IRS. We also define a few other graphs that we mention in Section 2.3.

One graph that we investigate in Chapter 3 is a *k-tree*. The notion of k-trees is one generalization of the concept of trees [BLS99]. We know that any tree can be generated by starting with a single vertex and then repeatedly adding a new vertex, making the new vertex adjacent to another vertex in the existing tree. In a k-tree, instead of starting with *one* vertex and making each new vertex adjacent to *one* other vertex, we start with a k-clique and make each new vertex adjacent to the vertices of a k-clique in the existing k-tree. We give a formal definition of a k-tree in Section 3.2.1.

For example, the graph in Figure 2.1 can be constructed as follows. We start with the 3-clique formed by $\{v_1, v_2, v_3\}$ and then add vertex v_4, making it adjacent to the

vertices in the 3-clique $\{v_1, v_2, v_3\}$. Then we add vertex v_5, making it adjacent to the vertices in the 3-clique $\{v_1, v_3, v_4\}$. Following the same way, we add the vertices v_6, v_7 and v_8 one by one in this order such that at the time each of them is added, its neighbors form a 3-clique. Therefore, the graph is a 3-tree. Note that the same 3-tree can be formed by starting with a different 3-clique or adding the vertices in a different order.

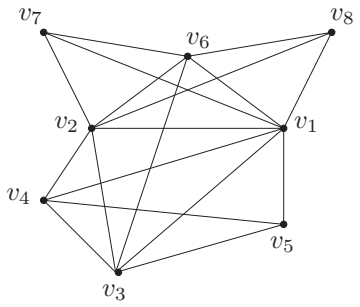


Figure 2.1: A 3-tree

We now introduce a D-dimensional torus, which is shown in Chapter 3 to have smaller compactness in an OIRS than in an IRS. We first give the idea of a D-dimensional grid. A *D-dimensional grid of dimensions d_1, d_2, ..., d_D* [BvLT91] has its vertices at integer coordinates $(x_1, x_2, \ldots, x_D)$ such that $0 \leq x_i < d_i$ for all $1 \leq i \leq D$. There is an edge between two vertices if and only if the coordinates of the two vertices differ in exactly one index by one. Figure 2.2(a) illustrates a 2-dimensional grid. A *D-dimensional hypercube* is a special case of the D-dimensional grid where $d_i = 2$ for all i [BvLT91].

In *a D-dimensional torus of dimensions d_1, d_2, ..., d_D* [BvLT91], there are additional edges that connect vertices on the "opposite sides" of the grid. More precisely, a torus has an edge between vertices $v = (x_1, x_2, \ldots, x_D)$ and $w = (y_1, y_2, \ldots, y_D)$ if and only if there exists an i such that $x_i = (y_i \pm 1) \bmod d_i$ and $x_j = y_j$ for all $j \neq i$. Clearly, it has $n = \prod_{i=1}^{D} d_i$ vertices and $m = Dn$ edges. Each of its vertices has degree $2D$. Figure 2.2(b) shows a 2-dimensional torus of dimensions 5 and 7. It has

35 vertices and 70 edges and the degree of any vertex is 4.

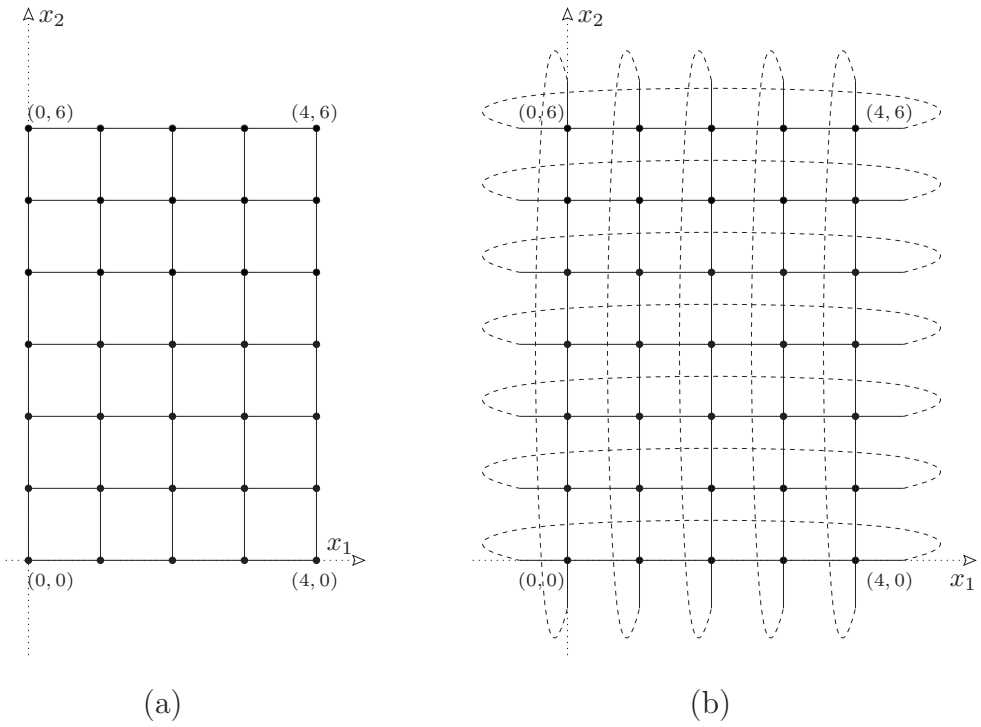


Figure 2.2: A 2-dimensional grid and a 2-dimensional torus, each of dimensions 5 and 7

In Chapter 3, we define a new class of graphs called *k-garland graphs*. A k-garland graph has k *base* vertices $b_1, b_2, \ldots, b_k$ that form a simple path. All other vertices are adjacent to either one or two of the base vertices. If a non-base vertex has two base vertices as its neighbor, then the two base vertices are adjacent to each other. The graph in Figure 2.3 is a 4-garland graph. We give the formal definition in Section 3.4 where we show that this graph has an optimal 2-OLIRS, although it may not have an optimal 2-IRS.

The graph in Figure 2.4 is called the *Petersen graph* [GM84]. This is another graph we investigate in Chapter 3.

In the rest of this section, we introduce a few other graphs that are known to have

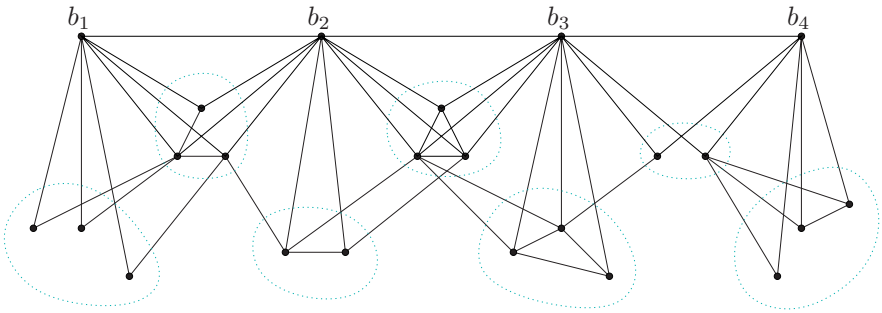


Figure 2.3: A 4-garland graph

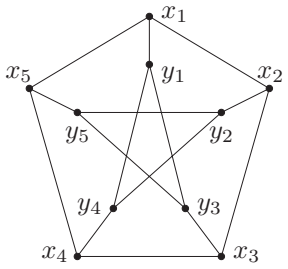


Figure 2.4: The Petersen graph

optimal IRS's of compactness less than three. The definitions of these graphs can be found in many books of graph theory [BLS99, BM76, Die00]. We mention the exact compactness of the optimal IRS's of these graphs in Section 2.3.

A *bipartite graph* is one whose vertices can be partitioned into two subsets so that the two endpoints of each edge are in different subsets. In a *complete bipartite graph*, each vertex in one subset is adjacent to all vertices in the other subset.

A graph is called an *interval graph* if each vertex of the graph can be associated with an interval so that any two vertices are adjacent if and only if the corresponding intervals are overlapping. If each of the intervals associated with the vertices of an interval graph is of unit length, the graph is called a *unit interval graph*.

A bridge of a connected graph G is an edge whose removal disconnects G. A *lithium graph* is a graph with three bridges that connect a connected component with

three other distinct connected components of at least two vertices [FG98]. Figure 2.5 shows one (in fact, the smallest) lithium graph and the structure of a lithium graph.

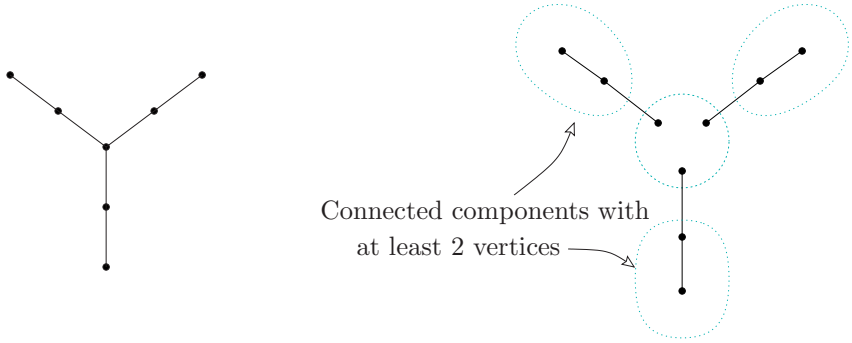


Figure 2.5: A lithium graph and the structure of a lithium graph

A graph is said to be *planar* if it can be drawn in the plane so that its edges intersect only at their endpoints. An *outerplanar graph* is a planar graph that can be drawn on a plane in such a way that all vertices lie on the boundary of the outer face.

2.3 Related Work

In this section, we give a brief survey of results on the compactness of interval routing schemes, emphasizing the results pertaining to the graphs investigated in Chapter 3.

When routing along a shortest path is not required, it is easy to determine an IRS of compactness one for *any* graph. Using a depth first search to assign vertex labels, Santoro and Khatib [SK85] show that every acyclic digraph has a 1-SIRS. Obviously, this technique can be used to get a 1-SIRS for a tree. Hence, by labeling the spanning tree of a graph, we can show that every connected graph has a 1-SIRS. In fact, any graph can be shown to have a 1-SIRS such that all the edges have non-empty labels [vLT87]. However, the same is not true for LIRS's. Fraigniaud and

Gavoille [FG98] prove that a graph has a LIRS of compactness one if and only if it is not a lithium graph. In Section 3.5, we show three lithium graphs having OLIRS's of compactness one.

For the case of optimal routing, the dependence of the compactness of an IRS (and its variants) on the graph is much more complicated. The problem of characterizing graphs which have optimal IRS's (LIRS's, SIRS's, SLIRS's) of compactness one or two is NP-complete [EMZ02, FGS96]. Given a graph G, determining the minimum k such that G has an optimal k-IRS is NP-hard [FGS96]. A short survey on similar problems can be found in a paper by Eilam et al. [EMZ02].

One implication of our results in Chapter 3 is that for certain values of k, the set of graphs having optimal k-IRS's (or its variants) is a subset of the the set of graphs having optimal k-OIRS's (or its variants). In other words, optimal routing using k intervals per edge label is possible for a larger set of graphs in OIRS than in IRS. In this context, we mention the following results to give the reader an idea of the set of graphs admitting k-IRS's or its variants. We assume that the reader is familiar with the graphs mentioned but not defined here. The definitions of these graphs can be found in the books of Bondy and Murty [BM76] and Brandstädt et al. [BLS99]. The classes of graphs that are known to have optimal LIRS's of compactness one include paths, complete graphs, D-dimensional hypercubes and grids [BvLT91], complete bipartite graphs with each partition of size at least two [KKR94] and unit interval graphs [FG98]. Graphs with optimal 1-IRS's include trees (in fact, any outerplanar graph) [FJ88], cycles [SK85], interval graphs and complete bipartite graphs [NS98]. All D-dimensional tori have optimal 2-LIRS's, but not all of them have optimal 1-IRS's [FG98]. The Petersen graph [Gav00] and 3-trees [NN98] need at least three intervals per edge label in any IRS. Any k-tree has a IRS of compactness 2^{k+1}, but it is not known whether the bound is tight or not [NN98]. In a number of subclasses of planar graphs [FJ88, GP96, KRŠ00], there exist graphs that need compactness proportional to $\sqrt{n}$ in any IRS. Many other results concerning different interval routing

schemes on various graph classes can be found in the survey by Gavoille [Gav00].

Now we elaborate on the results related to the graphs investigated in the next chapter.

Narayanan and Nishimura [NN98] show that there is an optimal 2^{k+1}-IRS for any k-tree. Their vertex and edge labeling depends on an ordering of the vertices that reflects the step-by-step construction of the k-tree. First, all the vertices are hierarchically partitioned into *clusters* using that vertex ordering. Vertices are labeled by 1, 2, ..., n in an order which is a function of the clustering. The edge labels are assigned using both the vertex order and cluster hierarchy in a non-trivial way. Details of their scheme are given in Section 3.2, where we utilize the concepts of Narayanan and Nishimura to devise an OIRS of compactness 2^{k-1}. For a 2-tree, Narayanan and Nishimura modify the above scheme to devise an optimal 3-IRS. They also prove that three is a tight bound on the number of intervals required for IRS's of 2-trees. They prove this by showing a 2-tree that does not have an optimal 2-IRS. In Section 3.4, we use this particular 2-tree as a counterexample to show that for a k-garland graph, the compactness needed in an OLIRS is less than the compactness needed in an IRS.

Bakker et al. [BvLT91] prove that none of the graphs in Figure 2.6 has an optimal 1-LIRS. They also prove that a cycle of more than four vertices has no optimal 1-LIRS. In fact, according to Bakker et al. [BvLT91], a graph that contains any of the above graphs as a subgraph of shortest paths does not have optimal 1-LIRS's. A graph G' is a *subgraph of shortest paths* of a graph G if and only if G' is a subgraph of G, and all the shortest paths of G between any pair of vertices of G' are contained in G'. If G has an optimal k-IRS (or optimal k-LIRS), an optimal k-IRS (or optimal k-LIRS) of G' can be obtained by first labeling the vertices and edges of G' with the corresponding labels of G and then modifying the labels to omit the "extra" integers [FG98]. We show in Section 3.5 that all cycles and all the graphs in Figure 2.6 have optimal 1-OLIRS's. Consequently, although a graph that contains any of the graphs of Figure 2.6 or a

cycle of more than four vertices as a subgraph of shortest paths has no optimal LIRS of compactness one, it may have an optimal OLIRS of compactness one.

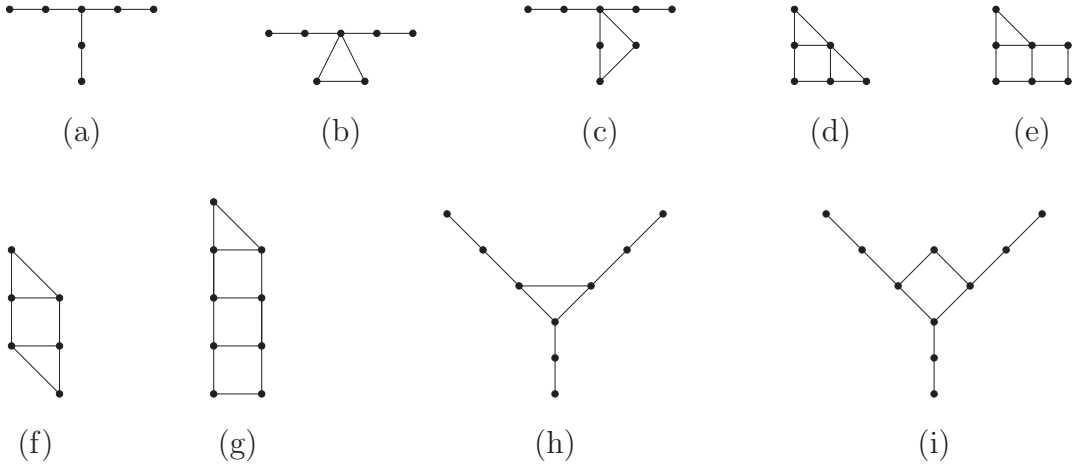


Figure 2.6: Graphs with no optimal 1-LIRS's

A D-dimensional torus has an optimal 1-LIRS (also an optimal 1-SLIRS) if and only if its largest dimension is of size at most four [BvLT91]. One direction of the proof is obvious: if any dimension of a torus has size greater than four, the torus contains a cycle of more than four vertices as a subgraph of shortest paths, and a graph containing such a cycle has no optimal 1-LIRS. The proof for the other direction is not so straightforward and not relevant to our discussion, hence omitted. Fraigniaud and Gavoille [FG98] extend this result using the concept of Cartesian product. A graph $G = (V, E)$ is called the *Cartesian product* of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ if $V = V_1 \times V_2$ and $E = \{(u, v) : u = (u_1, u_2), v = (v_1, v_2)$ and either $u_1 = v_1$ and $(u_2, v_2) \in E_2$ or $u_2 = v_2$ and $(u_1, v_1) \in E_1\}$ [Gav00]. Fraigniaud and Gavoille give several formulas to compute the compactness of the Cartesian product of the graphs G_1 and G_2 from the compactness of G_1 and G_2. Using the fact that a torus can be expressed as a Cartesian product of cycles, they show that a torus has an optimal 1-IRS (also an optimal 1-SIRS) if and only if its second largest dimension is of size at most four, and that any torus has an optimal 2-SLIRS. All these results

imply that if any dimension of a torus has size greater than four, then the torus has no optimal LIRS of compactness one, and if any two of its dimensions have sizes greater than four, then it has no optimal IRS of compactness one. In Section 3.3, we show that *any* torus has an optimal 1-OLIRS, which means that it has also an optimal 1-OIRS.

By exhaustive computation of all possible labelings, Gavoille [Gav00] shows that any IRS or LIRS or SIRS or SLIRS of the Petersen graph needs at least three intervals per edge labels. We give an optimal 2-OLIRS of this graph in Section 3.1.

One major difference between an IRS and an OIRS is that an IRS does not allow any overlap in the labels of the outgoing edges at a vertex, but an OIRS allows such overlaps. One variant of IRS that allows non-disjoint edge labels at each vertex is an *α-adaptive k-IRS*, in which every destination address appears in exactly α different outgoing edges at each vertex v (provided the outdegree of v is at least α), and a packet randomly chooses any of the edges containing its destination address. Gavoille and Zemmari [GZ03] prove that every graph has an *optimal α-adaptive k-IRS* of compactness $\frac{n}{\alpha}$. An α-adaptive k-IRS allows more flexible routing for dynamic traffic in a network. Another variant of IRS that allows non-disjoint edge labels at each vertex is *all shortest paths IRS* that represents all possible shortest paths between every pair of vertices [FGNT01]. We omit details of these variants, since they are completely different from an OIRS. The survey by Gavoille [Gav00] covers these variants also.

In the next chapter, we elaborate on optimal OIRS's of k-trees, k-garland graphs, D-dimensional tori, the graphs in Figure 2.6 and the Petersen graph. We show that for each of these graphs, the number of intervals needed for an OIRS (or OLIRS) is less than the number of intervals needed for an IRS (or LIRS).

Chapter 3

OIRS's of Selected Graphs

In this chapter, we investigate ordered interval routing schemes of selected graphs. Our aim is to show that the compactness of an optimal OIRS or OLIRS is less than that of an optimal IRS or LIRS. We start with an easy case: optimal routing in the Petersen graph. Then we study optimal routing in k-trees, D-dimensional tori and k-garland graphs. Finally we show optimal 1-OLIRS's of ten simple graphs that are known to have no optimal 1-LIRS's.

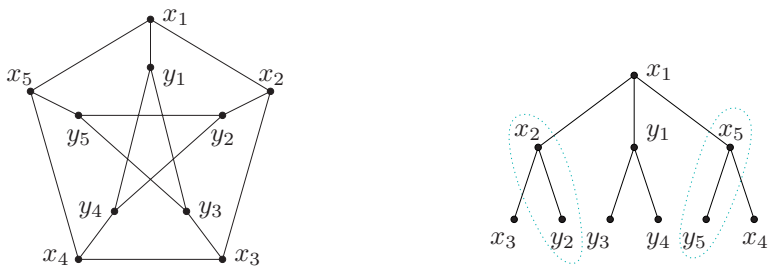
3.1 The Petersen Graph

The graph in Figure 3.1(a) is called the Petersen graph [GM84]. Gavoille [Gav00] shows that any optimal IRS of the Petersen graph has compactness at least three. In the following, we show that the graph has an optimal 2-OLIRS.

The vertex labels in the Petersen graph of Figure 3.1(a) are assigned in such a way that for all i, x_i and y_i have consecutive labels. Consider the shortest paths from x_1 to all other vertices, which is depicted in the shortest path tree of Figure 3.1(b). We label the edge (x_1, x_2) with two intervals: one containing only x_3 and another

containing only x_2 and y_2. The edge (x_1, x_5) is also labeled with two intervals: one containing only x_4 and another containing only x_5 and y_5. The label of the edge (x_1, y_1) contains the labels of *all* vertices. It is obvious that the intervals in the labels of the edges (x_1, x_2) and (x_1, x_5) are mutually disjoint and provides shortest path routing for all vertices except y_1, y_3 and y_4. If the edge ordering π_{x_1} at x_1 ends with the edge (x_1, y_1), then regardless of the relative order of the two other edges, we get shortest path routing from x_1 to all vertices. The case is similar when the shortest paths from any other vertex is considered.

Since each of the edges is labeled with at most two linear intervals, our scheme is an optimal 2-OLIRS.



(a) (b)

Figure 3.1: The Petersen Graph and the shortest path tree at vertex x_1

3.2 k-trees

In this section, we show that any k-tree has an optimal 2^{k-1}-OIRS. Our scheme is based on the optimal interval routing scheme of Narayanan and Nishimura [NN98]. In the following, we give our result following definitions and the labeling scheme of Narayanan and Nishimura.

3.2.1 Definitions

In this subsection, we give the formal definition of a k-tree. We also illustrate here the concepts of *cluster* and *cousin* of a vertex as defined by Narayanan and Nishimura [NN98]. Our labeling scheme in Section 3.2.3 uses these concepts.

Definition 3 (k-Tree [BLS99]) *For $k > 0$:*

(i) A clique with k vertices is a k-tree.

(ii) If G is a k-tree of n vertices and K is a clique with k vertices in G, then the $n+1$ vertex graph G' formed by adding a new vertex v to G and making v adjacent to all vertices in K is a k-tree.

(iii) There are no further k-trees.

It is clear from the definition that any k-tree has a vertex order that reflects its step-by-step construction. More precisely, the vertices of a k-tree have an order $v_1, v_2, \ldots, v_n$ such that:

(i) $v_1, v_2, \ldots, v_k$ is a k-clique; and

(ii) each vertex v_i $(k < i \leq n)$ has exactly k neighbors before it in the ordering and those neighbors form a clique.

The rest of the discussion in this section assumes that we know a specific vertex order with these properties *a priori*.

Given such an order, Narayanan and Nishimura [NN98] define the following terms. The first k vertices in the vertex order are called the *original vertices*; the rest are called *non-original vertices*. The clique formed by the k neighbors of a vertex v that are before it in the ordering is called the *attachment clique of* v, denoted by $AC(v)$.

Any vertex in $AC(v)$ is a *parent* of v. The vertex v is a *child* of each vertex in $AC(v)$. Given a set S of vertices, $children(S)$ is the set of vertices v such that v is a child of each vertex in S. In other words, $children(S) = \{v : S \subseteq AC(v)\}$. The set of vertices with attachment clique $AC(v)$ are *siblings* of v, denoted by $siblings(v)$. The term *descendant of* v is defined recursively as follows: w is a descendant of v if either $w = v$ or w is a descendant of a child of v.

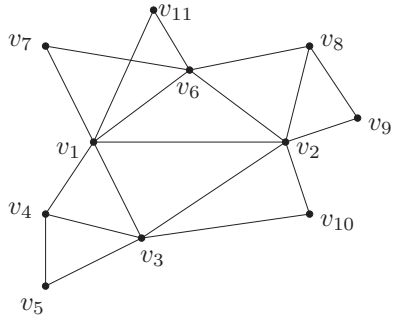


Figure 3.2: A 2-tree

For example, one vertex order corresponding to the step-by-step construction of the 2-tree in Figure 3.2 is $v_1, v_2, \ldots, v_{11}$, although there are many other possibilities. For this order, v_1 and v_2 are the original vertices and the rest are the non-original vertices. Moreover, $AC(v_4) = \{v_1, v_3\}$, v_3 is a parent of v_4, v_4 is a child of v_3, $children(\{v_1, v_6\}) = \{v_7, v_{11}\}$, $sibling(v_7) = \{v_{11}\}$, and v_5 is a descendant of v_1.

Now we focus on the concepts of cluster and cousin. First, we introduce a few relevant terms from Narayanan and Nishimura. The *depth* of a vertex v is zero if it is an original vertex; otherwise it is one greater than the maximum depth of vertices in $AC(v)$. We assign *rank* to the vertices in the following way: the k original vertices are given distinct ranks in the range $1 \ldots k$ arbitrarily. The rank of a non-original vertex is $depth(v) + k$. For any vertex v, the parent with the minimum rank is called the *oldest parent of* v (denoted by $op(v)$).

For the 2-tree of Figure 3.2, using the same vertex order as before, the depths of the vertices v_1, v_2, ..., v_{11} are 0, 0, 1, 2, 3, 1, 2, 2, 3, 2 and 2, respectively, and their ranks are 1, 2, 3, 4, 5, 3, 4, 4, 5, 4 and 4, respectively. Moreover, $op(v_4) = v_1$ and $yp(v_4) = v_3$.

For a non-original vertex v and $p \in \{AC(v) - op(v)\}$, we define the *cousins* of v and p, denoted by $cous(v, p)$, to be the set of vertices b such that p is the vertex with the minimum rank in $AC(v) \cap AC(b)$. For an original vertex v, we define $N(v)$ to be the set of vertices for which v is the oldest parent. For a non-original vertex v, the *cluster of* v, denoted by $cluster(v)$, is the set of descendants of v that are equidistant from all parents of v. For a set S of vertices, $cluster(S) = \bigcup_{v \in S} cluster(v)$.

Again, for the above example, $cous(v_4, v_3) = \{v_5, v_{10}\}$ because the vertex with minimum rank in $AC(v_4) \cap AC(v_5)$ is $\{v_3\}$, and the same argument holds for v_{10} and no other vertices. Since v_2 is the oldest parent of v_8, v_9 and v_{10} and of no other vertices, $N(v_2) = \{v_8, v_9, v_{10}\}$. Also, $cluster(v_3) = \{v_3, v_5\}$ because v_3 has three descendants v_3, v_4 and v_5, and v_4 is the only descendant not equidistant from the vertices in $AC(v_3) = \{v_1, v_2\}$.

3.2.2 Labeling of Narayanan and Nishimura

Narayanan and Nishimura [NN98] give an interval routing scheme that induces a shortest path between every pair of vertices of a k-tree using at most 2^{k+1} intervals per edge label. We split the description into two parts: labeling of the vertices and labeling of the edges.

The vertex labeling scheme of Narayanan and Nishimura uses two orderings of vertices in a non-trivial manner [NN98, Section 4]. We briefly discuss the labeling to clarify two aspects of it that are relevant to our scheme described in Section 3.2.3. Let $f : V \rightarrow \mathcal{N}$ be any one-to-one function that respects the partial order defined by the ranks of the vertices. In other words, f is such that if $rank(v) < rank(w)$,

then $f(v) < f(w)$. Narayanan and Nishimura define an ordering of the non-original vertices as follows: for any two non-original vertices v and w with attachment cliques $(v_1, v_2, \ldots, v_k)$ and $(w_1, w_2, \ldots, w_k)$ respectively, if

$$(f(v_1), f(v_2), \ldots, f(v_k))$$

is lexicographically smaller than

$$(f(w_1), f(w_2), \ldots, f(w_k)),$$

then v comes before w in the ordering, and if they are lexicographically equal, then v and w can be in any order. We call this ordering the *intermediate ordering*. Now, let x_1, x_2, ..., x_k be an arbitrary order of the original vertices. Another vertex ordering defined by Narayanan and Nishimura involves all the vertices. We call this ordering the *final ordering*. In the final ordering, vertices are in the order given below:

$$x_1, cluster(N(x_1)), x_2, cluster(N(x_2)), \ldots, x_k, cluster(N(x_k))$$

where the ordering of the vertices within each set $cluster(N(x))$ is as follows. For any two distinct vertices v and w in $N(x)$, if v comes before w in the intermediate ordering, then $cluster(v)$ precedes $cluster(w)$ in the final ordering. The ordering of the vertices in a cluster is defined (recursively) as follows: for any two distinct children v and w of u that are in $cluster(u)$, if v comes before w in the intermediate ordering, then $cluster(v)$ precedes $cluster(w)$ in the final ordering. For any (non-original) vertex v, v is the first vertex in $cluster(v)$ in the final ordering. From the final ordering, the vertex labels are assigned as follows: the vertices are labeled with 0, 1, ..., $n-1$ in the same order they appear in the final ordering.

We note the following observations about the vertex labeling which are obvious from the way the labels are assigned. We refer to them in the proof of Lemma 3.2.2.

Observation 3.2.1 *For any original vertex x, the labels of the vertices in the set $\{x\} \cup cluster(N(x))$ occupy a single interval.*

Observation 3.2.2 *For any non-original vertex v, the labels of the vertices in the set cluster(v) occupy a single interval.*

Another observation about the above vertex labeling that we use in the analysis of our scheme in Section 3.2.3 is as follows. It is proven in the first part of Lemma 19 of Narayanan and Nishimura [NN98].

Observation 3.2.3 *For any non-original vertex v and any one of its parents $w \in AC(v) - \{op(v)\}$, the labels of the vertices in $\{w\} \cup cluster(cous(v, w))$ occupy at most 2^{k-1} intervals.*

Before going through the details of the edge labeling scheme of Narayanan and Nishimura, we first make a classification of the edges for ease of discussion, as follows. We categorize an outgoing edge (v, w) at vertex v into one of the following four classes:

Class A**:** w is a child of v

Class B**:** both v and w are original vertices

Class C**:** $w \in AC(v) - \{op(v)\}$

Class D**:** $w = op(v)$

Note that for a class A edge (v, w), w is a non-original vertex since all child vertices are non-original by definition. Also, if (v, w) is a class C or class D edge, v is non-original.

It is easy to show that each outgoing edge (v, w) at v belongs to exactly one of the classes. When v is an original vertex, if w is non-original, then (v, w) is in class A; otherwise (v, w) is in class B. On the other hand, when v is a non-original vertex, w can be either its child or its parent. If w is a child of v, the edge belongs to class A. If w is a parent of v, the edge belongs to class D or class C depending on whether

w is the oldest parent of v or not. Thus, the four classes form a partition of the set of all outgoing edges at a given vertex v.

The edge labeling scheme of Narayanan and Nishimura assigns the label of an edge (v, w) as follows, depending on the class it belongs to. Note that in the rest of Section 3.2.2 and also in Section 3.2.3, we mention an edge label as a set of vertices for simplicity, although the edge label actually consists of the *labels* of the vertices of that vertex set.

Class A**:** $cluster(w)$

Class B**:** $\{w\} \cup cluster(N(w)) - cluster(children(v))$

Class C**:** $\{w\} \cup cluster(cous(v, w)) - cluster(children(v))$

Class D**:**

$$V - \left(\{v\} \cup cluster(children(v)) \cup \bigcup_{u \in P_v} (\{u\} \cup cluster(cous(v, u))) \right)$$

where $P_v = AC(v) - \{op(v)\}$

Narayanan and Nishimura prove that the above labeling gives an IRS that induces a shortest path between any pair of vertices, and that each edge label contains at most 2^{k+1} intervals [NN98, Theorem 1].

3.2.3 Labeling and Ordering in the OIRS

To show that any k-tree has an optimal 2^{k-1}-OIRS, we first introduce our labeling and edge ordering scheme and then prove that it induces a shortest path between every pair of vertices of a k-tree and that it has at most 2^{k-1} intervals per edge label.

In our optimal 2^{k-1}-OIRS, we use the same vertex labels as in the work of Narayanan and Nishimura (discussed in Section 3.2.2). Now, using the edge classification we introduced in Section 3.2.2, we define the label of an edge (v, w) as follows, depending on the class it belongs to:

Class A**:** $cluster(w)$

Class B**:** $\{w\} \cup cluster(N(w))$

Class C**:** $\{w\} \cup cluster(cous(v, w))$

Class D**:** V

Finally, we order the outgoing edges (v, w) at each vertex v as follows: the edges in class A comes first, followed by the edges in class B, followed by class C edges and then the single edge in class D. Within a class, edges are ordered arbitrarily.

3.2.4 Proving the Optimality of Our Scheme

To prove that any k-tree has an optimal 2^{k-1}-OIRS, we first show that both our scheme and the scheme of Narayanan and Nishimura induce the same routing function. Then we determine the number of intervals needed in each edge label of our scheme.

Lemma 3.2.1 *Between any pair of vertices, the path induced by our scheme is the same as the one induced by the routing scheme of Narayanan and Nishimura.*

Proof: We prove this by modifying the scheme of Narayanan and Nishimura step by step in such a way that the induced path between any pair of vertices remains the same.

First, we start with the scheme of Narayanan and Nishimura and impose an order on the outgoing edges at each vertex in the same way as our scheme, i.e., class A edges come first followed by edges of classes B, C and D in this order. Since the edge labels at each vertex are mutually disjoint, the imposed edge order keeps all the induced paths unchanged. We call this modified scheme the *old scheme*.

Now we modify the labels of all outgoing edges at v in the old scheme to get the *new scheme*: The labels of all class A edges remain the same; we add the vertices $cluster(children(v))$ to all class B and class C edge labels; the class D edge is given the label V. Because the class A edge labels are unchanged and they are considered first, any packet that goes out of vertex v following a class A edge in the old scheme follows the same edge in the new labeling scheme. Note that all the labels of class A edges contain only the vertices in $\bigcup_{w \in children(v)} cluster(w) = cluster(children(v))$. Therefore, no vertex in the set $cluster(children(v))$ can be the destination of a packet that is not forwarded by a class A edge. Hence, if a packet goes out of vertex v following a class B or class C edge in the old scheme, no vertex in the set $cluster(children(v))$ can be its destination. So, such a packet follows the same edge in the new scheme. All the packets not forwarded by any class A, B or C edge are going to follow the single edge in class D anyway. So, setting this label to V in the new scheme has no effect on routing of these packets.

Finally, note that the new scheme is exactly the same as our scheme. This implies that all the paths induced by the scheme of Narayanan and Nishimura are unchanged in our scheme. □

Lemma 3.2.2 *In our edge labeling scheme, one edge label consists of at most* 2^{k-1} *intervals.*

Proof: The label of a class A edge consists of the cluster of a single vertex. This, along with Observation 3.2.2, implies that a class A edge label forms a single interval. Also, for any original vertex x, $\{x\} \cup cluster(N(x))$ forms a single interval by Observation 3.2.1, which implies that a class B edge forms a single interval. The size of a class C edge label is obtained from Observation 3.2.3: for the edge (v, w) $(w \in AC(v) - \{op(v)\})$, the set $\{w\} \cup cluster(cous(v, w))$ occupies at most 2^{k-1} intervals. The label of the class D edge clearly consists of one interval. Therefore, an edge label in our scheme consists of at most 2^{k-1} intervals. □

Theorem 1 *Every k-tree has an optimal 2^{k-1}-OIRS.*

Proof: The path induced by the scheme of Narayanan and Nishimura between any pair of vertices is a shortest path [NN98, Lemmas 10 to 15], and it is the same as the one induced by our scheme by Lemma 3.2.1. So, our scheme, too, induces shortest paths. Therefore, by Lemma 3.2.2, our scheme is an optimal 2^{k-1}-OIRS. □

3.3 Tori

According to Bakker et al. [BvLT91], if any dimension of a torus has size greater than four, then the torus has no optimal 1-LIRS. Also, according to Fraigniaud and Gavoille [FG98, Theorem 14], if any two dimensions of a torus have sizes greater than four, then the torus has no optimal 1-IRS. In this section, we prove that *any* torus has an optimal OLIRS of compactness one, which implies that it has also an optimal OIRS of compactness one. The section is organized as follows. First of all, we give the formal definition of a torus. Then in Sections 3.3.1 and 3.3.2, we elaborate on assigning vertex and edge labels and edge orders at the vertices of a torus that gives an OLIRS. The proof that the OLIRS is an optimal 1-OLIRS appears in Section 3.3.3.

Definition 4 (D-dimensional torus [BvLT91]) *A D-*dimensional torus *of dimensions d_1, d_2, ..., d_D is a graph consisting of $n = \prod_{i=1}^{D} d_i$ vertices at coordinates $(x_1, x_2, \ldots, x_D)$ with $0 \leq x_i < d_i$ for each $1 \leq i \leq D$, which has an edge between vertices $v = (x_1, x_2, \ldots, x_D)$ and $w = (y_1, y_2, \ldots, y_D)$ iff there exists an i such that $x_i = (y_i \pm 1) \bmod d_i$ and $x_j = y_j$ for all $j \neq i$.*

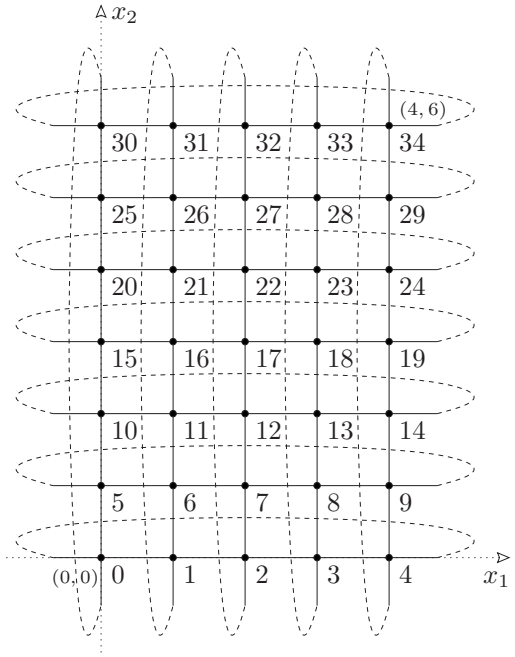
3.3.1 Labeling Vertices in the OLIRS

In the OLIRS of a torus, the vertex labels are assigned in the lexicographic order implied by the coordinates of the vertices, considering coordinates from right to left.

More precisely, the label $L(s)$ of the vertex $s = (x_1, x_2, \ldots, x_D)$ can be computed as follows:

$$L(s) = x_1 + d_1 x_2 + d_1 d_2 x_3 + \ldots + d_1 d_2 \ldots d_{D-1} x_D = \sum_{i=1}^{D} W_i x_i$$

where $W_1 = 1$ and $W_i = W_{i-1} d_{i-1}$ for $1 < i \leq D$. Figure 3.3 shows the vertex labeling of two tori using our scheme. In the 2-dimensional torus of Figure 3.3(a), for example, the vertex at coordinate $(1, 4)$ is assigned the label $4 + 5 \times 1 = 21$.



(a) A 2-dimensional torus of dimensions 5 and 7 and its vertex labels

$L(0,0,0) = 0$ $L(0,0,1) = 9$
$L(1,0,0) = 1$ $L(1,0,1) = 10$
$L(2,0,0) = 2$ $L(2,0,1) = 11$
$L(0,1,0) = 3$ $L(0,1,1) = 12$
$L(1,1,0) = 4$ $L(1,1,1) = 13$
$L(2,1,0) = 5$ $L(2,1,1) = 14$
$L(0,2,0) = 6$ $L(0,2,1) = 15$
$L(1,2,0) = 7$ $L(1,2,1) = 16$
$L(2,2,0) = 8$ $L(2,2,1) = 17$

(b) Vertex labels of a 3-dimensional torus of dimensions 3, 3 and 2.

Figure 3.3: Vertex labeling of two tori in the OLIRS

3.3.2 Ordering and Labeling Edges in the OLIRS

For ordering and labeling the outgoing edges at any vertex $s = (x_1, x_2, \ldots, x_D)$, we classify the edges into D classes. Each class C_i, $1 \leq i \leq D$, consists of the two edges

connecting s to the following vertices:

$$\begin{aligned} u_i &= (x_1, x_2, \ldots, x_{i-1}, x_i^-, x_{i+1}, x_{i+2}, \ldots, x_D) \\ w_i &= (x_1, x_2, \ldots, x_{i-1}, x_i^+, x_{i+1}, x_{i+2}, \ldots, x_D) \end{aligned}$$

where x_i^- and x_i^+ denote $(x_i - 1) \bmod d_i$ and $(x_i + 1) \bmod d_i$ respectively.

In the edge ordering at vertex s, the edges of class C_1 come first, followed by those in classes C_2, C_3, ..., C_D in this order. Within class C_i, the order of the two edges depends on the corresponding coordinate: if x_i is greater than or equal to $\lfloor \frac{d_i}{2} \rfloor$, the edge (s, u_i) comes before the edge (s, w_i); otherwise, (s, u_i) follows (s, w_i) in the ordering. From now on, for simplicity, we use h_i to denote $\lfloor \frac{d_i}{2} \rfloor$.

The edge labels are assigned in such a way that at vertex s, the packet addressed to some vertex t is forwarded by a class C_1 edge if only the first coordinates of s and t are different, by a class C_2 edge if the second coordinates of s and t are different but their third and later coordinates are the same, by a class C_3 edge if the third coordinates of s and t are different but their fourth and later coordinates are the same, and so on. In other words, for each $i \in \{1, 2, \ldots, D\}$, class C_i edges at vertex $(x_1, x_2, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_D)$ forward only the packets addressed to vertices $(y_1, y_2, \ldots, y_{i-1}, y_i, x_{i+1}, \ldots, x_D)$, where $y_i \neq x_i$. Within each class C_i, each of the edges (s, u_i) and (s, w_i) covers about half of all the addresses covered by both of them.

Before giving details of our edge labeling, we illustrate the idea with a small example. We show how labels are assigned to the outgoing edges of the vertex labeled with 21 in the 2-dimensional torus of Figure 3.3(a). For ease of discussion, we name the vertices by their labels. Now, $C_1 = \{(21, 20), (21, 22)\}$ and $C_2 = \{(21, 16), (21, 26)\}$. The ordering π_{21} of these edges depends on the coordinates of vertex 21. Since this vertex is at $(1, 4)$, we have $x_1 = 1 < \lfloor \frac{5}{2} \rfloor$ and $x_2 = 4 > \lfloor \frac{7}{2} \rfloor$. These inequalities imply that in π_{21}, the edge $(21, 22)$ comes before $(21, 20)$ and the edge $(21, 16)$ comes before

$(21, 26)$, i.e., $\pi_{21} = ((21, 22), (21, 20), (21, 16), (21, 26))$. The labels of class C_1 edges are such that they forward only the packets with destination vertices having the first coordinates different from that of vertex 21, but second coordinates the same as that of vertex 21. The first edge $(21, 22)$ is labeled with $[22, 23]$ which optimally routes the packets with addresses in this interval. The second edge $(21, 20)$ forwards packets addressed only to 20 and 24. However, because this edge is considered after the edge $(21, 22)$ during routing, we label it with $[20, 24]$ to achieve the same effect. The third edge $(21, 16)$ should carry only the packets with destination vertices in $[5, 19]$. So, we label the edge $(21, 16)$ with the interval $[5, 19]$. Finally, the edge $(21, 26)$ should forward packets addressed only to vertices in $[0, 4] \cup [25, 34]$; we achieve this by labeling the edge with $[0, 34]$ because this edge is considered after all the other edges. The four shaded regions in Figure 3.4 show the labels of the four edges in π_{21}.

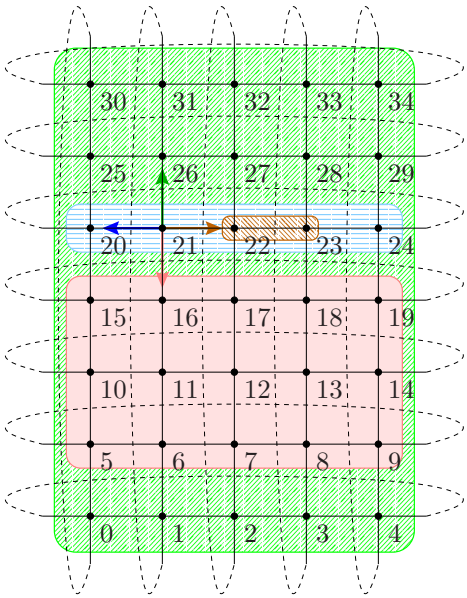


Figure 3.4: The labels of the edges at vertex 21

Now we give the details of the edge labeling. For each $i \in \{1, 2, \ldots, D\}$, the label of the class C_i edge at s that comes after the other edge of the same class in the

ordering is as follows:

$$[\, L(0,0,\ldots,0,0,x_{i+1},x_{i+2},\ldots,x_D),$$
$$L(d_1-1,d_2-1,\ldots,d_{i-1}-1,d_i-1,x_{i+1},x_{i+2},\ldots,x_D)\,].$$

We call this interval the *larger interval of class C_i at s*. In other words, the larger interval of class C_i at s is $I(s,u_i)$ if x_i is less than h_i; otherwise, it is $I(s,w_i)$. Later on, in Lemma 3.3.2, we show the justification for using the name "larger interval". The label of the other class C_i edge depends on the ith coordinate of s. Edge labels of each class C_i are as follows:

Class C_1: If $x_1 \geq h_1$, then the first edge of the class (i.e. (s,u_1)) is labeled with the interval

$$[\, L(x_1-h_1,x_2,x_3,\ldots,x_D),\ L(x_1-1,x_2,x_3,\ldots,x_D)\,]$$

and the second edge (i.e. (s,w_1)) is labeled with the larger interval of class C_1, that is,

$$[\, L(0,x_2,x_3,\ldots,x_D),\ L(d_1-1,x_2,x_3,\ldots,x_D)\,].$$

Otherwise, the first edge of the class (i.e. (s,w_1)) is labeled with the interval

$$[\, L(x_1+1,x_2,x_3,\ldots,x_D),\ L(x_1+h_1,x_2,x_3,\ldots,x_D)\,]$$

and the second edge (i.e. (s,u_1)) is labeled with the larger interval of class C_1.

Class C_2: If $x_2 \geq h_2$, then the first edge of the class (i.e. (s,u_2)) is labeled with the interval

$$[\, L(0,x_2-h_2,x_3,x_4,\ldots,x_D),\ L(d_1-1,x_2-1,x_3,x_4,\ldots,x_D)\,]$$

and the second edge (i.e. (s,w_2)) is labeled with the larger interval of class C_2, that is,

$$[\, L(0,0,x_3,x_4,\ldots,x_D),\ L(d_1-1,d_2-1,x_3,x_4,\ldots,x_D)\,].$$

Otherwise, the first edge of the class (i.e. (s, w_2)) is labeled with the interval

$$L(0, x_2 + 1, x_3, x_4, \ldots, x_D),\ L(d_1 - 1, x_2 + h_2, x_3, x_4, \ldots, x_D)\ ]$$

and the second edge (i.e. (s, u_2)) is labeled with the larger interval of class C_2.

And, in general, for $i = 1, 2, \ldots, D$:

Class C_i: If $x_i \geq h_i$, then the first edge of the class (i.e. (s, u_i)) is labeled with the interval

$$I(s, u_i) = [\ L(0, 0, \ldots, 0, x_i - h_i, x_{i+1}, x_{i+2}, \ldots, x_D),$$
$$L(d_1 - 1, d_2 - 1, \ldots, d_{i-1} - 1, x_i - 1, x_{i+1}, x_{i+2}, \ldots, x_D)\ ]$$

and the second edge (i.e. (s, w_i)) is labeled with the larger interval of class C_i.

Otherwise, the first edge of the class (i.e. (s, w_i)) is labeled with the interval

$$I(s, w_i) = [\ L(0, 0, \ldots, 0, x_i + 1, x_{i+1}, x_{i+2}, \ldots, x_D),$$
$$L(d_1 - 1, d_2 - 1, \ldots, d_{i-1} - 1, x_i + h_i, x_{i+1}, x_{i+2}, \ldots, x_D)\ ]$$

and the second edge (i.e. (s, u_i)) is labeled with the larger interval of class C_i.

3.3.3 Proving the Optimality of Our Scheme

This subsection elaborates the correctness of the above scheme, i.e., the proof that the above scheme is an optimal 1-OLIRS. The outline of the proof is as follows. We first establish two properties of our labeling scheme. Using these properties, we then determine the edge a packet follows to leave the current vertex on the way to its destination. Finally we show that our scheme is a valid OLIRS and that the path followed by a packet is a shortest path.

The following lemma establishes a property of our vertex labels. Intuitively, by this lemma, we can compare the labels of two vertices by comparing only the "rightmost" coordinate at which the two vertices differ.

Lemma 3.3.1 *For any two vertices $s = (x_1, x_2, \ldots, x_D)$ and $t = (y_1, y_2, \ldots, y_D)$ of the torus:*

(i) if there exists some $r \in \{1, 2, \ldots, D\}$ such that $x_r > y_r$ and $x_i = y_i$ for all $i > r$, then $L(s) > L(t)$;

(ii) if $x_i = y_i$ for all $i \in \{1, 2, \ldots, D\}$, then $L(s) = L(t)$;

(iii) if $L(s) > L(t)$, then there exists some $r \in \{1, 2, \ldots, D\}$ such that $x_r > y_r$ and $x_i = y_i$ for all $i > r$; and

(iv) if $L(s) = L(t)$, then $x_i = y_i$ for all $i \in \{1, 2, \ldots, D\}$.

Proof:

(i) We prove this part of the lemma by showing that $L(s) - L(t) \geq 1$:

$$\begin{aligned}
L(s) - L(t) &= \sum_{i=1}^{D} W_i x_i - \sum_{i=1}^{D} W_i y_i = \sum_{i=1}^{D} W_i (x_i - y_i) \\
&= W_r (x_r - y_r) + \sum_{i=1}^{r-1} W_i (x_i - y_i) \text{ because } x_i = y_i \text{ for all } i > r \\
&\geq W_r + \sum_{i=1}^{r-1} W_i (0 - (d_i - 1)) \\
&\quad \text{because } x_r - y_r \geq 1 \text{ and } 0 \leq x_i, y_i \leq d_i - 1 \\
&= W_r + \sum_{i=1}^{r-1} W_i - \sum_{i=1}^{r-1} W_i d_i \\
&= W_r + \sum_{i=1}^{r-1} W_i - \sum_{i=2}^{r} W_i \text{ because } W_i d_i = W_{i+1} \\
&= W_1 = 1
\end{aligned}$$

(ii) This is obvious from the definition of vertex labels.

(iii) We prove this part by contradiction. Suppose that there exists no r such that $x_r > y_r$ and $x_i = y_i$ for all $i > r$. Consequently, exactly one of the following is true:

(a) $x_i = y_i$ for all i; or

(b) there exists some r such that $x_r < y_r$ and $x_i = y_i$ for all $i > r$.

In the first case, by part (ii) of the lemma, $L(s) = L(t)$. In the second case, $L(s) < L(t)$ by part (i). Both are impossible since $L(s) > L(t)$.

(iv) Suppose there exists some r such that $x_r \neq y_r$. Without loss of generality, we can assume that r is the largest such index. Therefore, $x_i = y_i$ for all $i > r$. Now, if $x_r < y_r$, then $L(s) < L(t)$ by part (i) of the lemma, which is a contradiction. The case is similar when $x_r > y_r$.

□

One property of our edge labeling is that the larger interval of any class includes the labels of all the edges that come before it in the edge ordering. The following lemma validates this claim.

Lemma 3.3.2 *For any vertices $s = (x_1, x_2, \ldots, x_D)$ and $t = (y_1, y_2, \ldots, y_D)$, if $L(t)$ is in the interval of a class C_r edge at s, then $L(t)$ is in the larger interval of class C_i at s, for all $i \geq r$.*

Proof: We prove the lemma in two steps. In the first step, we show that $y_i = x_i$ for all $i > r$. We use this fact in the second step to show that $L(t)$ lies between the left and right endpoints of the larger interval of class C_i for all $i \geq r$. This completes the proof.

To show that $y_i = x_i$ for all $i > r$ in the first step, suppose instead that $y_i \neq x_i$ for some $i > r$. Without loss of generality, we can assume that i is the largest index with this property. Therefore, $y_j = x_j$ for all $j > i$. Now, consider the following cases:

Case 1: $y_i < x_i$**:** The left endpoint of the interval of a class C_r edge is at least

$$\begin{aligned} & \min\{L(0,0,\ldots,0,x_r-h_r,x_{r+1},x_{r+2},\ldots,x_D), \\ & \quad L(0,0,\ldots,0,0,x_{r+1},x_{r+2},\ldots,x_D), \\ & \quad L(0,0,\ldots,0,x_r+1,x_{r+1},x_{r+2},\ldots,x_D)\} \\ = \; & L(0,0,\ldots,0,0,x_{r+1},x_{r+2},\ldots,x_D) \end{aligned}$$

by Lemma 3.3.1 because $0 \leq x_r - h_r < x_r + 1$. But, since $y_i < x_i$ and $y_j = x_j$ for all $j > i$, $L(t) = L(y_1, y_2, \ldots, y_D)$ is less than $L(0,0,\ldots,0,0,x_{r+1},x_{r+2},\ldots,x_D)$ (by Lemma 3.3.1). Thus, $L(t)$ is less than the left endpoint of any interval of C_r edge labels, which implies that $L(t)$ is not in the interval of any class C_r edge label. This is a contradiction.

Case 2: $y_i > x_i$**:** Using similar argument, $L(t)$ can be shown to be greater than the right endpoint of any interval of class C_r edge labels, which implies that $L(t)$ is not in the interval of any C_r edge label. This is a contradiction.

This completes the first step of our proof. Now we show that $L(t)$ is in the larger interval of class C_i for all $i \geq r$. The left endpoint of the larger interval of class C_i is as follows:

$$L(0,0,\ldots,0,0,x_{i+1},x_{i+2},\ldots,x_D).$$

When $i \geq r$, this endpoint is less than or equal to $L(t) = L(y_1, y_2, \ldots, y_D)$ by Lemma 3.3.1, because $y_j \geq 0$ for all $j \leq i$ and $y_j = x_j$ for all $j > i$. The right endpoint of the larger interval of class C_i is as follows:

$$L(d_1 - 1, d_2 - 1, \ldots, d_{i-1} - 1, d_i - 1, x_{i+1}, x_{i+2}, \ldots, x_D).$$

When $i \geq r$, this endpoint is greater than or equal to $L(t) = L(y_1, y_2, \ldots, y_D)$ by Lemma 3.3.1, because $y_j \leq d_j - 1$ for all $j \leq i$ and $y_j = x_j$ for all $j > i$. Therefore, $L(t)$ lies between the endpoints (inclusive) of the larger interval of class C_i. □

The following two lemmas characterize the paths defined by our routing scheme. When a packet P with destination vertex t is at another vertex s, P uses an edge e to leave vertex s. We know from the following lemma the class to which edge e belongs.

Lemma 3.3.3 *If the destination vertex of a packet P is $t = (y_1, y_2, \ldots, y_D)$ and P is currently at a vertex $s = (x_1, x_2, \ldots, x_D)$ such that for some $r \in \{1, 2, \ldots, D\}$, $x_r \neq y_r$ and $x_i = y_i$ for all $i > r$, then P uses a class C_r edge to leave vertex s.*

Proof: We have to prove that P does not use a class C_j edge for all $j \neq r$. First we show that this holds for all $j > r$.

The left endpoint of the larger interval of class C_r is:

$$L(0, 0, \ldots, 0, 0, x_{r+1}, x_{r+2}, \ldots, x_D).$$

By Lemma 3.3.1,

$$L(0, 0, \ldots, 0, 0, x_{r+1}, x_{r+2}, \ldots, x_D) \ \leq \ L(t)$$

since $y_i \geq 0$ for all $i \leq r$ and $y_i = x_i$ for all $i > r$. The right endpoint of the larger interval of class C_r is:

$$L(d_1 - 1, d_2 - 1, \ldots, d_{r-1} - 1, d_r - 1, x_{r+1}, x_{r+2}, \ldots, x_D).$$

Again, by Lemma 3.3.1,

$$L(t) \ \leq \ L(d_1 - 1, d_2 - 1, \ldots, d_{r-1} - 1, d_r - 1, x_{r+1}, x_{r+2}, \ldots, x_D)$$

since $y_i \leq d_i - 1$ for all $i \leq r$ and $y_i = x_i$ for all $i > r$. These facts imply that $L(t)$ is in the larger interval of class C_r. Hence, P does not use any edges in classes C_j, $j > r$, since these edges follow the larger edge of class C_r in the edge ordering.

Now we show that for all $j < r$, there is no possibility that P uses any class C_j edge. The larger interval of C_{r-1} is as follows:

$$\begin{aligned}[& L(0, 0, \ldots, 0, 0, x_r, x_{r+1}, \ldots, x_D), \\ & L(d_1 - 1, d_2 - 1, \ldots, d_{r-2} - 1, d_{r-1} - 1, x_r, x_{r+1}, \ldots, x_D) \ ].\end{aligned}$$

We know from the statement of the lemma that $x_r \neq y_r$ and $x_i = y_i$ for all $i > r$. If $y_r < x_r$, then the left endpoint of the above interval is less than $L(t)$; and if $y_r > x_r$, then the right endpoint of the above interval is greater than $L(t)$ (by Lemma 3.3.1). Therefore, the larger interval of class C_{r-1} does not include $L(t)$. Finally, the contrapositive of Lemma 3.3.2 implies that none of the intervals of the edges of class C_j, $j \leq r-1$, contains $L(t)$.

Therefore, P uses a class C_r edge to leave s. □

The above lemma determines the class C_r of the edge through which a packet P leaves its current vertex. Lemma 3.3.4 below identifies the *exact* edge in C_r that P uses. In fact, the edge used by P is the one that ensures that P follows the shortest path to its destination. Therefore, before the lemma, we determine the length of the shortest path between two vertices of a torus. Let a packet P whose destination vertex is $t = (y_1, y_2, \ldots, y_D)$ be currently at a different vertex $s = (x_1, x_2, \ldots, x_D)$. Because the coordinates of any neighbor of s differ from those of s in exactly one coordinate x_i by one (modulo d_i), exactly one coordinate of the current position of P changes by 1 (modulo the length of the corresponding dimension) whenever P "crosses" an edge. Therefore, the length of the shortest path from s to t is equal to the minimum number of times we need to change the coordinates of s, one at a time and by amount one, so that the corresponding coordinates of s and t become the same. When $x_i < y_i$, if we repeatedly add 1 (modulo d_i) to x_i, the number of steps needed to make x_i equal to y_i is $|x_i - y_i|$, and if we repeatedly subtract 1 (modulo d_i) from x_i, the number of steps needed is $d_i - |x_i - y_i|$. The minimum number of changes required is $\min\{|x_i - y_i|, d_i - |x_i - y_i|\}$, which is clearly the same when $x_i > y_i$. So, we get

$$dist(s,t) = \sum_{i=1}^{D} \min\{|x_i - y_i|, d_i - |x_i - y_i|\}.$$

Before proving the next lemma, we note the following observation about the expression $\min\{|x_i - y_i|, d_i - |x_i - y_i|\}$:

Observation 3.3.1 *The value of* $\min\{|x_i - y_i|, d_i - |x_i - y_i|\}$ *is equal to* $|x_i - y_i|$ *if* $|x_i - y_i| < h_i$*, and equal to* $d_i - |x_i - y_i|$ *if* $|x_i - y_i| > h_i$*.*

Proof: Because each of $|x_i - y_i|$ and $d_i - |x_i - y_i|$ is an integer, and because the minimum of them cannot be greater than their average, we get

$$\min\{|x_i - y_i|, d_i - |x_i - y_i|\} \leq \left\lfloor \frac{|x_i - y_i| + d_i - |x_i - y_i|}{2} \right\rfloor = \left\lfloor \frac{d_i}{2} \right\rfloor = h_i.$$

Also, since the sum of $|x_i - y_i|$ and $d_i - |x_i - y_i|$ is equal to d_i and $d_i \geq 2h_i$, it is always the case that exactly one of $|x_i - y_i|$ and $d_i - |x_i - y_i|$ is less than or equal to h_i, and if one of them is less than h_i, the other is greater than h_i. The conclusion follows immediately. □

We make use of this observation repeatedly in the proof of the following lemma.

Lemma 3.3.4 *Let* $t = (y_1, y_2, \ldots, y_D)$ *be the destination vertex of a packet* P *and* $s = (x_1, x_2, \ldots, x_D)$ *be the current position of* P *such that for some* $r \in \{1, 2, \ldots, D\}$*,* $x_r \neq y_r$ *and* $x_i = y_i$ *for all* $i > r$*. If* P *uses the edge* (s, s_1) *to leave vertex* s*, then* $dist(s_1, t) = dist(s, t) - 1$*.*

Proof: By Lemma 3.3.3, (s, s_1) is a class C_r edge. Consequently, s_1 is either u_r or w_r. In the proof below, we only consider the case $s_1 = u_r$, because the proof is similar for the other case.

Because the coordinates of u_r and s differ in only the rth coordinate, we need to consider only the rth coordinates when comparing $dist(u_r, t)$ and $dist(s, t)$. More precisely, it is sufficient to show that

$$\min\{|x_r^- - y_r|, d_r - |x_r^- - y_r|\} = \min\{|x_r - y_r|, d_r - |x_r - y_r|\} - 1.$$

We split our proof into two cases: $x_r \geq h_r$ and $x_r < h_r$.

Case 1: $x_r \geq h_r$**:** The edge (s, u_r) is before (s, w_r) in the edge ordering at s. Now, the fact that P uses edge (s, u_r) implies $L(t) = L(y_1, y_2, \ldots, y_D)$ is contained in the interval of the label of (s, u_r), i.e.,

$$L(y_1, y_2, \ldots, y_D) \geq L(0, 0, \ldots, 0, x_r - h_r, x_{r+1}, x_{r+2}, \ldots, x_D)$$

and

$$L(y_1, y_2, \ldots, y_D) \leq L(d_1 - 1, d_2 - 1, \ldots, d_{r-1} - 1, x_r - 1, x_{r+1}, x_{r+2}, \ldots, x_D).$$

From these inequalities, by Lemma 3.3.1: $y_r \geq x_r - h_r$ and $y_r \leq x_r - 1$. Therefore, $1 \leq x_r - y_r \leq h_r$ which implies, by Observation 3.3.1,

$$\min\{|x_r - y_r|, d_r - |x_r - y_r|\} = x_r - y_r.$$

Also, in this case, $x_r^- = (x_r - 1) \bmod d_r = x_r - 1$. Therefore, from the inequality $1 \leq x_r - y_r \leq h_r$, we get $0 \leq x_r^- - y_r \leq h_r - 1$ which implies, by Observation 3.3.1,

$$\begin{aligned} \min\{|x_r^- - y_r|, d_r - |x_r^- - y_r|\} &= x_r^- - y_r \\ &= x_r - y_r - 1. \end{aligned}$$

Case 2: $x_r < h_r$**:** The edge (s, w_r) is before (s, u_r) in the edge ordering at s. Now, the fact that P uses edge (s, u_r) implies $L(t) = L(y_1, y_2, \ldots, y_D)$ is *not* contained in the interval of the label of (s, w_r), i.e., either

$$L(y_1, y_2, \ldots, y_D) < L(0, 0, \ldots, 0, x_r + 1, x_{r+1}, x_{r+2}, \ldots, x_D)$$

or

$$L(y_1, y_2, \ldots, y_D) > L(d_1 - 1, d_2 - 1, \ldots, d_{r-1} - 1, x_r + h_r, x_{r+1}, x_{r+2}, \ldots, x_D).$$

From these inequalities, by Lemma 3.3.1, either $y_r < x_r + 1$ or $y_r > x_r + h_r$. We consider the cases separately:

Case 2a: $y_r < x_r + 1$**:** Since $y_r \neq x_r$ and $x_r < h_r$, we conclude that $y_r < x_r < h_r$, which implies $0 < x_r - y_r < h_r$. From Observation 3.3.1,

$$\min\{|x_r - y_r|, d_r - |x_r - y_r|\} = x_r - y_r.$$

Also, because $x_r > y_r \geq 0$, $x_r^- = x_r - 1$. Therefore, from the inequality $1 \leq x_r - y_r \leq h_r$: $0 \leq x_r^- - y_r < h_r - 1$. Again, from Observation 3.3.1,

$$\begin{aligned} \min\{|x_r^- - y_r|, d_r - |x_r^- - y_r|\} &= x_r^- - y_r \\ &= x_r - y_r - 1. \end{aligned}$$

Case 2b: $y_r > x_r + h_r$**:** In this case, $y_r - x_r > h_r$. So, from Observation 3.3.1,

$$\min\{|x_r - y_r|, d_r - |x_r - y_r|\} = d_r - y_r + x_r.$$

Now, we consider the cases $x_r > 0$ and $x_r = 0$ separately.

When $x_r > 0$, $x_r^- = x_r - 1$ and $y_r - x_r^- = y_r - x_r + 1 > h_r + 1$. Therefore, from Observation 3.3.1,

$$\begin{aligned} \min\{|x_r^- - y_r|, d_r - |x_r^- - y_r|\} &= d_r - y_r + x_r^- \\ &= d_r - y_r + x_r - 1. \end{aligned}$$

When $x_r = 0$, $x_r^- = (x_r - 1) \bmod d_r = d_r - 1$. Since $y_r > x_r + h_r \geq h_r$ and $y_r \leq d_r - 1$, the range of possible values of y_r is as follows: $h_r < y_r \leq d_r - 1$. By subtracting the maximum and the minimum possible value of y_r from x_r^-, we get $0 \leq x_r^- - y_r < d_r - 1 - h_r$. This bound can be simplified to $0 \leq x_r^- - y_r < h_r$ using the fact that d_r is at most $2h_r + 1$. Therefore, from Observation 3.3.1,

$$\begin{aligned} \min\{|x_r^- - y_r|, d_r - |x_r^- - y_r|\} &= x_r^- - y_r \\ &= d_r - 1 - y_r + x_r \text{ since } x_r = 0. \end{aligned}$$

In all the cases, we have shown that

$$\min\{|x_r^- - y_r|, d_r - |x_r^- - y_r\} = \min\{|x_r - y_r|, d_r - |x_r - y_r|\} - 1,$$

which proves the lemma. □

In fact, Lemma 3.3.4 fully characterizes the path followed by a packet, because given a packet P with a predetermined destination t, the statement of the lemma applies to all vertices of the torus except the destination vertex t, allowing repeated application of the lemma to trace the whole path from any "source" vertex to t. The following theorem formalizes the idea to prove the optimality of our routing scheme.

Theorem 2 *The above routing scheme is an optimal* 1*-OLIRS of the torus.*

Proof: In the following, we first show that the above routing scheme is a valid 1-OLIRS. Then we prove its optimality.

By definition, each edge label consists of exactly one linear interval. Moreover, the larger interval of class C_D is as follows:

$$[\, L(0, 0, \ldots, 0), L(d_1 - 1, d_2 - 1, \ldots, d_D - 1)\,]$$

which covers the label $L(t)$ of any vertex $t = (y_1, y_2, \ldots, y_D)$ of the torus. This follows from Lemma 3.3.1 since $0 \leq y_i \leq d_i - 1$ for all $i \in \{1, 2, \ldots, D\}$. So, the scheme is a valid 1-OLIRS.

To prove the optimality of the routing scheme, suppose a packet P whose destination is vertex t is at another vertex s_0. Because the larger interval of class C_D at s_0 contains all vertex labels (as shown in the last paragraph), vertex s_0 has (at least) one edge containing $L(t)$ in its label. So, packet P must leave vertex s_0 through an edge (s_0, s_1). If s_1 and t are not the same vertices, the same argument applies to s_1 also. So, P must leave vertex s_1 through an edge (s_1, s_2). Let $(s_0, s_1, s_2, \ldots, s_i, \ldots)$

be the path thus followed by P. Now we have to show that this path is guaranteed to reach t and that it does so in the smallest possible number of steps. From the statement of Lemma 3.3.4, it is obvious that vertex s mentioned in that lemma can be any vertex of the torus except the destination of P. Therefore, by Lemma 3.3.4, $dist(s_{i+1}, t) = dist(s_i, t) - 1$ for all edge (s_i, s_{i+1}) on that path, provided $s_i \neq t$. Thus, at each step P decreases its distance to t by exactly one until it reaches t. So, the total number of steps to reach t from s is $dist(s, t)$, which is optimal. □

3.4 k-garland Graphs

In this section, we define and study a class of graphs, called k-garland graphs, that have optimal 2-OLIRS's, but do not always have optimal 2-IRS's. The section is organized as follows. Following the formal definition of a k-garland graph, we give in Section 3.4.1 a counterexample showing that not all k-garland graphs have optimal 2-IRS's. In Section 3.4.2, we determine a set of characteristics of shortest paths in a k-garland graph. Finally, in Sections 3.4.3 and 3.4.4, we present an optimal 2-OLIRS of the graph and prove its optimality.

First of all, we define a k-garland graph as follows. Intuitively, a k-garland graph has a chain of k "special" vertices; all other vertices are connected to exactly one or two of these special vertices. Moreover, there is no common neighbor of two non-adjacent special vertices.

Definition 5 (k-garland graph) *A graph is called a* k-garland graph *if its vertices can be partitioned into the sets B, C_i for $i \in \{1, 2, \ldots, k\}$ and D_i for $i \in \{1, 2, \ldots, k-1\}$ such that:*

(i) set B consists of k vertices b_1, b_2, ..., b_k, called the base vertices*, such that b_i is adjacent to b_j iff $j = i + 1$ $(1 \leq i, j \leq k)$;*

(ii) *set C_i, $i \in \{1, 2, \ldots, k\}$, consists of all the vertices that are adjacent to b_i and not adjacent to b_j for all $j \neq i$;*

(iii) *set D_i, $i \in \{1, 2, \ldots, k-1\}$, consists of all the vertices that are adjacent to both b_i and b_{i+1} and not adjacent to b_j for all $j \notin \{i, i+1\}$;*

(iv) *no vertex in C_i is adjacent to any vertex in C_j for all $j \neq i$;*

(v) *no vertex in D_i is adjacent to any vertex in D_j for all $j \neq i$; and*

(vi) *each vertex $v \in C_i$ is adjacent to at most one vertex in D_{i-1}, at most one vertex in D_i and no vertex in D_j, $j \notin \{i-1, i\}$.*

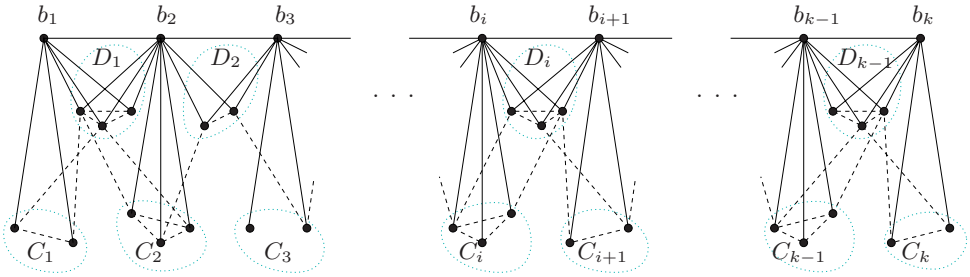


Figure 3.5: An outline of the structure of a k-garland graph

Figure 3.5 outlines the structure of a k-garland graph. Note that for all $i \in \{2, 3, \ldots, k-1\}$, any path from a vertex in $C_{i-1} \cup \{b_{i-1}\} \cup D_{i-1}$ to a vertex in $D_i \cup C_{i+1} \cup b_{i+1}$ goes through a vertex in $\{b_i\} \cup C_i$. Extending this idea to a larger set of vertices, we get the following observation, which we use in the following subsections.

Observation 3.4.1 *For all $i \in \{2, 3, \ldots, k-1\}$, any path from a vertex in $\bigcup_{j=1}^{i-1}(C_j \cup \{b_j\} \cup D_j)$ to a vertex in $\bigcup_{j=i+1}^{k}(D_{j-1} \cup C_j \cup \{b_j\})$ goes through a vertex in $C_i \cup \{b_i\}$.*

3.4.1 Compactness of an IRS of a k-garland Graph

In this subsection, we give a counterexample to show that not all k-garland graphs have an optimal 2-IRS.

Narayanan and Nishimura [NN98] give a counterexample to prove that not every 2-tree has an optimal 2-IRS. The 2-tree T used in the counterexample is as follows. It has 77 vertices: x, y and a_i, b_i, c_i, d_i, e_i for $i \in [1, 15]$. The edges of T are (x, a_i), (x, b_i), (x, c_i), (y, c_i), (y, d_i), (y, e_i), (a_i, b_i), (b_i, c_i), (c_i, d_i), (d_i, e_i) for $i \in [1, 15]$ and (x, y). One component of the graph is shown in Figure 3.6.

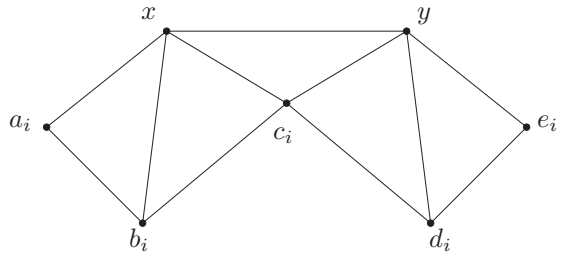


Figure 3.6: The ith component of T

It is easy to see that graph T is a 2-garland graph by partitioning its vertices into the following sets: $B = \{x, y\}$, $C_1 = \{a_i, b_i : i \in [1, 15]\}$, $C_2 = \{d_i, e_i : i \in [1, 15]\}$ and $D_1 = \{c_i : i \in [1, 15]\}$.

Narayanan and Nishimura [NN98, Theorem 3] prove that graph T does not have an optimal 2-IRS. This implies that not all k-garland graphs have optimal 2-IRS's, since T is a k-garland graph. In the next two subsections, we show that every k-garland graph has an optimal 2-OLIRS.

3.4.2 Shortest Paths in k-garland Graphs

This subsection gives five lemmas concerning the length of a shortest path in a k-garland graph. In fact, the lemmas in this subsection determine the distances between

almost all pairs of vertices in a k-garland graph. The pairs of vertices not covered by these lemmas consist of adjacent vertices or vertices with a common neighbor; the distance between each such pair is either two or three, hence easy to determine.

In the proofs of the lemmas in this subsection, we utilize the following observation about shortest paths in a graph:

Observation 3.4.2 *Given two vertices u and v and a vertex set $W \subseteq V$, if every path from u to v passes through a vertex in W, then $dist(u,v) = \min_{w \in W}\{dist(u,w) + dist(w,v)\}$.*

Proof: For any $w \in W$, the path formed by concatenating a shortest path from u to w with a shortest path from w to v has the length at least $dist(u,v)$. So,

$$dist(u,v) \leq \min_{w \in W}\{dist(u,w) + dist(w,v)\}.$$

Since every path from u to v passes through a vertex in W, the shortest path from u to v passes through a vertex $w_0 \in W$. Because any subpath of a shortest path is a shortest path [CLR90, Lemma 25.1], $dist(u,v) = dist(u,w_0) + dist(w_0,v)$. Therefore, we conclude that $dist(u,v) = \min_{w \in W}\{dist(u,w) + dist(w,v)\}$. □

In the rest of this section, we use D_0 and D_k to denote two empty sets of vertices for ease of discussion.

The next two lemmas establish the formula for computing the distance of a base vertex from all other vertices. Lemma 3.4.1 gives the distance of a base vertex from all vertices in sets B and C_i, $1 \leq i \leq k$.

Lemma 3.4.1 *For any base vertex b_i and any $j \in \{1, 2, \ldots, k\}$, $dist(b_i, b_j) = |j - i|$ and $dist(b_i, v) = |j - i| + 1$ if $v \in C_j$.*

Proof: We prove the case $j \geq i$ by induction on $j - i$. The basis of the induction is $j = i$, which is trivial. For the induction step, suppose that, for all $j \in \{i, i+1, \ldots, r\}$, $dist(b_i, b_j) = |j - i|$ and $dist(b_i, v) = |j - i| + 1$ if $v \in C_j$.

Now, by Observation 3.4.1, any path from b_i to b_{r+1} passes through a vertex in $C_r \cup \{b_r\}$. So, Observation 3.4.2 implies

$$\begin{aligned} dist(b_i, b_{r+1}) &= \min_{w \in C_r \cup \{b_r\}} \{dist(b_i, w) + dist(w, b_{r+1})\} \\ &= \min\{dist(b_i, b_r) + dist(b_r, b_{r+1}), \\ &\qquad \min_{w \in C_r} \{dist(b_i, w) + dist(w, b_{r+1})\}\} \end{aligned}$$

and hence, by the induction hypothesis

$$\begin{aligned} dist(b_i, b_{r+1}) &= \min\{|r - i| + |(r+1) - r|, \min_{w \in C_r} \{|r - i| + 1 + dist(w, b_{r+1})\}\} \\ &= \min\{r - i + 1, r - i + 1 + \min_{w \in C_r} \{dist(w, b_{r+1})\}\}, \end{aligned}$$

since $r > i$. Therefore, $dist(b_i, b_{r+1}) = r + 1 - i = |(r+1) - i|$.

For any $v \in C_{r+1}$, any path from b_i to v passes through a vertex in $C_r \cup \{b_r\}$ by Observation 3.4.1. Therefore, by Observation 3.4.2,

$$\begin{aligned} dist(b_i, v) &= \min_{w \in C_r \cup \{b_r\}} \{dist(b_i, w) + dist(w, v)\} \\ &= \min\{dist(b_i, b_r) + dist(b_r, v), \min_{w \in C_r} \{dist(b_i, w) + dist(w, v)\}\} \\ &= \min\{|r - i| + |(r+1) - r| + 1, \min_{w \in C_r} \{|r - i| + 1 + dist(w, v)\}\} \end{aligned}$$

by the induction hypothesis. Since $r > i$,

$$dist(b_i, v) = \min\{r - i + 2, r - i + 1 + \min_{w \in C_r} \{dist(w, v)\}\}.$$

Now, for any $w \in C_r$, $dist(w, v)$ is at least two because w and v are non-adjacent by the definition of a k-garland graph. So, $dist(b_i, v) = r - i + 2 = |(r+1) - i| + 1$.

Therefore, the lemma is true for all $j \geq i$.

For the case $j \leq i$, the base case $j = i$ is the same as before. Suppose the lemma is true for all $j \in \{i, i-1, \ldots, r\}$. Because any path from b_i to b_{r-1} or from b_i to a vertex $v \in C_{r-1}$ passes through a vertex in $C_r \cup \{b_r\}$ (by Observation 3.4.1), we can construct a similar proof for the induction step, proving the case $j \leq i$. □

It follows directly from the above proof that the shortest path from b_i to b_j $(j \geq i)$ is $(b_i, b_{i+1}, b_{i+2}, \ldots, b_j)$ and that the shortest path from b_i to a vertex v in C_j $(j \geq i)$ is $(b_i, b_{i+1}, b_{i+2}, \ldots, b_j, v)$. We get similar shortest paths for the case $j < i$. For a vertex $v \in D_j$ $(j \geq i)$, the shortest path from b_i to v is $(b_i, b_{i+1}, b_{i+2}, \ldots, b_j, v)$, which (as well as the similar fact for the case $j < i$) follows from the proof of the following lemma that establishes the distance of a base vertex from all vertices in sets D_i, $1 \leq i < k$.

Lemma 3.4.2 *For any base vertex b_i and any vertex $v \in D_j$, $dist(b_i, v) = j - i + 1$ if $j \geq i$, and $dist(b_i, v) = i - j$ if $j < i$.*

Proof: We split the proof into four cases for different values of j, as follows.

Case 1: $j = i$**:** In this case, because b_i and v are adjacent, $dist(b_i, v) = 1 = j - i + 1$.

Case 2: $j > i$**:** Any path from b_i to v passes through a vertex in $C_j \cup \{b_j\}$. Therefore, by Observation 3.4.2,

$$\begin{aligned} dist(b_i, v) &= \min_{w \in C_j \cup \{b_j\}} \{dist(b_i, w) + dist(w, v)\} \\ &= \min\{dist(b_i, b_j) + dist(b_j, v),\ \min_{w \in C_j} \{dist(b_i, w) + dist(w, v)\}\} \\ &= \min\{|j - i| + dist(b_j, v),\ \min_{w \in C_j} \{|j - i| + 1 + dist(w, v)\}\} \end{aligned}$$

by Lemma 3.4.1. Since $j > i$ and $dist(b_j, v) = 1$,

$$\begin{aligned} dist(b_i, v) &= \min\{j - i + 1,\ j - i + 1 + \min_{w \in C_j} \{dist(w, v)\}\} \\ &= j - i + 1. \end{aligned}$$

Case 3: $j = i - 1$**:** Since b_i and v are adjacent, $dist(b_i, v) = 1 = i - j$.

Case 4: $j < i - 1$**:** Any path from b_i to v passes through a vertex in $C_{j+1} \cup \{b_{j+1}\}$. So, by Observation 3.4.2,

$$\begin{aligned} dist(b_i, v) &= \min_{w \in C_{j+1} \cup \{b_{j+1}\}} \{dist(b_i, w) + dist(w, v)\} \\ &= \min\{dist(b_i, b_{j+1}) + dist(b_{j+1}, v), \\ &\qquad \min_{w \in C_{j+1}} \{dist(b_i, w) + dist(w, v)\}\} \\ &= \min\{|j + 1 - i| + dist(b_{j+1}, v), \\ &\qquad \min_{w \in C_{j+1}} \{|j + 1 - i| + 1 + dist(w, v)\}\} \end{aligned}$$

by Lemma 3.4.1. Since $j < i - 1$ and $dist(b_{j+1}, v) = 1$,

$$\begin{aligned} dist(b_i, v) &= \min\{i - 1 - j + 1, i - 1 - j + 1 + \min_{w \in C_j} \{dist(w, v)\}\} \\ &= i - j. \end{aligned}$$

□

The remaining lemmas in this subsection determine the distance between vertices in sets C_i, $1 \leq i \leq k$, and D_i, $1 \leq i < k$. The next lemma handles the case when both vertices are in sets C_i, $1 \leq i \leq k$.

Lemma 3.4.3 *If u and v are two non-adjacent vertices such that for some $i, j \in \{1, 2, \ldots, k\}$, $u \in C_i$, $v \in C_j$ and no vertex is adjacent to both u and v, then $dist(u, v) = |j - i| + 2$.*

Proof: The case $j = i$ is impossible, since in this case, b_i is adjacent to both u and v.

When $|j - i| = 1$, $dist(u, v) \leq 3$ because of the path (u, b_i, b_j, v) of length three. Moreover, since u and v have no common neighbor, $dist(u, v) > 2$. So, $dist(u, v) = 3 = |j - i| + 2$ in this case. This forms the basis of our proof by induction.

When $|j-i|>1$, we have the following two cases. If $j>i+1$, then by Observation 3.4.1, any path from u to v passes through the vertices in $C_{j-1}\cup b_{j-1}$. On the other hand, if $j<i-1$, then any path from u to v passes through the vertices in $C_{j+1}\cup b_{j+1}$. Let $l=j-1$ if $j>i+1$ and $l=j+1$ if $j<i-1$. Note that $|l-i|=|j-i|-1$. By Observation 3.4.2,

$$\begin{aligned} dist(u,v) &= \min_{w\in C_l\cup\{b_l\}}\{dist(u,w)+dist(w,v)\} \\ &= \min\{dist(u,b_l)+dist(b_l,v),\ \min_{w\in C_l}\{dist(u,w)+dist(w,v)\}\} \\ &= \min\{|i-l|+1\ +\ |j-l|+1,\ \min_{w\in C_l}\{dist(u,w)+dist(w,v)\}\} \end{aligned}$$

by Lemma 3.4.1. Since $|j-l|=1$, we have

$$dist(u,v) \ = \ \min\{|i-l|+3,\ \min_{w\in C_l}\{dist(u,w)+dist(w,v)\}\}.$$

If, for all $w\in C_l$, u and w have no common neighbor, then by the induction hypothesis, $dist(u,w)=|i-l|+2$. Moreover, for all $w\in C_l$, $dist(w,v)\geq 2$ as w and v are not adjacent. So, $dist(u,v)=|i-l|+3=|j-i|-1+3=|j-i|+2$.

On the other hand, if for some $w\in C_l$, u and w have a common neighbor, then by the definition of a k-garland graph, l and i are consecutive integers. So, $|i-l|=1$ i.e., $|i-l|+3=4$. For any $w\in C_l$, $dist(u,w)\geq 2$ because u and w are not adjacent, and $dist(w,v)\geq 2$ similarly. As a result, $dist(u,v)=4=|i-l|+3=|j-i|-1+3=|j-i|+2$. □

The following two lemmas establish the distance of a vertex in set D_i, $1\leq i<k$, from all vertices in the graph except the base vertices.

Lemma 3.4.4 *If u and v are two non-adjacent vertices such that for some $i\in\{1,2,\ldots,k\}$ and some $j\in\{1,2,\ldots,k-1\}$, $u\in C_i$ and $v\in D_j$, then*

(i) $dist(u,v)=j-i+2$ when $j\geq i$; and

(ii) $dist(u,v) = i - j + 1$ *when* $j < i$.

Proof: In the following, we split the case $j \geq i$ into two sub-cases.

Case 1: $j = i$**:** Since u and v are not adjacent, $dist(u,v) > 1$. Because of the path (u, b_i, v) of length two, $dist(u,v) \leq 2$. So, $dist(u,v) = 2 = j - i + 2$.

Case 2: $j > i$**:** By Observation 3.4.1, any path from u to v passes through a vertex in $C_j \cup \{b_j\}$. So, Observation 3.4.2 implies

$$\begin{aligned} dist(u,v) &= \min_{w \in C_j \cup \{b_j\}} \{dist(u,w) + dist(w,v)\} \\ &= \min\{dist(u,b_j) + dist(b_j,v),\ \min_{w \in C_j}\{dist(u,w) + dist(w,v)\}\} \\ &= \min\{|i-j| + 1 + dist(b_j,v),\ \min_{w \in C_j}\{dist(u,w) + dist(w,v)\}\} \end{aligned}$$

by Lemma 3.4.1. Since b_j and v are adjacent by the definition of a k-garland graph, $dist(b_j, v) = 1$. So, we have

$$dist(u,v) = \min\{|i-j| + 2,\ \min_{w \in C_j}\{dist(u,w) + dist(w,v)\}\}.$$

If, for all $w \in C_j$, u and w have no common neighbor, then by Lemma 3.4.3 $dist(u,w) = |i-j| + 2$. So, $dist(u,v) = |i-j| + 2 = j - i + 2$.

On the other hand, if for some $w \in C_j$, u and w have a common neighbor, then by the definition of a k-garland graph, $j = i + 1$ i.e., $|i-j| + 2 = 3$. For any $w \in C_j$, $dist(u,w) \geq 2$ because u and w are not adjacent, and $dist(w,v) \geq 1$. As a result, $dist(u,v) = 3 = j - i + 2$.

This proves the case $j \geq i$. We can prove the other case in a similar way. □

The next lemma handles the same case as the last lemma except that both vertices are in sets D_i, $1 \leq i < k$ in this lemma.

Lemma 3.4.5 *If u and v are two non-adjacent vertices such that for some $i, j \in \{1, 2, \ldots, k-1\}$ $(i \neq j)$, $u \in D_i$ and $v \in D_j$, then $dist(u, v) = |j - i| + 1$.*

Proof: We can prove this lemma is the same way as that of Lemma 3.4.3 except that we split the proof into the cases $j > i$ and $j < i$ and make use of Lemmas 3.4.2 and 3.4.4 instead of Lemma 3.4.1 in each case. □

As mentioned earlier, the pairs of vertices not covered by Lemmas 3.4.1 to 3.4.5 are the ones consisting of adjacent vertices or vertices with a common neighbor; the distance between each such pair is easy to determine.

3.4.3 Labeling and Ordering in the OLIRS

For labeling the vertices of a k-garland graph, we first take an ordering π of the vertices as follows. Let C'_i denote the set $C_i \cup \{b_i\}$ for all $i \in \{1, 2, \ldots, k\}$. The ordering π starts with the vertices in set C'_1, followed by those in the sets D_1, C'_2, D_2, C'_3, …, C'_{k-1}, D_{k-1} and C'_k in that order. In π, b_i is either the first or the last vertex within each set C'_i, $1 \leq i \leq k$. The vertices within each of the sets C_i, $1 \leq i \leq k$, and D_i, $1 \leq i < k$, can be in any arbitrary order. Vertices are labeled with 0, 1, …, $n-1$ in the same order they appear in π.

Before giving details of ordering and labels of the outgoing edges at a vertex, we give an idea of our routing scheme. In the following, we illustrate how all the vertices of a k-garland graph are covered by the outgoing edges at a vertex in each of the sets B, C_i, $1 \leq i \leq k$, and D_i, $1 \leq i < k$.

For base vertex b_i, an edge that connects b_i to a vertex v in D_{i-1} or C_i or D_i covers only v. The edge (b_i, b_{i-1}) covers the vertices to the "left" of b_i (in Figure 3.7), excluding those in D_{i-1}; more precisely, it covers the vertices in C'_1, D_1, C'_2, D_2, …, C'_{i-1}. The edge (b_i, b_{i+1}) covers the vertices that are to the "right" of b_i excluding those in D_i; more precisely, it covers the vertices in C'_{i+1}, D_{i+1}, C'_{i+2}, D_{i+2}, …, C'_k.

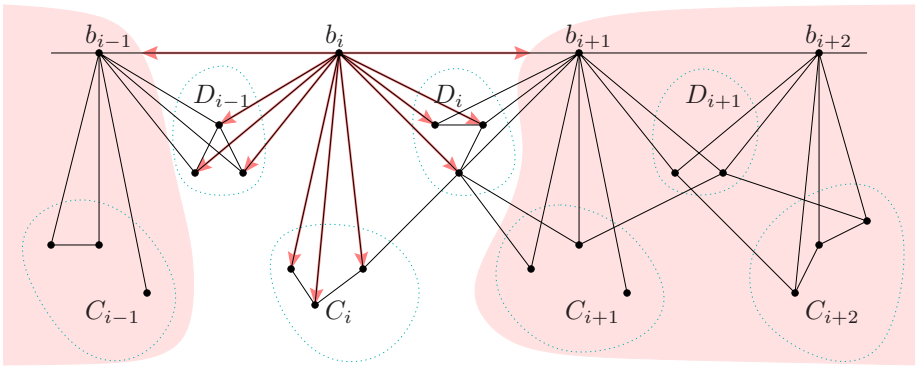


Figure 3.7: Vertices covered by the outgoing edges at base vertex b_i

For a vertex $c \in C_i$, each edge that connects c to a vertex v in C_i covers only v. If an edge connects c to $v \in D_{i-1}$, then the edge covers v and the vertices in C_{i-1}. Similarly, if an edge connects c to $v \in D_i$, then the edge covers v and the vertices in C_{i+1}. All the vertices not covered by any other edges is covered by (c, b_i). This includes the vertices in C_{i-1}, C_i, C_{i+1}, D_{i-1} and D_i not covered by any other edges at c. This case is illustrated in Figure 3.8.

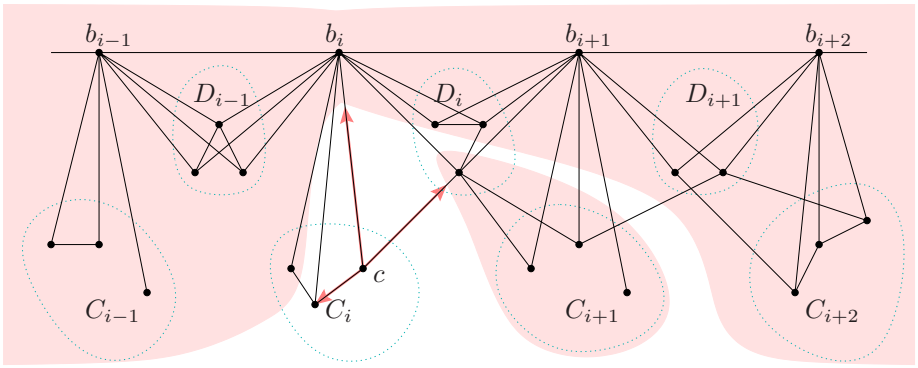


Figure 3.8: Vertices covered by the outgoing edges at vertex $c \in C_i$

For a vertex $d \in D_i$, an edge connecting d to a vertex v in C_i or D_i or C_{i+1} covers only the destination v. All the vertices to the "left" of D_i (in Figure 3.9) as well as the non-adjacent vertices in D_i are covered by (d, b_i). All the vertices to the "right"

of d (i.e., the rest of the vertices) are covered by (d, b_{i+1}).

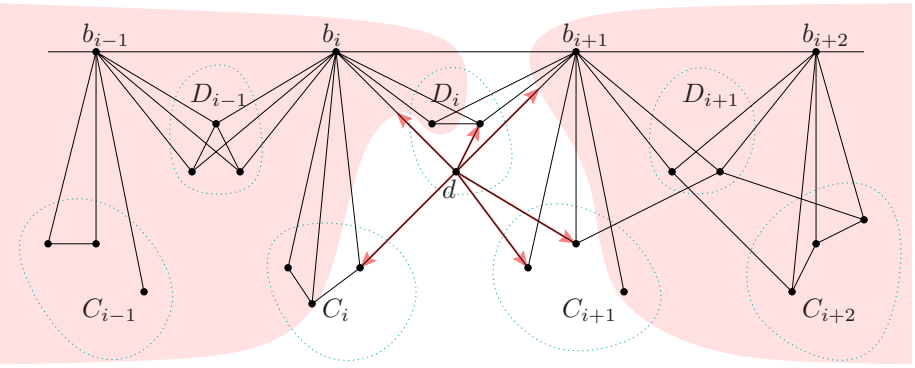


Figure 3.9: Vertices covered by the outgoing edges at vertex $d \in D_i$

Now we formalize ordering and labels of the outgoing edges at each vertex in each of the subsets B, C_i for $i \in \{1, 2, \ldots, k\}$ and D_i for $i \in \{1, 2, \ldots, k-1\}$.

Set B: For each vertex $b_i \in B$, the edge order π_{b_i} starts with the edges (b_i, v), $v \in D_{i-1} \cup C_i \cup D_i$, in any arbitrary order, followed by the edge (b_i, b_{i-1}) and then the edge (b_i, b_{i+1}).

The label of an edge (b_i, v) is:

- $L(v)$ if $v \in D_{i-1} \cup C_i \cup D_i$;
- the set of labels of the vertices in $C'_1 \cup D_1 \cup C'_2 \cup D_2 \cup \ldots \cup D_{i-2} \cup C'_{i-1}$ if $v = b_{i-1}$; and
- the set of labels of the vertices in $C'_{i+1} \cup D_{i+1} \cup C'_{i+2} \cup D_{i+2} \cup \ldots \cup D_{k-1} \cup C'_k$ if $v = b_{i+1}$.

Set C_i: For each $i \in \{1, 2, \ldots, k\}$, the edge order π_c of each vertex $c \in C_i$ starts with the edges (c, v), $v \in C_i$, in any arbitrary order, followed by the edge (c, u) if vertex c has a neighbor $u \in D_{i-1}$ and then the edge (c, w) if c has a neighbor $w \in D_i$. The last edge in π_c is (c, b_i).

The label of an edge (c, v) is:

- $L(v)$ if $v \in C_i$;
- the set of labels of the vertices in $\{v\} \cup C_{i-1}$ if $v \in D_{i-1}$;
- the set of labels of the vertices in $\{v\} \cup C_{i+1}$ if $v \in D_i$; and
- $[1, n]$ if $v = b_i$.

Set D_i: For each $i \in \{1, 2, \ldots, k-1\}$, the edge order π_d of each vertex $d \in D_i$ starts with the edges (d, v), $v \notin B$, in any arbitrary order, followed by the edge (d, b_i) and then the edge (d, b_{i+1}).

The label of an edge (d, v) is:

- $L(v)$ if $v \notin B$;
- the set of labels of the vertices in $C'_1 \cup D_1 \cup C'_2 \cup D_2 \cup \ldots \cup C'_i \cup D_i$ if $v = b_i$; and
- the set of labels of the vertices in $C'_{i+1} \cup D_{i+1} \cup C'_{i+2} \cup D_{i+2} \cup \ldots \cup D_{k-1} \cup C'_k$ if $v = b_{i+1}$.

We note the following observation about the vertex labels, which is obvious from the way we assign the labels.

Observation 3.4.3 *For $1 \leq i \leq j < k$, the labels of the vertices in each of the following sets form a single linear interval:*

(i) $\bigcup_{l=i}^{j}(C'_l \cup D_l)$

(ii) $\bigcup_{l=i}^{j}(C'_l \cup D_l) \cup C'_{j+1}$

3.4.4 Proving the Optimality of Our Scheme

This subsection elaborates the correctness of the scheme presented in Section 3.4.3, i.e., the proof that the scheme is an optimal 2-OLIRS. We first determine the edge

a packet follows to leave the current vertex on the way to its destination, using Lemmas 3.4.1 to 3.4.4. Then we show that our scheme is a valid 2-OLIRS and that it induces shortest paths between any pair of vertices.

Lemma 3.4.6 *Let t be the destination vertex of a packet P and s be the current position of P such that $s \neq t$. If s and t are adjacent, then P uses the edge (s,t) to leave vertex s.*

Proof: We split the proof into the following three cases.

Case 1: $s = b_i$ $(1 \leq i \leq k)$: In this case, $t \in D_{i-1} \cup C_i \cup D_i \cup \{b_{i-1}, b_{i+1}\}$.

If $t \in D_{i-1} \cup C_i \cup D_i \cup \{b_{i-1}\}$, then for any edge (b_i, w) that comes before (b_i, t) in π_{b_i}, $w \in D_{i-1} \cup C_i \cup D_i - \{t\}$ and hence $L(t) \notin I(b_i, w) = L(w)$. Moreover, $L(t) \in I(b_i, t)$. So, P uses the edge (b_i, t).

If $t = b_{i+1}$, then for any edge (b_i, w) such that $w \in D_{i-1} \cup C_i \cup D_i$, $L(b_{i+1}) \notin I(b_i, w) = L(w)$. Also $L(b_{i+1}) \notin I(b_i, b_{i-1})$. So, P uses the edge (b_i, b_{i+1}).

Case 2: $s \in C_i$ $(1 \leq i \leq k)$: In this case, $t \in C_i \cup D_{i-1} \cup D_i \cup \{b_i\}$.

If $t \in C_i$, then for any edge (s, w) that comes before (s, t) in π_s, $w \in C_i - \{t\}$ and hence $L(t) \notin I(s, w) = L(w)$. Moreover, $L(t) \in I(s, t)$. So, P uses the edge (s, t).

If $t \in D_{i-1}$, then $t \notin C_i$ which implies that $L(t) \notin (s, w)$ for all $w \in C_i$. Moreover, $L(t) \in I(s, t)$. Because only the edges (s, w), where $w \in C_i$, can come before (s, t) in π_s, P uses the edge (s, t).

If $t \in D_i$, then $t \notin C_i$ which implies that $L(t) \notin (s, w)$ for all $w \in C_i$. Also, $t \notin C_{i-1} \cup D_{i-1}$ which implies that $L(t)$ is not in the label of the edge connecting s to a vertex in D_{i-1}, if such an edge exists at all. Moreover, $L(t) \in I(s, t)$. Therefore, P uses the edge (s, t).

If $t = b_i$, then $t \notin C_i \cup C_{i-1} \cup D_{i-1} \cup D_i \cup C_{i+1}$, which implies that $L(t)$ is not in the label of any edge that comes before (s,t) in π_s. So, P uses the edge (s,t) since $L(t) \in I(s,t)$.

Case 3: $s \in D_i$ $(1 \leq i < k)$**:** This case is similar to Case 1.

In all the cases above, P follows the edge (s,t) to leave vertex s. □

When P moves from s to t in case of the above lemma, the distance of P from its destination t changes from one to zero. In fact, we can show that for any current vertex s and any destination vertex t, if P uses the edge (s, s_1) to leave vertex s, then $dist(s_1, t) = dist(s, t) - 1$. Lemmas 3.4.7 to 3.4.9 prove this claim for the case when s and t are not adjacent. Lemma 3.4.7 covers the case when s is a base vertex.

Lemma 3.4.7 *Let t be the destination vertex of a packet P and b_i $(1 \leq i \leq k)$ be the current position of P such that b_i and t are distinct and not adjacent. If P uses the edge (b_i, s_1) to leave vertex b_i, then $dist(s_1, t) = dist(b_i, t) - 1$.*

Proof: We prove the following three cases separately, depending on which of the subsets B, C_j $(1 \leq j \leq k)$ and D_j $(1 \leq j < k)$ contains t. In each of the following cases, $j \neq i$ as b_i and t are not adjacent.

Case 1: $t = b_j$**:** If $j > i$, then from the edge order and labels at b_i (as defined in page 53), we infer that the only edge in π_{b_i} that contains $L(t)$ in its label is (b_i, b_{i+1}). So, $s_1 = b_{i+1}$ in this case. Therefore, by Lemma 3.4.1,

$$dist(b_i, t) = |j - i| = j - i$$

and

$$dist(s_1, t) = |j - (i+1)| = j - i - 1.$$

If $j < i$, then the only edge in π_{b_i} that contains $L(t)$ in its label is (b_i, b_{i-1}) (as before, from page 53). Therefore, $s_1 = b_{i-1}$. By Lemma 3.4.1,

$$dist(b_i, t) = |j - i| = i - j$$

and

$$dist(s_1, t) = |j - (i-1)| = i - j - 1.$$

Case 2: $t \in C_j$**:** If $j > i$, then from the edge order and labels at b_i (as defined in page 53), we infer that the only edge in π_{b_i} that contains $L(t)$ in its label is (b_i, b_{i+1}). So, $s_1 = b_{i+1}$ in this case. Therefore, by Lemma 3.4.1,

$$dist(b_i, t) = |j - i| + 1 = j - i + 1$$

and

$$dist(s_1, t) = |j - (i+1)| + 1 = j - i.$$

If $j < i$, then the only edge in π_{b_i} that contains $L(t)$ in its label is (b_i, b_{i-1}) (as before, from page 53). So, $s_1 = b_{i-1}$. Therefore, by Lemma 3.4.1,

$$dist(b_i, t) = |j - i| + 1 = i - j + 1$$

and

$$dist(s_1, t) = |j - (i-1)| + 1 = i - j.$$

Case 3: $t \in D_j$**:** In this case, $j \notin \{i-1, i\}$ as b_i and t are not adjacent.

If $j > i$, then the edge order and labels at b_i (as defined in page 53) imply that the only edge in π_{b_i} that contains $L(t)$ in its label is (b_i, b_{i+1}). Therefore, $s_1 = b_{i+1}$. By Lemma 3.4.2,

$$dist(b_i, t) = j - i + 1$$

and

$$dist(s_1, t) = j - (i+1) + 1 = j - i.$$

If $j < i-1$, then the only edge in π_{b_i} that contains $L(t)$ in its label is (b_i, b_{i-1}) (as before, from page 53). So, $s_1 = b_{i-1}$. Therefore, by Lemma 3.4.2,

$$dist(b_i, t) = i - j$$

and

$$dist(s_1, t) = (i-1) - j = i - j - 1.$$

In all the cases, we have shown that $dist(s_1, t) = dist(b_i, t) - 1$, which proves the lemma. □

The next lemma proves the same claim as the above lemma, but for the case when the current position of P is a vertex in C_i for some $i \in \{1, 2, \ldots, k\}$.

Lemma 3.4.8 *Let t be the destination vertex of a packet P and s be the current position of P such that s and t are distinct and not adjacent. If $s \in C_i$ $(1 \leq i \leq k)$ and P uses the edge (s, s_1) to leave vertex s, then $dist(s_1, t) = dist(s, t) - 1$.*

Proof: We prove the following three cases separately, depending on which of the subsets B, C_j $(1 \leq j \leq k)$ and D_j $(1 \leq j < k)$ contains t.

Case 1: $t = b_j$: From the edge order and labels at s (as defined in page 53), we infer that the only edge in π_s that contains $L(t)$ in its label is (s, b_i). So, $s_1 = b_i$ in this case. Therefore, by Lemma 3.4.1,

$$dist(s, t) = |j - i| + 1$$

and

$$dist(s_1, t) = |j - i|.$$

Case 2: $t \in C_j$: We split this case into the following sub-cases:

Case 2a: When $j = i - 1$ and s is adjacent to a vertex $v \in D_{i-1}$, the edge order and labels at s (as defined in page 53) imply that the first edge in π_s that contains $L(t)$ in its label is (s, v). So, $s_1 = v$ in this case. If v is adjacent to t, then clearly,

$$dist(s, t) = 2$$

and

$$dist(s_1, t) = dist(v, t) = 1.$$

If, on the other hand, v is not adjacent to t, then no vertex is adjacent to both s and t. So, Lemma 3.4.3 implies

$$dist(s, t) = |j - i| + 2 = 3$$

and Lemma 3.4.4 implies

$$dist(s_1, t) = dist(t, v) = (i - 1) - j + 2 = j - j + 2 = 2.$$

Case 2b: When $j = i + 1$ and s is adjacent to a vertex $v \in D_i$, the first edge in π_s that contains $L(t)$ in its label is (s, v) (as before, from page 53). So, $s_1 = v$ in this case. As before, if v is adjacent to t, then

$$dist(s, t) = 2$$

and

$$dist(s_1, t) = dist(v, t) = 1;$$

otherwise, by Lemmas 3.4.3 and 3.4.4,

$$dist(s, t) = 3$$

and

$$dist(s_1, t) = 2.$$

Case 2c: In the remaining sub-cases (i.e. when $j \notin \{i-1, i+1\}$; or $j \neq i-1$ and s has no neighbor in D_i; or $j \neq i+1$ and s has no neighbor in D_{i-1}; or s has no neighbor in $D_{i-1} \cup D_i$), the only edge in π_s that contains $L(t)$ in its label is (s, b_i) (as before, from page 53). So, $s_1 = b_i$. Therefore, by Lemma 3.4.3,

$$dist(s,t) = |j-i| + 2$$

and by Lemma 3.4.1,

$$dist(s_1,t) = |j-i| + 1.$$

Case 3: $t \in D_j$**:** Since s and t are not adjacent, the only edge in π_s that contains $L(t)$ in its label is (s, b_i) which can be verified from the edge order and labels at s as defined in page 53. So, $s_1 = b_i$. If $j \geq i$, then by Lemma 3.4.4,

$$dist(s,t) = j - i + 2$$

and by Lemma 3.4.2,

$$dist(s_1,t) = j - i + 1.$$

On the other hand, if $j < i$, then by Lemma 3.4.4,

$$dist(s,t) = i - j + 1$$

and by Lemma 3.4.2,

$$dist(s_1,t) = i - j.$$

In all the cases, we get $dist(s_1,t) = dist(s,t) - 1$, which proves the lemma. □

Lemma 3.4.9 below handles the case not covered by the last two lemmas, i.e., when the current position of P is a vertex in D_i $(1 \leq i < k)$.

Lemma 3.4.9 *Let t be the destination vertex of a packet P and s be the current position of P such that s and t are distinct and not adjacent. If $s \in D_i$ $(1 \leq i < k)$ and P uses the edge (s, s_1) to leave vertex s, then $dist(s_1, t) = dist(s, t) - 1$.*

Proof: As in the last two lemmas, we prove the following three cases separately, depending on which of the subsets B, C_j $(1 \leq j \leq k)$ and D_j $(1 \leq j < k)$ contains t.

Case 1: $t = b_j$**:** Since s and t are not adjacent, $j \notin \{i, i+1\}$.

If $j > i+1$, then from the edge order and labels at s (as defined in page 54), we infer that the only edge in π_s that contains $L(t)$ in its label is (s, b_{i+1}). So, $s_1 = b_{i+1}$ in this case. Therefore, by Lemma 3.4.2,

$$dist(s, t) = j - i$$

and by Lemma 3.4.1,

$$dist(s_1, t) = |j - (i+1)| = j - i - 1.$$

If $j < i$, then the only edge in π_s that contains $L(t)$ in its label is (s, b_i) (as before, from page 54). Therefore, $s_1 = b_i$. By Lemma 3.4.2,

$$dist(s, t) = i - j + 1$$

and by Lemma 3.4.1,

$$dist(s_1, t) = |j - i| = i - j.$$

Case 2: $t \in C_j$**:** If $j > i$, then the edge order and labels at s (as defined in page 54) imply that the only edge in π_s that contains $L(t)$ in its label is (s, b_{i+1}). Therefore, $s_1 = b_{i+1}$. Lemma 3.4.4 implies

$$dist(s, t) = j - i + 1$$

and Lemma 3.4.1 implies

$$dist(s_1, t) = |j - (i + 1)| + 1 = j - i.$$

If $j \leq i$, then the only edge in π_s that contains $L(t)$ in its label is (s, b_i) (as before, from page 54). Therefore, $s_1 = b_i$. By Lemma 3.4.4,

$$dist(s, t) = i - j + 2$$

and by Lemma 3.4.1,

$$dist(s_1, t) = |j - i| + 1 = i - j + 1.$$

Case 3: $t \in D_j$**:** If $j > i$, then the edge order and labels at s (as defined in page 54) imply that the only edge in π_s that contains $L(t)$ in its label is (s, b_{i+1}). Therefore, $s_1 = b_{i+1}$. Lemma 3.4.5 implies

$$dist(s, t) = |j - i| + 1 = j - i + 1$$

and Lemma 3.4.2 implies

$$dist(s_1, t) = j - (i + 1) + 1 = j - i.$$

If $j < i$, then the only edge in π_s that contains $L(t)$ in its label is (s, b_i) (as before, from page 54). So, $s_1 = b_i$. Therefore, $s_1 = b_i$. By Lemma 3.4.5,

$$dist(s, t) = |j - i| + 1 = i - j + 1$$

and by Lemma 3.4.2,

$$dist(s_1, t) = i - j.$$

If $i = j$, then the only edge in π_s that contains $L(t)$ in its label is (s, b_i) (as before, from page 54). So, $s_1 = b_i$. Since both s and t are adjacent to b_i, clearly

$$dist(s, t) = 2$$

and

$$dist(s_1, t) = 1.$$

In all the cases, we get $dist(s_1, t) = dist(s, t) - 1$, which proves the lemma. □

Although Lemmas 3.4.6 to 3.4.9 determine only the first edge used by P to leave its current position s, we can apply the lemmas repeatedly at each intermediate vertex to trace the whole path from s to the destination of P. The following theorem uses this idea to prove the optimality of our scheme after showing that the scheme is a valid 2-OLIRS.

Theorem 3 *The above scheme is an optimal 2-OLIRS of the k-garland graph.*

Proof: We first show that the above routing scheme is a valid OLIRS and that it has at most two intervals per edge label. Then we prove that the scheme is optimal.

To show that our scheme is a valid OLIRS, we have to prove that from each vertex, there is an outgoing edge for every destination vertex. At each vertex u in set B or in set D_i, $1 \leq i < k$, all the edge labels together cover the labels of the vertices in $\bigcup_{i=1}^{k-1}(C'_i \cup D_i) \cup C'_k$ which is the whole vertex set. At each vertex u in set C_i, $1 \leq i \leq k$, the last edge in π_u has the label $[1, n]$ which covers all vertex labels. So, from every vertex, there is an outgoing edge for every destination vertex.

Now we show that our scheme has at most two intervals per edge label. By Observation 3.4.3, the labels of the vertices in the set $\bigcup_{i=i_1}^{i_2}(C'_i \cup D_i)$ form a single linear interval for $1 \leq i_1 \leq i_2 < k$. The same is true for the set $\bigcup_{i=i_1}^{i_2}(C'_i \cup D_i) \cup C'_{i_2+1}$ for $1 \leq i_1 \leq i_2 < k$. From the edge labeling, it is clear that the label of each outgoing edge of a vertex in set B or in set D_i, $1 \leq i < k$, consists of only one linear interval. For a vertex $c \in C_i$, $1 \leq i \leq k$, each edge connecting c to a vertex in $C_i \cup \{b_i\}$ consists of one linear interval and each edge connecting c to a vertex in $D_{i-1} \cup D_i$ consists of two linear intervals. So, our scheme is a valid 2-OLIRS.

To prove the optimality of the routing scheme, suppose a packet P whose destination is vertex t is at another vertex s_0. Because the labels of all the outgoing edges

at each vertex cover all the vertex labels of the graph, vertex s_0 has (at least) one edge containing $L(t)$ in its label. So, packet P must leave vertex s_0 through some edge (s_0, s_1). If s_1 and t are not the same vertices, the same argument applies to s_1 also. So, P must leave vertex s_1 through some edge (s_1, s_2). Let $(s_0, s_1, s_2, \ldots, s_i, \ldots)$ be the path thus followed by P. Now, from the statements of Lemmas 3.4.6 to 3.4.9, it is obvious that vertex s mentioned in those lemmas can be any vertex of the k-garland graph except t. Therefore, by those lemmas, for all edge (s_i, s_{i+1}) on the path $(s_0, s_1, s_2, \ldots, s_i, \ldots)$, $dist(s_{i+1}, t) = dist(s_i, t) - 1$, provided $s_i \neq t$. Hence, at each step P decreases its distance to t by exactly one until it reaches t. Therefore, the path is guaranteed to reach t and it does so in $dist(s, t)$ steps, which is optimal. □

It follows directly from the above proof that if a k-garland graph has no edge connecting any vertex in C_i to any vertex in $D_{i-1} \cup D_i$ $(1 \leq i \leq k)$, then the graph has an optimal 1-OLIRS.

3.5 Graphs without Optimal 1-LIRS's

In this section, we give optimal 1-OLIRS's of the graphs in Figure 2.6 that are shown by Bakker et al. [BvLT91] to have no optimal 1-LIRS's.

In Figures 3.10 to 3.15, we present the vertex and edge labels and edge ordering of all the graphs in Figure 2.6. In the figures in this section, each vertex v is marked by a circled integer, where the integer denotes the label $L(v)$. The mark of the form "$i : [x, y]$" near vertex v along an edge (v, w) is used to indicate that (v, w) is the ith edge in π_v and that the label of (v, w) is the interval $[x, y]$.

Figure 3.10 illustrates an optimal 1-OLIRS of the graph in Figure 2.6(a). We have omitted the edge labels and orderings at the vertices labeled with 3, 4, 5 and 6 because the edge labels and orderings of vertices 4 and 6 are exactly the same as

those of vertex 2 and the edge labels and orderings of vertices 3 and 5 are similar to those of vertex 1.

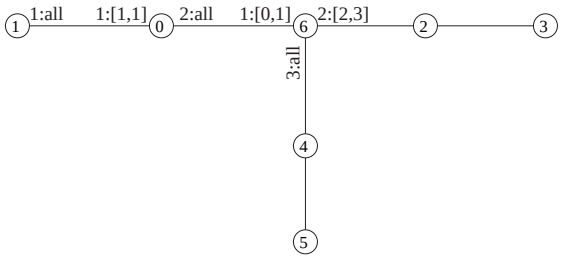


Figure 3.10: An optimal 1-OLIRS of the graph in Figure 2.6(a)

Figures 3.11 to 3.15 show optimal 1-OLIRS's of the graphs in Figures 2.6(b) to 2.6(i), respectively. None of these graphs have optimal 1-LIRS's [BvLT91]. As in the case of Figure 3.10, we have marked the edge labels and orderings of selected vertices of each of the following graphs for simplicity. The edge labels and orderings of the rest of the vertices are either trivial or symmetric to other vertices.

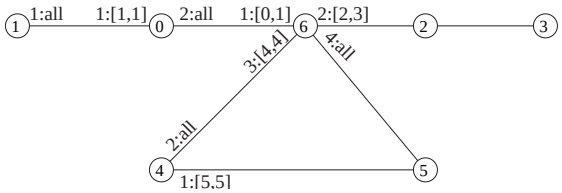


Figure 3.11: An optimal 1-OLIRS of the graph in Figure 2.6(b)

Bakker et al. [BvLT91] also prove that a cycle of more than four vertices has no optimal 1-LIRS. Since a cycle of n vertices is a 1-dimensional torus, it has an optimal 1-OLIRS by Theorem 2.

Note that Bakker et al. actually prove that a graph G that contains any of the graphs discussed in this section as a subgraph of shortest paths has no optimal LIRS of compactness one. However, it is obvious that G may have an optimal OLIRS of compactness one.

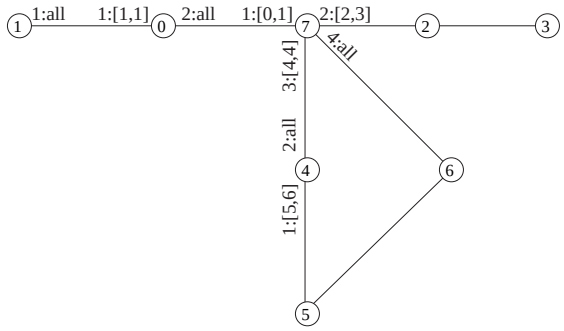


Figure 3.12: An optimal 1-OLIRS of the graph in Figure 2.6(c)

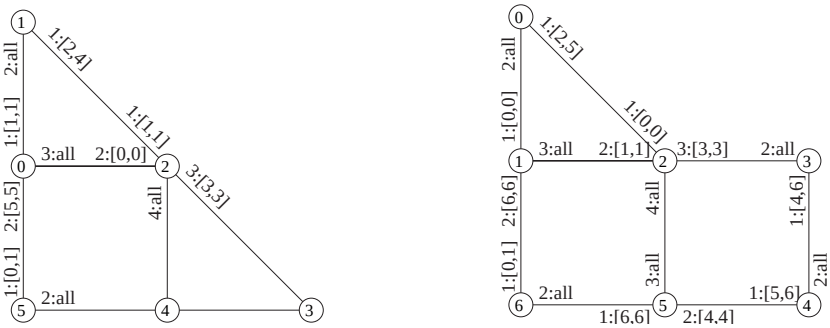


Figure 3.13: Optimal 1-OLIRS's of the graphs in Figures 2.6(d) and 2.6(e)

Also note that the three graphs in Figures 3.10 and 3.15 are lithium graphs and have no 1-LIRS's even when we do not consider optimality [FG98].

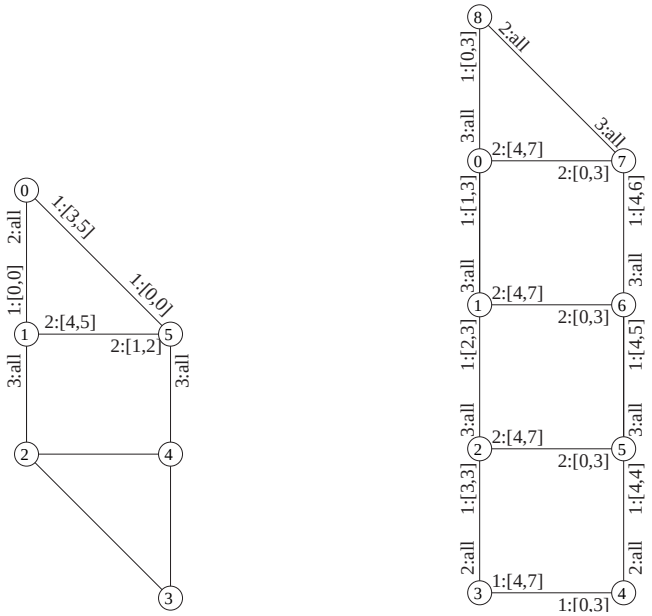


Figure 3.14: Optimal 1-OLIRS's of the graphs in Figures 2.6(f) and 2.6(g)

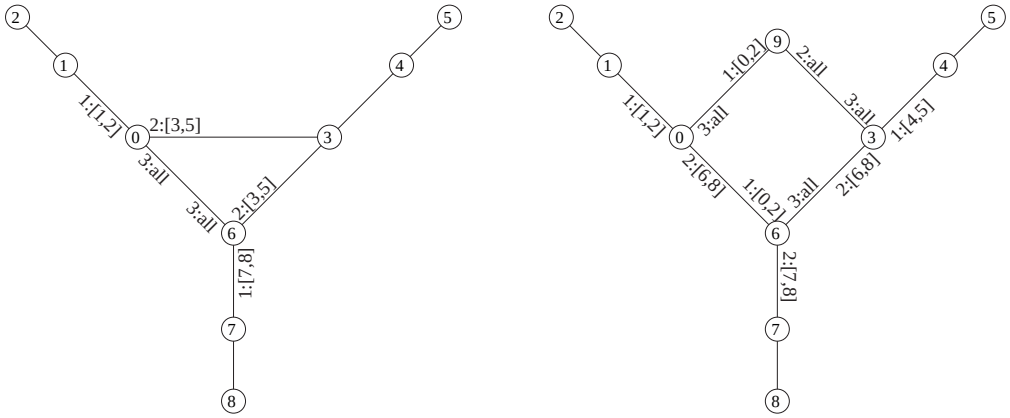


Figure 3.15: Optimal 1-OLIRS's of the graphs in Figures 2.6(h) and 2.6(i)

Chapter 4

Conclusion

In this thesis, we have proposed the concept of an Ordered Interval Routing Scheme (OIRS), a new type of interval routing scheme that is different from an IRS in two ways. An OIRS allows non-disjoint edge labels on the outgoing edges at a vertex which is not allowed in an IRS. Secondly, the processor at each vertex considers the labels of the outgoing edges in a predetermined order. In an IRS, such orderings are insignificant because of the disjoint edge labels at each vertex.

For certain graph classes, we have shown improvements in the size of routing tables achieved by OIRS's compared to IRS's. Using IRS's, we can guarantee optimal routing in any k-tree using 2^{k+1} intervals per edge label, and this is the best known result for k-trees. However, we have shown that any k-tree has an optimal 2^{k-1}-OIRS. For a D-dimensional torus, we have presented an optimal OLIRS of compactness one; this compactness can be achieved by an IRS only if the size of at most one dimension of the torus is greater than four. We have also defined the class of k-garland graphs that have optimal 2-OLIRS's, but not always have optimal 2-IRS's. A similar result has been shown for the Petersen graph. Finally, ten small graphs without optimal 1-LIRS's have been shown to have optimal 1-OLIRS's.

We have shown our results for selected graph classes for which the compactness of an IRS does not depend on the number of vertices n. For many graphs, the compactness of an IRS depend on n. For example, certain subclasses of planar graphs [FJ88, GP96, KRŠ00] need compactness proportional to $\sqrt{n}$ in any IRS. Examining OIRS's of such graphs is an obvious extension of our work.

Another interesting problem is to fully characterize the graphs that have optimal k-OIRS's for some fixed value of k. The problem is NP-complete for the case of optimal IRS's even for small values of k [EMZ02, FGS96]. Determining whether the case of OLIRS's is solvable in polynomial time or not is also an interesting open problem.

We can also consider the problem of determining the minimum k such that a given graph has an optimal k-OIRS. This is likely to be an intractable problem, considering the fact that the corresponding problem for IRS is NP-hard [FGS96]. For the case of k-trees, an easier problem is to determine whether it is possible to ensure optimal routing in any k-tree with an OIRS of compactness less than 2^{k-1}. The question is yet unsolved for the case of IRS, but the bound is 2^{k+1} in this case [NN98].

In this thesis, we have represented a network using an unweighted graph. Interval routing schemes have been investigated for weighted graphs also, where the weight of an edge models the cost of communication through the corresponding link. Two models of weighted graphs have been studied [BvLT91, EMZ02]. One is the *fixed cost model* where each edge has some constant weight. In the *dynamic cost model*, the aim is to determine an IRS that ensures optimal routing for every possible values of edge weights. Investigating OIRS's for the fixed cost and the dynamic cost models is another possible extension of our work.

Bibliography

[BLS99] A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph Classes: a Survey*. SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1999.

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

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

[CLR90] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1990.

[Die00] R. Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, second edition, 2000.

[EMZ02] T. Eilam, S. Moran, and S. Zaks. The complexity of the characterization of networks supporting shortest-path interval routing. *Theoret. Comput. Sci.*, 289(1):85–104, 2002.

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

[FGNT01] M. Flammini, G. Gambosi, U. Nanni, and R. B. Tan. Characterization results of all shortest paths interval routing schemes. *Networks*, 37(4):225–232, 2001.

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

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

[Gav00] C. Gavoille. A survey on interval routing. *Theoret. Comput. Sci.*, 245(2):217–253, 2000.

[GM84] M. Gondran and M. Minoux. *Graphs and Algorithms*. John Wiley & Sons Ltd., Chichester, 1984.

[GP96] C. Gavoille and S. Pérennès. Lower bounds for interval routing on 3-regular networks. In Nicola Santoro and Paul Spirakis, editors, 3^{rd} *International Colloquium on Structural Information & Communication Complexity (SIROCCO)*, pages 88–103. Carleton University Press, 1996.

[GZ03] C. Gavoille and A. Zemmari. The compactness of adaptive routing tables. *J. Discrete Algorithms*, 1(2):237–254, 2003.

[INM91] *The T9000 Transputer Products Overview Manual, INMOS*. SGS-Thomson Microelectronics, 1991.

[KKR94] E. Kranakis, D. Krizanc, and S. S. Ravi. On multi-label linear interval routing schemes. In *Proceedings of the 19th International Workshop on Graph-Theoretic Concepts in Computer Science*, volume 790 of *Lecture Notes in Comput. Sci.*, pages 338–349, Berlin, 1994. Springer-Verlag.

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

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

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

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

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

[WMT93] P. H. Welch, M. D. May, and P. W. Thompson. *Networks, Routers and Transputers: Function, Performance and Application*. IOS Press, Netherlands, February 1993.