Technical Report CS-2001-04

Optimum Multi-dimensional Interval Routing Schemes on Networks with Dynamic Cost Links

Yashar Ganjali

yganjali@math.uwaterloo.ca

http://www.cs.uwaterloo.ca/~yganjali

Department of Computer Science, University of Waterloo
Waterloo, Ontario N2L 3G1, Canada

February 2001

Abstract

One of the fundamental tasks in any distributed computing system is routing messages between pairs of nodes. An Interval Routing Scheme (IRS) is a space efficient way of routing messages in a network. The problem of characterizing graphs that support an IRS is a well-known problem and has been studied for some variants of IRS. It is natural to assume that the costs of links may vary over time (*dynamic cost links*) and to try to find an IRS which routes all messages on shortest paths (*optimum IRS*). In this paper, we study this problem for a variant of IRS in which the labels assigned to the vertices are d-ary integer tuples (d-*dimensional IRS*). The only known results in this case are for specific graphs like hypercubes, n-dimensional grids, or for the 1-dimensional case. We give a complete characterization for the class of networks supporting multi-dimensional strict and linear (which is a variation of IRS) interval routing schemes with dynamic cost links.

Keywords: Interval routing, networks, routing algorithms, dynamic, multi-dimensional, characterization.

1 Introduction

One of the fundamental tasks in any distributed computing system is routing messages between pairs of nodes (processors). The classic method for routing messages in a network is to store a routing table at each node of the network. A routing table has one entry for each destination node that indicates which outgoing link should be used to forward a message going to that destination. This is an example of a *routing scheme* which in general is a strategy that determines which path a message, originating from a known source and going to a known destination, should take in the network.

Using routing tables is not always efficient, because each routing table requires $\Omega(n)$ worst case space in an n-node network. An *interval routing scheme* (IRS), which was originally proposed by Khatib and Santoro [SK85], is a more efficient routing scheme. In this method each node v of the network is assigned a unique integer label, $L(v)$, taken from $\{1, 2, ..., n\}$ (n is the number of nodes in the network). Each outgoing link e at a node v is also associated with a cyclic interval I_e, which denotes the set of destinations reachable through e. This routing method was used in the C104 Router Chip of INMOS T9000 transputer design [INM91].

Throughout this paper, a network is modeled by a graph $G = (V, E)$. The set of vertices V represents the nodes of the network and the set of edges E represents the links between the nodes. Whenever there is no ambiguity, we will use the terms nodes and vertices and also links and edges interchangeably. We assume that the graph is simple, connected and does not have any self-loops. For any edge $(u, v) \in E$ we will use both (u, v) and (v, u) in order to assign two unidirectional labels to the edge.

A *Linear Interval Routing Scheme* (LIRS) is a variant of IRS in which the intervals assigned to the links are not cyclic. Another variation is a *Strict Interval Routing Scheme* (SIRS). In SIRS, no interval associated with a link e, which is adjacent to a node v, can contain the label of v. For example, a node with label 4 cannot have a link that is labeled with the interval [2..8]. An IRS which is both linear and strict is denoted by SLIRS.

In any IRS, the routing is completed in a distributed way. At each intermediate node p, the routing process ends if the destination of the message, $dest$, is p. Otherwise, the message is forwarded through a link e labeled by an interval I_e, such that $dest \in I_e$. This method requires $O(l)$ space at each node (l is the number of links at that node), which is a more efficient memory allocation than required by routing tables.

The class of networks which have an LIRS or SLIRS such that each message eventually reaches its destination has been characterized by Fraigniaud and Gavoille [FG98]. There, the routes traversed by messages are not necessarily shortest paths. If an IRS routes all messages on shortest paths, the IRS is called an *optimum IRS*. Given a graph G and an IRS defined on G, it is reasonable to assume that the labels of the vertices remain fixed over time, but the cost of the links may vary. Assuming that the costs of the links are non-negative numbers that vary over time, the question is: can we always relabel the links of G such that the path traversed by any message always remains optimum? In other words, does a given graph G having dynamic cost links, have an optimum IRS?

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

Here, a natural question is: can we slightly change IRS (LIRS, SLIRS, etc.) to expand the class of graphs supporting LIRS, SIRS and SLIRS, or improve the length of routing paths?

One way to make IRS more flexible and the routing more efficient is to assign more than one label to each link. Bakker, Leeuwen and Tan have proved that the class of graphs supporting LIRS with k ($k \geqslant 1$) intervals per link is a strict subset of the class of graphs supporting LIRS with $k+1$ links [vLT87, BvLT91]. Hence, increasing the number of intervals at each link increases the power of interval routing and allows a larger class of networks to support IRS. Narayanan and Nishimura study the problem of finding the number of intervals needed at each link for the case of optimum IRS on k-trees [NN98]. Using the notion of tree-width, Bodlaender *et al* also give an interesting characterization of the graphs supporting optimum IRS with dynamic cost links and k intervals per link [BvLTT97].

Another interesting variant is to assign multi-dimensional labels to the nodes and multi-dimensional interval labels to the links of the network. This extension of IRS is called a *Multi-dimensional Interval Routing Scheme (MIRS)* and was originally proposed by Flammini *et al.* [FGNT98]. More formally, in a d-dimensional MIRS we will assign a d-dimensional label which is a d-ary tuple of the form $(p_1, p_2, ..., p_d)$, $1 \leqslant p_i \leqslant n$, for $1 \leqslant i \leqslant d$, to each node of the network. We will also assign a d-dimensional intervals to each outgoing link (at each node) where a d-dimensional interval, denoted by $I = [a_1..b_1, a_2..b_2, ..., a_d..b_d]$ ($a_i, b_i \in 1, 2, ..., n$ for $1 \leqslant i \leqslant d$) is the set of all d-ary tuples, like $P = (p_1, p_2, ..., p_d)$, such that $a_i \leqslant p_i \leqslant b_i$, for every i, $1 \leqslant i \leqslant d$.

The routing process in an MIRS is quite similar to the routing process in a 1-dimensional IRS. A *Linear MIRS (MLIRS)*, *a Strict MIRS (MSIRS)*, and *a Strict and Linear MIRS (MSLIRS)*, are defined analogously to 1-dimensional LIRS, SIRS and SLIRS, respectively.

Let us consider a graph G which has a d-dimensional MIRS with k intervals associated with each link. If for any pair of nodes s and t in $V(G)$, the message originating from s eventually reaches t, we say that *G is in $\langle k, d\rangle$-MIRS* or *G supports $\langle k, d\rangle$-MIRS.* Classes of networks supporting $\langle k, d\rangle$-MLIRS and $\langle k, d\rangle$-MSLIRS are defined similarly.

The class of networks supporting $\langle 1, d\rangle$-MLIRS and $\langle 1, d\rangle$-MSLIRS (not an optimum IRS) has already been characterized [Gan01]. In this paper, we completely characterize the class of networks supporting an optimum $\langle 1, d\rangle$-MSLIRS with dynamic cost links. This is a natural generalization of the characterization results (for the 1-dimensional case) mentioned above.

The rest of this paper is organized as follows: in Section 2 we introduce some preliminary concepts. Then, in Section 3 we show which graphs can have an optimum $\langle 1, d\rangle$-MSLIRS with dynamic cost links. Section 3.1 demonstrates how to assign d-dimensional labels to the vertices of a given graph and Section 3.2 shows how to assign labels to the links of the graph, for a given set of link costs. Finally in Section 4 we conclude and give a list of open problems.

2 Preliminaries

In this section we just briefly mention some of the important definitions used throughout this paper. For basic graph theoretic definitions and notation we refer the reader to standard texts [BM76, Wes96]. Any graph considered in this paper is assumed to be connected, simple and undirected. If removing a vertex v disconnects a graph G, v is called a *cut-vertex* in G. A connected graph having no cut-vertex is called a *block*. Every block which has at least three vertices is 2-connected. A *block of a graph* is a maximal subgraph that is a block. A vertex joining two blocks of a graph G is called an *articulation point of* G. A vertex which is not an articulation point is called a *non-articulation* point.

Observation 1. *If a graph G is connected, removing any of its articulation points will disconnect the graph. It follows directly from this observation that any two blocks of a graph share at most one vertex (which is an articulation point).*

Any d-dimensional label associated with a node of a network denotes a point in d-dimensional Cartesian space with integetr coordinates. We will use this point and the label interchangeably. Let us consider a set of points P in d-dimensional space. If for any dimension i, $1 \leqslant i \leqslant d$, the ith coordinate of a point b in P is less than or equal to the ith coordinate of every other point in P, b is called a *minimum point for the ith dimension.* A maximum point is defined similarly. A *boundary set B of P* is a minimal set of points in P containing a minimum and a maximum point for each dimension $i, 1 \leqslant i \leqslant d$, where one point can be both the minimum and the maximum point for some or many dimensions.

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

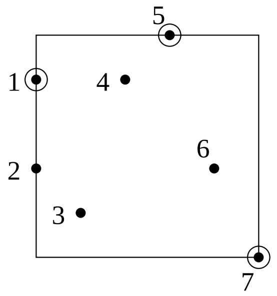

Figure 1: An example of a boundary set in 2-dimensional space.

Clearly, for any set of points in d-dimensional space, the number of points in any boundary set is at most $2d$.

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

Example 2. *The region R containing all the points in the second quadrant in the plane, such that $x_1 \leq -1$ and $x_2 \geq 1$ is denoted by $(\overleftarrow{-1}, \overrightarrow{1})$ (Figure 2).*

For a region R and the ith dimension, if R contains points with infinitely large positive (negative) values in the ith dimension, the region is said to be *open in the positive (negative) side of the ith axis* and the positive (negative) direction of the ith axis is said to be an *open direction for R* and will be denoted by $\overrightarrow{x_i}$ ($\overleftarrow{x_i}$). It is worth mentioning that a region is defined by the origin and the set of open directions. The negative direction of the first axis and the positive direction for the second axis are open in the region shown in example 2, so this

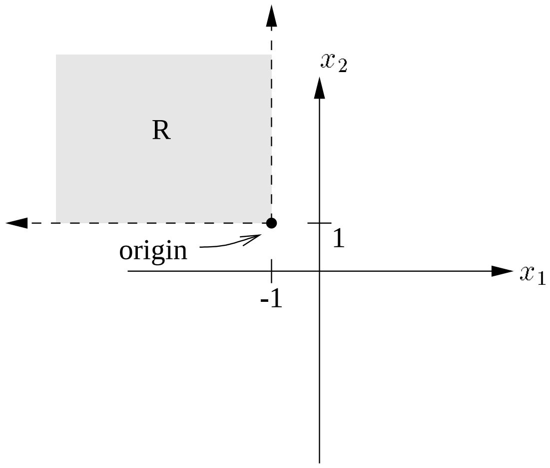

Figure 2: The region R with two open directions in 2-dimensional space.

region has two open directions. We can consider d-dimensional space as a region with origin $(0, 0, ..., 0)$ and call it *the universal region*. This region has $2d$ open directions (one positive and one negative direction for each of the d axes) and can be denoted by $(\overleftrightarrow{0}, \overleftrightarrow{0}, ..., \overleftrightarrow{0})$.

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

Example 3. *The complement of the region* $R = (\overleftarrow{-1}, \overrightarrow{1})$ *is the region* $(\overrightarrow{-1}, \overleftarrow{1})$.

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

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

3 Characterization

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

We start with an observation about boundary sets.

Observation 2. *For a set of points P in d-dimensional space and a boundary set B of P, any d-dimensional interval I containing all the points in B contains all of P. This is true because if $Q = (q_1, q_2, ..., q_d)$ is an arbitrary point in P, then for each dimension i, $1 \leqslant i \leqslant d$, there is a minimum point m_i and a maximum point M_i in P (m_i and M_i can be the same) such that $m_i \leqslant q_i \leqslant M_i$. Since I contains these minimum and maximum points, the ith dimension of I covers the ith dimension of Q and so I contains Q. Therefore, I contains all of the points in P.*

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

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

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

G is a connected and non-trivial graph, so v has at least two adjacent vertices. We let u be an arbitrarily chosen neighbor of v. Recalling that the links have dynamic costs, there must be a labeling of the links of G for any assignment of costs. We can consider a case in which the cost of the link (v, u) is 1 and the cost of any other link adjacent to v is arbitrarily large, say M (e.g. M is at least n^2). The cost of any other link of the graph is set to be 1 (Figure 3).

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

This argument together with Observation 2 shows that the interval I assigned to (v, u) contains all other points including the points in the boundary set B. Therefore, I contains v, which contradicts the fact that the IRS is strict. ∎

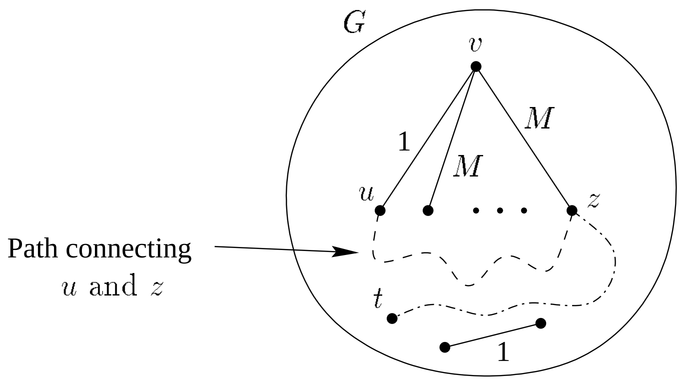

Figure 3: Costs assigned to the links of the graph G. Here, M is at least n^2.

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

3.1 Labels of Vertices

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

We will use a structure, which we call *the block tree of a graph*, in order to find such a labeling of vertices. This structure defines an ordering of the vertices of the graph, based on which we will assign the labels to the vertices. By using this ordering we will assign labels of vertices such that all the non-articulation points will be in a boundary set and articulation points are placed so that they are not contained in any boundary set. In other words, when we assign a point in d-dimensional space, this assignment is done in a way that in some direction (one of the $2d$ directions of the d-dimensional space) this point is a minimum or a maximum point and we will not place any other point beyond this one in that specific direction.

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

Whenever there is no ambiguity, we will use the same name for a block in G and its corresponding vertex in $BT(G)$ and also for any articulation point in G and its corresponding vertex in $BT(G)$. Figure 4 depicts an example of a block tree. The graph indicated in this example, has four blocks B_0, B_1, B_2, B_3 and two articulation points u_1 and u_2. The vertex u_1 is connects B_0, B_1 and B_2 in G, so in the block tree $BT(G)$ the vertex representing u is connected to the vertices representing B_0, B_1 and B_2.

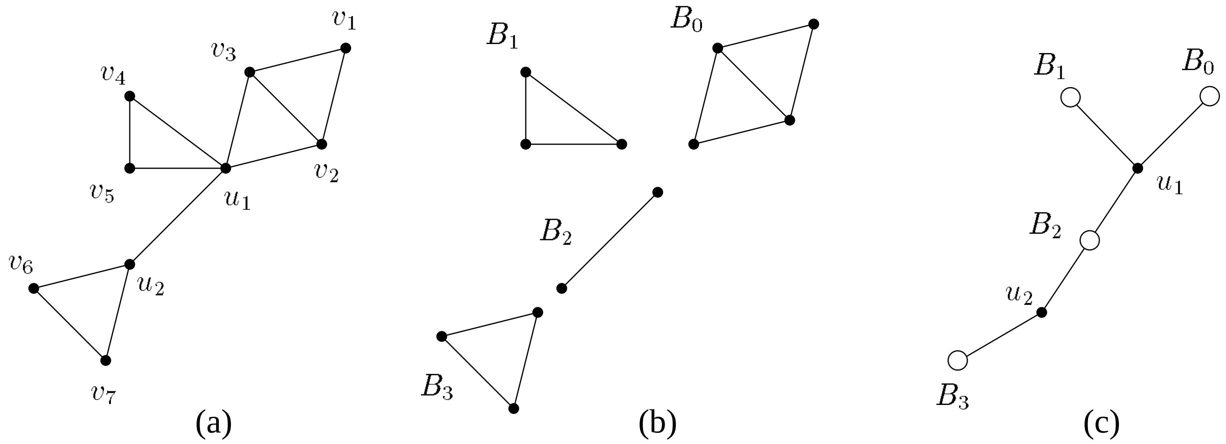

Figure 4: (a) A graph G (b) blocks of G and (c) the block tree of G.

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

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

Intuitively, we will assign the whole d-dimensional space to the root B_0 of the block tree. The regions assigned to the vertices in a subtree are subregions of the region assigned to the root of that subtree. Also, in a node v which is the root of more than one subtrees, the regions assigned to different subtrees will be disjoint. This property will allow us to assign intervals to the links of the graph without any conflicts, as we will see later in this paper.

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

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

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

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

Example 5. *For the graph G denoted in Figure 4 (a), the block tree is depicted in Figure 4 (b). Here, B_0 is the root of the block tree. In this graph, $OD(u_1) = OD(B_2) = \{\overleftarrow{x_2}, \overleftarrow{x_3}\}$ which is the same as $OD(v_6) \cup OD(v_7)$ (the non-articulation points in the subtree rooted at B_2 or u_1).*

The next step is to assign an origin to each set of open directions associated with a vertex in $BT(G)$. The origin of the region assigned to v (any vertex in $BT(G)$) will be denoted by $X(v)$. This will be used for calculating the origin of each non-articulation point later.

We start with the block B_0 and let $X(B_0) = (0, 0, ..., 0)$. If G has $2d$ non-articulation points, recalling that we have already assigned all $2d$ open directions to B_0, the region assigned to B_0 would be the universal region, $(\overset{\leftrightarrow}{0}, \overset{\leftrightarrow}{0}, ..., \overset{\leftrightarrow}{0})$. To compute the origin of the region assigned to u, a child of a vertex v with a known origin, we let $X'(v, u) =$

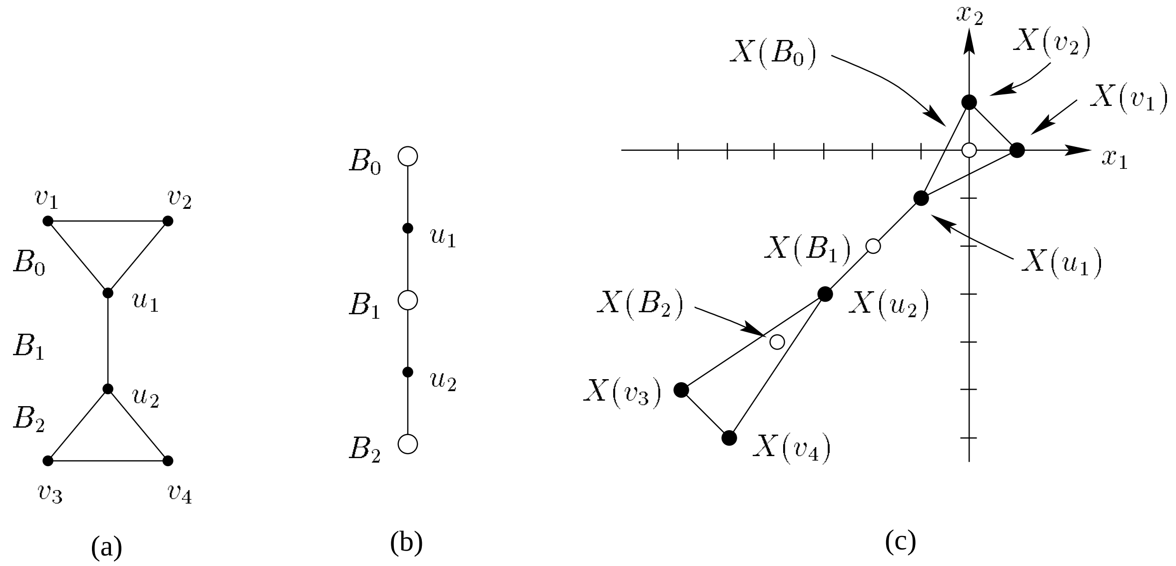

Figure 5: (a) A graph G (b) The block tree of G rooted at B_0 (c) The origin of each region assigned to each block and each vertex in G.

move($X(v), OD(v)$). Then we let $X(u) =$ move($X'(v,u), OD(u)$). This is the origin of the region associated with u. Since the root of the tree $BT(G)$ has a known origin, by repeating this step every vertex in $BT(G)$ will eventually have an origin.

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

Example 6. *In the graph G depicted in Figure 5 the region associated with B_0 is $(\overset{\leftrightarrow}{0}, \overset{\leftrightarrow}{0})$. To find $X(v_1)$ we move $X(B_0) = (0,0)$ in the direction of $OD(B_0)$ (which includes all four directions) and get $(0,0)$. Then this point is moved in the direction of $OD(v_1)$ which is $\overrightarrow{x_1}$. Therefore, $X(v_1) = (1,0)$.*

In the optimum $\langle 1, d\rangle$-MLIRS defined on G, we let the label assigned to each vertex v, denoted by $L(v)$, be the same as the origin of the region assigned to that vertex ($L(v) = X(v)$. Figure 6 is the pseudo-code for labeling the vertices of a graph G. We can verify that this algorithm can be executed in $O(n^2)$ time.

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

Observation 3. *For a block B in G, we let $X'(B) = (a_1, a_2, \ldots, a_d)$ be the point resulting from moving $X(B)$ in the direction of $OD(B)$. If v is a non-articulation point in B, then,*

$L(v) = (b_1, b_2, ..., b_d)$ where $b_i = a_i$ for all i, $1 \leqslant i \leqslant d$ except for one dimension j such that $OD(v) = \overleftarrow{x_j}$ or $OD(v) = \overrightarrow{x_j}$. In this dimension $b_j = a_j + 1$ or $b_j = a_j - 1$ based on the direction of $OD(v)$.

Lemma 2. If v is a vertex in a block B or in the subtree of $BT(G)$ rooted at B, the region assigned to v, R_v by the VL algorithm, is a subregion of R_B, the region assigned to B.

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

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

If R_v is the region assigned to a vertex v in $BT(G)$, the previous lemma shows that all vertices in the subtree of $BT(G)$ rooted at v are in R_v. With an argument similar to that of the proof of Lemma 2 we can verify the following lemma.

Lemma 3. Any vertex not in the subtree of $BT(G)$ rooted at v is in the region $\overline{R_v}$.

Lemma 4. For $e = (u, v)$ an edge in block B and z a vertex contained in a block $B' \neq B$ which is in the subtree of $BT(G)$ rooted at B, if the shortest path from u to z goes through e, then there is a shortest path from u to any other vertex t in the subtree of $BT(G)$ rooted at B which goes through e.

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

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

3.2 Labels of Links

In this section we show that for a given graph G and the labels assigned to the vertices of G using the Vertex Labeling (VL) algorithm, introduced in Section 3.1, we can always find labels for the links of G for any set of link costs. By this labeling of links, any message from any source vertex to any destination vertex in G will be routed on a shortest path.

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

Lemma 5. With the labels assigned by the VL algorithm, for any subset C of the non-articulation vertices in a block B, we can always find a d-dimensional interval containing the vertices in C and no other vertex in B.

Proof. We assume that $X'(B)$ is the point resulting from moving $X(B)$ in the direction of $OD(B)$. Without loss of generality, we can assume that $X'(B) = (0, 0, ..., 0)$ (otherwise we can shift every label by $-X'(B)$). By Observation 3 one can verify that the label of each non-articulation vertex v in B is of the form $(0, 0, ..., 1, ..., 0)$ or $(0, 0, ..., -1, ..., 0)$ (exactly one coordinate is 1 or -1 and the rest of coordinates are all 0).

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

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

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

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

Lemma 6. With the labels assigned by the VL algorithm, for any subset C of the vertices in a block B which does not include the parent of B in BT(G), we can always find a d-dimensional interval containing exactly the vertices in C and no other vertex.

Proof. We let B' be the set resulting from replacing each articulation point z in B with the set of non-articulation points in the subtree(s) of $BT(G)$ rooted at z say $z_1, z_2, ..., z_t$

($B' = B - \{z\} \cup \{z_1, z_2, ..., z_t\}$ for any articulation point z in B). These new vertices ($\{z_1, z_2, ..., z_t\}$) all together represent the articulation point z in B.

Lemma 5 shows that for any subset C' of B' there is an interval containing exactly the vertices in C'. If C contains the articulation point z, we let $C' = C - \{z\} \cup \{z_1, z_2, ..., z_t\}$. By Lemma 5 there is an interval I containing exactly the vertices in C'. If I contains aal of the points $z_1, z_2, ..., z_t$, it will also contain z, so this interval contains all the vertices in C. On the other hand, any vertex v in $B - C$ is in $B' - C'$. It means if I contains a vertex v in $B - C$, it also contains a point from $B' - C'$ which is impossible by Lemma 5. ■

Example 7. *In graph G shown in Figure 5, the origin of region associated with B_2 is $(-4, -4)$ and moving this point in the direction of $OD(B_2)$ results in the point $(-5, -5)$, because $OD(B_2)$ contains the negative direction of both axes. If we consider this point as the origin, the coordinates of $X(v_3)$ and $X(v_4)$ (v_3 and v_4 are non-articulation points in B_2) are $(-1, 0)$ and $(0, -1)$ respectively. If $C = \{v_3, v_4\}$ then the interval covering C would be $I = [-1..0, -1..0]$.*

In the VL algorithm, each block has at most one articulation point as its parent in $BT(G)$. For a block B and v the vertex in $BT(G)$ which is the parent of B, Lemma 2 shows that all the vertices in the subtree of $BT(G)$ rooted at B are contained in the region R_B. Lemma 3 states that any other vertex is in $\overline{R_B}$.

Lemma 7. If I is an interval in the region $\overline{R_B}$ containing the articulation point v, we can find another interval I' such that I' contains all the vertices in the subtree rooted at v and the same set of points in $\overline{R_B}$ as I. Also, if I is an interval in R_B containing v, we can find another interval I' such that I' contains all the vertices which are not in the subtree rooted at v and the same set of points in R_B as I.

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

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

Lemma 8. For any assignment of costs to the edges of a graph G, and with labels assigned to the vertices of the graph by the VL algorithm, we can always find suitable intervals for the links so that the result is an optimum $\langle 1, d\rangle$-MSLIRS.

Proof. First, we will consider a link $e = (u, v)$ and the set of vertices S_e reachable (by a shortest path) through e. The link e is in a block, say B, of G. We let $Q_1 = B \cup S_e$ that is Q_1 is the subset of vertices in B that are contained in S_e. Let us consider a vertex z which is a vertex in the subtree of $BT(G)$ rooted at B. If z is not in B but is contained in S_e, by Lemma 4, S_e also contains all the vertices in the subtree of $BT(G)$ containing z (we denote the vertices in this subtree by the set Q_2). Finally, if z is not in a child block of B, S_e must contain all the vertices that are not a child of B (we denote the set containing all these vertices by Q_3).

Lemma 6 shows that we can always find an interval covering exactly the vertices in Q_1. If there is any point in Q_2 (or Q_3) then the articulation point joining B to the vertices in Q_2 (Q_3 respectively) must also be in Q_1. This is because this articulation point is the only vertex connecting B to the child subtree (or the vertices of G that are not contained in the subtree rooted at B) and so the only way to reach those vertices. Lemma 7 shows that we can always find an interval containing the same set of points in Q_1 at the previously assigned interval, and covering all the points in Q_2 (Q_3). This completes the proof. ■

Example 8. *In the graph G of Figure 5, let us assume that the cost of the edge (v_1, v_2) is extremely large and the cost of any other edge is 1. The interval assigned to the edge $e = (v_1, u_1)$ should contain all the vertices, except v_1. The interval $[-1..0, -1..1]$ contains $Q_1 = \{v_2, u_1\}$. Therefore we can find another interval which contains all vertices in the subtree of $BT(G)$ rooted at u_1 (Lemma 7). This interval is $[-6..0, -6..1]$.*

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

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

Proof. Lemma 1 states that any graph having more than $2d$ non-articulation points can not support an optimum $\langle 1, d\rangle$-MSLIRS with dynamic cost links. By Lemma 8 if a graph G has at most $2d$ non-articulation points we can always find a fixed labeling for the vertices such that for any costs assigned to the links, we can find intervals for each link to support an optimum $\langle 1, d\rangle$-MSLIRS. ■

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

4 Conclusion and open problems

Characterizing the class of graphs which support different variations of IRS is a well-known problem. Assuming that the costs of links (and therefore the intervals assigned to links) may vary over time for a fixed set of labels assigned to nodes seems quite natural. In this paper, we completely characterized the class of networks supporting an optimum $\langle 1, d\rangle$-MSLIRS with dynamic cost links. Theorem 1 shows that adding the number of dimensions strictly increases the power of IRS. In other words, for any $d \in \mathbb{N}$ there is a is a class of graphs which does not support optimum $\langle 1, d\rangle$-MSLIRS with dynamic cost links, but supports an optimum $\langle 1, d+1\rangle$-MSLIRS.

Characterizing the class of graphs supporting an optimum $\langle k, d\rangle$-MSLIRS with dynamic cost links which is a generalization of the result of this paper, is a very interesting open problem. Even if we consider the case with fixed cost links the problem is still open. Assuming the case in which the IRS is not necessarily optimum, another open problem is to find lower bounds on the length of paths traversed by messages. We can also consider the variants in which the IRS is not necessarily linear or strict. For more open problems in this area, we refer the reader to a comprehensive survey by Gavoille [Gav00].

5 Acknowledgments

I would like to thank my supervisor, Prof. Naomi Nishimura, for her thoughtful comments, guidance and support. I am also grateful to Mohammadtaghi Hajiaghayi for his useful comments.

References

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

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

[BvLTT97] Hans L. Bodlaender, Jan van Leeuwen, Richard Tan, and Dimitrios M. Thilikos. On interval routing schemes and treewidth. *Inform. and Comput.*, 139(1):92–109, 1997.

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

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

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

[Gan01] Yashar Ganjali. Characterization of networks supporting multi-dimensional linear interval routing schemes. *Technical Report CS-2001-03, Department of Computer Science, University of Waterloo, Waterloo, Canada*, 2001.

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

[INM91] INMOS. The T9000 Transputer overview manual. 1991.

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

[SK85] Nicola Santoro and Ramez Khatib. Routing without routing tables. *Technical Report SCS-TR-6 School of Computer Science, Carleton University, Ottawa*, 1985.

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

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

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

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

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

Figure 6: Algorithm for labeling the vertices of a given graph G.