The Complexity of the Characterization of Networks Supporting Shortest-Path Interval Routing

T. Eilam S. Moran * S. Zaks
Department of Computer Science
The Technion,
Haifa 32000, Israel
email: {eilam,moran,zaks}@cs.technion.ac.il

Abstract

Interval Routing is a routing method that was proposed in order to reduce the size of the routing tables by using intervals and was extensively studied and implemented. Some variants of the original method were also defined and studied.

The question of characterizing networks which support *optimal* (i.e., shortest path) Interval Routing was thoroughly investigated for each of the variants and under different models. Only partial answers, both positive and negative, were given.

In this paper we study the characterization problem under the most basic model - the one unit cost, and with the most restrictive memory requirements - one interval per edge. We prove that this problem is NP-hard (even for the restricted class of graphs of diameter at most 3). Our results hold for all variants of Interval Routing. They significantly extend the related NP-hardness results of [FGS95], and imply that, unless P=NP, the results of [BvLT91, NS96, ZTRP93], which gave partial characterizations for some classes of networks which support shortest path Interval Routing, cannot be pushed further to lead to efficient characterizations for these classes.

*The research of this author was supported by the fund for promoting research in the Technion, and by the Bernard Elkin Chair in Computer Science.

1 Introduction

In a distributed network, where processors communicate by sending and receiving messages, a *routing scheme* is employed in order to determine the path that a message will traverse from its source to its destination. A classical routing method is to keep in each node a table with n entries (where n is the number of nodes in the network); the jth entry in the table determines the link on which to send a message destined to Node j. This routing method is *optimal* in the sense that it is always possible to construct routing tables that will guarantee a delivery of every message on a shortest length path. However, it has a prominent drawback; the space required in each node is proportional to the size of the network, which makes it very costly for large graphs.

During the last decade, strategies to reduce the amount of space required for routing have been thoroughly investigated. It was shown that there exists a trade-off between the length of the routing paths and the memory which is required for the routing (see, e.g., [PU89, AP90]). An *optimal* routing scheme is a routing method that ensures that every message is delivered from its source to its destination on a shortest length path. Many researchers focused on the problem of designing space-efficient optimal routing schemes. A common approach was to label the nodes and links of the network in a way that encodes some of the information on the network topology, and then to route messages according to the labeling.

One of the most popular such methods is Interval Routing, which was introduced in [SK85, vLT86] and was implemented in the C104 Router Chip of the INMOS T9000 Transputer design [Inm91]. (See [Gav97b] for a surveys on Interval Routing.) The basic idea of Interval Routing is to reduce the amount of memory needed in a node by encoding sets of destinations using intervals. An interval is a consecutive sequence of nodes which might wrap-around over the end of the name segment to its beginning. Under Interval Routing, every node is assigned a unique number from $\{0, ..., n-1\}$, and each link in each direction is labeled by a set of intervals. The routing of a message is performed according to the labeling; a message is sent on the link labeled by an interval which contains its destination. Some variants of the original method were proposed by posing more constraints on the interval labels. In Linear Interval Routing wrap-around intervals are not allowed. In strict Interval Routing (and strict Linear Interval Routing) an interval on an edge outgoing a node must not include the label of that node.

Though Interval Routing can be implemented on any network, for some networks, in order to achieve optimality, we might have to use $\Theta(n)$ intervals on some of the edges ([Gav97a]). The *compactness* of an interval routing scheme is the maximum size of a set of intervals which labels an edge. Interval routing schemes with compactness k are termed k-interval routing schemes (in short, k-IRS). The abbreviation k-LIRS, k-SIRS, and k-SLIRS, are used for the other variants (linear, strict and strict linear, respectively). Fundamental questions are to characterize for a constant integer k the class of graphs for which there exists an interval routing scheme with compactness k (for each of the variants), or in general, to determine the minimal compactness for which there exists an interval routing scheme for a given graph. These questions were investigated quite extensively, under various models. So far, only partial answers (both positive and negative) were given.

The class of graphs which admit optimal interval routing schemes with compactness k was termed k-$\mathcal{IRS}$ (respectively, k-$\mathcal{SIRS}$, k-$\mathcal{LIRS}$, k-$\mathcal{SLIRS}$ for the other variants). It was first shown that many familiar classes of graphs such as complete graphs, meshes, hypercubes [BvLT91], complete bipartite graphs [KKR93], and proper interval graphs [FG94] are in the class 1-$\mathcal{LIRS}$. Other graphs such as trees, tori, and unit-circular arc graphs are in the class 1-$\mathcal{IRS}$ [SK85, KKR93, FG94, NS96].

These positive results correspond to the basic and familiar model termed *one unit cost* which assumes a uniform cost on all the links. Another common model (which is a generalization of the first one) is the *fixed link cost* under which every link of the network is assigned a positive number, termed *its cost*. Costs of links reflect attributes such as propagation delay and congestion. The length of a path in this model is the sum of the costs of its links (optimality of a labeling for a network depends on these costs).

Yet another model is the *dynamic link cost* under which a labeling of the nodes of a network is optimal if it enables optimal routing for every assignment of costs to the links of the network. Under the last model few characterization results were established. [BvLT91] fully characterized graphs with dynamic link cost in the class 1-$\mathcal{LIRS}$. In [FJ88] a characterization of graphs with dynamic link cost in the class 1-$\mathcal{SIRS}$ was shown. This result was extended [BTvL94] to non-strict schemes and was generalized in [BTTvL95], where it was shown that for each $k \geq 1$, the set of connected graphs in the class k-$\mathcal{IRS}$ under the dynamic link cost model are closed under taking minors, which implies, using the classical results in [RS85], that there are linear time algorithms recognizing these sets, though there might be no constructive way to design these algorithms. Furthermore, the model of dynamic link cost is considered too restrictive. A network admits an optimal labeling only if it does so regardless of the costs of its links; therefore most of the graphs that admit an optimal labeling under the one unit cost or fixed link cost models do not fall into that category when dynamic link costs are assumed.

Under the one unit cost model, [NS96] and [GG95] gave the following partial characterization of graphs in the class k-$\mathcal{IRS}$: a necessary and sufficient condition for a network to belong to k-$\mathcal{IRS}$ is that each of its biconnected components belongs to this class. This result reduces the characterization question for interval routing to the class of biconnected graphs. In the same spirit, [NS96] gave a characterization of the graphs which in the class 1-$\mathcal{SLIRS}$ in terms of its biconnected components. These characterization results hold also under the more general fixed link cost model. However, none of these results imply a polynomial time algorithm which recognizes the corresponding graphs. Last, in an attempt to characterize one-unit cost graphs in the class 1-$\mathcal{LIRS}$ some conditions were given in [BvLT91]. [ZTRP93] conjectured that the class 1-$\mathcal{SLIRS}$ is constructible in the sense that it can be well described using simple graph operators and gave some properties of graphs in this class.

On the negative side, it was shown ([FGS95]) that the optimization problem of determining the minimal k such that a given network with fixed link costs belongs to k-$\mathcal{IRS}$ (resp., k-$\mathcal{LIRS}$, etc...), is NP-hard. They also showed that the related problem of determining the minimal integer K such that there is an optimal interval labeling which uses a total of at most K intervals on all the edges is NP-hard. Recently, it was proved in [Fla97] that the problem of characterizing the class of graphs 2-$\mathcal{IRS}$ (and respectively, 2-$\mathcal{LIRS}$, 2-$\mathcal{SIRS}$ and 2-$\mathcal{SLIRS}$) is NP-Hard under the one unit cost model (and hence under the fixed link cost model). In that paper the problem of characterizing the class 1-$\mathcal{IRS}$ (and its variants) is mentioned as the main open problem in this area.

In this paper we study the characterization question under the most basic model - the one unit cost, and with the most restrictive memory requirements - one interval per edge - and settle the above problem. Specifically, we prove that the characterization of graphs which admit optimal (strict or non-strict) interval labeling and optimal (strict or non-strict) linear interval labeling with compactness 1 in this model is NP-hard (these results are easily generalized to the fixed-cost model. Our results significantly extend the related NP-hardness results of [FGS95], and imply that, unless P=NP, the results of [BvLT91, NS96, ZTRP93] cannot be further pushed to lead to efficient characterizations of these classes of graphs. It is worth noting that, at least for the (strict or non-strict) 1-$\mathcal{LIRS}$ with unit-cost case, our result could come as a surprise, since it was shown in [BvLT91] that this class of graphs is very poor; for example, among other restrictions, a graph in this class may not contain a sub-graph of minimum paths which is a cycle of length greater than four.

Model / Problem	One unit cost	Fixed link cost	Dynamic link cost
$G \in 1\text{-}\mathcal{IRS}$?	NPC this paper	NPC this paper	P [BTvL94]
$G \in 1\text{-}\mathcal{LIRS}$?	NPC this paper	NPC this paper	P [BvLT91]
$G \in 2\text{-}\mathcal{IRS}$?	NPC [Fla97]	NPC [Fla97]	P [BTTvL95]
$G \in 2\text{-}\mathcal{LIRS}$?	NPC [Fla97]	NPC [Fla97]	P [BTTvL95]
$G \in k\text{-}\mathcal{IRS}$? for any fixed $k > 2$	?	?	P [BTTvL95]
$G \in k\text{-}\mathcal{LIRS}$? for any fixed $k > 2$	?	?	P [BTTvL95]
Minimal k s.t. $G \in k\text{-}\mathcal{IRS}$	NPH this paper, [Fla97]	NPH [FGS95]	?
Minimal k s.t. $G \in k\text{-}\ \mathcal{LIRS}$	NPH this paper, [Fla97]	NPH [FGS95]	?

Figure 1: Complexity results under the various models

The complexity results under the various models are summarized in Figure 1, where every entry refers to both the strict and non-strict versions of the problem. Following is a brief explanation of it. Entry "Minimal k s.t. $G \in k\text{-}\mathcal{IRS}$" (and "minimal k s.t. $G \in k$ - $\mathcal{LIRS}$") refers to the optimization problem of determining the minimal k for which a given graph admits an optimal labeling with compactness k. The entry "$G \in k\text{-}\mathcal{IRS}$? for any fixed $k > 2$" (and "$G \in k\text{-}\mathcal{LIRS}$? for ...") actually refers to a family of problems - the kth problem is to recognize the graphs in the class $k\text{-}\mathcal{IRS}$. To the best of our knowledge, these problems are open for all $k > 2$. In the table NPH denotes an NP-hard optimization problem and NPC denotes an NP-complete problem.

The rest of the paper is organized as follows. In Section 2 we give a precise definition of the model and the characterization problems. In Section 3 we prove that two problems which we define in the sequel, the Acyclic Graph Orientation problem and the Constrained Order problem are NP-complete. We then use these results in Section 4 where we prove the NP-completeness of both the 1-$\mathcal{LIRS}$ and the 1-$\mathcal{SLIRS}$ problems under the one unit cost model (and hence under the fixed link cost model) and in Section 5 where we prove the same result for the 1-$\mathcal{IRS}$ and 1-$\mathcal{SIRS}$ problems. Implications of our results and directions for further research are discussed in Section 6.

2 Preliminaries, Definitions and Notations

2.1 The Model

We consider two different models of a network; the *one unit cost* and the *fixed link cost.* Under both, the network is modeled as an undirected finite graph $G = (V, E)$, $|V| = n$, where the set of vertices represent the nodes of the network and the set of edges represent the bidirectional links. Graphs are connected, loopless and do not contain multi-edges. Under the fixed link cost model, every edge of the graph is associated with a positive number, which is its *cost*; under the one-unit cost model we assume that the cost of every edge is one unit. The length of the path in the graph under both models is the sum of costs of its edges (for the one unit cost model this is equal to the number of edges in it). The basic idea of Interval Routing is to label each vertex of the graph by a unique number from the set $N = \{0, ..., n-1\}$, and then at each vertex v to label each adjacent edge by a set of intervals over the set N, such that intervals assigned to edges emanating v are mutually disjoint and their union covers the set N. The routing is performed according to the labeling; in a vertex v, a message will be sent on the (unique) edge whose label contains an interval which contains the destination of the message. The *compactness* of an interval routing scheme is the maximum size of a set of intervals that labels an edge. In this paper we concentrate only on interval routing schemes with compactness 1 (i.e., every edge in each direction is labeled with 1 interval). Formally we define an *interval* over N as follows.

Definition 1 *An* interval of N *is one of the following:*

1. *A* linear interval $\langle p, q\rangle = \{p, p+1, \cdots, q\}$, *where* $p, q \in N$ *and* $p \leq q$.
2. *A* wrap-around interval $\langle p, q\rangle = \{p, p+1, \cdots, n-1, 0, \cdots, q\}$, *where* $p, q \in N$ *and* $p > q$.
3. *The* null interval $\langle\rangle$ *is the empty set.*

We say that a vertex $u \in V$ is *contained* in an interval $\langle p, q\rangle$ if $u \in \langle p, q\rangle$. An interval labeling scheme with compactness 1 is termed 1-interval labeling scheme and is formally defined as follows.

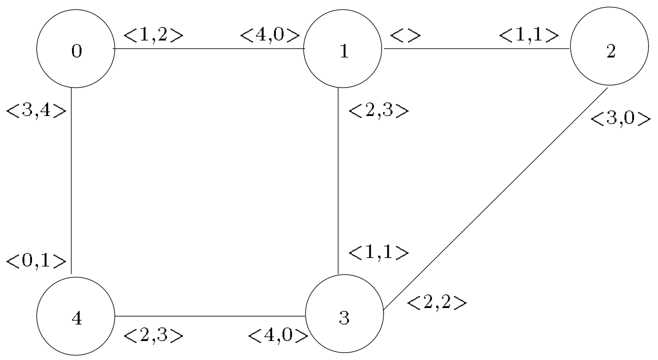
Definition 2 *A* 1-interval labeling scheme *(in short, 1-ILS),* $\mathcal{L}_G = (L, \{I_v\}_{v\in V})$, *of a graph* $G = (V, E)$, *is defined by:*

1. *A one-to-one function* $L : V \to N$ *that labels the vertices of* V.
2. *for every vertex* v, *an edge labeling function* $I_v : E_v \to I$, *where* E_v *is the set of edges adjacent to* v *and* I *is the set of intervals of* N, *that satisfies the following properties for every* $v \in V$:

   union property $N \setminus \{L(v)\} \subseteq \bigcup_{e\in E_v} I_v(e)$.
   disjunction property *For any two distinct edges* e_1, e_2 *in* E_v, $I_v(e_1) \bigcap I_v(e_2) = \emptyset$.

   (In other words, the non-empty intervals associated with all the edges outgoing any vertex v *form a partition of* N *or of* $N - \{L(v)\}$.*)*

For simplicity, in the sequel we will not distinguish between a vertex v and its label $L(v)$.



Example 1 *Figure 2.1 shows a graph together with a* 1*-ILS for it. A message with source* 0 *and destination* 2 *will traverse the path* $0-1-3-2$.

Few variants of Interval Routing are *Linear Interval Routing* in which wrap-around intervals are not allowed, and *strict* Interval Routing (respectively, *strict* Linear Interval Routing) in which an interval on an edge outgoing a vertex may not contain the label of this vertex. Their definitions follows.

Definition 3 *A* 1-linear-interval labeling scheme *(in short,* 1*- LILS), of a graph* $G=(V,E)$*, is a* 1*-interval labeling scheme in which for every* $v \in V$ *and for every* $e \in E_v$*, the interval* $I_v(e)$ *is either linear or null.*

Definition 4 *A* 1 strict interval labeling scheme *(in short,* 1*-SIRS) (respectively,* 1*-SLIRS) , of a graph* $G=(V,E)$*, is a* 1*-interval labeling scheme (respectively,* 1*-linear interval labeling scheme) in which for every* $v \in V$ *and for every* $e \in E_v$*,* $L(v) \notin I_v(e)$*.*

By *a labeling scheme* we mean 1-ILS or any of its variants. Note that the path a message with source u and destination v traverses is completely determined by the labeling scheme of the graph. If the labeling is arbitrary then this path could contain a cycle; in other words, the message can cycle forever without reaching its destination. A *valid* labeling scheme is a labeling that guarantees that every message will eventually arrive at its destination. A valid labeling scheme induces, for every ordered pair of vertices (u,v) in the graph, a simple path from u to v. A Valid 1-ILS is termed 1-IRS (Interval Routing Scheme) and respectively, valid 1-LILS, valid 1-SILS and valid 1-SLILS are termed 1-LIRS, 1-SIRS and 1-SLIRS. An *optimal* labeling scheme is a labeling scheme that guarantees that for every ordered pair of vertices, the induced path is a shortest length path in the graph (that is, the labeling guarantees that every message will traverse a shortest length path).

We define four classes of graphs; 1-$\mathcal{IRS}$ contains all graphs for which there exists an optimal 1-IRS, and 1-$\mathcal{LIRS}$ contains all the graphs for which there exists an optimal 1-LIRS. (Clearly, 1-$\mathcal{LIRS} \subseteq$ 1-$\mathcal{IRS}$.) Correspondingly, 1-$\mathcal{SIRS}$ and 1-$\mathcal{SLIRS}$ are the classes of graphs which admit optimal 1-SIRS and optimal 1-SLIRS. (Clearly, 1-$\mathcal{SIRS} \subseteq$ 1-$\mathcal{IRS}$ and 1-$\mathcal{SLIRS} \subseteq$ 1-$\mathcal{LIRS}$.)

The problems of determining, for a given graph, whether it belongs to either of these classes, are formally defined as follows.

The 1-$\mathcal{LIRS}$ problem
Input: A graph G.
Question: does $G \in$ 1-$\mathcal{LIRS}$?

and,

The 1-$\mathcal{IRS}$ problem
Input: A graph G.
Question: does $G \in$ 1-$\mathcal{IRS}$?

The problems 1-$\mathcal{SIRS}$ *and* 1-$\mathcal{SLIRS}$ are defined similarly. Also, if the model assumed is the fixed link cost then the input to each of the above four problems is a graph $G = (V, E)$ and a cost function $cost : E \rightarrow R^+$, which assigns a positive number to every edge in the graph. Clearly, all of these problems are in the class NP; given a graph (with or without a cost function) and any labeling for it, it can be verified in polynomial time whether the labeling is optimal for the graph.

In this paper we prove the following two theorems.

Theorem 1 *The 1-$\mathcal{LIRS}$ and the 1-$\mathcal{SLIRS}$ problems are NP-complete under the one unit cost model (and hence under the fixed link cost model).*

Theorem 2 *The 1-$\mathcal{IRS}$ and the 1-$\mathcal{SIRS}$ problems are NP-complete under the one unit cost model (and hence under the fixed link cost model).*

We will prove Theorem 1 and Theorem 2 by a sequence of polynomial transformations from the 3-SAT problem. In the next subsection we will define two problems that will be used in the proofs.

2.2 The Acyclic Graph Orientation and the Constrained Order Problems

Informally, the input to the Acyclic Graph Orientation problem is a directed graph and a partition of its edges to equivalence classes. The problem is to determine whether there exists a subset of the equivalence classes such that switching the directions of the edges in them will result in a directed acyclic graph (DAG). Formally,

The Acyclic Graph Orientation (AGO) Problem
Input: A directed graph $G = (V, E)$ and a partition of its edges to equivalence classes $\{E_i\}_{0 \leq i \leq t}$, (the sets E_i are pairwise disjoint and $E = \bigcup_{0 \leq i \leq t} E_i$).
Question: Is there a subset of indices $J \subseteq \{0, ..., t\}$ such that the directed graph $G_J = (V, E_J)$, which is obtained by reversing the directions of all the edges in $\bigcup_{j \in J} E_j$, is a DAG?

In the Constrained Order problem, the question is to determine whether there exists a partial order $<_o$ over a given set of items U that satisfies a given set of *order constraints* (in short *constraints*).[1] A constraint is an ordered pair of sets of items of the form $(Set_1 \mid Set_2)$. Such a constraint is satisfied by a partial order $<_o$ iff either $Set_1 <_o Set_2$ or $Set_2 <_o Set_1$ ($A <_o B$ means that $u <_o v$ for every $u \in A$ and every $v \in B$). Formally,

The Constrained Order (CO) Problem
Input: A set of constraints $Cons = \{Cons_i\}_{1 \leq i \leq t}$, over a set of items U, where the constraint

[1] In this paper, a partial order is a transitive, asymmetric relation.

$cons_i$ is of the form $Cons_i = (Set_i^1 \mid Set_i^2)$, $Set_i^1, Set_i^2 \subseteq U$, $Set_i^1, Set_i^2 \neq \emptyset$, and $Set_i^1 \cap Set_i^2 = \emptyset$, for every $i = 1, ..., t$.
Question: Is there a partial order $<_o$ over the set U such that for every constraint $Cons_i$, $i = 1, ..., t$, either $Set_i^1 <_o Set_i^2$ or $Set_i^2 <_o Set_i^1$.
Note that both the AGO problem and the CO problem are in NP. In the course of proving Theorem 1 and Theorem 2 we prove that the AGO problem and the CO problem are NP-complete. Specifically, the proof of Theorem 1 is by a sequence of polynomial transformations. The first transformation is from the well-known NP-complete 3-SAT problem (see, e.g., [GJ79]) to the AGO problem. The second transformation is from the AGO problem to the CO problem. The last transformation is from the CO problem to the 1-$\mathcal{LIRS}$ problem. In fact, the last transformation will imply that both 1-$\mathcal{LIRS}$ and 1-$\mathcal{SLIRS}$ are NP-complete under both the fixed cost model and the one unit cost model.

In order to prove Theorem 2, we show a polynomial transformation from the CO problem. Again, the same transformation will imply the NP-completeness of both variations and under both models. The structure of the proofs of Theorem 1 and Theorem 2 is illustrated in Figure 2.

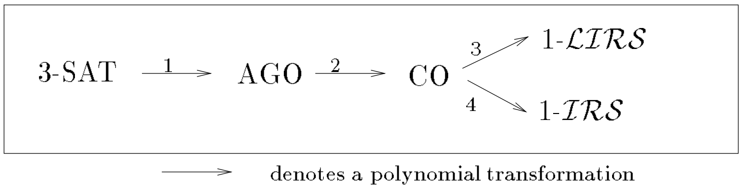


Figure 2: The structure of the proofs.

3 The NP-Completeness of the AGO and the CO Problems

We prove that the AGO problem is NP-complete by a polynomial transformation from the 3-SAT problem. We then prove the NP-completeness of the CO problem by a polynomial transformation from the AGO problem.

3.1 A Transformation from 3-SAT to AGO

An instance of the 3-SAT problem is a CNF formula φ over a set of variables $Var = \{v_1, \cdots, v_t\}$, that is, $\varphi = \bigwedge_{i=1,\cdots,m} C_i$ where $C_i = (l_i^1 \vee l_i^2 \vee l_i^3)$ is a clause of three literals, the literal l_i^j is either a variable v_k or a negation of a variable $\neg v_k$ for some $v_k \in Var$. The question is whether there is a truth assignment to the variables in Var which satisfies φ. This instance is transformed to an instance of the AGO problem, which consists of a directed graph $G = (V, E)$ and a partition of E to equivalence classes $\{E_j\}_{j \in \{0,\cdots,t\}}$ as follows.

The graph $G = (V, E)$ consists of m components $H_1, \cdots, H_m$, where H_i corresponds to clause $C_i = (l_i^1 \vee l_i^2 \vee l_i^3)$. H_i contains four directed edges e_i, e_i^1, e_i^2, e_i^3, whose underlying graph is a cycle; The orientations of the edges are determined as follows: the edge e_i, $i = 1, \cdots, m$, is a *compass* edge which is always oriented counterclockwise. An edge $e_i^j (1 \leq j \leq 3)$ is oriented clockwise if the literal l_i^j is a variable v_k and it is oriented counterclockwise if l_i^j is a negation of a variable $\neg v_k$.

The partition of the edges to equivalence classes is as follows. $E_0 = \{e_i\}_{i=1,\cdots,m}$ consists of all the compass edges. In addition, there is a class E_k for each variable $v_k \in Var$, which consists of all edges e_i^j, where l_i^j is either the variable v_k or its negation $\neg v_k$.

Following is an example of the transformation.

Example 2 *Figure 3 (a) depicts an example of the transformation from a CNF formula φ to a directed graph and a set of equivalence classes on its set of edges. The vertices of H_i, $i = 1, 2$, are a_i, b_i, c_i and d_i; The compass edge e_i is the edge (a_i, d_i). The edge e_i^1 (which correspond to the literal l_i^1) connects the vertices a_i and b_i, similarly the edges e_i^2 and e_i^3 connect the vertices b_i, c_i and c_i, d_i respectively. The edge e_1^1, for example, is oriented clockwise (i.e., from a_1 to b_1) since the corresponding literal l_1^1 is a variable (v_1). Similarly, e_1^2 is oriented counterclockwise since the corresponding literal l_1^2 is a negation of a variable ($\neg v_2$). All other edges are oriented by similar rules. The numbers on the edges $(0, 1, 2, 3)$ denote the equivalence class E_i, $(i = 0, 1, 2, 3)$ to which the edge belongs. For example, the literal l_1^1 is v_1 and the literal l_2^2 is $\neg v_1$ and therefore $E_1 = \{(a_1, b_1), (c_2, b_2)\}$.*

$$\varphi = (v_1 \vee \neg v_2 \vee v_3) \wedge (\neg v_2 \vee \neg v_1 \vee \neg v_3)$$

$$\begin{aligned} V &= \{a_1, b_1, c_1, d_1, a_2, b_2, c_2, d_2\} \\ E_0 &= \{(a_1, d_1), (a_2, d_2)\} \\ E_1 &= \{(a_1, b_1), (c_2, b_2)\} \\ E_2 &= \{(c_1, b_1), (b_2, a_2)\} \\ E_3 &= \{(c_1, d_1), (d_2, c_2)\} \end{aligned}$$

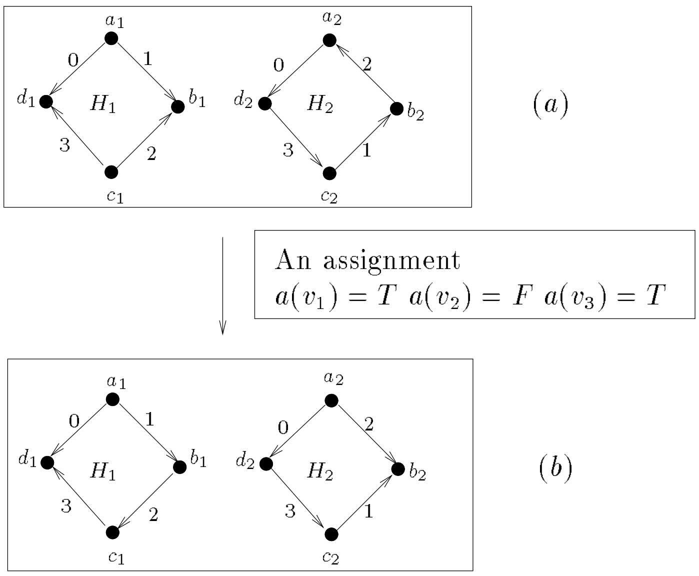


Figure 3: An example of the transformation. (a) The directed graph G and the equivalence classes of its edges, constructed by the transformation from the formula φ. (b) The legal orientation of G's edges which corresponds to the truth-assignment a.

In this transformation, we interpret a clockwise orientation of an edge e_i^j as a truth assignment a under which the truth value $\overline{a}(l_i^j)$ of the literal l_i^j is T and a counterclockwise orientation of this edge as an assignment a such that $\overline{a}(l_i^j) = F$. Thus, the initial orientation of the edges in the graph G corresponds to an assignment of the value T to all the variables (therefore, a literal which is a variable is oriented clockwise and a literal which is a negation of a variable is oriented counterclockwise). Denote an orientation which is obtained by reversing all the edges in a set of equivalence classes $\{E_j\}_{j\in J}$ as *legal* orientation. Then the definition of the equivalence classes E_j

guarantees that in each legal orientation, all the edges which correspond to a variable v_k have the same orientation, which is the opposite orientation of the edges which correspond to $\neg v_k$. The counterclockwise orientation of the compass edges e_i guarantees that in the initial graph G a cycle H_i is a directed cycle iff clause C_i contains only negated variables (thus, this clause is not satisfied by the initial assignment). For a set $J \subseteq \{1, \cdots, t\}$, the graph G_J defined by reversing the edges in the classes $\{E_j\}_{j \in J}$ corresponds to a truth assignment which assigns the value F to each variable in the set $\{v_j\}_{j \in J}$, and the value T to all other variables.

Example 3 *Figure 3(b) is the graph G_J where $J = \{2\}$. It corresponds to the assignment of the value T to v_1 and v_3 and the value F to v_2. Note that φ is satisfied under this assignment and indeed the corresponding graph G_J is a DAG.*

We now prove the correctness of the transformation.

Proposition 1 *A CNF formula $\varphi = \bigwedge_{i=1,\cdots,m} C_i$ over $Var = \{v_1, \cdots, v_t\}$ has a satisfying truth assignment iff there exists a subset $J \subseteq \{0, ..., t\}$ such that the directed graph $G_J = (V, E_J)$ is a DAG.*

Proof: Assume first that there is a truth-assignment $a : Var \rightarrow \{F, T\}$ which satisfies φ. We take J to be the set of indices $J = \{k | \ a(v_k) = F\}$. By the discussion above, an edge e_i^j in G_J is oriented clockwise iff $\overline{a}(l_i^j) = T$. By the assumption that a satisfies φ, each component H_i in G_J has at least one edge which is oriented clockwise. Since the compass edge of H_i is oriented counterclockwise in G_J, we have that G_J contains no directed cycle, hence it is acyclic, as required.

Assume now that there is a set of indices J such that the graph $G_J = (V, E_J)$, is acyclic. We can assume without loss of generality that $0 \notin J$ (otherwise, replace J by $\bar{J} = \{0, 1, \cdots, t\} \setminus J$, noting that $G_{\bar{J}}$ is obtained by reversing all the edges in G_J, and hence it is also acyclic). Thus, for each component H_i in G_J, the compass edge e_i is directed counterclockwise. Now, consider the truth assignment a that assigns F to all variables $\{v_j\}_{j \in J}$ (and T to the rest of the variables). By the definition of G_J, $\bar{a}(l_i^j) = T$ only if in the graph G_J, the edge e_i^j is directed clockwise. Since the graph G_J is acyclic, each component H_i contains an edge which is oriented clockwise. This edge corresponds to a literal in clause C_i with value T under the assignment a, hence all clauses are satisfied by a, as claimed. ∎

3.2 A Transformation from AGO to CO

Let a directed graph $G = (V, E)$ and a partition of its edges to equivalence classes $\{E_j\}_{j=0,\cdots,t}$ be an instance to the AGO problem, where $V = \{v_1, ..., v_n\}$. We transform it to a set of constraints S over a set of items U as follows.
The set U is given by $U = \{u_1, ..., u_n\} \cup \{l_j, r_j | 0 \leq j \leq t\}$. The set of constraints is defined as follows. For every directed edge $(x, y) \in E$ there exists a unique j, $0 \leq j \leq t$ such that the edge (x, y) belongs to the equivalence class E_j. We add the constraint $(\{l_j, x\} \mid \{r_j, y\})$ to the set of constraints (i.e., the total number of constraints is $\sum_{j=0}^{t} |E_j| = |E|$). The transformation is demonstrated by the following example.

Example 4 *Let the instance of the AGO be as constructed in Example 2 (Figure 3(a)). The transformation will construct from it the following set of constraints over the set of items $U = \{a_1, b_1, c_1, d_1, a_2, b_2, c_2, d_2, l_0, r_0, l_1, r_1, l_2, r_2, l_3, r_3\}$.*

$$
\begin{array}{lcccr}
(& \{l_0, a_1\} & | & \{r_0, d_1\} &) \\
(& \{l_0, a_2\} & | & \{r_0, d_2\} &) \\
(& \{l_1, a_1\} & | & \{r_1, b_1\} &) \\
(& \{l_1, c_2\} & | & \{r_1, b_2\} &) \\
(& \{l_2, c_1\} & | & \{r_2, b_1\} &) \\
(& \{l_2, b_2\} & | & \{r_2, a_2\} &) \\
(& \{l_3, c_1\} & | & \{r_3, d_1\} &) \\
(& \{l_3, d_2\} & | & \{r_3, c_2\} &)
\end{array}
$$

Proposition 2 *There exists a subset $J \subseteq \{0, ..., t\}$ such that the directed graph $G_J = (V, E_J)$ is a DAG iff there exists a partial order of the items in U which satisfies all the constraints in S.*

Proof: Assume that there is a partial order $<_o$ that satisfies all the constraints in S. We claim that if we orient the edges in E such that an edge is oriented from v_i to v_j if $u_i <_o u_j$, we obtain the desired acyclic orientation. Consider the set of constraints which corresponds to the class E_j. If $l_j <_o r_j$ then $x <_o y$ for each edge $(x, y) \in E_j$, hence these edges preserve their original orientation; otherwise ($r_j <_o l_j$), all the edges in E_j are reversed. This means that the resulted graph is G_J, where $J = \{j | r_j <_o l_j\}$. Since $<_o$ is a partial order, G_J cannot contain a cycle, as required.

Conversely, given a set of indices $J \subseteq \{0, ..., t\}$ such that G_J is acyclic, we define the partial order $<_o$ on the items of U to be the transitive closure of the following relation R: For each $j \notin J$, and for each $(x, y) \in E_j$, $x\ R\ y, l_j\ R\ y, x\ R\ r_j$, and in addition $l_j\ R\ r_j$ For each $j \in J$, and for each $(x, y) \in E_j$, $y\ R\ x, y\ R\ l_j, r_j\ R\ x$, and in addition $r_j\ R\ l_j$. Clearly, the relation R satisfies all the constraints in S. Moreover, since the graph G_J is acyclic, it is easily seen that the transitive closure of the relation R is a partial order on the items of U. ■

4 The NP-Completeness of the 1-$\mathcal{LIRS}$ Problem

In this section we complete the proof that the 1-$\mathcal{LIRS}$ and the 1-$\mathcal{SLIRS}$ problems are NP-complete (Theorem 1), by a polynomial transformation form the CO problem.

4.1 Polynomial Transformation from the Constrained Order Problem

For a set of constraints $S = \{Cons_1, Cons_2, ..., Cons_t\}$ over a set of items $U = \{u_1, u_2, ..., u_n\}$, where $Cons_i = (Set_i^1 | Set_i^2)$, $i = 1, \cdots, t$, we construct the graph $G_S = (V_S, E_S)$ as follows. For each constraint $Cons_i \in S$ we will have three vertices in V, a_i, b_i, and m_i which are termed the *ith triple*. Additionally, to every item $u_i \in U$ we will have in G a vertex v_i. Formally,
$V_S = \{m_i, a_i, b_i | 1 \leq i \leq t\} \cup \{v_j | 1 \leq j \leq n\}$.
The vertices $\{m_i\}_{i=1,\cdots,t}$ are termed *high-level* vertices, the vertices $\{a_i, b_i\}_{i=1\cdots,t}$ are termed *intermediate* vertices, and the vertices $\{v_j\}_{j=1,\cdots,n}$ are termed *low-level* vertices.

The set of edges E_S is defined so that the following holds. The subgraph induced by the intermediate vertices $\{a_i, b_i\}_{i=1,\cdots,t}$ and the subgraph induced by the low-level vertices $\{v_j\}_{j=1,\cdots,n}$ are both complete graphs. In addition, each high level vertex m_i is connected only to the two other vertices in the ith triple, a_i and b_i; a_i is also connected to all low-level vertices except those which correspond to the set Set_i^2, and b_i is connected to all low-level vertices except those which correspond to the set Set_i^1. Formally, $E = \cup_{i=1}^4 E_i$, where
$E_1 = \{\{m_i, a_i\}, \{m_i, b_i\} | 1 \leq i \leq t\}$,

$E_2 = \{\{a_i, b_j\} | 1 \leq i, j \leq t\} \cup \{\{a_i, a_j\}, \{b_i, b_j\} | 1 \leq i, j \leq t, i \neq j\}$,
$E_3 = \{\{a_i, v_j\} | u_j \notin Set_i^2\} \cup \{\{b_i, v_k\} | u_k \notin Set_i^1\}$, and
$E_4 = \{\{v_i, v_j\} | 1 \leq i, j \leq n, i \neq j\}$.
The transformation is demonstrated by the following example.

Example 5 *Figure 4 is an example of a set of constraints S over a set of items $U = \{u_1, u_2, \cdots, u_6\}$ and the corresponding graph G_S. For simplicity the complete graph induced by the set of vertices $\{a_i\} \cup \{b_i\}$ and the complete graph induced by the set $\{v_j\}$ are depicted only implicitly.*

A set of constraints S

$Cons_1 : \ \{u_1, u_2, u_3\} \mid \{u_4, u_5\}$
$Cons_2 : \ \{u_1, u_5, u_6\} \mid \{u_2, u_3\}$

The graph G_S

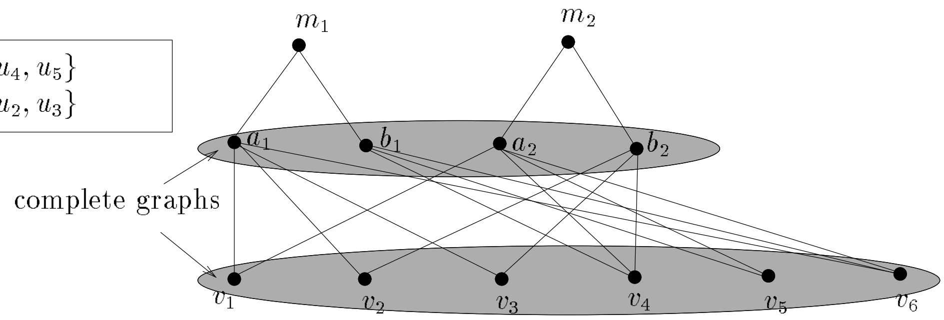


Figure 4: A set of constraints S over $U = \{u_1, ..., u_6\}$ and the corresponding graph G_S

Informally, the transformation is based on the following idea. For each constraint $Cons_i = (Set_i^1 | Set_i^2)$, all the shortest paths from m_i to vertices that correspond to the items in Set_i^1 start with the edge $\{m_i, a_i\}$, and all the shortest paths from m_i to vertices that corresponds to the items in Set_i^2 starts with the edge $\{m_i, b_i\}$. It follows that for each $i = 1, \cdots, t$, the order on the low-level vertices $\{v_j\}$ induced by an optimal 1-LIRS for the graph must satisfy the constraint $Cons_i$. Therefore, if there is an optimal 1-LIRS for the graph, there is an order of the items in S that satisfies all the constraints in the set S. Conversely, we will show that, given a partial order $<_o$ which satisfies all the constraints in S, we can construct an optimal 1-LIRS for the graph. It is worth to note that it is not hard to prove the first direction using a simpler construction; connect every vertex a_i to all vertices which correspond to items in Set_i^1 and every vertex b_i to all vertices that correspond to items in Set_i^2, however, we do not know how to construct an optimal 1-LIRS for this graph.

4.2 Correctness of the Transformation

First note that the transformation is polynomial. Given a set of constraints S over a set of items U, the number of vertices in the graph G_S is exactly $|V_S| = n + 3t$, where n is the number of items in the set U and t is the number of constraints in the set S. We prove the following proposition.

Proposition 3 *There exists a partial order which satisfies all the constraints in S iff $G_S \in 1\text{-}\mathcal{LIRS}$ (that is, there exists an optimal 1-LIRS for G_S).*

The rest of this section is the proof of Proposition 3.

We first prove that if $G_S \in$1-$\mathcal{LIRS}$ then there exists a partial order that satisfies all the constraints in S. Let $\mathcal{L}_{G_S} = (L, \{I_v\})$ be an optimal 1-LIRS for the graph G_S. Then the vertex labeling L induces a natural order on the items in U, namely $u_i <_o u_j$ iff $L(v_i) < L(v_j)$. We argue that $<_o$ is a partial (in fact - total) order on U which satisfies all the constraints in S.

Consider the constraint $Cons_i = (Set_i^1 | Set_i^2)$. By the construction of the graph G_S, every shortest path from m_i to any vertex v_j that corresponds to an item u_j in Set_i^1 starts with the edge $\{m_i, a_i\}$ (in fact, $\langle m_i, a_i, v_j \rangle$ is the only shortest path from m_i to v_j). It follows that $\{v_j | u_j \in Set_i^1\} \subseteq I_{m_i}(m_i, a_i)$. By the same arguments, $\{v_j | u_j \in Set_i^2\} \subseteq I_{m_i}(m_i, b_i)$. Thus, there are two disjoint intervals, one of which contains all vertices that correspond to items in Set_i^1 and the other contains all vertices that correspond to the set Set_i^2. Since the order of the low-level vertices is equivalent to the order of the corresponding items in U, either $Set_i^1 <_o Set_i^2$ or $Set_i^2 <_o Set_i^1$; in either case, the constraints $Cons_i$ is satisfied by the order $<_o$.

The proof of the other direction is more intricate. Given a partial order $<_o$ which satisfies the constraints in S we will construct an optimal 1-LIRS $\mathcal{L}_{G_S}$ for G_S.

The labeling of the vertices

First we extend the partial order $<_o$ on the items in U which satisfies all the constraints in S to a total order $<_{to}$ in an arbitrary manner. We then construct a vertex-labeling function $L : V_S \rightarrow \{0, ..., |V_S| - 1\}$ that satisfies the following conditions.

1. For every two low-level vertices v_i and v_j: $L(v_i) < L(v_j)$ iff $u_i <_{to} u_j$. That is, the order of the low-level vertices is equivalent to the order $<_{to}$ on the items in the set U. Having this in mind, we will identify the set of items Set_i^j with the set of low-level vertices $\{v_j | u_j \in Set_i^j\}$.

2. For every ith triple $\{m_i, a_i, b_i\}$, $1 \leq i \leq t$, there are two cases to consider.

   (a) $Set_i^1 <_{to} Set_i^2$. Then $L(Set_i^1) < L(a_i) < L(m_i) < L(b_i) < L(Set_i^2)$ (where $L(A) = \{L(v) | v \in A\}$ for a set of vertices A). (See Figure 5.)

   (b) $Set_i^2 <_{to} Set_i^1$. Then $L(Set_i^2) < L(b_i) < L(m_i) < L(a_i) < L(Set_i^1)$.

   Note that by the assumption that the order $<_{to}$ satisfies the constraints in S, exactly one of the two cases (a) and (b) holds.

3. For $i = 1, \cdots, t$, the vertices of the ith triple (m_i, a_i, b_i) are ordered consecutively. Formally, $\{L(m_i), L(a_i), L(b_i)\} = \{k_i, k_i + 1, k_i + 2\}$ for some integer k_i.

Note that clearly a labeling that satisfies all three conditions exists: first arrange all low-level vertices by the order $<_{to}$, and then insert every ith triple anywhere between Set_i^1 and Set_i^2.

An observation which we use later in the proof is that for the ith triple $\{m_i, a_i, b_i\}$, if case 2(a) holds then a_i is adjacent (in G_S) to all low-level vertices that are smaller than it (and possibly to some low-level vertices that are larger than it) and b_i is adjacent to all low-level vertices which are larger than it. Case 2(b) implies the opposite observation.

The Labeling of the Edges

Now we show how to label the edges so as to obtain an optimal 1-LIRS (actually, 1-SLIRS). In our proof we use the observation that in order to show optimality of a labeling scheme, it suffices to show that for each pair of vertices u, v, the (unique) edge e (e adjacent to u) for which $L(v) \in I_u(e)$, lies on a shortest path from u to v.

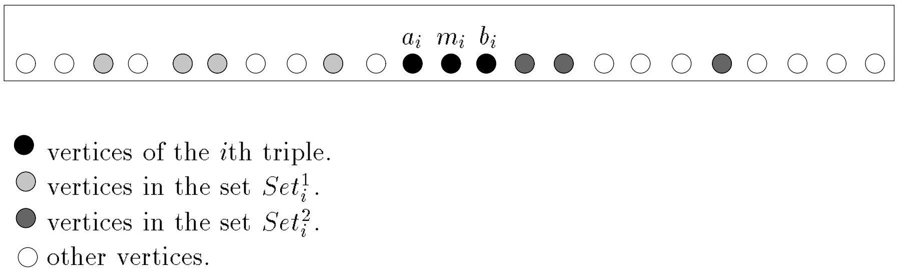


Figure 5: The order of the vertices in the case: $Set_i^1 < Set_i^2$.

For brevity, we will not distinguish between a vertex v and the number assigned to it $L(v)$. We will denote by min the vertex with minimal label and by max the vertex with maximal label.

1. **labeling of the edges of a high-level vertex.** Consider any vertex m_i $(1 \le i \le t)$. m_i is at distance 3 from any other high-level vertex m_j, at distance 1 from a_i and b_i, and at distance 2 from all other vertices. Moreover, the vertex labeling implies that the edge (m_i, a_i) [resp., (m_i, b_i)] lies on a shortest path from m_i to each of the other vertices, except b_i [resp., a_i] and except the low-level vertices in Set_i^2 [resp., Set_i^1]. Thus, if $a_i < m_i < b_i$, (case 2(a)), then the labeling $I_{m_i}(m_i, a_i) = \langle min, a_i \rangle$ and $I_{m_i}(m_i, b_i) = \langle b_i, max \rangle$ guarantees optimal routing from m_i. In the other case $(b_i < m_i < a_i)$, the labeling $I_{m_i}(m_i, a_i) = \langle a_i, max \rangle$ and $I_{m_i}(m_i, b_i) = \langle min, b_i \rangle$ guarantees optimal routing from m_i.

2. **labeling of the edges of a low-level vertex.** Consider any vertex v_j, $1 \le j \le n$. Then v_j is adjacent to all other low-level vertices; also, for any ith triple $\{m_i, a_i, b_i\}$ $(1 \le i \le t)$, v_j is adjacent to one or two vertices in $\{a_i, b_i\}$ (since it cannot be both in Set_i^1 and Set_i^2). Moreover, an edge from v_j to any adjacent vertex in the ith triple is on a shortest path from v_j to any non-adjacent vertex in this triple.
   Therefore, optimal routing for any other low-level vertex v_k, is guaranteed by the labeling $I_{v_j}(v_j, v_k) = \langle v_k, v_k \rangle$. Optimal routing to vertices in any ith triple $\{m_i, a_i, b_i\}$ $(1 \le i \le t)$ is guaranteed by the following labeling: Assume w.l.o.g. that $a_i < m_i < b_i$ (the other case is symmetric). If v_j is adjacent to both a_i and b_i then $I_{v_j}(v_j, a_i) = \langle a_i, m_i \rangle$ and $I_{v_j}(v_j, b_i) = \langle b_i, b_i \rangle$, otherwise, if v_j is connected only to a_i then $I_{v_j}(v_j, a_i) = \langle a_i, b_i \rangle$, otherwise, v_j is connected only to b_i and $I_{v_j}(v_j, b_i) = \langle a_i, b_i \rangle$.

3. **labeling of the edges of an intermediate vertex.** Consider any intermediate vertex a_i (a vertex b_i is treated similarly). a_i is adjacent to m_i, to all other intermediate vertices, and to all low-level vertices except those in Set_i^2, and it is at distance 2 from all other vertices. A shortest path from a_i to a non-adjacent high level vertex m_j passes through either a_j or b_j, and the shortest paths to a non-adjacent low-level vertex v_j passes through a low-level vertex or an intermediate-level vertex which is adjacent to both a_i and v_j. Using these facts, the labeling is constructed as follows:
   First, $I_{a_i}(a_i, m_i) = \langle m_i, m_i \rangle$. For a vertex a_j $(j \ne i)$ the interval $I_{a_i}(a_i, a_j)$ will include a_j and m_j and for a vertex b_j (possibly $j = i$) the interval $I_{a_i}(a_i, b_j)$ will include the vertex b_j. This guarantees optimal labeling for all high-level and intermediate vertices. Now we extend this labeling so as to guarantee optimal routing from a_i to all low-level vertices. Assume w.l.o.g that $a_i < m_i < b_i$ (the other case is treated similarly). We have to construct a labeling which

will send messages destined to a neighbor v_j of a_i directly to v_j, and messages destined to a vertex $v_k \in Set_i^2$ to a neighbor of v_k. Thus, we first insert vertex v_j to $I_{a_i}(a_i, v_j)$ for every neighbor v_j of a_i. Next we extend the labeling obtained so far to include also the vertices of Set_i^2 as follows. Denote as a *block* a maximal set of vertices in Set_i^2 which are ordered consecutively by L. Consider such a block B, and assume that its vertices are mapped by L on the integers $l, l+1, \cdots, l+|B|-1$. Since B does not contain all the vertices of G, there is a vertex $x \notin Set_i^2$ which is mapped on either $l-1$ or on $l+|B|$ (i.e., x appears immediately before or immediately after the vertices in B). If x appears immediately before B, then $x = v_k$ for some low-level vertex v_k, or x is an intermediate-level vertex (i.e., a_j or b_j) which is adjacent to all the low-level vertices which are larger than it. Similarly, if x appears immediately after B, then $x = v_k$ for some low-level vertex v_k, or x is an intermediate-level vertex (i.e., a_j or b_j) which is adjacent to all the low-level vertices which are smaller than it. In both cases, x is adjacent to both a_i and to all the vertices in B. In both cases, we set $I_{a_i}(a_i, x)$ to include the segment containing $B \cup \{x\}$. This procedure is repeated for every block B.

This completes the proof of Proposition 3.

Note that the above constructed labeling is strict. Thus, if the set of constraints S has the desired partial order, then the graph G_S has an optimal strict linear interval routing. Also, since the graph constructed admits no non-uniform costs on the links, we proved the NP-completeness of the 1-$\mathcal{LIRS}$ problem and the 1-$\mathcal{SLIRS}$ problem under the one unit cost model and hence under the fixed link cost model (Theorem 1).

5 The NP-completeness of the 1-$\mathcal{IRS}$ Problem

In this section we prove that the 1-$\mathcal{IRS}$ and the 1-$\mathcal{SIRS}$ problems are NP-complete under the one unit cost model (and hence under the fixed link cost model) by a polynomial transformation from the CO problem.

5.1 Polynomial Transformation from the CO Problem

The transformation is accomplished in two steps; given any set of constraints S we first construct from it a new set of constraints S', we then use exactly the same transformation we used in Section 4 to construct from S' the graph $G_{S'}$. We prove that the original set of constraints S is satisfiable if the graph $G_{S'}$ has an optimal interval routing scheme and that $G_{S'}$ has an optimal strict interval routing scheme if S is satisfiable.

Step 1:

Let U be a set of items and $S = \{Cons_i\}_{1 \le i \le t}$ a set of constraints over U such that $Cons_i = (Set_i^1 | Set_i^2)$ $(i = 1, ..., t)$. We construct a set of constraints S' over the set $U' = U \cup \{a\}$ $(a \notin U)$ as follows. For every constraint $Cons_i \in S$ there are two constraints in S';

$Cons_i^l = (\ Set_i^1 \cup \{a\} \mid Set_i^2\ )$

$Cons_i^r = (\ Set_i^1 \mid Set_i^2 \cup \{a\}\ )$

The set of constraints S' is:

$S' = \{Cons_i^l, Cons_i^r | 1 \le i \le t\}$.

Example 6 *The following set of constraints is constructed from the set of constraints presented in Example 5.*

$$\begin{array}{llcll}
(& \{u_1, u_2, u_3, a\} & | & \{u_4, u_5\} &) \\
(& \{u_1, u_2, u_3\} & | & \{u_4, u_5, a\} &) \\
(& \{u_1, u_5, u_6, a\} & | & \{u_2, u_3\} &) \\
(& \{u_1, u_5, u_6\} & | & \{u_2, u_3, a\} &)
\end{array}$$

Step 2:
Now we construct from U' and S' the graph $G_{S'} = (V_{S'}, E_{S'})$ by the transformation of Section 4.

5.2 Correctness of the Transformation

It is easily seen that the transformation is polynomial. We prove the following proposition.

Proposition 4 *There is a partial order on the items in U that satisfies the constraints in the set S iff $G_{S'} \in$ 1-$\mathcal{IRS}$.*

Proof: We use the following well-known (see, e.g., [vLT87]) observation. Given any 1-IRS $\mathcal{L}_G$ for a graph G with n vertices, there are $n-1$ equivalent 1-IRSs that are obtained from $\mathcal{L}_G$ by cyclically shifting the labels of the vertices and intervals. That is, the ith 1-interval labeling is obtained by assigning every vertex v the label $L(v) + i(mod\ n)$ and replacing the intervals on the edges correspondingly. The paths that a message will traverse under any of the n equivalent 1-IRSs are identical. Note that this observation enables us to assume, given a 1-IRS for the graph G that a specific vertex, say a, is the vertex with maximal label ($L(a) = n-1$).

Assume that there is an optimal 1-IRS $\mathcal{L}_G$ for $G_{S'}$. Then (by the observation above) we can assume w.l.o.g. that the vertex v_a which corresponds to the item a in U' is the vertex with maximal label. We denote by $\{m_i^l, a_i^l, b_i^l\}$ the triple of vertices which correspond to the constraint $Cons_i^l$ in S' and by $\{m_i^r, a_i^r, b_i^r\}$ the triple which corresponds to the constraint $Cons_i^r$. By the same considerations of the proof in Section 4, we get $Set_i^1 \cup \{v_a\} \subseteq I_{m_i^l}(m_i^l, a_i^l)$, $Set_i^2 \subseteq I_{m_i^l}(m_i^l, b_i^l)$, $Set_i^1 \subseteq I_{m_i^r}(m_i^r, a_i^r)$ and $Set_i^2 \cup \{v_a\} \subseteq I_{m_i^r}(m_i^r, b_i^r)$. We argue that the order of the vertices determined by the 1-IRS $\mathcal{L}_G$ satisfies either $Set_i^1 < Set_i^2$ or $Set_i^2 < Set_i^1$ ($i = 1, ..., t$). Assume that this is not the case. Then either there are three low-level vertices $v_x, v_y \in Set_i^1$ and $v_z \in Set_i^2$ such that $L(v_x) < l(v_z) < l(v_y)$, or a similar inequality holds where $v_x, v_y \in Set_i^2$ and $v_z \in Set_i^1$. Assume w.l.o.g that the first case holds. Then by the assumption that v_a is the vertex with maximal label we get $L(v_x) < L(v_z) < L(v_y) < L(v_a)$ and clearly this implies that the intervals $I_{m_i^r}(m_i^r, a_i^r)$ and $I_{m_i^r}(m_i^r, b_i^r)$ on the edges outgoing the vertex m_i^r are not disjoint. The other case will imply, by the same arguments, that the intervals on the edges outgoing m_i^l are not disjoint. It follows that the order of the low-level vertices by the labeling $\mathcal{L}_G$ naturally induces a total order on the items of the set U that satisfies all the constraints in S.

In the other direction we have to construct, given a partial order $<_o$ that satisfies all the constraints in the set U, an optimal 1-IRS for the graph $G_{S'}$. The construction proceeds as follows: First, the vertex a and all its adjacent edges are ignored, and the construction is done as in Section 4 (note that when a is ignored, the graph $G_{S'}$ is similar to the graph G_S constructed in Section 4, except that now each constraint is replaced by two identical constraints, which are represented by two triples, $\{m_i^l, a_i^l, b_i^l\}$ and $\{m_i^r, a_i^r, b_i^r\}$). We set $L(a) = max + 1$ where max is the maximal label of all other vertices. Next we assign the labels $I_a(a,x)$ to the edges leaving a by the same rules as the labeling I_v for any other low-level vertex v. Then, for each vertex x which is adjacent to a, $I_x(x,a)$ is set to $\langle a,a\rangle$. Last, we modify the edge labeling of high-level and intermediate-level vertices which are not adjacent to a. For each edge leaving a high-level vertex m_i which was labeled by $\langle *, max\rangle$ we replace max by $max+1 = L(a)$. Each intermediate-level vertex x ($x \in \{a_i^l, a_i^r, b_i^l, b_i^r\}$) which is not adjacent to a (there is exactly one such vertex in each triple) must be adjacent to a vertex y such that $L(y) = min$ or $L(y) = max$; we add $L(a) = max+1$ to the interval $I_x(x,y)$ (if $L(a) = min$ then this transforms $I_x(x,y)$ to a cyclic interval). It is easy to see that the resulting labeling is optimal and uses one interval per edge. ■

As in Section 4, here also we actually construct an optimal 1-SIRS for the graph G, thus proving that both the 1-$\mathcal{IRS}$ and the 1-$\mathcal{SIRS}$ are NP-complete under the one unit cost model and hence under the fixed link cost model. This completes the proof of Theorem 2.

6 Summary

We proved that recognizing networks that admit either optimal 1-IRS or optimal 1-LIRS (strict or non-strict), is NP-hard under the one-unit cost model (and hence under the fixed link cost model). These results clearly imply that the problems of determining the minimal k such that a network G belongs to k-$\mathcal{IRS}$ or to k-$\mathcal{LIRS}$ (entries "Minimal k s.t. $G \in k$-$\mathcal{IRS}$" and "Minimal k s.t. $G \in k$-$\mathcal{LIRS}$", respectively, in Figure 1) are NP hard. They also imply that determining the minimal integer K, for which a network G has an optimal labeling (IRS or LIRS) which uses a total of at most K intervals is NP hard: Under the one unit cost model, a simple graph $G = (V,E)$ admits an optimal 1-LIRS (respectively, 1-IRS) iff it admits an optimal LIRS (respectively, IRS) which uses a total of at most $2|E|$ intervals.

There are still many open questions. Our transformations from the AGO problem to the 1-$\mathcal{LIRS}$ problem and the 1-$\mathcal{IRS}$ problem imply that the same problems remain NP-complete even for the restricted class of graphs with diameter at most 3. Do these problems remain NP-complete for other restricted classes of graphs, e.g., bounded degree graphs? In a different direction, recently it was proved ([Fla97]) that the same characterization problems are NP-hard also for $k = 2$; it remains an open question to determine the complexity of the problems k-$\mathcal{IRS}$ and k-$\mathcal{LIRS}$ for fixed k, $k > 2$. It seems that our technique does not directly imply that any of these problems is NP-hard. Another question is to find an approximation algorithm to the optimization problem of finding the minimal k such that a graph belongs to k-$\mathcal{IRS}$ (resp., k-$\mathcal{LIRS}$ etc...). Our results imply that there is no such algorithm with approximation ratio less than 2. The existence of such algorithm with constant ratio is another interesting open question.

A different direction of research, which is motivated by our results, is to study networks that admit near optimal interval labeling schemes, that is, networks which admit a labeling scheme (of a given type) that guarantees a small stretch factor (where *a stretch factor* of a routing scheme is the maximum ratio between the length of a shortest path and the length of a routing path between two endpoints). Recently, it was shown in [EGP98] that every graph admits an interval routing scheme with stretch factor at most 5 (and an avarage stretch factor at most 3) and with compactness $2\sqrt{n(1+\ln n)}$. It was also shown in [EGP98] that there are some graphs for which the compactness of any interval routing scheme that guarantees any constant strech factor is at least $\sqrt{n}$. It remains an open question to recognize networks that acheive a better trade-off.

References

[AP90] B. Awerbuch and D. Peleg. Sparse partitions. In *Proc. 31st IEEE Symp. on Foundations of Computer Science*, pages 514–522, 1990.

[BTTvL95] H. Bodlaender, R. Tan, D. Thilikos, and J. van Leeuwen. On interval routing schemes and treewidth. In *Workshop on Graphs Algorithms*, 1995.

[BTvL94] E. M. Bakker, R. B. Tan, and J. van Leeuwen. Manuscript, 1994.

[BvLT91] E.M. Bakker, J. val Leeuwen, and R. B. Tan. Linear interval routing. *Algorithms Review*, 2:45–61, 1991.

[EGP98] T. Eilam, C. Gavoille, and D. Peleg. Compact routing schemes with low stretch factor. In ACM PRESS, editor, *17th Annual ACM Symp. on Principles of Distributed Computing (PODC)*, pages 11–20, 1998. Appeared also as: Research Report RR-1195-98, LaBRI, University of Bordeaux, 351, cours de la Libération, 33405 Talence Cedex, France, January 1998.

[FG94] P. Fraigniaud and C. Gavoille. Optimal interval routing. In *Parallel Processing, Vol. 854 of Springer-Verlag Lecture Notes in Computer Science*, pages 785–796, 1994.

[FGS95] M. Flammini, G. Gambosi, and S. Salomone. Interval routing schemes. In *Proceedings of STACS*, pages 279–290, 1995.

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

[Fla97] M. Flammini. On the hardness of devising interval routing schemes. To appear in PPL, 1997.

[Gav97a] Cyril Gavoille. On the dilation of interval routing. In 22^{nd} *International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 1300 of Lecture Notes in Computer Science, pages 259–268. Springer-Verlag, August 1997.

[Gav97b] Cyril Gavoille. A survey on interval routing scheme. Research Report RR-1182-97, LaBRI, University of Bordeaux, 351, cours de la Libération, 33405 Talence Cedex, France, October 1997. To Appear in Theoretical Computer Science.

[GG95] C. Gavoille and E. Guévremont. Worst case bounds for shortest path interval routing. Technical Report 95-02, LIP, January 1995.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability.* Freeman, San Francisco, 1979.

[Inm91] Inmos. The t9000 transputer products overview manual, 1991.

[KKR93] E. Kranakis, D. Krizanc, and S. Ravi. On multi-label linear interval routing schemes. In *Workshop on Graph-Theoretic Concepts in Computer Science (WG) Vol. 790 of Springer-Verlag Lecture Notes in Computer Science*, pages 338–349, 1993.

[NS96] L. Narayanan and S. Shende. Characterization of networks supporting shortest path interval routing schemes. In *Proceedings of 3rd International Colloquium on Structural Information and Communication Complexity*, 1996. also as: Technical Report, Concordia University, 1995.

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

[RS85] N. Robertson and P. D. Seymour. Graphs minors - a survey. In I. Anderson, editor, *Surveys in Combinatorics*, pages 153–171. Cambridge University Press, 1985.

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

[vLT86] J. van Leeuwen and R. B. Tan. Routing with compact routing tables. In G. Rozemberg and A. Salomaa, editors, *The book of L.* Springer Verlag, 1986.

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

[ZTRP93] B. Zerrouk, S. Tricot, B. Rottembourg, and L.M. Patnaik. Proper linear interval routing schemes. Technical report, Institut Blaise Pascal, Univ. Pierre et Marie Curie, Paris, 1993.