On the Inverse Shortest Paths Problem

Pål Nilsson
Department of Communication Systems, Lund University, Sweden

paln@telecom.lth.se

Abstract

In this paper we survey a variant of the inverse shortest paths problem. Given is a network in which certain node-pairs require connections over specific predefined single paths. The aim is to find link weights such that the desired paths are uniquely induced by a shortest-path algorithm. Further we require that the link weights fit existing shortest-path based routing protocols. We give a motivation for this problem, discuss a number of its solvability aspects, and show how a feasible solution can be obtained from its linear relaxation. A network operator who wants to obtain a specific routing pattern by adjusting link weights, will find practical use of the solution to the considered problem.

1 Introduction

Currently, the vast majority of research in communication network design is devoted to networks carrying IP traffic. In many cases it is assumed that routing patterns can be chosen almost arbitrarily, to optimize some given traffic handling objectives. Although such an assumption may be advantageous for the resulting network performance, it is realistic only in the presence of certain advanced routing techniques, such as using end-to-end MPLS tunnels. Even if the idea of MPLS tunneling is attractive from several aspects, it will result in dropping the beneficial property of statistical multiplexing, as well as simplicity of the routing mechanism. Regardless, numerous methods and algorithms addressing network optimality emerge, practically all of which are relying on systematic assignment of routes and bandwidth to communication demands. Clearly, a great deal of such research efforts are exclusively destined for the framework of MPLS. A significant share of these approaches are hardly realizable by the current IP network routing mechanisms, such as OSPF, IS-IS etc., which are based on single shortest-path routing techniques, computed from a set of administrative network link weights. However, as these techniques are very simple from administrative point of view, and also since their proposed stiffness may be relaxed by interaction with lower

layers, their importance remains. Uncertainty of the penetration of MPLS and similar routing techniques, actually makes it highly relevant to ask which types of routing patterns can be accomplished by simpler, easily implemented and commonly used routing techniques. These are the type of problems that we address in this paper. We assume a given network, in which a set of pairs of nodes require communication. Each such pair of nodes we call a demand and assume that it is given with an associated, single connection path. It is reasonable to assume that these paths are set by the operator, who is in turn relying on some network optimization tool. Specifically, we address the question for which such instances it is possible to assign weights to links such that the predefined paths are induced by application of a simple shortest-path algorithm, and if realizable, what is the complexity of the required computation of weights. If the weights are allowed to be fractional the problem is solved in polynomial time as a linear programme. However, it is the case that existing protocols allow only for strictly positive weights, restricted to integrality [1]. Due to representation, these integer weights are furthermore most likely bounded in magnitude. Thus in reality, the problem of finding appropriate weights can be substantially more difficult.

Example 1: Consider the simple network given in Figure 1. This instance has two demanded connections, between nodes 1-3 and nodes 2-3, with associated required paths (dashed lines). Undoubtedly, the required routing pattern is obtainable using MPLS tunneling. However, this is not the case if a classical routing protocol, based on calculation of shortest paths, is employed. The reason for this is that there is no assignment of weights to links such that a shortest-path algorithm will compute the required unique shortest paths. Generally, a routing pattern intersecting two paths at two nodes, must intersect entirely between those nodes to be compatible with some set of link weights.

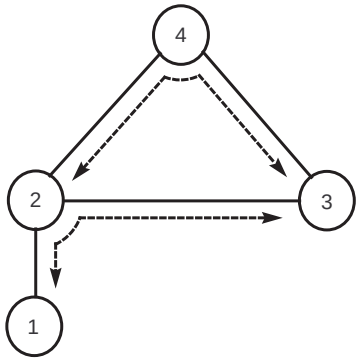


Figure 1: Routing pattern not inducable by link weights.

One might argue that the routing pattern of Example 1 may indeed be obtainable if the more sophisticated versions of OSPF are used. Certainly, this is not the case even though OSPF featuring the equal cost multi-path (ECMP) principle admits splitting of

flows on equal-length (in terms of link weights) shortest paths. It should be realized that the splitted flow shares in this case must be equal. Note that the instance of Example 1 admits no splitting of flows and that the parallel flows could differ in magnitude. Moreover, if ECMP is used, resequencing of a splitted flow of packets must be performed at some unifying point. It is furthermore considered a disadvantage that a flow which exploits several paths increases the size of the routing table. In addition to this it has been shown that determining an optimal weight system accounting for ECMP is NP-hard [2]. To the bottomline, ECMP is a method of circumventing the ambiguity implied by several shortest paths of the same length. We will attack this inherited problem at an earlier stage, enforcing for each demand a *unique shortest path*.

The primary interest of this study is the problem that we refer to as WIPS (Weights Inducing a Path System), assuming desired predefined paths. A somewhat less realistic but closely related problem is encountered if demands are given solely as node pairs, i.e., paths are not predefined but selected (WIPS-PS).

The (ongoing) studies reported in this paper are mainly addressing the same problems as those of Ben-Ameur and Gourdin in [3]. A study with identical purpose and considerations, but with a different setting was presented in [4]. Also, research considering assignment of weights to links, but with a different objective, was conducted in [5]. The general single-path routing problem, regardless of objective, is known to be difficult in terms of computational complexity [6].

2 Problem Formulation

We start by formulating the problem of WIPS which consists in, given a set of demands and associated paths in a specified network, how do we assign link weights such that (i) the weights fit the routing protocol, and (ii) the predefined paths are uniquely determined by a shortest-path algorithm. Usually, routing protocols restrict link weights to be strictly positive integers. Moreover, there is an upper bound on how large a link weight may be. Therefore, (i) is considered satisfied if link weights are positive integers and if the maximal weight is chosen as small as possible (note that for certain instances, even though we cannot do better, this might be insufficient).

Consider a network represented by a graph, $G = (V, E)$, where V is the set of vertices (nodes) and E is the set of edges (links). Let D be a set of demands with each demand identifying an s-d vertex pair, (u, v), $u, v \in V$, $u \neq v$, between which there is a connection to be established. For each demand $d \in D$, let $P(d)$ be the corresponding set of possible simple paths. Each path $p \in P(d)$, is a sequence of adjacent edges connecting the s-d pair of demand d. The weight of link e, $e \in E$ is denoted by w_e. We may now express WIPS as an integer programme;

$$\begin{aligned}
\min \quad & t && && (1)\\
\text{subject to} \quad & w_e \leq t && e \in E && (2)\\
& \sum_{e \in q} w_e + 1 - s_{dp} \leq \sum_{e \in p} w_e && p \in P(d),\ d \in D && (3)\\
& w_e \in \mathbb{Z}^+ && e \in E\ , && (4)
\end{aligned}$$

where $q \in P(d)$ is the desired path for demand d, and $s_{dp} = 1$ if $p = q$, and $s_{dp} = 0$ otherwise. The objective function (1), and the constraint (2) assure that the largest weight is minimal. They may therefore be replaced by the objective

$$\min_{e \in E} ||w_e||_\infty,$$

using the l_∞-norm. Furthermore, this integer programme states that link weights, w_e, $e \in E$, are such that there is just one shortest path for each demand (3), and that the weights are integral and strictly positive (4). For brevity, in the sequel we will denote by $\delta(p; \boldsymbol{w})$, the length of path p with respect to link weights $\boldsymbol{w} = [w_e]_{e \in E}$, and by $\lambda(p)$ the length of path p in terms of hops.

3 Feasibility

Adopting the terminology of [3], we say that a set of link weights is compatible with a set of predefined paths if the shortest paths with respect to the link weights are equal to the predefined paths. Clearly, there is no guarantee that a specific instance of the integer programme (1)-(4) (nor its linear relaxation) is feasible. There is for instance a risk of that the predefined paths violate the suboptimality condition illustrated by Example 1. Formally, this necessary condition reads as follows:

Condition 1 *A set of link weights is compatible with a set of predefined paths only if any two predefined paths that intersect at two different nodes follow the same (sub-)path between these nodes.*

The necessity of this condition is obvious; there cannot be two shortest paths between two nodes. The condition is furthermore easily checked — for each predefined path, verify that the condition holds for every intersection with another path. Unfortunately, it is not a sufficent condition, as at least two more necessary conditions, none of which are implicitly satisfied by Condition 1, can be listed [3]. However, both these additional conditions, based upon the notion of cyclic compatibility, are harder to check. The most straightforward way of checking feasibility of an instance of (1)-(4) is probably to check feasibility of its linear relaxation, i.e., to find any element of the set defined by (3)-(4) with (4) replaced by

$$w_e \in \mathbb{Q}^*, \ e \in E \ , \tag{5}$$

where $\mathbb{Q}^* = \{x : x \geq 1, x \in \mathbb{Q}\}$. Certainly, since $\mathbb{Z}^+ \subset \mathbb{Q}^*$, the linear relaxation gives a necessary condition – if no rational weights exist, there cannot exist integral weights. Now let $\boldsymbol{w} = [w_e]_{e \in E}$ be a vector of link weights, such that $w_e \in \mathbb{Q}^*$ for all $e \in E$. Let $\pi(D, \boldsymbol{w})$ be the set of (single) shortest paths corresponding to the demands of set D, with respect to the link weights, $\boldsymbol{w}$. It is obvious that for any $r \in \mathbb{Q}^+$, $\pi(D, \boldsymbol{w}) = \pi(D, r \cdot \boldsymbol{w})$, i.e., multiplying link weights by a strictly positive constant preserves the length ordering of paths. It is a simple algebraic property that if there exists any vector of weights (whose elements are strictly greater than 1) satisfying (3), then there exists a vector of weights, $\boldsymbol{w}$, such that $w_e \in \mathbb{Q}^*$ for all $e \in E$, also satisfying (3). Furthermore, indeed there exists a smallest $r \in \mathbb{Q}^+$ such that $r \cdot w_e \in \mathbb{Z}^+$, for all $e \in E$. Thus we can conclude that the integer programme of (1)-(4) is feasible if and

only if its linear relaxation is feasible. This property may be checked in polynomial time by an Interior-Point Method (IPM) solver. However, in the following we will neglect the rare exponential running times and assume that the feasibility check is based on solving the linear relaxation by means of Simplex.

4 The linear relaxation

Substituting (4) by (5) gives a linear relaxation of WIPS. To solve this Linear Programme (LP) by means of application of an LP-solver, forces us to stress (3) for each demand, d, and for each existing non-desired path $p \in P(d)$. For a reasonable-sized instance, this will result in an LP with a large number of constraints. Questions that arise are how all these paths are generated, and if every possible path for each demand really need to be included. Certainly, there will be paths that are not needed to be included in $P(d)$, as their lengths in terms of hops are very large. Intuitively, a demand d for which the desired path q, has a large corresponding $\lambda(q)$, will require the corresponding set $P(d)$ be large. The most appealing way of solving the linear relaxation seems to be to solve a sequence of LPs, interleaved with appropriate shortest-path calculations. One variant is as follows:

initialization: Assign $w_e := 1$, for all $e \in E$. Let $\phi : \mathbb{Z}^+ \cup \{0\} \longrightarrow \mathbb{Z}^+$ be some appropriate increasing function. For each demand $d \in D$ solve the shortest path problem. Let $p_s(d)$ denote the shortest path and $q(d)$ the desired path. Define $z_d = \lambda(q(d)) - \lambda(p_s(d))$, and let $k_d = \phi(z_d)$. Solve a k_d shortest paths problem and let the obtained paths constitute the set $P(d)$ for demand $d \in D$.

step 1: Solve the linear relaxation of WIPS for the current sets of paths to obtain new link weights $\boldsymbol{w}$.

step 2: For each demand $d \in D$ solve a 2 shortest paths problem for the current $\boldsymbol{w}$. Any of the two obtained paths, say path p, that satisfies $\delta(p; \boldsymbol{w}) \leq \delta(q(d); \boldsymbol{w})$ is added to $P(d)$. If for each demand $d \in D$, no extensions of $P(d)$ are performed, the current $\boldsymbol{w}$ is optimal. Otherwise go to step 1.

Provided that the feasibility check turns out positive, i.e, that the solution space defined by the linear relaxation is non-empty, we will implicitly obtain an integral solution. As will be shown, this vector of integer-valued link weights is guaranteed to possess no other properties than feasibility and integrality. Consider the linear relaxation of (1)-(4). Denote by M the number of constraints in this formulation. The number of variables is $|E| + 1$. Now we want to write this linear programme in the standard linear programming form. Define a decision vector $\boldsymbol{x}$ of length $|E| + M + 1$, such that each of the $|E|$ first elements are defined as $x_e = w_e - 1$, $e \in E$, and $x_{|E|+1} = t$. The rest of the M elements will be the slack variables. The linear relaxation can be written as

$$\begin{aligned} \min \quad & x_{|E|+1} && (6) \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} && (7) \\ & \boldsymbol{x} \geq \boldsymbol{0}\ , && (8) \end{aligned}$$

where (7) expresses the constraints of (2) and (3), and (8) corresponds to (5). This formulation is in the standard form, and if Simplex is applied, the solution will be of the form

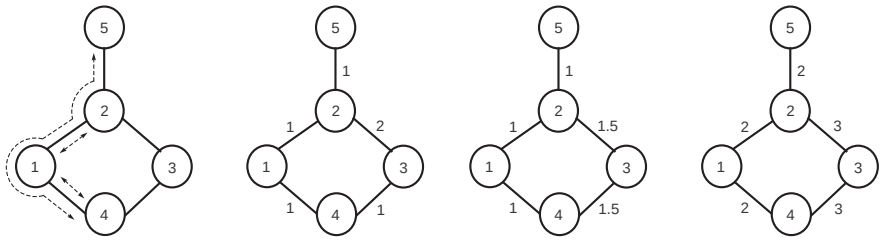
$$\boldsymbol{x}_B = \boldsymbol{B}^{-1}\boldsymbol{b} \ ,$$

where $\boldsymbol{x}_B$ is a basic solution of (6)-(8), and $\boldsymbol{B}$ is a $M \times M$ (basis) submatrix of $\boldsymbol{A}$. Now if we let $\boldsymbol{x}_{|E|}$ denote the subvector obtained from the $|E|$ first entries of $\boldsymbol{x}_B$, we have that our fractional weight vector, $\boldsymbol{w}_q$, is obtained as $\boldsymbol{w}_q = \boldsymbol{x}_{|E|} + \boldsymbol{e}$, where $\boldsymbol{e}$ is a vector of ones of length $|E|$. Usage of Cramer's rule along with the fact that the entries of $\boldsymbol{B}$ are integers, imply that the vector

$$\boldsymbol{w}_i = \det(\boldsymbol{B})\boldsymbol{w}_q \ ,$$

is integer-valued in each element.

Example 2: Consider the instance given in Figure 2. Demands are given between nodes 1-2, 1-4, and 4-5. Corresponding desired paths are shown in the figure. As can be seen, an integral solution carried over from a fractional one, will generally be non-optimal.



(a) A simple instance (b) An optimal solution (not unique) (c) The fractional solution (d) Multiplied fractional solution

Figure 2: Comparison of optimal integer-valued solution and multiplied fractional solution.

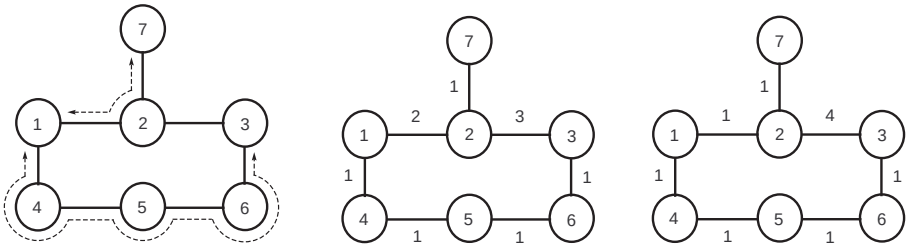
There exist of course other methods for converting a fractional solution to an integral one, not necessarily using multiplication by a constant [3].

5 Passive links

Since our purpose is to model the demand in a realistic backbone network, it is not reasonable to assume that there are demands between each pair of nodes. Indeed, in a backbone network, some of the nodes will be transit nodes, just forwarding packets. Also, it is very natural to assume that some links are left *passive*, meaning that they are not included in any of the desired paths. Thus any solution procedure for WIPS must allow for the possibility of difference between the graph representing the network and the supply graph (constituted by the desired paths). A legitimate question that arises

is if passive links can be excluded from the optimization process, and accounted for later. One idea is for instance to solve WIPS for the supply graph and then assigning each of the passive links' weights the same, sufficiently large, integer. Such a sufficiently large number should then be chosen such that (i) if a shortest-path algorithm is applied, none of the passive links are traversed and (ii) if the number is decreased by 1, at least one of the passive links is traversed, or there is an ambiguity in which path is the shortest for some demand. However, the following example shows that this type of approach will not be on target.

Example 3: Consider the instance given in Figure 3. Demands are given between nodes 1-3 and 1-7. Corresponding desired paths are shown in the figure. As can be seen, the largest weight produced by the "two-step approach", optimizing subject to the supply graph and then chosing the passive link's weight sufficiently large, will not attain the optimum.



(a) A simple instance

(b) An optimal solution (non-unique)

(c) The two-step approach solution

Figure 3: An example of that the "two-step approach" will not work

6 Computational complexity considerations

To the best of our knowledge, it is an open question whether WIPS is NP-hard or not. Related problems have indeed shown to possess this type of computational difficulties [5]. Formally, WIPS will be shown NP-hard if the following decision problem (which is actually more interesting from a practical point of view) can be proven NP-complete:

INSTANCE. Given a graph $G = (V, E)$, a set of node-pairs D, an integer K, and for each $d \in D$ a simple path $q(d)$.

QUESTION. For each link $e \in E$, is there a weight $w_e \in \mathbb{Z}^+,\ w_e \leq K$, such that for each demand, $d \in D$, the length with respect to these weights of any path $p(d),\ p(d) \neq q(d)$, connecting node-pair d, is larger than that of $q(d)$?

Clearly, this decision problem is in NP – application of any (polynomial-time) k

shortest-path algorithm, with $k = 2$, confirms the uniqueness of the induced paths. It is checked by simple inspection that link weights are bounded from above.

7 Conclusion

This paper has introduced and overviewed a variant of the inverse shortest paths problem, originally stated in [3]. The problem consists in, given a network and a set of demands with predefined single routes, how to assign integer-valued weights to each link, such that the routing pattern is uniquely obtained by a shortest-path algorithm. We have considered aspects of the problem's linear relaxation, from which a feasible integer-valued solution has been shown deducable. As the topics presented in this paper are part of an on-going study, there are several possible further work directions. Undoubtedly, important issues are the computational complexity of the problem, as well as to solve its linear relaxation in an efficient way.

References

[1] B. Fortz, J. Rexford and M. Thorup, "Traffic engineering with traditional IP routing protocols", *IEEE Communications Magazine*, 40(10), pp118-124, 2002.

[2] M. Pióro, A. Szentesi, J. Harmatos, A. Juttner, P. Gajowniczek and S. Kozdrowski, "On open shortest path first related network optimisation problems", *Performance Evaluation*, 48(1-4), pp201-223, 2002.

[3] W. Ben-Ameur and E. Gourdin, "Internet Routing and Related Topology Issues", *SIAM Journal of Discrete Mathematics*, 17(1), pp18-49, 2003.

[4] A. Farago, A. Szentesi and Szviatovszki, "Allocation of administrative weights in PNNI", in proceedings of Networks '98, pp621-625.

[5] D. Burton, B. Pulleyblank and P. Toint, "The inverse shortest path problem with upper bounds on shortest path costs", Internal Report, FUNDP, (submitted to ORSA Journal on Computing), 1993.

[6] M. Pióro and D. Medhi, "Routing, Flow and Capacity Design in Communication of Computer Networks", Morgan Kaufmann Publishers, June 2004.