Complexity of Some Inverse Shortest Path Lengths Problems

Tingting Cui
Department of Industrial Engineering and Operations Research, University of California, Berkeley, California

Dorit S. Hochbaum
Department of Industrial Engineering and Operations Research, Walter A. Haas School of Business, University of California, Berkeley, California

The input to an inverse shortest path lengths problem (ISPL) consists of a graph *G* with arc weights, and a collection of source-sink pairs with prescribed distances that do not necessarily conform to the shortest path lengths in *G*. The goal is to modify the arc weights, subject to a penalty on the deviation from the given weights, so that the shortest path lengths are equal to the prescribed values. We show that although ISPL is an NP-hard problem, several ISPL classes are polynomially solvable. These cases include ISPL where the collection of the pairs share a single source and all other nodes as destinations (the single-source all-sink problem SAISPL). For the case where the collection contains a single node pair (the single-source single-sink problem SSISPL), we identify conditions on the uniformity of the penalty functions and on the original arc weights, which make SSISPL polynomially solvable. These results cannot be strengthened significantly as the general single-source ISPL is NP-hard and the all-sink case, with more than one source, is also NP-hard. We further provide a convex programming formulation for a relaxation of ISPL in which the shortest path lengths are only required to be no less than the given values (LBISPL). It is demonstrated how this compact formulation leads to efficient algorithms for ISPL. © 2009 Wiley Periodicals, Inc. NETWORKS, Vol. 56(1), 20–29 2010

Keywords: inverse optimization; shortest path; complexity; polynomial algorithm

1. INTRODUCTION

When solving an optimization problem, it is usually assumed that problem parameters such as costs or capacities are known exactly. Frequently, however, the input parameters are often only estimates and errors in the estimation may

lead to poor decisions. In certain circumstances, additional information may be available, such as the optimal solution or optimal objective value. We would like to use this information to attain better estimates for the unknown parameters.

The idea of inverse optimization is to adjust the values of the parameters so that the observed solutions or objective values are indeed optimal and the adjusted values differ from the given estimates by as little as possible. The difference is measured in terms of a penalty function. The goal is to adjust the parameters while minimizing the total penalty. In this article, we study the inverse shortest path lengths problem (ISPL), in which the weights of the arcs in a network are adjusted so that the lengths of the shortest paths between a collection of source-sink pairs are equal to the prescribed observed values.

Burton and Toint [6] were the first to investigate the inverse version of the shortest path problem. Since then, different inverse optimization problems have been considered by various authors (see [12] for a comprehensive survey). We distinguish between two categories of inverse problems: inverse solution optimization and inverse objective value optimization. The inverse solution optimization problem is to adjust the cost vector in order to make a prescribed given solution optimal. Numerous combinatorial problems have been studied in this category including inverse shortest path [4, 6, 7, 24], inverse minimum spanning tree [2, 13, 19], and other inverse combinatorial problems [8, 9, 17, 22, 23]. The second category, inverse objective value problem, has drawn less attention than the first one. In an inverse objective value problem, a desired solution is not provided in advance but rather a desired objective value. The problem here is to find a cost vector under which the optimal objective value is equal to the prescribed value. In general, problems of the first category are comparatively easier than problems of the second category. It is shown in [3] that certain types of inverse solution problems (restricted in terms of the penalty functions) are polynomially solvable if the corresponding (noninverse) optimization problems are polynomially solvable. On the

other hand, Ahmed and Guan [1] have shown that the inverse objective value problem for a linear program is NP-hard. The ISPL belongs to the second category: for prescribed distances between a collection of source-sink pairs, it finds the minimum cost modification of the arc weights so that the shortest path lengths are equal to the prescribed values.

The work by Fekete et al. [10] is one of the few that address combinatorial inverse objective value problems. The problem they investigate is to assign arc weights (rather than modify or adjust weights) for a set of arcs in a given graph such that certain shortest path lengths are equal to the prescribed values. In this problem, there is no penalty for modifying the weight of an arc. As such, this problem is a special case of ISPL with zero modification cost for each arc. We refer to this problem as the feasible inverse shortest path lengths problem (FISPL). FISPL is shown to be NP-complete in [10]. Additionally, they suggest restricted situations in which FISPL becomes tractable: where there is only one source node (single-source FISPL), or each source node is paired up with all other nodes as sinks (all-sink FISPL).

Hung [15] also studies the FISPL problem. Hung identifies polynomially solvable cases based on the structure of the physical network. He also develops a heuristic for FISPL using an implicit formulation which, in general, has exponentially many constraints. Hung's algorithm relies on constraint generation techniques.

Burton et al. [5] study a relaxation of ISPL in which the shortest path lengths are required to be no more than certain upper bounds. This problem is referred to as the upper bounding inverse shortest path lengths problem (UBISPL). They prove that even this restricted case of ISPL is NP-hard. They also propose a heuristic to find a local optimum and prove that it is finitely convergent.

Zhang et al. [25–27] address a special case of UBISPL in which there is a single source. They show that the problem is polynomially solvable under the ℓ_∞ norm but NP-hard even to achieve an approximate solution under either the ℓ_1 or the ℓ_2 norm. They also provide a heuristic method to solve the problem approximately on a spanning tree.

The focus of this paper, like [10], is on the complexity of special cases of ISPL. We prove that ISPL is strictly harder than FISPL by demonstrating that the single-source and the all-sink problems, the two polynomially solvable cases of FISPL, are NP-hard as ISPL problems. Another contribution is in devising polynomial time algorithms for some classes of the single source ISPL problem. We also study the lower bounding inverse shortest path lengths problem (LBISPL), in which the shortest path lengths are required to be no less than certain lower bounds. We show that LBISPL is polynomial time solvable using a compact convex programming formulation. This formulation can be especially useful in designing efficient heuristics for ISPL.

The rest of this article is organized as follows. In Section 2, we provide a formal definition of the ISPL problem and introduce some notation. Section 3 discusses complexity issues where we prove that two polynomial classes for FISPL are NP-complete for ISPL. We also identify a further restricted class for which ISPL becomes polynomially solvable. In Section 4, we provide the formulation of LBISPL as a convex programming problem with a polynomial number of linear constraints. We further discuss how this formulation can be used to design efficient algorithms for ISPL. The last section is devoted to a restricted case of ISPL, in which only one source-sink pair is specified (SSISPL). Using the polynomial time solvability of LBISPL, we prove that SSISPL is polynomial time solvable if certain technical conditions hold on the parameters and the penalty function.

2. PRELIMINARIES AND AN EXAMPLE

We denote a vector by boldface letters, so $\mathbf{x}$ is a vector, and x_i is the ith component of $\mathbf{x}$. An arc is indicated by an ordered pair (i, j) and an edge by $[i, j]$. A directed path from i_1 through $i_2, \ldots, i_k$ is presented as $(i_1, i_2, \ldots, i_k)$.

An instance of the ISPL consists of two directed graphs defined on the same set of nodes V. The graph $G = (V, A)$ has a nonnegative weight vector $\mathbf{c}$ indicating the original estimates of weights for each arc. The paths between the sources and the destinations are routed in G. A second graph $G_d = (V, A_d)$ provides the pairwise distance $d_{ij} \geq 0$ as the prescribed shortest path length between the pair i, j, for every source-sink pair $i, j \in V$. We refer to the first graph as the network and the second as the distance graph of the instance. For ease of notation, the ISPL instance is written as $G(\mathbf{c}) = (V, A, \mathbf{c})$ and $G_d(\mathbf{d}) = (V, A_d, \mathbf{d})$. In addition, we let $P_{st}(\mathbf{x})$ denote a shortest path from s to t in $G(\mathbf{x})$, $D_{st}(\mathbf{x})$ denote the length of $P_{st}(\mathbf{x})$, and $D(P, \mathbf{x})$ denote the length of path P in $G(\mathbf{x})$.

The input to ISPL also includes the penalty function for the deviation of arc weights in the network. For each arc $(i, j) \in A$, there is a function $f_{ij} : \Re \mapsto \Re^+$, where $f_{ij}(x_{ij} - c_{ij})$ is the cost of modifying the cost of arc (i, j) from c_{ij} to x_{ij}. We assume that $f_{ij}()$ is a convex function which is monotonically decreasing on $(-\infty, 0]$ and monotonically increasing on $[0, \infty)$. It is commonly assumed that the penalty functions satisfy $f_{ij}(0) = 0$, for all $(i, j) \in A$, but this is not necessary for the analysis in this article to hold.

Given the network $G(\mathbf{c})$ and the distance graph $G_d(\mathbf{d})$, a weight vector $\mathbf{x} : A \mapsto \Re_+^{|A|}$ is said to be G_d-satisfying if $D_{st}(\mathbf{x}) = d_{st}$ for all arcs $(s, t) \in A_d$. The ISPL is to find a G_d-satisfying weight assignment $\mathbf{x}$ on G that minimizes the total penalty cost $F(\mathbf{x}) = \sum_{(i,j) \in A} f_{ij}(x_{ij} - c_{ij})$.

The distance graph $G_d = (V, A_d)$ is said to be a source star (sink star) from (to) s, if all arcs in A_d have s as their common tail (head). A star is either a source star or a sink star. G_d is called a complete star if $A_d = \{s\} \times (V \setminus \{s\})$ or $A_d = (V \setminus \{s\}) \times \{s\}$.

An instance of ISPL is said to be a *Single-Source ISPL* (SISPL) if G_d is a source star. Similarly, an *All-Sink ISPL* (AISPL) is an ISPL instance whose distance graph is a union of complete stars. A *Single-Source All-Sink* ISPL (SAISPL) is both an SISPL and an AISPL with a distance graph that is a complete star. A *Single-Source Single-Sink* ISPL (SSISPL) is

TABLE 1. Abbreviations of problem classes.

Abbreviation	Problem description
ISPL	Inverse shortest path lengths problem
FISPL	Feasible ISPL
SISPL	Single-source ISPL
AISPL	All-sink ISPL
SAISPL	Single-source all-sink ISPL
SSISPL	Single-source single-sink ISPL
SFISPL	Single-source FISPL
AFISPL	All-sink FISPL
LBISPL	Lower bounding ISPL
UBISPL	Upper bounding ISPL

an ISPL with a distance graph containing a single arc. Special cases of FISPL are defined analogously.

Table 1 lists the abbreviations of these problem classes.

The FISPL problem is shown to be NP-complete in [10]. It follows that ISPL is NP-hard, since it is a generalization of FISPL. The NP-completeness result of FISPL is proved through a reduction of the 3-SAT problem to a constructed instance of FISPL. We will not repeat the proof here but rather provide an example illustrating the difficulty of solving FISPL.

Consider an FISPL instance with the physical graph G as shown in Figure 1. The distance graph G_d has five arcs in $A_d = \{(a,c), (d,f), (g,i), (m,n), (p,q)\}$, with the prescribed shortest path lengths $d_{ac} = d_{df} = d_{gi} = 1$ and $d_{mn} = d_{pq} = 0$. A G_d-satisfying weight assignment $\mathbf{x}$ must satisfy $x_{ab} + x_{bc} = 1$ since (a,b,c) is the only path from a to c. Similarly, we have $x_{de} + x_{ef} = 1$ and $x_{gh} + x_{hi} = 1$. We introduce boolean variables y_1, y_2, and y_3, defined as follows:

$$y_1 = \begin{cases} F & \text{if } x_{ab} = 0, \\ T & \text{if } x_{bc} = 0, \\ \text{unspecified} & \text{otherwise.} \end{cases}$$

$$y_2 = \begin{cases} F & \text{if } x_{de} = 0, \\ T & \text{if } x_{ef} = 0, \\ \text{unspecified} & \text{otherwise.} \end{cases}$$

$$y_3 = \begin{cases} F & \text{if } x_{gh} = 0, \\ T & \text{if } x_{hi} = 0, \\ \text{unspecified} & \text{otherwise.} \end{cases}$$

Since (m,a,b,n), (m,d,e,n), and (m,g,h,n) are the only three directed paths from m to n, and the length of the shortest path from m to n is $d_{mn} = 0$, at least one of x_{ab}, x_{de}, and x_{gh} must be zero, hence the logical statement $\bar{y}_1 \bigvee \bar{y}_2 \bigvee \bar{y}_3$ must be true. Similarly, $d_{pq} = 0$ implies that $y_1 \bigvee y_2 \bigvee y_3$ is true. Therefore, this FISPL instance has a G_d-satisfying weight assignment if and only if there exists a truth assignment for the logical statement $(\bar{y}_1 \bigvee \bar{y}_2 \bigvee \bar{y}_3) \bigwedge (y_1 \bigvee y_2 \bigvee y_3)$. Using a similar reduction, any 3-SAT problem can be solved through a G_d-satisfying weight assignment for an FISPL instance.

3. COMPLEXITY OF ISPL AND RESTRICTED CASES

3.1. NP-Hardness of ISPL

Fekete et al. [10] conducted a thorough study of the complexity of the FISPL. We note that FISPL can be taken as a special case of ISPL where $f_{ij}() = 0$, for all $(i,j) \in A$. Since ISPL is at least as hard as FISPL, and the latter is NP-complete, ISPL is NP-hard.

3.2. NP-Hardness of SISPL and AISPL

Here, we delineate the difference in complexity between ISPL and FISPL problems: while both problems are NP-hard, ISPL is strictly harder than FISPL. Fekete et al. [10] identified two polynomial cases for FISPL, where the distance graph is a star (SFISPL) or a union of complete stars (AFISPL). In contrast, we show here that both SISPL and AISPL are NP-hard.

Theorem 3.1. *SISPL is NP-hard.*

Proof. We give a reduction from the Minimum Weight Steiner Tree Problem (MStT), defined as follows. Given a graph $G = (V, E)$, a weight function $\mathbf{c} : E \mapsto \Re_+^{|E|}$, and a set of terminal nodes $R \subseteq V$ where $|R| \geq 3$, is there a subtree of G that contains all nodes in R and has a total weight of less than or equal to W? Without loss of generality, we assume that $c_{ij} > 0$ for all $[i,j] \in E$.

For a given instance $G = (V, E, \mathbf{c})$, $R \subseteq V$ of MStT, we construct an instance $\hat{G}(\hat{\mathbf{c}}) = (V, \hat{A}, \hat{\mathbf{c}})$, $\hat{G}_d(V, \hat{A}_d, \hat{\mathbf{d}})$ of SISPL as follows:

$$\hat{A} = \{(i,j), (j,i) \mid [i,j] \in E\},$$
$$\hat{c}_{ij} = \hat{c}_{ji} = w_{ij}, \ \forall [i,j] \in E.$$

For s an arbitrary node in R, define:

$$\hat{A}_d = \{(s,t) \mid t \in R \setminus \{s\}\};$$
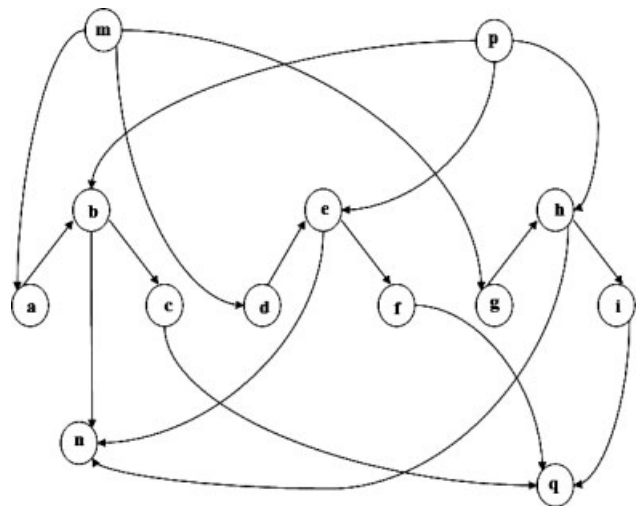$$\hat{d}_{st} = 0, \ \forall (s,t) \in \hat{A}_d.$$

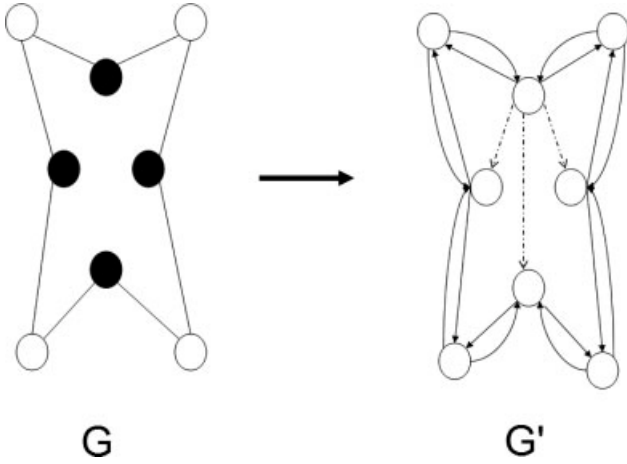FIG. 1. An instance of ISPL.

FIG. 2. Constructing an SISPL instance from an MStT instance.

Finally, let the penalty function be

$$f_{ij}(x_{ij}) = |x_{ij} - \hat{c}_{ij}|, \ \forall (i,j) \in \hat{A}.$$

Figure 2 illustrates how to construct an SISPL instance from an MStT instance. The graph on the left is an instance of MStT, with all terminal nodes shown in black. The graph on the right is the SISPL instance, where the solid lines stand for arcs in the network and the dotted lines represent arcs in the distance graph. The zeros along the lines indicate the weights of the arcs in the distance.

We claim that there is a Steiner tree with weight no more than W if and only if there is a solution to the SISPL instance whose modification cost does not exceed W.

Let $\mathbf{x}$ be a solution of the SISPL instance. We assume that

$$x_{ij} = \begin{cases} 0 & \text{if } (i,j) \text{ is on a shortest path from } s \text{ to,} \\ & \quad \text{a terminal node,} \\ \hat{c}_{ij} & \text{otherwise.} \end{cases}$$

This assumption does not affect the generality of the proof since if there exist $(i,j) \in \hat{A}$ such that $x_{ij} \neq \hat{c}_{ij}$, and (i,j) is not on any shortest path, then we can always change x_{ij} to $\hat{c}_{ij}$ without affecting the feasibility, and this new weight assignment can only cost less.

Define $\hat{T} = \{(i,j) \in \hat{A} \mid x_{ij} = 0\}$. We make a further assumption that $\hat{T}$ is a directed tree rooted at s. If not, we can find an arc (i,j) whose deletion from $\hat{T}$ does not affect the connectivity from s to any other terminal node. By letting $x_{ij} = 0$, we obtain a feasible weight assignment with a lower penalty cost.

Let $T = \{[i,j] \in E \mid (i,j) \in \hat{T} \text{ or } (j,i) \in \hat{T}\}$. T is a Steiner tree of G since it connects s with every other terminal node. Finally, we show that $F(\mathbf{x}) = \sum_{(i,j) \in \hat{A}} f_{ij}(x_{ij} - \hat{c}_{ij}) = \sum_{[i,j] \in T} c_{ij}$. The only nontrivial part of the proof is that at most one of x_{ij} and x_{ji} can be zero, due to the fact that $\hat{T}$ is acyclic.

Conversely, if the MStT instance has a solution T, we define the following weight assignment for the SISPL instance:

$$x_{ij} = \begin{cases} 0 & \text{if } [i,j] \in T \text{ and } i \text{ is closer to } s \text{ than } j \text{ in } T, \\ & \quad \text{than } j \text{ in } T, \\ \hat{c}_{ij} & \text{otherwise.} \end{cases}$$

This weight assignment yields a path of zero length in $\hat{G}$ between s and every other terminal node. These paths are the shortest because of the nonnegativity of arc weights. The penalty cost associated with x is $F(\mathbf{x}) = \sum_{(i,j) \in \hat{A}} f_{ij}(x_{ij} - \hat{c}_{ij}) = \sum_{[i,j] \in T} c_{ij}$.

The above is a reduction of the MStT instance to an ISPL instance whose distance graph is a source star. To attain the reduction to an ISPL instance whose distance graph is a sink star, one needs only to reverse all the arcs in the network and distance graph. ∎

Theorem 3.2. *AISPL is NP-hard.*

Proof. The NP-hardness of AISPL is derived from the reduction of the Hamiltonian Cycle problem (HC): given a graph $G = (V, E)$, is there a cycle in G that visits each node exactly once?

Let $G = (V, E)$ be an instance of HC. We construct an instance $\hat{G}(\hat{\mathbf{c}}) = (V, \hat{A}, \hat{\mathbf{c}}), \hat{G}_d(V, \hat{A}_d, \hat{\mathbf{d}})$ of AISPL as follows:

$$\hat{A} = \{(i,j), (j,i) \mid [i,j] \in E\};$$
$$\hat{c}_{ij} = 1, \ \forall (i,j) \in \hat{A};$$
$$\hat{A}_d = \{(u,v) \mid u, v \in V, \ u \neq v\};$$
$$\hat{d}_{uv} = 0, \ \forall (u,v) \in \hat{A}_d.$$

The penalty function is defined as:

$$f_{ij}(x_{ij}) = |x_{ij} - \hat{c}_{ij}|, \ \forall (i,j) \in \hat{A}.$$

We claim that there is a Hamiltonian cycle in G, if and only if the AISPL instance has a solution that costs no more than $|V|$.

Suppose $\mathbf{x}$ is a solution to the AISPL instance and $F(\mathbf{x}) = \sum_{(i,j) \in \hat{A}} f_{ij}(x_{ij}) \leq |V|$. Define $\hat{C} = \{(i,j) \in E \mid x_{ij} = 0\}$. Since there is a zero length path between each pair of nodes in $\hat{C}$ and $|\hat{C}| \leq \sum_{(i,j) \in \hat{A}} f_{ij}(x_{ij}) \leq |V|$, $\hat{C}$ must be a directed cycle that visits each node exactly once. We can easily construct a Hamiltonian cycle of G by letting $C = \{[i,j] \mid (i,j) \in \hat{C} \text{ or } (j,i) \in \hat{C}\}$.

Conversely, if there exists a Hamiltonian cycle C in G, we can define a feasible weight assignment $\mathbf{x}$ for $\hat{G}$ as follows. Order the nodes in the cyclical order that they appear on C, starting from an arbitrary node $s \in V$ and in an arbitrary direction. Let $x_{ij} = 0$, if node j is visited right after node i, otherwise $x_{ij} = 1$. Under this weight assignment, there is a path of length 0 between each ordered pair of nodes in $\hat{G}$. These paths must be shortest due to the nonnegativity of arc weights. The total penalty cost associated with this weight assignment is equal to $|V|$. ∎

3.3. A Polynomially Solvable Case

From the proof of Theorem 3.1 it is evident that the single-source ISPL problem shares some similarity with the Steiner tree problem in the sense that the union of shortest paths forms a tree that connects the source node with all the sink nodes. The minimum Steiner tree problem reduces to the tractable minimum spanning tree problem if all nodes in the graph are terminal nodes. Analogously, we will show that ISPL can be solved in polynomial time if its distance graph is a complete star. It is interesting to note that this special case is in the intersection of the single-source and all-sink case.

The proof of the complexity of SAISPL is based on the following lemma.

Lemma 3.1. *Let $G(\mathbf{c}) = (V, A, \mathbf{c})$, $G_d = (V, A_d, \mathbf{d})$ be an instance of SAISPL, where $A_d = \{(s, j) \mid j \in V, j \neq s\}$ forms a complete star. An optimal weight assignment $\mathbf{x}$ satisfies:*

$$x_{ij} = \begin{cases} d_{sj} - d_{si} & \text{if } c_{ij} < d_{sj} - d_{si} \text{ or } (i,j) \text{ is on a shortest,} \\ & \qquad \text{path from } s \text{ to } j, \\ c_{ij} & \text{otherwise.} \end{cases}$$

Proof. A necessary condition for $D_{si}(\mathbf{x}) = d_{si}$ and $D_{sj}(\mathbf{x}) = d_{sj}$ is that $x_{ij} \geq d_{sj} - d_{si}$, for each $(i,j) \in A$. If the original weight $c_{ij} < d_{sj} - d_{si}$, then $x_{ij} = d_{sj} - d_{si}$ satisfies the above condition, as well as minimizes the penalty cost on (i,j). On the other hand, if $c_{ij} \geq d_{sj} - d_{si}$, $x_{ij} = d_{sj} - d_{si}$ only if (i,j) is on a shortest path from s to j. If (i,j) is not on any shortest path, x_{ij} should equal to c_{ij}, otherwise the solution is suboptimal. ∎

Theorem 3.3. *SAISPL is polynomially solvable.*

Proof. Let $G = (V, A, \mathbf{c})$, $G_d = (V, A_d, \mathbf{d})$ be an instance of SAISPL, where $A_d = \{(s, j) \mid j \in V, j \neq s\}$ forms a complete star. Without loss of generality, we assume that $c_{ij} \geq d_{sj} - d_{si}$ for all $(i,j) \in A$. If there is $(i,j) \in A$

such that $c_{ij} < d_{sj} - d_{si}$, then from Lemma 3.1 an optimal solution $\mathbf{x}$ always satisfies $x_{ij} = d_{sj} - d_{si}$, and the solution does not change if we let $c_{ij} = d_{sj} - d_{si}$. We just need to add the penalty on (i,j) to the total cost.

Under this assumption, the weight on (i,j) in an optimal solution is either $d_{sj} - d_{si}$ or c_{ij}. An arc (i,j) is said to be admissible if $x_{ij} = d_{sj} - d_{si}$, and then (i,j) is on a shortest path from s to j. The above discussion reduces our problem to finding the cheapest way of making a subset of arcs that connect s to every other node in the graph admissible. The cost of making (i,j) admissible is:

$$w_{ij} = \begin{cases} f_{ij}(d_{sj} - d_{si} - c_{ij}) & \text{if } d_{sj} - d_{si} \geq 0, \\ \infty & \text{otherwise.} \end{cases}$$

This problem is equivalent to the Minimum Weight Arborescence Problem studied in [20].

Figure 3 illustrates an example of a reduction of an SAISPL instance (G, G_d) to a Minimum Weight Arborescence instance G'. Numbers along lines indicate the arc weights. The penalty functions used in this example are $f_{ij}(x_{ij} - c_{ij}) = |x_{ij} - c_{ij}|$ for all arcs (i,j).

The algorithm to find a solution $\mathbf{x}$ to the SAISPL instance is summarized below. Let a routine, which returns the minimum weight arborescence rooted at s among the set of arcs A and with respect to the weight function $\mathbf{w}$ be called $MWA(A, s, \mathbf{w})$.

STEP 0. Let $A' = \Phi$.

STEP 1. For each $(i,j) \in A$, calculate $\tilde{x}_{ij} = d_{sj} - d_{si}$. If $\tilde{x}_{ij} \geq 0$, $A' = A' \cup \{(i,j)\}$.

STEP 2. For each $(i,j) \in A'$, let

$$w_{ij} = \begin{cases} f_{ij}(\tilde{x}_{ij} - c_{ij}) & \text{if } c_{ij} \geq \tilde{x}_{ij} \\ 0 & \text{if } c_{ij} < \tilde{x}_{ij}. \end{cases}$$

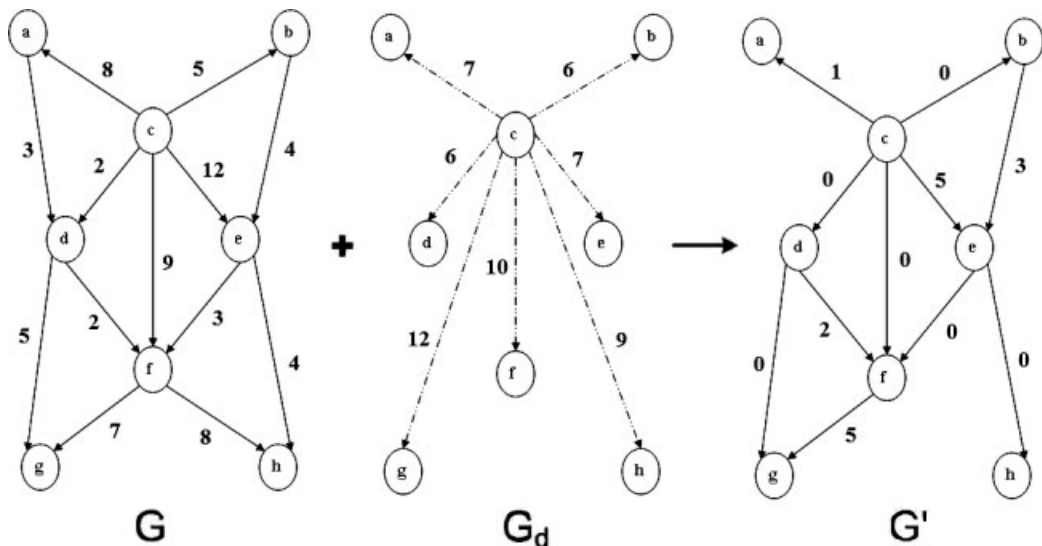STEP 3. Find $T = MWA(A', s, \mathbf{w})$.

FIG. 3.   Reducing an SAISPL instance to the Minimum Weight Arborescence Problem.
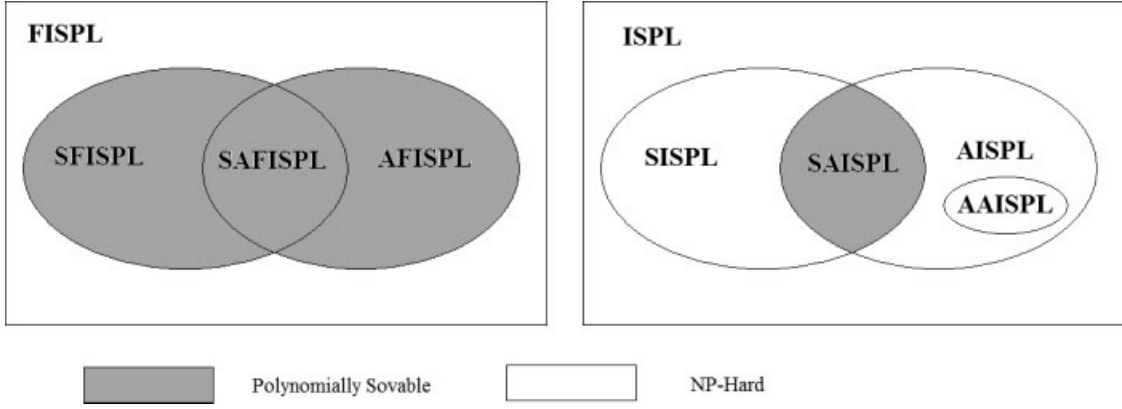
FIG. 4.   Complexities of FISPL and ISPL.

STEP 4. Set

$$x_{ij} = \begin{cases} \tilde{x}_{ij} & \text{if } c_{ij} \leq \tilde{x}_{ij} \text{ or } (i,j) \in T, \\ c_{ij} & \text{otherwise.} \end{cases}$$

Note that the minimum modification cost is not equal to the total weight on the minimum weight arborescence, because we do not count the cost to "bring up" arc weights, which is a constant. The actual modification cost is

$$\begin{aligned} F(\mathbf{x}) &= \sum_{(i,j) \in A} f_{ij}(x_{ij} - c_{ij}) \\ &= \sum_{(i,j) \in T} w_{ij} + \sum_{(i,j) \in A \mid c_{ij} < \tilde{x}_{ij}} f_{ij}(\tilde{x}_{ij} - c_{ij}). \end{aligned}$$

The complexity of finding a minimum weight arborescence dominates all other steps. Tarjan's algorithm runs in $O(|A| \cdot \log |V|)$ time (see [20]), which is clearly polynomial. ∎

Recall that FISPL is a special case of ISPL where the modification cost functions are zero. Although both problems are NP-hard, they differ in complexity when the distance graphs have special structures. Specifically, SFISPL and AFISPL are polynomially solvable, whereas SISPL and AISPL are NP-hard. A polynomial case of ISPL is SAISPL, which belongs to both of the SISPL and the AISPL cases. Figure 4 provides a diagram to compare the complexities of these problems.

4. LBISPL AND AN ALGORITHM FOR ISPL

In this section, we study a relaxation of ISPL, in which the shortest path lengths are required to be no less than certain lower bounds (LBISPL). LBISPL and its complementary problem, upper bounding ISPL (UBISPL), are studied by Burton and Toint [5] and the latter is shown to be NP-complete. Although it is admitted in [5] that LBISPL should be easier, their formulation for LBISPL consists of an exponential number of constraints. We provide a compact convex programming formulation for LBISPL and demonstrate how this formulation can be applied to develop an efficient approximate algorithm for ISPL.

4.1. A Compact Formulation for LBISPL

Let $G = (V, A, \mathbf{c})$, $G_d = (V, A_d, \mathbf{d})$ be an instance of LBISPL, where $A_d = \{(s_k, t_k) \mid k \in K\}$ and $L_k = d_{s_k, d_k}$ is the lower bound on D_{s_k, t_k}. We use a formulation of LBISPL given in [14]:

$$\text{Min } F(\mathbf{x}) = \sum_{(i,j) \in A} f_{ij}(x_{ij} - c_{ij}) \tag{1a}$$

$$\text{s.t. } p_j^k - p_i^k \leq x_{ij} \quad \forall k \in K, \ \forall (i,j) \in A \tag{1b}$$

$$p_{s_k}^k = 0 \quad \forall k \in K \tag{1c}$$

$$p_{t_k}^k \geq L_k \quad \forall k \in K \tag{1d}$$

$$p^k \geq 0 \quad \forall k \in K \tag{1e}$$

To see that this formulation is indeed valid for LBISPL, note that for any $(\mathbf{x}, \mathbf{p})$ satisfying (1b)–(1e), $\mathbf{x}$ is a feasible solution to the LBISPL instance. This is because $D_{s_k, i}(\mathbf{x}) \geq p_i^k$ for all $k \in K$ and $i \in V$. We prove this by induction on the number of arcs in the shortest path.

First, if the shortest path from s_k to i consists of only one arc, then $D_{s_k, i}(\mathbf{x}) = x_{s_k, i} \geq p_i^k - p_{s_k}^k = p_i^k$. Next, suppose $D_{s_k, i}(\mathbf{x}) \geq p_i^k$ holds for all $k \in K$ and $i \in V$, given that all shortest paths from s_k to i use no more than ℓ arcs. For a shortest path from s_k to j using $\ell + 1$ arcs, let i be the last node on the path visited before j. It follows that the path from s_k to i is also the shortest. Therefore, $D_{s_k, j}(\mathbf{x}) = D_{s_k, i}(\mathbf{x}) + x_{ij} \geq p_i^k + p_j^k - p_i^k = p_j^k$, where the inequality follows from the inductive assumption and constraint (1c).

Consequently, we have that the inequalities $D_{s_k, t_k}(\mathbf{x}) \geq p_{t_k}^k \geq L_k$ hold for all $k \in K$; i.e., $\mathbf{x}$ is a feasible solution to the LBISPL instance.

Conversely, if $\mathbf{x}$ is a feasible weight assignment for the LBISPL instance, define $p_i^k = D_{s_k, i}(\mathbf{x})$ for all $k \in K$ and $i \in V$. Then $(\mathbf{x}, \mathbf{p})$ satisfies (1b)–(1e), since $p_j^k - p_i^k = D_{s_k, j}(\mathbf{x}) - D_{s_k, i}(\mathbf{x}) \leq x_{ij}$, for all $k \in K$ and $(i, j) \in A$.

Formulation (1a)–(1e) is a convex programming problem with linear constraints and separable objective function. This type of problem can be solved in polynomial time using the

interior-point method, given that the objective function satisfies a self-concordance condition, in which its third derivative is assumed to be bounded in terms of its second derivative (see [11], [18], and [21]). Specifically, Kortanek and Zhu [16] introduce an $O(n \log \epsilon)$ algorithm to find ϵ-approximate solutions for linearly constrained convex programing problems with n variables. Their algorithm imposes a scaled Lipschitz condition on the objective function, which is satisfied by a linear, convex quadratic, or the entropy function.

4.2. An Algorithm for ISPL

Hung [15] has proposed a heuristic algorithm for FISPL, which can be easily adapted to address the ISPL problem. The idea is to take the LBISPL relaxation of an ISPL instance, by imposing a penalty on the gap between the lengths of the shortest paths and the desired values. Because of the unavailability of a compact formulation, the relaxed problem is solved by row generation in [15]. Since the solution of LBISPL is a repeatedly called subroutine, the use of the convex programming formulation ((1a)–(1e)) greatly improves the efficiency of this algorithm.

Another difficulty is due to the lack of knowledge of the shortest paths. We use the initiation of the estimates of the shortest paths lengths as in [15], to be the shortest paths in $G(\mathbf{c})$. After solving the LBISPL relaxation, we update the estimated paths to the shortest paths under the current weight assignment and reoptimize the problem. This procedure is repeated until the estimated paths are indeed shortest with respect to the modified arc weights. The detailed algorithm is stated below, for an ISPL instance $G = (V, A, \mathbf{c})$, $G_d = (V, A_d, \mathbf{d})$, and $A_d = \{(s_k, t_k) \mid k \in K\}$. Note that α is the penalty multiplier on the infeasibility gap, and flag is used to indicate whether or not the set of estimated paths is modified in an iteration.

Relaxation Heuristic

STEP 0. Set $\ell = 0$ and flag $= 0$. For each $k \in K$, set P_k^ℓ to be a shortest path from s_k to t_k under weight assignment $\mathbf{c}$.

STEP 1. Find a solution $\mathbf{x}^\ell$ to the following problem LBISPL(ℓ):

$$\text{LBISPL}(\ell) \ \text{Min} \quad \sum_{(i,j) \in A} f_{ij}(x_{ij} - c_{ij})$$

$$+ \sum_{k \in K} \alpha_k \left(\sum_{(i,j) \in P_k^\ell} x_{ij} - d_{s_k, t_k} \right) \quad (2a)$$

$$\text{s.t.} \quad p_j^k - p_i^k \leq x_{ij} \quad \forall k \in K, \ (i,j) \in A \quad (2b)$$

$$p_{s_k}^k = 0 \quad \forall k \in K \quad (2c)$$

$$p_{t_k}^k \geq d_{s_k, t_k} \quad \forall k \in K \quad (2d)$$

$$p^k \geq 0 \quad \forall k \in K. \quad (2e)$$

STEP 2. For each $k \in K$, if P_k^ℓ is the shortest under weight assignment $\mathbf{x}^\ell$, set $P_k^{\ell+1} = P_k^\ell$; otherwise set $P_k^{\ell+1}$ to be an

arbitrary shortest path from s_k to t_k under $\mathbf{x}^\ell$. If $P_k^{\ell+1} \neq P_k^\ell$ for some $k \in K$, set flag$= 1$; otherwise set $flag = 0$.

STEP 3. If flag $= 0$, terminate and output $\mathbf{x}^\ell$. Otherwise set $\ell = \ell + 1$ and go to Step 1.

Next, we discuss the finite convergence of the algorithm.

Lemma 4.1. *The optimal objective value of (2a)-(2e) decreases at each iteration.*

Proof. Define $z^\ell(\mathbf{x})$ to be the objective value of LBISPL(ℓ) associated with weight assignment $\mathbf{x}$. The following assertion holds:

$$z^{\ell+1}\left(\mathbf{x}^{\ell+1}\right) \leq z^{\ell+1}(\mathbf{x}^\ell) = \sum_{(i,j) \in A} f_{ij}\left(x_{ij}^\ell - c_{ij}\right)$$

$$+ \sum_{k \in K} \alpha_k \left[D\left(P_k^\ell, \mathbf{x}^\ell\right) - d_k \right]$$

$$< \sum_{(i,j) \in A} f_{ij}\left(x_{ij}^\ell - c_{ij}\right)$$

$$+ \sum_{k \in K} \alpha_k \left[D\left(P_k^{\ell+1}, \mathbf{x}^\ell\right) - d_k \right] = z^\ell(\mathbf{x}^\ell),$$

where the first inequality follows from the fact that $\mathbf{x}^{\ell+1}$ is optimal to LBISPL($\ell + 1$). The second (strict) inequality is due to the fact that $P_k^{\ell+1}$ is a shortest path between s_k and t_k under the weight assignment $\mathbf{x}^\ell$, and $P_k^{\ell+1}$ is strictly shorter than P_k^ℓ for at least one $k \in K$. ∎

Theorem 4.1. *The Relaxation Heuristic is finitely convergent.*

Proof. The number of paths between two nodes is finite, since there are finite number of arcs in the graph. As a consequence, the number of combinations of the estimated shortest paths used in an iteration is also finite. Furthermore, each of these combinations is considered at most once because the objective function is strictly decreasing from Lemma 4.1. Therefore, the algorithm terminates in a finite number of steps. ∎

5. SINGLE-SOURCE SINGLE-SINK ISPL

One major difficulty associated with solving ISPL is that the modification of the weight of a single arc can affect the lengths of shortest paths between more than one source-sink pair. This issue is not present if the distance graph consists of only one source-sink pair. In this section, we show that single-source single-sink ISPL (SSISPL) is polynomial time solvable under certain assumptions on the parameters and the cost function.

The discussion in this section addresses the SSISPL instance $G = (V, A, \mathbf{c})$, $G_d = (V, A_d, \mathbf{d})$ where $A_d = \{(s, t)\}$. To simplify the discussion we define the length of a shortest $s - t$ path in $G(\mathbf{x})$ as $D(\mathbf{x}) = D_{st}(\mathbf{x})$ and the desired length

as $d = \mathbf{d}_{st}$. The complexity results of SSISPL are based on the following lemmas.

Lemma 5.1. *The SSISPL instance is equivalent to its LBISPL relaxation if $D(\mathbf{c}) < d$.*

Proof. Suppose that $\mathbf{x}$ is an optimal solution to the LBISPL relaxation. Then it must be true that $D(\mathbf{x}) = d$ and thus $\mathbf{x}$ is optimal for SSISPL. Otherwise suppose $D(\mathbf{x}) > d$. Since $D(\mathbf{c}) < d$, there must be an arc $(i,j) \in A$ such that $x_{ij} > c_{ij}$. We define a new weight assignment $\mathbf{x}'$ as follows:

$$x_e' = \begin{cases} x_e - \varepsilon & e = (i,j) \\ x_e & e \in A \setminus \{(i,j)\}, \end{cases}$$

where $\varepsilon = \min\{D(\mathbf{x}) - d, x_{ij} - c_{ij}\}$.

Since $D(\mathbf{x}') \geq D(\mathbf{x}) - \varepsilon \geq d$, $\mathbf{x}'$ is also a feasible solution to the LBISPL relaxation. Furthermore, the penalty cost associated with $\mathbf{x}'$ is strictly less than that associated with $\mathbf{x}$. This contradicts the assumption that $\mathbf{x}$ is optimal to the LBISPL relaxation. ∎

Next we show that if $D(\mathbf{c}) > d$, then SSISPL is equivalent to a new problem CPP, defined as follows. Let P be a path from s to t. For a constant $d \geq D(\mathbf{c})$, we define $\mathbf{x}(P,d) = argmin_{\mathbf{x}}\{F(\mathbf{x})| D(P,\mathbf{x}) = d, \mathbf{x} \geq \mathbf{0}\}$ and $C(P,d) = F(\mathbf{x}(P,d))$. The cheapest path problem (CPP) is to find an $s - t$ path that minimizes $C(P,d)$; i.e., the problem is to find an $s - t$ path whose length is cheapest to reduce to d.

Lemma 5.2. *The SSISPL instance is equivalent to the CPP problem if $D(\mathbf{c}) > d$.*

Proof. Suppose that P' is an $s - t$ path that minimizes $C(P,d)$. We define $\mathbf{x}' = \mathbf{x}(P',d)$, and claim that $\mathbf{x}'$ is an optimal solution to the SSISPL instance.

To prove the feasibility of $\mathbf{x}'$, we only need to show that P' is a shortest path in $G(\mathbf{x}')$, since by construction we have $D(P',\mathbf{x}') = d$. If P' is not the shortest under $\mathbf{x}'$, let P'' be an $s - t$ path such that $D(P'',\mathbf{x}') < D(P',\mathbf{x}') = d$, and let $D(P'',\mathbf{x}') = d - \varepsilon$ for some $\varepsilon > 0$. Because $D(P'',\mathbf{c}) \geq D(\mathbf{c}) > d$, $C(P'',d)$ must be decreasing in d; i.e., given that the original length of P'' is higher than the target length, the modification cost is lower as the target length increases. Therefore, $C(P'',d) < C(P'',d - \varepsilon)$, and $C(P'',d - \varepsilon) \leq F(\mathbf{x}')$, following from the definition of $C(P,d)$. Putting these together, we have $C(P'',d) < F(\mathbf{x}') = C(P',d)$. This contradicts the assumption that P' minimizes $C(P,d)$.

The optimality of $\mathbf{x}'$ is straight forward, since $F(x') = C(P',d)$ is the lowest cost to modify any $s - t$ path to have length d. This completes the proof. ∎

We next identify some polynomial cases of CPP.

Lemma 5.3. *CPP is polynomially solvable if $f_{ij}(x_{ij} - c_{ij}) = |x_{ij} - c_{ij}|, \forall(i,j) \in A$.*

Proof. If $f_{ij}(x_{ij}) = |x_{ij} - c_{ij}|$ for all $(i,j) \in A$, the cost of reducing the length of a path is equal to the difference between the path length and the desired value. Therefore, the problem is reduced to finding a shortest $s - t$ path in $G(\mathbf{c})$. Since $\mathbf{c} \geq \mathbf{0}$, Dijkstra's algorithm can be used to find the shortest path. The complexity is $O(|A| + |V|\log|V|)$. ∎

We now consider a wider range of penalty functions. We say that a penalty function $\mathbf{f}$ is *uniform* to a convex function $g : \mathfrak{R} \mapsto \mathfrak{R}_+$ if $f_{ij}(x_{ij} - c_{ij}) = g(x_{ij} - c_{ij})$, for all $(i,j) \in A$ and $\mathbf{x} \in \mathfrak{R}_+^{|A|}$. The following lemmas characterize solutions to the cheapest path with such uniform penalty functions.

Lemma 5.4. *Let P be an $s - t$ path and $|P|$ be the number of arcs it uses. If $\mathbf{f}$ is uniform to g and $c_{ij} - \frac{D(P,\mathbf{c})-d}{|P|} \geq 0$ for all $(i,j) \in A$, then $C(P,d) = |P| \cdot g(\frac{L(P)-d}{|P|})$.*

Proof. From the definition, $C(P,d)$ is the optimum objective value of the following mathematical programming problem (3a)–(3c):

$$\text{Min} \sum_{(i,j)\in A} g(x_{ij} - c_{ij}) \tag{3a}$$

$$\text{s.t.} \sum_{(i,j)\in P} x_{ij} = d \tag{3b}$$

$$\mathbf{x} \geq \mathbf{0}. \tag{3c}$$

Substituting $t_{ij} = c_{ij} - x_{ij}$ and dropping the nonnegativity constraint, we get

$$\text{Min} \sum_{(i,j)\in A} g(t_{ij}) \tag{4a}$$

$$\text{s.t.} \sum_{(i,j)\in P} t_{ij} = D(P,\mathbf{c}) - d. \tag{4b}$$

The Karush-Kuhn-Tucker conditions for the above problem are:

$$g'(t_{ij}) + \lambda = 0, \ \forall(i,j) \in P$$
$$g'(t_{ij}) = 0, \ \forall(i,j) \in A \setminus P.$$

A solution satisfying these conditions is

$$t_{ij}^* = \begin{cases} \frac{D(P,\mathbf{c})-d}{|P|} & \text{if } (i,j) \in P \\ 0 & \text{otherwise,} \end{cases}$$

and

$$x_{ij}^* = \begin{cases} c_{ij} - \frac{D(P,\mathbf{c})-d}{|P|} & \text{if } (i,j) \in P \\ c_{ij} & \text{otherwise.} \end{cases}$$

Since $c_{ij} - \frac{D(P,\mathbf{c})-d}{N(P)} \geq 0$, for all $(i,j) \in A$, $\mathbf{x}^*$ is also a solution to (3). Therefore, $C(P,d) = F(\mathbf{x}^*) = |P| \cdot g(\frac{D(P,\mathbf{c})-d}{|P|})$. ∎

Lemma 5.5. *CPP is polynomially solvable if f is uniform to g and $c_{\max} - c_{\min} \leq \frac{d}{|V|-1}$.*

Proof. Let P be an arbitrary $s - t$ path with $|P|$ arcs. For each arc $(i, j) \in P$, it follows that

$$c_{ij} - \frac{D(P, \mathbf{c}) - d}{|P|} = \left[c_{ij} - \frac{D(P, \mathbf{c})}{|P|} \right] + \frac{d}{|P|}$$

$$\geq (c_{\min} - c_{\max}) + \frac{d}{|P|}$$

$$\geq (c_{\min} - c_{\max}) + \frac{d}{|V| - 1} \geq 0.$$

The first inequality holds since the difference between the weight of one arc and the average weight on the path cannot exceed the difference between $c_{\min}$ and $c_{\max}$. The second inequality follows as a simple path consists of no more than $|V| - 1$ arcs.

From Lemma 5.4 we know that $C(P, d) = |P| \cdot g(\frac{D(P, \mathbf{c}) - d}{N(P)})$, which depends only on $D(P, \mathbf{c})$ and $|P|$. Since $D(P, \mathbf{c}) > d$, among all $s - t$ paths using a fixed number of arcs, the shortest one minimizes $C(P, d)$.

Finding the shortest simple path length from s to t using exactly k arcs is an NP-hard problem (for $k = n - 1$ it would solve the Hamiltonian path problem). However, as we show here, it is sufficient to find the shortest, among all paths (simple and nonsimple), path length using exactly k arcs, $S_k(t)$. For $S_k(t)$, the length of a simple path of exactly k arcs, the minimum of $C(P, d)$ is equal to

$$\min_{1 \leq k \leq |V|-1} \left\{ k \cdot g \left(\frac{S_k(t) - d}{k} \right) \right\}.$$

The value of $S_k(t)$ for paths that could be nonsimple can be evaluated using dynamic programming: Let $S_k(j)$ be the shortest path length from s to j using exactly k arcs, for all $j \in V \setminus \{s\}$ and $k = 0, 1, \ldots, |V| - 1$. The recursive equations and the boundary conditions used in the dynamic programming are

$$S_k(j) = \min \left\{ \min_{(i,j) \in A} \{ S_{k-1}(i) + c_{ij} \}, \ \infty \right\},$$

$$\forall \, 1 \leq k \leq |V| - 1, \ j \in V;$$

$$S_0(s) = 0, \ S_0(j) = \infty, \ \forall j \in V \setminus \{s\}.$$

It is easy to see, by induction, that $S_k(t)$ is the length of a path with k arcs from s to t, which has value ∞ if there is no path with exactly k arcs: This is true initially and then assuming it is true for $k - 1$, the dynamic programming recursion guarantees that the length of the path with k arcs to any node j is $S_k(j)$. For each $1 \leq k \leq |V| - 1$, either $S_k(t) = \infty$, i.e., there is no $s - t$ path with exactly k arcs, or the dynamic programming algorithm yields an $s - t$ path P^k with exactly k arcs, which is the shortest among all $s - t$ paths using the same number of arcs (the path P^k may be non-simple).

Define $k^* = \text{argmin}\{ k \cdot g((D(P^k, \mathbf{c}) - d)/k) : 1 \leq k \leq |V| - 1\}$ (by convention, we let $D(P^k, \mathbf{c}) = \infty$ if there is no

$s - t$ path using exactly k arcs). We claim that $P^* = P^{k^*}$ is a simple path, and it is the cheapest among all simple paths from s to t. The proof is by contradiction, as shown next.

Suppose that P^* is nonsimple, so we can find a different $s - t$ path P' that only uses a subset of arcs in P^*. Let $k' < k^*$ be the number of arcs in P'. Also let $\mathbf{x}^* = \mathbf{x}(P^*, d)$ be the cheapest modification on P^*; i.e., $x_{ij}^* = c_{ij} - (D(P^*, \mathbf{c}) - d)/k^*$ if $(i, j) \in P^*$, and $x_{ij}^* = c_{ij}$ otherwise. It follows that

$$C(P^*, d) = k^* g((D(P^*, \mathbf{c}) - d)/k^*)$$

$$> k' g((D(P^*, \mathbf{c}) - d)/k^*)$$

$$= C(P', D(P', \mathbf{x}^*))$$

$$\geq C(P', d).$$

where the first inequality follows directly from $k' < k^*$; the second equation is based on the definition of the function C; and the last inequality is due to the fact that $\mathbf{x}^* \geq \mathbf{0}$ and $D(P', \mathbf{x}^*) \leq D(P^*, \mathbf{x}^*) = d$, and that $C(P, d)$ is decreasing in d given that $D(P, \mathbf{c}) > d$. It is also clear that $C(P', d) \geq C(P^{k'}, d)$, since $D(P', \mathbf{c}) \geq D(P^{k'}, \mathbf{c})$. This leads to the contradiction that $C(P^*, d) > C(P^{k'}, d)$.

Next, we show that P^* is the cheapest among all simple $s - t$ paths. Let $\hat{P}$ be an arbitrary simple path that uses $\hat{k}$ arcs. It follows that

$$C(\hat{P}, d) \geq C(P^{\hat{k}}, d) \geq C(P^*, d).$$

The complexity of the dynamic programming procedure is $O(|A| \cdot |V|)$ and it dominates all other steps. The algorithm stated above thus runs in polynomial time. ∎

Finally, we summarize the complexity results of SSISPL:

Theorem 5.1. *SSISPL is polynomially solvable if one or more of the following conditions are satisfied:*

- *$D(\mathbf{c}) < d$;*
- *$D(\mathbf{c}) > d$ and $f_{ij}(x_{ij} - c_{ij}) = |x_{ij} - c_{ij}|$ for all $(i, j) \in A$;*
- *$D(\mathbf{c}) > d$, $\mathbf{f}$ is uniform to a convex function, and $c_{\max} - c_{\min} \leq \frac{d}{|V|-1}$.*

Proof. The correctness of the above theorem follows from Lemmas 5.1–5.5. ∎

6. CONCLUSIONS

In this article we study the complexity of certain ISPL, as a generalization of the FISPL investigated in [10]. We show that ISPL is strictly harder than FISPL as demonstrated in the special cases where the distance graph is a star or a union of complete stars. We also provide a polynomial time algorithm for the more restricted case where the distance graph is a complete star. We then consider a relaxation of ISPL, in which the shortest path lengths are required to be no less than some prescribed lower bounds (LBISPL). We provide a compact convex programming formulation for LBISPL and demonstrate how it can be utilized to develop a heuristic algorithm

for ISPL. Also based on this formulation for LBISPL, we prove that the single-source ISPL problem is polynomially solvable if certain conditions hold on the penalty function and the original weight assignment.

REFERENCES

[1] S. Ahmed and Y. Guan, The inverse optimal value problem, Math Program 102 (2005), 91–110.

[2] R.K. Ahuja and J.B. Orlin, A faster algorithm for the inverse spanning tree problem, J Algorithms 34 (2000), 177–193.

[3] R.K. Ahuja and J.B. Orlin, Inverse optimization, Oper Res 49 (2001), 171–783.

[4] D. Burton, On the inverse shortest path problem, PhD Thesis, Facultes Universitaries Notre-Dame de la Paix de Namur, 1993.

[5] D. Burton, W.R. Pulleyblank, and Ph.L. Toint, "The inverse shortest path problem with upper bounds on shortest path costs", Lecture Notes in Economics and Mathematical Systems, P. Pardalos, D.W. Hearn, and W.H. Hager (Editors), Vol. 450, Network Optimization, Springer, 1997, pp. 156–171.

[6] D. Burton and Ph.L. Toint, On an instance of the inverse shortest paths problem, Math Program 53 (1992), 45–61.

[7] D. Burton and Ph.L. Toint, On the use of an inverse shortest paths algorithm for recovering linearly correlated costs, Math Program 63 (1994), 1–22.

[8] M. Cai and Y. Li, Inverse matroid intersection problem, Math Methods Oper Res 45 (1997), 235–243.

[9] M. Cai, X. Yang, and J. Zhang, The complexity analysis of the inverse center location problem, J Global Optim 15 (1999), 213–218.

[10] S.P. Fekete, W.H. Hochstattler, S. Kromberg, and C. Moll, "The complexity of an inverse shortest paths problem", Contemporary trends in discrete mathematics: From DIMACS and DIMATIA to the future, R.L. Graham, J. Kratochvil, J. Nesetrol, and F.S. Roberts (Editors), American Mathematical Society, DIMATIA-DIMACS Conference, Stirin Castle, Czech Republic, 1997, pp. 49–113.

[11] A.V. Fiacco and G.P. McCormick, Nonlinear programming: Sequential unconstrained minimization techniques, Wiley, New York, 1968.

[12] C. Heuberger, Inverse combinatorial optimization: A survey on problems, methods, and results, J Combin Optim 8 (2004), 329–361.

[13] D.S. Hochbaum, Efficient algorithms for the inverse spanning-tree problem, Oper Res 51 (2003), 785–797.

[14] D.S. Hochbaum, Polynomial time algorithms for the inverse shortest paths problems, Working Paper, 2005.

[15] C.H. Hung, On the inverse shortest path length problem, PhD Thesis, Georgia Institute of Technology, 2003.

[16] K.O. Kortanek and J. Zhu, A polynomial barrier algorithm for linearly constrained convex programming problems, Math Oper Res 18 (1993), 116–127.

[17] Z. Liu and J. Zhang, On inverse problem of maximum perfect matching, J Combin Optim 7 (2003), 215–228.

[18] Y.E. Nesterov and A.S. Nemirovskii, Interior point polynomial methods in convex programming, SIAM Publications, Philadelphia, 1994.

[19] P.T. Sokkalingam, R.K. Ahuja, and J.B. Orlin, Solving inverse spanning tree problems through network flow techniques, Oper Res 47 (1999), 291–298.

[20] R.E. Tarjan, Finding optimum branchings, Networks 7 (1997), 25–35.

[21] Y. Ye, Interior-point algorithms: Theory and analysis, John Wiley and Sons, New York, 1997.

[22] J. Zhang and Z. Liu, A general model of some inverse combinatorial optimization problems and its solution method under ℓ-infinity norm, J Combin Optim 6 (2002), 207–227.

[23] J. Zhang, Z. Liu, and Z. Ma, Some reverse location problems, Eur J Oper Res 124 (2000), 77–88.

[24] J. Zhang, Z. Ma, and C. Yang, A column generation method for inverse shortest paths problems, Math Methods Oper Res 41 (1995), 347–385.

[25] J. Zhang, X. Yang, and M. Cai, "Reverse center location problem", Lecture Notes in Computer Science, A. Aggarwal and C. Pandu Rangan (Editors), Algorithms and Computation, Vol. 1741, [Proc ISAAC'99, Chennai, India] Springer, New York, 1999, pp. 279–294.

[26] J. Zhang, X. Yang, and M. Cai, A network improvement problem under different norms, Comput Optim Appl 27 (2004), 305–319.

[27] J. Zhang, X. Yang, and M. Cai, Inapproximability and a polynomially solvable special case of a network improvement problem, Eur J Oper Res 155 (2004), 251–257.