

# The Complexity of Preprocessing

Tsung-Chyan Lai

Department of Business Administration

National Taiwan University

[tclai@ntu.edu.tw](mailto:tclai@ntu.edu.tw)

and

James B. Orlin

Sloan School of Management

MIT

[jorlin@mit.edu](mailto:jorlin@mit.edu)

October, 2003.

**Abstract.** Suppose that  $\min (c(S) : S \in \mathbf{F})$  is an instance  $\mathbf{I}$  of an optimization problem, where  $\mathbf{F}$  is a collection of subsets of  $J$ , and  $c$  is a linear cost function. Suppose that  $e \in J$ . One natural question within optimization is the following: Is there any optimal solution  $S'$  for  $\mathbf{I}$  such that  $e \in S'$ . In preprocessing, we do not assume that  $c$  is known precisely, but that each coefficient is known within an additive factor of  $K$ . The corresponding question in preprocessing is the following: Is there a cost vector  $d$  and a solution  $S' \in \mathbf{F}$  such that (i)  $e \in S'$ , (ii)  $|d(e') - c(e')| \leq K$  for each  $e' \in J$ , and (iii)  $S'$  is optimal for  $\min (d(S) : S \in \mathbf{F})$ ? (It is called preprocessing, because the complementary question determines whether  $e$  is no optimal solution regardless of the choice of  $d$ , and thus can be eliminated from the problem.) In this paper, we show that the preprocessing problem is NP-complete if the original problem is the shortest path problem on an acyclic network, the assignment problem, the minimum cut problem, or the minimum cost arc (or node) disjoint cycle problem. We show that the preprocessing problem is solvable in polynomial time if the original problem is either the minimum cost spanning tree problem or the maximum weight closure problem. We also discuss relationships between the preprocessing problem and inverse optimization.

**Key Words.** Preprocessing, Inverse Optimization, Network Optimization

## 1. INTRODUCTION

We define preprocessing problems in terms of combinatorial optimization problems. Let  $\mathbf{P}$  denote a 0-1 combinatorial optimization problem, and let  $\mathbf{I}$  denote an instance of  $\mathbf{P}$  with  $\mathbf{F}$  as the set of feasible solutions and  $c$  as the cost vector; that is,  $\mathbf{I} = \min\{c(S) : S \in \mathbf{F}\}$ , and  $\mathbf{F} \subseteq \{0, 1\}^n$ , where  $c(S) = \sum_{i \in S} c_i$ . For another cost vector  $d$ , we let  $\mathbf{I}(d) = \min\{d(S) : S \in \mathbf{F}\}$ . We assume that the ground set of the feasible solutions is  $J$ . That is, for each  $S \in \mathbf{F}$ ,  $S \subseteq J$ .

Let  $\|\cdot\|_\infty$  denote the sup norm or the  $L_\infty$  norm. For example,  $\|c\|_\infty = \max(|c_j| : j \in J)$ . Let  $\|\cdot\|_1$  denote the  $L_1$  norm. For example,  $\|c\|_1 = \sum_{j \in J} |c_j|$ .

### The Preprocessing Problem for $\mathbf{P}$ .

INSTANCE. An instance  $\mathbf{I}$  of  $\mathbf{P}$ , a element  $j^*$  of  $J$  and an integer  $K$ .

QUESTION. Is there a subset  $S \in \mathbf{F}$  and a cost vector  $d$  such that

- i.  $j^* \in S$ ;
- ii.  $\|d - c\|_\infty \leq K$ ; and
- iii.  $d(S) \leq d(S')$  for all  $S' \in \mathbf{F}$ ?

Thus the Preprocessing Problem for  $\mathbf{P}$  is the problem of determining whether there is an optimal solution  $S$  for  $\mathbf{I}(d)$  such that  $j^* \in S$  and such that  $\|d - c\|_\infty \leq K$ .

Lai and Sotskov [23] and Lai, Sotskov, Sotskova, and Werner [24] proposed a variant of preprocessing problems as a way of modeling situations in which there is an estimated cost function  $c$  at some time  $T$ , but the real objective function  $d$  does not become known until some later time  $T'$ . All that is known at time  $T$  is that  $\|c - d\|_\infty \leq K$  for some value  $K$ . Suppose that one can solve the Preprocessing Problem at time  $T$ . If the answer to the preprocessing problem is no, then one can eliminate  $j^*$  from  $J$ , and obtain a smaller optimization problem. Moreover, one can solve the Preprocessing Problem for each variable  $j \in J$ , and thus eliminate any variable that is in no optimal solution  $\mathbf{I}(d)$  for any possible choice of  $d$ . At time  $T'$ , when the solution time is perhaps more urgent, one can solve the real problem  $\mathbf{I}(d)$  but with fewer variables because of the preprocessing, and thus an improved running time.

The variant of the preprocessing problem considered by Lai and Sotskov [23] and by Lai, Sotskov, Sotskova, and Werner [24] is the one in which the condition (ii) above is replaced by the condition  $\|w(d - c)\|_\infty \leq K$ , where  $w$  is a vector of weights that is specified as part of the input. More precisely, they specified that  $l_i \leq d_i \leq u_i$  for each  $i \in J$ , which is mathematically equivalent to specifying a vector  $w$  of weights, a cost vector  $c$ , and a value  $K$ , and requiring that  $\|w(d - c)\|_\infty \leq K$ .

In this paper, we prove that the Preprocessing Problem is NP-complete for the Shortest Path problem, even if restricted to acyclic graphs. We also show that the Preprocessing Problem is NP-complete for the following problems: the Assignment Problem, the Minimum Cost Node Disjoint Cycle Problem, the Minimum Cost Arc Disjoint Cycle Problem, and the Minimum Cut Problem. We show that the Preprocessing Problem is solvable in polynomial time if  $\mathbf{F}$  is the collection of independent sets of a matroid (such as in the minimum cost spanning tree problem) or if  $\mathbf{F}$  is a lattice<sup>1</sup>. An example of a lattice is the set of closed sets in a graph.

### Relation to Inverse Optimization and Partial Inverse Optimization.

The Weighted Partial Inverse Optimization for  $\mathbf{P}$  can be described as follows;

#### Weighted Partial Inverse Optimization Problem for $\mathbf{P}$ .

INSTANCE. An instance  $\mathbf{I}$  of  $\mathbf{P}$ , disjoint subsets  $J_1$  and  $J_0$  of  $J$  and an integer  $K$ .

QUESTION. Is there a subset  $S \in \mathbf{F}$  and a cost vector  $d$  such that

- i.  $J_1 \subseteq S; J_0 \cap S = \emptyset;$
- ii.  $\|w(d - c)\|_1 \leq K;$  and
- iii.  $d(S) \leq d(S')$  for all  $S' \in \mathbf{F}.$

The *Partial Inverse Optimization Problem* for  $\mathbf{P}$  is the special case of the Weighted Partial Inverse Optimization Problem in which all weights are 1, and thus (ii) becomes  $\|d - c\|_1 \leq K$ . The *Weighted Minimax Partial Inverse Optimization Problem for  $\mathbf{P}$*  is the same as above, except that the  $L_1$  norm is replaced by the  $L_\infty$  norm. That is (ii) becomes  $\|w(d - c)\|_\infty \leq K$ . The *Minimax Partial Inverse Optimization Problem for  $\mathbf{P}$*  is the special case of the weighted problem in which all weights are 1, and thus (ii) becomes  $\|d - c\|_\infty \leq K$ .

Inverse problems are special cases of partial inverse problems in which  $J_1 \cup J_0 = J$ . Equivalently, the optimum solution is required to be  $J_1$ , and  $J_1$  must be optimal for  $\mathbf{I}(d)$  for some  $d$  that is a perturbation of  $c$  satisfying the norm inequality (ii).

The Preprocessing Problem for  $\mathbf{P}$  is a special case of the Minimax Partial Inverse Optimization Problem for  $\mathbf{P}$ , and thus the results in this paper establish the NP-completeness of the Minimax Partial Inverse Optimization Problem for the following problems: the Shortest Path Problem on acyclic graphs, the Assignment Problem, the Minimum Cost Node Disjoint Cycle Problem, the Minimum Cost Arc Disjoint Cycle Problem, and the Minimum Cut Problem. On the other hand, the Minimax Inverse Optimization Problem and the Inverse Optimization Problem for each of these problems is solvable in

---

<sup>1</sup> A lattice is a collection  $\mathbf{F}$  of subsets of  $J$  that are closed under both union and intersection.

polynomial time, as established in the following papers: Ahuja and Orlin [4], Yang, Zhang, and Ma [31], Zhang and Cai [32], and Zhang and Liu [33].

An additional motivation for analyzing the computational complexity of preprocessing problems is their close relationship with partial inverse optimization problems, which in turn generalize inverse optimization problems. In a companion paper, Orlin **Error! Reference source not found.** analyzes the computational complexity of each of the four types of partial inverse optimization problems with respect to the seven different network problems studied in this paper. Between this paper, and the paper of Orlin **Error! Reference source not found.**, 27 of the possible 28 problems are classified as either NP-complete or solvable in polynomial time.

The interest in inverse optimization problems for math programming problems was generated by the papers by Burton and Toint [9] and [10] who studied inverse shortest path problems arising in seismic tomography used in predicting the movement of earthquakes. In recent years, Inverse Optimization has been applied to analyzing user equilibrium in traffic analysis (see Dial [12] [13]), to the analysis of auctions (see Beil and Wein [6]), and Data Envelopment Analysis (see Wei, Zhang, and Zhang [27]). It is also closely connected to conjoint analysis in marketing in which marketers estimate utility functions of consumers by backwards inferring from product preferences (see, for example, Srinivasan [26] or Green and Srinivasan [19]). See Heuberger [20] for a survey of inverse optimization problems as well as closely related problems.

While inverse optimization problems have been studied widely in the past decade, the only papers that have dealt with partial inverse optimization are the following: Lai and Orlin [22], Gentry [17], Gentry, Saligrama, and Feron [18] and Yang [29]. Yang showed the NP-hardness of partial inverse assignment problem and the partial inverse minimum cut problem under the additional assumption that there are lower bounds and upper bounds on the objective  $d$ . Put into our format, he includes constraints of the form  $\|w(d - c)\|_\infty \leq K_1$  and also constraints of the form  $\|d - c\|_1 \leq K_2$ . In addition, he permits a subset of arcs to be required to be part of any optimal solution. As such, the NP-completeness results in this paper for the Preprocessing Assignment Problem and the Preprocessing Minimum Cut Problem are stronger than the results in Yang [29].

The outline of this paper is as follows. In Section 2, we state our graph theory notation, and prove some elementary results concerning preprocessing problems. In Section 3, we establish that the Preprocessing Shortest Path Problem is NP-complete. We also show that the following problems are NP-complete: the Preprocessing Minimum Cut Problem, the Preprocessing Minimum Cost Arc Disjoint Cycle Problem and the Preprocessing Minimum Cost Node Disjoint Cycle Problem. In Section 4, we show that the Preprocessing Shortest Path Problem remains NP-complete if restricted to acyclic graphs. We then establish the NP-completeness of the Preprocessing Assignment Problem. In Section 5, we give a polynomial time algorithm for a generalization of the Preprocessing Problem for  $\mathbf{P}$  where  $\mathbf{P}$  is the problem of finding a minimum cost basis of a matroid. This includes the Preprocessing Minimum Cost Spanning Tree Problem as a special case. In Section 6, we give a polynomial time algorithm for a

generalization of the Preprocessing Problem for  $\mathbf{P}$ , where  $\mathbf{P}$  is the problem of finding a minimum cost element of a lattice. Finally, in Section 7, we provide summary and conclusions.

## 2. ADDITIONAL NOTATION AND BACKGROUND

### Network and Graph Terminology

In this paper, if  $S$  and  $T$  are sets, then  $S \setminus T = \{j : j \in S \text{ and } j \notin T\}$ .

Let  $G = (N, A)$  be a (directed) network. Set  $N$  consists of a set of nodes. Unless stated otherwise, the arcs in set  $A$  are directed. We explicitly say *undirected network* if the arcs in  $A$  have no direction. Associated with each arc of the network are a cost  $c_{ij}$  and a capacity  $u_{ij}$ .

A *path* is an alternating sequence of nodes and arcs  $P = i_1, a_1, i_2, a_2, \dots, a_{k-1}, i_k$  such that  $a_j = (i_j, i_{j+1})$  or  $(i_{j+1}, i_j)$  for each  $j = 1$  to  $k-1$  and such that all nodes are distinct. A *directed path* is a path in which all arcs of the path are oriented in the same direction, that is  $a_j = (i_j, i_{j+1})$  for each  $j = 1$  to  $k-1$ . A network is *connected* if there is a path between every pair of nodes. A maximally connected subgraph of  $G$  is called a *connected component* of  $G$ . An *s-t path* is a directed path from node  $s$  to node  $t$ . The cost of a directed path  $P$  is the sum of the costs of its arcs. The *minimum cost s-t path problem* is to find a minimum cost path from an origin node  $s$  to a destination node  $t$ . We will also call this problem the *shortest path problem*, with the understanding that there is a designated origin node for the path and destination node for the path, and that the length of an arc is its cost.

The first node  $i_1$  of a path is called its *initial* node. The last node  $i_k$  of a path is called its *terminal* node. A *cycle* is a path  $P$  plus the arc  $(i_1, i_k)$  or the arc  $(i_k, i_1)$ . A *directed cycle* is a directed path  $P$  plus the arc  $(i_k, i_1)$ . An undirected network is *acyclic* if it has no undirected cycle. A directed network is *acyclic* if it has no directed cycle. The *minimum cost arc disjoint cycle problem* is to determine a collection  $(C_1, C_2, \dots, C_k)$  of directed cycles no two of which have an arc in common, and such that the total cost of the cycles is minimum.<sup>2</sup> The *minimum cost node disjoint cycle problem* is to determine a collection  $(C_1, C_2, \dots, C_k)$  of directed cycles no two of which have a node in common, and such that the total cost of the cycles is minimum.

A network  $G = (N, A)$  is called *bipartite* if it is possible to partition  $N$  into two parts  $N_1$ , and  $N_2$  so that every arc  $(i, j)$  has  $i \in N_1$  and  $j \in N_2$  or has  $i \in N_2$  and  $j \in N_1$ . Usually, a bipartite network will be represented as  $G = (N_1 \cup N_2, A)$ . In the case that  $|N_1| = |N_2|$ , an *assignment* (also called a *complete*

---

<sup>2</sup> It is equivalent to the minimum cost circulation problem on a network in which lower bounds on flows are 0, and upper bounds on flows are 1.

matching)  $S$  is a collection of  $|N_1|$  arcs so that each node of  $N_1$  and  $N_2$  is incident to exactly one of the arcs of  $S$ .<sup>3</sup> The *assignment problem* is to determine an assignment of minimum cost.

A *closed set* or *closure* in a network  $G = (N, A)$  is a set  $S$  of nodes with the property that no arc is directed out of  $S$ ; that is, if  $i \in S$ , and if  $(i, j) \in A$ , then  $j \in S$ . Let  $c_i$  denote the *cost* or *weight* of node  $i$  in  $G$ . The weight of a closure  $S$  is the sum of the weights of its nodes. The *maximum weight closure problem* is to determine a closure in  $G$  of maximum weight. The *minimum cost closure problem* is to determine a closure of minimum cost. The maximum weight closure problem is easily transformed into the minimum cost closure problem by replacing each weight by its negative. We will consider the minimum cost closure problem in Section 6. The maximum weight closure problem has a number of applications, (see, for example, Ahuja et al [1]) including the open pit mining problem, which was introduced by Johnson[21].

A subgraph  $G' = (N', A')$  of  $G$  is called *spanning* if  $N' = N$  and if  $G'$  is connected. A *spanning tree* is a spanning subgraph whose undirected version is acyclic. The cost of a spanning tree is the sum of the costs of its arcs. The *minimum cost spanning tree problem* is to determine a spanning tree of minimum cost.

In the network  $G$ , a *cut* is a subset  $A'$  of arcs whose deletion disconnects  $G$  into two or more components, and such that no strict subset of arcs of  $A'$  has this property. An *s-t cut* is a cut that partitions the node set into exactly two parts of which one part, say  $S$ , contains the node  $s$  and another part  $\bar{S} = N \setminus S$  contains node  $t$ . We let  $[S, \bar{S}]$  denote the cut. We let  $(S, \bar{S})$  represent the *forward arcs* of the cut; that is  $(S, \bar{S}) = \{(i, j) \in A : i \in S \text{ and } j \in \bar{S}\}$ . We let  $(\bar{S}, S)$  represent the *backward arcs* of the cut; that is  $(\bar{S}, S) = \{(i, j) \in A : i \in \bar{S} \text{ and } j \in S\}$ . The *capacity* of the *s-t cut*  $[S, \bar{S}]$  is the sum of the capacities of the forward arcs of the cut. We denote it by  $u[S, \bar{S}] = \sum_{(i,j) \in (S, \bar{S})} u_{ij}$ . The *minimum s-t cut problem* is to determine an *s-t cut* of minimum capacity.

## Preprocessing Problems

In this subsection, we state and prove a theorem showing that the preprocessing problems considered in this paper are all in the class NP. We first introduce another term that is widely used in proofs of NP-completeness (see Garey and Johnson [16]).

Let  $\mathbf{I} = \min\{c(S) : S \in \mathbf{F}\}$  be an instance of problem  $\mathbf{P}$ . Consider the instance  $\langle \mathbf{I}(c), j^*, K \rangle$  of the Preprocessing Problem for  $\mathbf{P}$ . A *certificate* for a yes-instance is a pair  $(d, S)$  such that  $d$  is a cost vector,  $S \in \mathbf{F}$  and such that the following are true.

- i.  $j^* \in S$ ;

---

<sup>3</sup> In an assignment, the directions of the arcs do not matter.

- ii.  $\|c - d\|_\infty \leq K$ ;
- iii.  $d(S) \leq d(S')$  for all  $S' \in \mathbf{F}$ .

**Theorem 1.** *Suppose that the combinatorial 0-1 optimization problem  $\mathbf{P}$  is solvable in polynomial time. Then the Preprocessing Problem for Problem  $\mathbf{P}$  is in the class NP.*

**Proof.** Let  $\mathbf{I} = \min\{c(S) : S \in \mathbf{F}\}$  be an instance of problem  $\mathbf{P}$ , and let  $\langle \mathbf{I}(c), J^*, K \rangle$  be a yes-instance of the Preprocessing Problem for  $\mathbf{P}$ . It follows that there is a certificate  $(d, S)$  that can be verified in time that is polynomial in the size of the input and in the size needed to represent the vector  $d$ . So, it remains to show that there is also a certificate  $(d', S)$  where  $d'$  can be expressed in time polynomial in  $|J|$ ,  $\log K$ , and  $\log c_{max}$ , where  $c_{max} = \max\{c_j : j \in J\}$ .

Suppose that  $(d^*, S)$  is a certificate. We can choose another certificate  $(d', S)$  by letting  $d'$  be the solution to the following system of linear inequalities:

$$d_i - c_i \leq K \quad \text{for each } i \in J \tag{1a}$$

$$c_i - d_i \leq K \quad \text{for each } i \in J \tag{1b}$$

$$\sum_{i \in S} d_i \leq \sum_{i \in S'} d_i \quad \text{for all } S' \in \mathbf{F}. \tag{1c}$$

The number of variables in the linear system (1) is  $|J|$ . By standard results concerning the size of solutions of systems of linear inequalities, (see for example, Bertsimas and Tsitsiklis [7], page 375), there is a rational solution for the system (1) such that the size of the numerators and denominators are polynomial in  $|J|$ ,  $\log c_{max}$ , and  $\log K$ . ♦

### 3. HARDNESS RESULTS FOR SOME PREPROCESSING PROBLEMS

In this section, we establish the NP-completeness results for (1) the Shortest Path Problem, (2) the Minimum Cut Problem (3) the Minimum Cost Node Disjoint Cycle Problem, and (4) the Minimum Cost Arc Disjoint Cycle Problem.

All of the transformations in this section rely on the following NP-complete problem.

#### Restricted Path Problem.

INSTANCE. A network  $G = (N, A)$ , a designated source node  $s$ , a designated destination node  $t$ , and an arc  $(i, j)$ .

QUESTION. Is there some path from  $s$  to  $t$  that passes through arc  $(i, j)$ ?

**Theorem 2.** (Fortune, Hopcroft, and Wyllie). *The Restricted Path Problem is NP-complete.*

**Proof.** Fortune et al. [15] proved that the following problem is NP-complete. Given two origin nodes,  $s$  and  $j$ , and two destination nodes  $i$  and  $t$ , are there node-disjoint paths from  $s$  to  $i$  and from  $j$  to  $t$ ? One can immediately see that this problem is equivalent to the Restricted Path Problem. ♦

**Theorem 3.** *The Preprocessing Shortest  $s$ - $t$  Path Problem is NP-complete.*

**Proof.** The Preprocessing Shortest  $s$ - $t$  Path Problem is in the class NP by Theorem 1. To show its NP-completeness we carry out a transformation from the Restricted Path Problem. Let  $G = (N, A)$  and  $(i, j)$  be an instance of the Restricted Path Problem. The transformed instance of the Preprocessing Shortest  $s$ - $t$  Path Problem is as follows:

INSTANCE. A network  $G = (N, A)$  with  $n$  nodes, a cost vector  $c$  for which  $c_{kl} = 0$  for all  $(k, l) \in A$ , a designated source node  $s$  and destination node  $t$ , an arc  $(i, j)$ , and  $K = 0$ .

QUESTION. Is there a cost vector  $d$  and an  $s$ - $t$  path  $P$  such that  $(i, j) \in P$ ,  $\|c - d\|_\infty \leq 0$ , and  $d(P) \leq d(P')$  for any other  $s$ - $t$  path  $P'$ ?

It is easily seen that the instance of the preprocessing problem is a yes instance if and only if there is a feasible solution to the Restricted Shortest Path Problem. ♦

**Theorem 4.** *The Preprocessing Minimum  $s$ - $t$  Cut Problem is NP-complete.*

**Proof.** The Preprocessing Minimum  $s$ - $t$  Cut Problem is in the class NP by Theorem 1. We now show its NP-completeness via a transformation from the Restricted Path Problem. Let  $G = (N, A)$  and arc  $(i, j)$  be an instance of the Restricted Path Problem. We create an instance of the minimum cut problem on the same network  $G = (N, A)$  as follows:  $u_{ij} = 2$ ; and  $u_{kl} = 1$  for  $(k, l) \in A(i, j)$ .

Now consider the preprocessing instance of determining whether there is a cut  $[S, \bar{S}]$  and a vector  $u'$  of capacities with the following properties: (i)  $(i, j) \in (S, \bar{S})$ ; (ii)  $\|u - u'\|_\infty \leq 1$ , and (iii)  $u'(S, \bar{S}) \leq u'(T, \bar{T})$  for any other  $s$ - $t$  cut  $[T, \bar{T}]$ .

We will show that the answer is yes for the instance of the Restricted Path Problem if and only if the answer is yes for the preprocessing instance, and this will establish the NP-completeness of the preprocessing problem.

We first consider the case that the answer is yes for the Restricted Path Problem. Then there is an  $s$ - $t$  path  $P$  with  $(i, j) \in P$ . Let  $u'_{kl} = 1$  for each arc  $(k, l) \in P$ ; and let  $u'_{kl} = 0$  for each arc in  $A \setminus P$ . It follows that  $\|u - u'\|_\infty = 1$ , and so (ii) is satisfied. Let  $S$  consist of node  $i$  plus all of the nodes that precede node  $i$  on  $P$ . Then  $[S, \bar{S}]$  is an  $s$ - $t$  cut with  $(i, j) \in (S, \bar{S})$ , and so (i) is satisfied. Moreover,  $u'(S, \bar{S}) = 1$ , and  $u'(T, \bar{T}) \geq 1$  for every other  $s$ - $t$  cut  $[T, \bar{T}]$ . Thus the answer to the instance of the Preprocessing Minimum Cut Problem is yes.

We next consider the case that the answer for the instance of the preprocessing problem is yes. So,  $(i, j) \in (S, \bar{S})$ ;  $\|u - u'\|_\infty \leq 1$ , and  $(S, \bar{S})$  is an  $s$ - $t$  cut that minimizes  $u'(S, \bar{S})$ . In particular  $u'_{ij} \geq 1$ . Every maximum  $s$ - $t$  flow sends  $u'_{ij}$  units of flow in arc  $(i, j)$  since  $(i, j)$  is in some minimum cut (by the complementary slackness conditions of linear programming.) Let  $x^*$  be a maximum  $s$ - $t$  flow. Using flow decomposition (see pages 79-83 of Ahuja et al [1]), one can express  $x^*$  as the sum of flows on paths from  $s$  to  $t$  and the sum of flows around directed cycles. Let  $y^*$  be obtained from  $x^*$  by deleting the flows around directed cycles. Then  $y^*$  is also a maximum  $s$ - $t$  flow, and at least one of the  $s$ - $t$  paths  $P$  in the flow decomposition contains arc  $(i, j)$ . So the answer to the restricted path problem is yes, completing the proof.  $\blacklozenge$

**Theorem 5.** *The Preprocessing Minimum Cost Node Disjoint Cycle Problem and the Preprocessing Minimum Cost Arc Disjoint Cycle Problem are both NP-complete.*

**Proof.** The Preprocessing Minimum Cost Node Disjoint Cycle Problem and the Preprocessing Minimum Cost Arc Disjoint Cycle Problem are both in class NP by Theorem 1. We now show the NP-completeness of the Preprocessing Minimum Cost Node Disjoint Cycle Problem via a transformation from the Restricted Path Problem. Let  $G = (N, A)$  and arc  $(i, j)$  be the input for the restricted path problem.

We create a network  $G'$  for the Minimum Cost Node Disjoint Cycle Problem as follows:  $G' = (N, A')$ , where  $A' = A \cup (t, s)$ . We let  $c(t, s) = -2$ ,  $c(i, j) = 2$ , and  $c(k, l) = 1$  for  $(k, l) \in A \setminus (i, j)$ . Now consider the instance of the Preprocessing Minimum Cost Node Disjoint Cycle Problem of determining whether there is a collection  $S$  of arcs and a cost function  $d$  satisfying the following additional properties: (i)  $(i, j) \in S$ , (ii)  $\|c - d\|_\infty \leq 1$ , and (iii)  $d(S) \leq d(S')$  for any other subset  $S'$  of node disjoint cycles of  $G'$ .

We will show that the answer is yes for the Restricted Path Problem if and only if the answer is yes for this instance of the preprocessing problem. We consider first the case that the answer is yes for the restricted path problem. Let  $P$  be an  $s$ - $t$  path containing  $(i, j)$ . Let  $S = P \cup \{(t, s)\}$ .

Let  $d$  be defined as follows:  $d(i, j) = 1$ ;  $d(t, s) = -2$ ; for each arc  $(k, l) \in P \setminus (i, j)$ ,  $d(k, l) = 0$ ; and for each arc  $(k, l) \in A \setminus P$ ,  $d(k, l) = 2$ .

We note that conditions (i) and (ii) are both satisfied by  $(P, d)$ . Moreover,  $S$  is a single cycle with  $d(S) = -1$ . We will next show that for any other cycle  $C$  of  $G$ ,  $d(C) \geq 0$ . To see that, note that any other cycle  $C$  must contain at least one arc  $a$  of  $A \setminus S$ , and  $d(a) \geq 2$ , and so  $d(C) \geq 0$ . It follows that  $d(S') \geq -1$  for any collection of node disjoint cycles of  $G'$ . So, if the answer is yes for the Restricted Path instance, then the answer to the preprocessing instance is also yes.

We now consider the case that the answer is yes for the instance of the preprocessing problem. Let  $(S, d)$  be a certificate that satisfies (i) to (iii). Then  $S$  is the union of node disjoint cycles. Let  $C$  be the cycle in  $S$  containing  $(i, j)$ . We will prove via a contradiction that  $(t, s) \in C$ . So, suppose that  $(t, s) \notin C$ . But then  $d(C) \geq 1$  because  $d(i, j) \geq 1$ , and  $d(a) \geq 0$  for all arcs  $a \in C$ . But then  $d(S \setminus C) < d(S)$ , contradiction that  $d(S)$  is minimum. So, we conclude that  $(t, s) \in C$ , and thus  $C \setminus (t, s)$  is a certificate for the Restricted Path instance. This completes the proof of the NP-completeness of the Preprocessing Minimum Cost Node Disjoint Cycle Problem. The exact same transformation and proof also establishes the NP-completeness of the Preprocessing Minimum Cost Arc Disjoint Cycle Problem.  $\blacklozenge$

In the next section, we establish the NP-completeness of the Preprocessing Shortest Path Problem on acyclic networks and the Preprocessing Assignment Problem. These two NP-completeness proofs are substantially more intricate than the three transformations given here in Section 3.

There is a standard transformation from the Minimum Cost Node-Disjoint Cycle Problem to the Assignment Problem which involves node-splitting. This node splitting transformation, does not succeed in transforming the Preprocessing Minimum Cost Node-Disjoint Cycle Problem into the Preprocessing Assignment Problem, nor does it appear that the transformation be patched. The difficulty in using the same transformation arises from the following reason. The node splitting transformation from the Minimum Cost Node-Disjoint Cycle Problem introduces  $n$  additional arcs for the assignment problem. The usual transformation relies on each of these  $n$  arcs having a cost of 0. However, in preprocessing problems we perturb the costs of arcs, and we cannot ensure that the perturbed cost  $d(e) = 0$  for each newly introduced arc, and the transformation fails.

#### **4. THE PREPROCESSING SHORTEST PATH PROBLEM ON ACYCLIC NETWORKS AND THE PREPROCESSING ASSIGNMENT PROBLEM**

In this section, we establish the NP-completeness of the Preprocessing Shortest Path Problem on acyclic networks and the Preprocessing Assignment Problem via transformations from 3-Satisfiability, also called 3-SAT. Our definition of 3-SAT is adapted from Garey and Johnson [16].

### 3-SAT

INSTANCE: Collection  $B = \{B_1, B_2, \dots, B_m\}$  of clauses, each a conjunction of 3 literals, where the literals are from a set  $X = \{x_1, \dots, x_n\}$  of variables. (The  $B$  stands for Boolean.)

QUESTION: Is there a truth assignment for  $X$  that satisfies all of the clauses in  $B$ ?

A *truth assignment* is a vector  $x^*$  of variables. If  $x_i^* = 1$ , then variable  $x_i$  is true. If  $x_i^* = 0$ , then variable  $x_i$  is false. The *literals* associated with the variable  $x_i$  are  $x_i$  and  $\bar{x}_i$ . In an instance of 3-SAT, we let clause  $B_j = \{b_{j1}, b_{j2}, b_{j3}\}$ , where  $b_{jr}$  is a literal for  $r = 1, 2$  and 3. For example, if  $B_j = (x_3, \bar{x}_5, \bar{x}_9)$ , then  $b_{j1} = x_3, b_{j2} = \bar{x}_5$ , and  $b_{j3} = \bar{x}_9$ . In this case, clause  $B_j$  is *satisfied* by truth assignment  $x^*$  if  $x_3^* = 1$  or  $x_5^* = 0$  or  $x_9^* = 0$ . The collection  $B$  is satisfied if every clause in  $B$  is satisfied.

**Theorem 6.** *The Preprocessing Shortest Path Problem on an acyclic network is NP-complete.*

**Proof.** The Preprocessing Shortest Path Problem on an acyclic network is in the class NP by Theorem 1 (and Theorem 3). We show its NP-completeness via a transformation from 3-SAT.

Let  $B = \{B_1, B_2, \dots, B_m\}$  and  $X = \{x_1, \dots, x_n\}$  be an instance of 3-SAT where  $B_j = \{b_{j1}, b_{j2}, b_{j3}\}$  for  $j = 1$  to  $m$ . From this instance of 3-SAT, we create an instance  $G = (N, A)$  with costs  $c$  of a shortest path problem on an acyclic network. For each pair  $x_i, \bar{x}_i$  of literals, we create four nodes of  $N$  labeled  $s_i, t_i$ , and  $x_i, \bar{x}_i$ . For each  $i = 1$  to  $n$ , there are arcs  $(s_i, x_i), (s_i, \bar{x}_i), (x_i, t_i), (\bar{x}_i, t_i)$  in  $A$ , and the cost of each of these arcs is 1. We illustrate this construction in Figure 1a.

For each clause  $B_j = \{b_{j1}, b_{j2}, b_{j3}\}$ , we create 5 nodes in  $N$ , labeled  $s_{n+j}, t_{n+j}, b_{j1}, b_{j2}$ , and  $b_{j3}$ . We also create the following arcs:  $(s_{n+j}, b_{j1}), (s_{n+j}, b_{j2}), (s_{n+j}, b_{j3}), (b_{j1}, t_{n+j}), (b_{j2}, t_{n+j}),$  and  $(b_{j3}, t_{n+j})$ . The cost of each of these arcs is also 1. We illustrate this construction in Figure 2b.

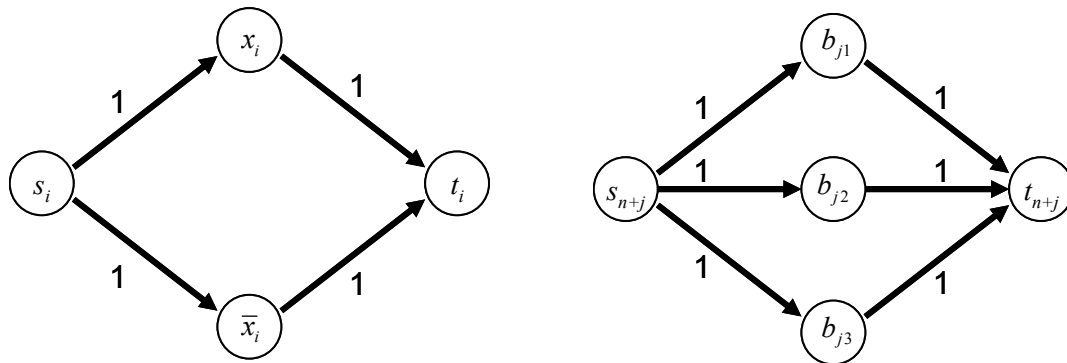


Figure 2a.

Figure 2b.

**Figure 2.** Figures 2a and 2b illustrates two subgraphs used in the construction in the proofs of Theorems 6 and 7. Figure 2a illustrates the four nodes associated with each pair of literals  $x_i \bar{x}_i$ . Figure 2b illustrates the subgraph with five nodes associated with the  $j$ -th clause.

For each  $j = 1$  to  $n + m - 1$ , there is an arc  $(t_j, s_{j+1})$  in  $A$  with  $c(t_j, s_{j+1}) = 1$ . Finally for each  $i = 1$  to  $n$  and for each  $j = 1$  to  $m$ , if  $b_{jr} = x_i$ , then there is an arc  $(\bar{x}_i, b_{jr})$  in  $A$ , with  $c(\bar{x}_i, b_{jr}) = -2$ . If  $b_{jr} = \bar{x}_i$ , then there is an arc  $(x_i, b_{jr}) \in A$  with  $c(x_i, b_{jr}) = -2$ . We refer to this collection of  $3m$  arcs (one for each literal  $b_{jr}$ ) as the *shortcut arcs*, and denote it as set  $A_{SC}$ .

The origin node for the shortest path problem is  $s_1$ . The destination node for this problem is  $t_{n+m}$ . Before proceeding further with the transformation, we make an observation, stated in the form of a lemma.

**Lemma 1.** *Every directed path  $P$  in  $G$  from node  $s_1$  to node  $t_{n+m}$  includes either arc  $(t_n, s_{n+1})$  or it includes exactly one shortcut arc.*

**Proof.** Let  $S = \bigcup_{i=1}^n \{s_i, x_i, \bar{x}_i, t_i\}$ . Then  $[S, \bar{S}]$  is cut that separates  $s_1$  from node  $t_{n+m}$ . The forward arcs of this cutset are  $(S, \bar{S}) = A_{SC} \cup \{(t_n, s_{n+1})\}$ , and there are no backward arcs; that is,  $(\bar{S}, S) = \emptyset$ . We claim that for any path  $P$  from  $s_1$  to node  $t_{n+m}$ ,  $|P \cap (S, \bar{S})| = 1$ . The proof will follow directly from this claim.

Note that for any cut  $[T, \bar{T}]$  separating  $s_1$  to node  $t_{n+m}$ , any path  $P$  from  $s_1$  to node  $t_{n+m}$  must contain exactly one more forward arc than backward arc in cut  $[T, \bar{T}]$ . Since  $[S, \bar{S}]$  has no backward arcs it follows that  $|P \cap (S, \bar{S})| = 1$ , completing the proof of the lemma.

We now return to the proof of Theorem 6. Consider the preprocessing instance of determining whether there is a path  $P$  from  $s_1$  to node  $t_{n+m}$  and a vector  $d$  of costs so that (i) path  $P$  contains  $(t_n, s_{n+1})$ , (ii)  $\|d - c\|_\infty \leq 1$ , and (iii)  $d(P) \leq d(Q)$  for any other path  $Q$  from  $s_1$  to node  $t_{n+m}$ . We will show that the answer is yes for this preprocessing instance if and only if the set of clauses is satisfiable, thus establishing the NP-completeness of the preprocessing problem.

We first consider the case that there is some satisfying truth assignment  $x^*$ . (There may be more than one.) To simplify notation, we reorder the literals in each clause so that  $b_{j1}$  is satisfied by  $x^*$  for each  $j = 1$  to  $m$ .

$$\text{Let } N^* = \{x_i : x_i^* = 1\} \cup \{\bar{x}_i : x_i^* = 0\} \cup \{b_{j1} : j = 1 \text{ to } m\}.$$

Note that every literal in  $N^*$  is true. Let  $P$  be the unique directed path of  $G$  from node  $s_1$  to node  $t_{n+m}$  that passes through each node of  $N^*$ . For each arc  $a \in A$ ,

$$d(a) = \begin{cases} 0 & \text{if } a \in P \\ -1 & \text{if } a \in A_{SC} \\ 1 & \text{otherwise.} \end{cases}$$

We now show that  $(P, d)$  is a certificate for the Preprocessing Shortest Path Problem. First, (i) is true because  $(t_n, s_{n+1}) \in P$ . Second (ii) is true because  $\|c - d\|_\infty \leq 1$ . We note that  $d(P) = 0$ . To show that (iii) is true, we will show that for any other path  $Q$  from  $s_1$  to  $t_{n+m}$ ,  $d(Q) \geq 0$ .

If  $Q$  has no shortcut arcs, then  $d(Q) \geq 0$ . So, suppose that  $Q$  has a shortcut arc. We will show that  $Q$  has at least one arc whose cost is 1, and so  $d(Q) \geq 0$ . By Lemma 1,  $Q$  has exactly one shortcut arc. Consider first the case that arc  $(x_i, b_{j1})$  is the shortcut arc of  $Q$ . By construction of the shortcut arcs,  $b_{j1} = \bar{x}_i$ , and  $x_i^* = 0$ . But then  $(s_i, x_i) \in Q$  since it is the only arc directed into  $x_i$ . Therefore,  $d(Q) \geq d(s_i, x_i) + d(x_i, b_{j1}) = 0$ , which is what we wanted to prove. We next consider the case that the shortcut arc of  $Q$  is the arc  $(\bar{x}_i, b_{j1})$ . Then  $b_{j1} = x_i$ , and  $x_i^* = 1$ , and  $d(Q) \geq d(s_i, \bar{x}_i) + d(\bar{x}_i, b_{j1}) = 0$ . Finally, if the shortcut arc of  $Q$  is arc  $(v, b_{jr})$  for  $r = 2$  or  $3$ , then  $d(Q) \geq d(v, b_{jr}) + d(b_{jr}, t_j) = 0$ . In all three cases for shortcut arcs,  $d(Q) \geq 0$ , and thus  $d(Q) \geq d(P)$ , completing the proof that the answer to the preprocessing problem is yes.

We now consider the case that the answer for this instance of the preprocessing problem is no. Suppose that there is a shortest path  $P$  with respect to costs  $d$  from node  $s_1$  to node  $t_{m+n}$  that passes through arc  $(t_n, s_{n+1})$ . By Lemma 1, the path  $P$  does not contain any shortcut arc. We now define the truth assignment  $x^*$  as follows: if node  $x_i \in P$ , let variable  $x_i^* = 1$ . If node  $\bar{x}_i \in P$ , let variable  $x_i^* = 0$ . We will show that  $x^*$  satisfies each of the clauses, and thus the instance of 3-SAT is a yes-instance. To see that  $x^*$  satisfies each of the clauses, consider clause  $B_j$ , and suppose that node  $b_{jr} \in P$ , where  $b_{jr} = \bar{x}_i$ . (The case that  $b_{jr} = x_i$  is proved in the same manner.) By construction,  $(x_i, b_{jr})$  is a shortcut arc, and  $d(x_i, b_{jr}) \leq -1$  because  $c(x_i, b_{jr}) = -2$ , and  $\|d - c\|_\infty \leq 1$ . If  $x_i \in P$ , then we could obtain a strictly shorter length path than  $P$  by replacing the subpath in  $P$  from  $x_i$  to  $b_{jr}$  by the shortcut arc  $(x_i, b_{jr})$ , contradicting that  $P$  is a minimum cost path. So,  $x_i \notin P$ , and by our choice of variables,  $x_i^* = 0$ , and the clause  $B_j$  is satisfied. Similarly, one can show that if  $b_{jr} = x_i$ , then  $x_i^* = 1$ , and the clause  $B_j$  is satisfied. We conclude that all clauses are satisfied, completing the proof.  $\blacklozenge$

## The Assignment Problem and Node-Splitting.

The transformation in the proof in Theorem 6 is the basis for transformation in the NP-completeness proof of Theorem 7 (below) as well. The latter transformation also includes “node-splitting.” Prior to stating Theorem 7 and its proof, we prove a lemma concerning “node-splitting”.

Consider an acyclic network  $G = (N, A)$  with costs  $c$  and with a designated source node  $s$  and destination node  $t$ . We create a network  $G' = (N', A')$  by “splitting” all nodes except  $s$  and  $t$ , and relabeling nodes  $s$  and  $t$ .

1.  $N' = \{s'', t'\} \cup \{i', i'' : i \in N \setminus \{s, t\}\}$
2.  $A' = A'_O \cup A'_{NS}$ , where  $A'_{NS} = \{(i', i'') : i \in N \setminus \{s, t\}\}$  is the collection of *node-splitting arcs* and  $A'_O = \{(i'', j') : (i, j) \in A\}$ .

Node-splitting is a standard transformation used in network optimization. See for example, Ahuja et al [1993, pages 41-43].

We let the symbol  $\oplus$  denote the *symmetric difference* operator. For any two subsets  $S$  and  $T$ ,  $S \oplus T$  are the elements in  $S$  or  $T$  but not both. Equivalently,  $S \oplus T = S \setminus T \cup T \setminus S$ .

**Lemma 2.** *Let  $G = (N, A)$  be an acyclic network with source node  $s$  and destination node  $t$ . Let  $G' = (N', A')$  be obtained from  $G$  by splitting all nodes in  $N \setminus \{s, t\}$ . Then there is a 1-1 correspondence between directed  $s$ - $t$  paths in  $G$  and assignments in  $G'$ .*

**Proof.** Suppose  $P$  is an  $s$ - $t$  path in  $G$ . Let  $P'$  be the corresponding path from  $s''$  to  $t'$  in  $G'$ . Let  $A'_{NS}$  be the set of node-splitting arcs of  $G'$ . We will show that  $M = P' \oplus A'_{NS}$  is an assignment of  $G'$ . We first note that exactly one arc of  $P'$  is incident to node  $s''$ , and thus one arc of  $M$  is incident to node  $s''$ . Also, one node of  $P'$  is incident to node  $t'$ , and thus one arc of  $M$  is incident to node  $t'$ . We now consider nodes  $i'$  and  $i''$ , and will show that each of these nodes is incident to one arc of  $M$ . If  $i \notin P \setminus \{s, t\}$ , then neither  $i'$  nor  $i''$  is a node of  $P'$ , and thus  $i'$  and  $i''$  are each only incident to  $(i', i'')$  in  $M$ . If  $i \in P \setminus \{s, t\}$ , then there is a subpath  $j, i, k$  in  $P$ , and  $(j'', i')$ ,  $(i', i'')$ , and  $(i'', k')$  are the arcs incident to  $i'$  and  $i''$  in  $P'$ . Then  $(j'', i')$  is the only arc of  $M$  that is incident to  $i'$  and  $(i'', k')$  is the only arc of  $M$  that is incident to  $i''$ . So, each node of  $N'$  is incident to exactly one arc of  $M$ , and  $M$  is an assignment.

We now consider the case that  $M$  is an assignment of  $G'$ . Let  $S = M \oplus A'_{NS}$ . We will show that  $S$  is a directed path from  $s''$  to  $t'$  in  $G'$ , and thus it induces a unique directed path from  $s$  to  $t$  in  $G$ . More precisely, we will show (i)  $s''$  has one outgoing arc in  $S$ , (ii)  $t'$  has one incoming arc in  $S$ , and (iii) every other node either has no incident arcs or it has one incoming arc and one outgoing arc. It will follow from flow decomposition theory that  $S$  may be decomposed into a directed path from node  $s''$  to node  $t'$  plus the flow around directed cycles. But  $G'$  is acyclic, and so  $S$  is a directed path from node  $s''$  to node  $t'$ .

The assignment  $M$  has one incident arc  $a$  to  $s''$ , and it must be outgoing. So  $a \in S$  and (i) is true. The assignment  $M$  has one incident arc to  $t'$ , and it must be incoming. So  $S$  has the same arc incoming to  $t'$ , and (ii) is true.

Next, consider nodes  $i'$  and  $i''$  and condition (iii). If  $(i', i'') \in M$ , then neither node  $i'$  nor node  $i''$  is incident to any arc of  $S$ , and (iii) is true in this case for both node  $i'$  and node  $i''$ . If  $(i', i'') \notin M$ , then  $i'$  is incident to an arc  $(j'', i') \in M$  for some node  $j \in N$ , and  $i''$  is incident to arc  $(i'', k')$  for some  $k \in N$ . (The arcs of the form  $(j'', i')$  are the only arcs incident to  $i'$  other than  $(i', i'')$ .) Thus  $(j'', i') \in S$  and  $(i', i'') \in S$ , and  $(i'', k') \in S$ . In this case (iii) is also true. Since (iii) is true in both cases, the theorem is true.  $\blacklozenge$

**Theorem 7.** *The Preprocessing Assignment Problem is NP-complete.*

**Proof.** The Preprocessing Assignment Problem is in the class NP by Theorem 1. In order to establish its NP-completeness we modify the transformation from the 3-SAT instance given in the proof of Theorem 6.

Let  $B = \{B_1, B_2, \dots, B_m\}$ ,  $X = \{x_1, \dots, x_n\}$  be an instance of 3-SAT where  $B_j = \{b_{j1}, b_{j2}, b_{j3}\}$  for  $j = 1$  to  $m$ . Let  $G = (N, A)$  with costs  $c$  be the instance of the shortest path problem on an acyclic network as created in the proof of Theorem 6, and as illustrated in part in Figure 1. Let  $G' = (N', A')$  be obtained from  $G$  by performing node-splitting on all nodes in  $N$  except for node  $s_1$  and node  $t_{n+m}$ . We let  $s_1''$  denote the source node of  $G'$ , and we let  $t_{n+m}'$  denote the destination node. Let  $A'_{NS}$  be the node-splitting arcs of  $A'$ , and let  $A'_{SC}$  denote the shortcut arcs of  $A'$ .

For each node-splitting arc  $a$ , let  $c'(a) = -1$ . If  $a \in A_0$ , and if  $a'$  is the corresponding arc of  $A'_0$ , then  $c'(a') = c(a)$ . For example, for all shortcut arcs  $a'$  of  $A'_0$ ,  $c'(a') = -2$ .

Consider the preprocessing instance of determining whether there is an assignment  $M$  and a vector  $d$  of costs so that (i)  $(t_n'', s_{n+1}') \in M$ , (ii)  $\|d - c'\|_\infty \leq 1$ , and (iii)  $d(M) \leq d(Q)$  for any other assignment  $Q$  of  $G'$ . We will show that the answer is yes for this preprocessing instance if and only if the set of clauses is satisfiable, thus establishing the NP-completeness of the preprocessing problem.

We first consider the case that there is a truth assignment  $x^*$  that satisfies all of the clauses. Let  $P$  be a directed path of  $G$  from node  $s_1$  to node  $t_{n+m}$  that passes through  $n + m$  true literals, as per the proof of Theorem 6. Let  $P'$  be the corresponding path from  $s_1''$  to node  $t_{n+m}'$  in  $G'$ .

$$\text{Let } d(i'', j') = \begin{cases} 0 & \text{if } (i, j) \in P \\ -1 & \text{if } (i, j) \in A_{SC} \\ 1 & \text{otherwise.} \end{cases}$$

Let  $d(i', i'') = 0$  for  $(i', i'') \in A'_{NS}$ .

By Lemma 2,  $M = P' \oplus A'_{NS}$  is an assignment of  $G'$ . We will show that  $M$  and  $d$  satisfy conditions (i) to (iii) above. Since  $P'$  contains no shortcut arcs, it follows that  $(t''_n, s'_{n+1}) \in P'$ , and thus  $(t''_n, s'_{n+1}) \in M$ , and (i) is true. It is obvious that (ii) is true. We now show that (iii) is also true.

Let  $Q$  be another assignment. Then, by Lemma 2,  $\hat{P} = Q \oplus A'_{NS}$  is a directed path from node  $s_1$  to node  $t_{n+m}$ . We claim that  $d(Q) = d(Q \oplus A'_{NS}) = d(\hat{P}) \geq d(P') = d(M)$ , and so  $d(Q) \geq d(M)$ . The first equality is true because all node-splitting arcs have a cost of 0. The inequality is true because  $P'$  is a shortest path from node  $s_1$  to node  $t_{n+m}$  with respect to costs  $d$  as established in the proof of Theorem 6. Thus a yes-instance for the 3-satisfiability Problem leads to a yes-instance for the Preprocessing Assignment Problem.

We now consider the case that the instance of the preprocessing problem is a yes instance. Let  $M$  be an assignment and  $d$  be the costs, where  $(M, d)$  satisfies conditions (i) to (iii). Let  $P' = M \oplus Q$ . Then by Lemma 2,  $P'$  is a path from  $s_1$  to node  $t_{n+m}$ . By (i),  $(t_n, s_{n+1}) \in M$ , and so  $(t_n, s_{n+1}) \in P'$ . By Lemma 1,  $P'$  has no shortcut arcs. Let  $P$  be the corresponding path in  $G$ , and let  $x^*$  be the induced truth assignment, as in the proof of Theorem 6. We will show via a proof by contradiction that  $x^*$  satisfies  $B$ , and thus the instance of 3-SAT is a yes instance, completing the proof.

We suppose that  $x^*$  does not satisfy  $B$  and will derive a contradiction. Suppose that  $b'_{jr} \in P'$ , and that  $b'_{jr}$  is not satisfied. So, suppose that  $b_{jr} = x_i$ , and  $x_i^* = 0$ . (The case that  $b_{jr} = \bar{x}_i$  is proved in the same manner.) Then  $\bar{x}''_i \in P'$ , and  $(\bar{x}''_i, b'_{jr})$  is a shortcut arc, and  $d(\bar{x}''_i, b'_{jr}) \leq -1$ . Let  $\hat{P}$  be obtained from  $P'$  by deleting the subpath in  $P'$  from  $\bar{x}''_i$  to  $b'_{jr}$ , and adding the shortcut arc  $(\bar{x}''_i, b'_{jr})$ . Let  $\hat{M} = \hat{P} \oplus A'_{NS}$ . By Lemma 2,  $\hat{M}$  is an assignment. We will next show that  $d(\hat{M}) < d(M)$ , which will be a contradiction. We note the following relationships.

1.  $M \cap A'_{NS} \subseteq \hat{M} \cap A'_{NS}$  because  $\hat{P} \cap A'_{NS} \subseteq P' \cap A'_{NS}$ ;
2.  $M \cap A'_o \subseteq \hat{M} \cap A'_o$  because  $P' \cap A'_o \subseteq \hat{P} \cap A'_o$ ;

From (1) and the fact that  $d(a) \leq 0$  for  $a \in A'_{NS}$ , it follows that  $d(M \cap A'_{NS}) - d(\hat{M} \cap A'_{NS}) \geq 0$ .

From (2) and the fact that  $d(a) \geq 0$  for  $a \in A'_o$ , it follows that  $d(M \cap A'_o) - d(\hat{M} \cap A'_o) \geq 0$ .

Thus,  $d(M) - d(\hat{M}) = [d(M \cap A'_{NS}) - d(\hat{M} \cap A'_{NS})] + [d(\hat{M} \cap A'_O) - d(M \cap A'_O)] - d(\bar{x}_i, b'_{jr}) \geq 1$ , and so  $M$  is not a minimum cost assignment, contrary to assumption. This completes the proof.  $\blacklozenge$

## 5. THE PREPROCESSING MINIMUM COST BASIS OF A MATROID PROBLEM.

Let  $\mathbf{I}$  denote an instance of  $\mathbf{P}$  with feasible set  $\mathbf{F}$ , and let  $c$  be the cost vector. So,  $\mathbf{I} = \min\{c(S) : S \in \mathbf{F}\}$ . The feasible collection of sets  $\mathbf{F}$  is called a *matroid* if the following three axioms are satisfied.

1.  $\emptyset \in \mathbf{F}$ ;
2. if  $S \in \mathbf{F}$ , and if  $T \subseteq S$ , then  $T \in \mathbf{F}$ ; and
3. if  $S \in \mathbf{F}$ , and if  $T \in \mathbf{F}$ , and if  $|T| > |S|$ , then there is an element  $e \in T \setminus S$  such that  $S \cup \{e\} \in \mathbf{F}$ .

Matroids have been widely studied (see, for example, Lawler [25]) and have a wide range of applications in combinatorial optimization. Edmonds [14] showed that the greedy algorithm solved matroids, and that matroids could be expressed in terms of the greedy algorithm.

If  $S \in \mathbf{F}$ , then  $S$  is said to be *independent*. Otherwise,  $S$  is said to be *dependent*. A maximum cardinality independent set is called a *basis* of the matroid. A *circuit*  $C$  is a subset of  $J$  that is minimally dependent, that is,  $C$  is dependent and no subset of  $C$  is dependent. For any subset  $S \subseteq J$ , the rank of  $S$  is the cardinality of the largest independent set in  $S$ , and we denote it as  $r(S)$ .

**Matroid Property 1.** *Suppose that  $(J, \mathbf{F})$  is a matroid. Suppose further that  $S \in \mathbf{F}$ , and  $S \cup \{j\} \notin \mathbf{F}$ . Then there is a unique circuit in  $S \cup \{j\}$ .*

**Matroid Property 2.** *Suppose that  $(J, \mathbf{F})$  is a matroid and that  $r(\cdot)$  denotes the rank function. The  $r(\cdot)$  is submodular. That is for any two subsets  $S$  and  $T$  of  $J$ ,  $r(S) + r(T) \geq r(S \cup T) + r(S \cap T)$ .*

Matroid Properties 1 and 2 are both well known.

We now consider the following generalization of the Preprocessing Problem for Problem  $\mathbf{P}$ .

### The Generalized Preprocessing Problem for $\mathbf{P}$ .

INSTANCE. An instance  $\mathbf{I} = \min\{c(S) : S \in \mathbf{F}\}$  of  $\mathbf{P}$ , a subset  $J'$  of  $J$ , a vector  $w$  of weights, and an integer  $K$ .

QUESTION. If there a subset  $S \in \mathbf{F}$  and a cost vector  $d$  such that

- i.  $J' \subseteq S$ ;

- ii.  $\|w(d - c)\|_\infty \leq K$ ;
- iii.  $d(S) \leq d(S')$  for all  $S' \in \mathbf{F}$ .

**Algorithm 1. The Generalized Preprocessing Problem for the Minimum Cost Basis of a Matroid.**

**begin**

$d'_i := c_i - K/w_i$  for all  $i \in J'$ ;

$d'_i := c_i + K/w_i$  for all  $i \notin J'$ ;

relabel the elements of  $J$  in non-decreasing order of  $d'_i$  (in case of a tie, choose elements of  $J'$  first);

$S' := \emptyset$ ;

**for**  $i = 1$  to  $n$ , **if**  $S' \cup \{i\} \in \mathbf{F}$ , **then**  $S' := S' \cup \{i\}$ ;

**if**  $S'$  contains  $J'$ , **then** the answer to Generalized Preprocessing Problem is yes; **else** the answer is no;

**end**

**Theorem 8.** *Algorithm 1 solves the Generalized Preprocessing Minimum Cost Matroid Basis Problem in polynomial time.*

**Proof.** Algorithm 1 first reorders the elements of  $J$  in non-decreasing order of their costs with respect to  $d'$  and then runs the greedy algorithm. As shown by Edmonds [14], the set  $S'$  is a minimum cost basis with respect to costs  $d'$  at the end of the algorithm.

If  $J' \subseteq S'$ , the answer is yes for the Generalized Preprocessing Problem. So, it remains to prove that the answer is no whenever  $J' \not\subseteq S'$ . Let  $q$  be the first element of  $J'$  that is not in  $S'$ . Let  $C$  be the unique circuit created by adding  $q$  to the existing independent set at iteration  $q$  of the for-loop of Algorithm 1. Then for any  $i \in C \setminus J'$ ,  $d'_i < d'_q$ , and thus  $c_i < c_q - K/w_i - K/w_q$ . Then for any cost function  $d$  with  $\|w(d - c)\|_\infty \leq K$ , it must be true that for any  $i \in C \setminus J'$ ,  $d_i < d_q$ .

If  $J'$  is part of some optimal basis with respect to  $d$ , then there must be an ordering of the elements of  $J$  such that the ordering is non-decreasing in costs with respect to  $d$ , and such that the greedy algorithm puts each element of  $J'$  in the basis. So consider the greedy algorithm as run with respect to costs  $d$ , and let  $k$  be the last element of  $C$  to be considered by the greedy algorithm. We will show that  $k \in J'$  and that  $k$  is not part of the minimum cost basis created by the greedy algorithm, and so the theorem is true. To see that  $k \in J'$ , note that for any  $i \in C \setminus J'$ ,  $d_i < d_j \leq d_k$ . To see that  $k$  is not part of the minimum cost basis selected by the greedy algorithm, let  $S$  be the elements considered strictly prior to  $k$  in running the greedy algorithm with respect to  $d$ . Then  $r(S \cup k) - r(S) \leq r(C) - r(C \setminus k) = 0$ , where the inequality follows by the submodularity of  $r$ . So  $k$  will not be added to  $S'$  by the greedy algorithm. ♦

Algorithm 1 can be used to solve the Weighted Minimax Inverse Minimum Cost Spanning Tree Problem, which was also solved by Ahuja and Orlin [3]. However, it does not solve the Weighted Minimax Partial Inverse Minimum Cost Spanning Tree Problem, which is shown to be NP-complete in Orlin **Error! Reference source not found.** Of related interest are polynomial time algorithms for the Inverse Minimum Cost Spanning Tree Problem by Ahuja, Orlin, and Sokkalingam [5], and Ahuja and Orlin [3], and Dell'Amico, Maffioli, and Malucelli [11].

## 6. THE PREPROCESSING MINIMUM COST CLOSURE PROBLEM.

In this section we give a polynomial time algorithm for the Generalized Preprocessing Minimum Cost Closure Problem. More generally, we solve the Generalized Preprocessing Minimum Cost Element of a Lattice Problem.

Let  $\mathbf{I}$  denote an instance of  $\mathbf{P}$  with  $\mathbf{F}$  as the set of feasible solutions and  $c$  as the cost vector; that is,  $\mathbf{I} = \min\{c(S) : S \in \mathbf{F}\}$ . The feasible collection of sets  $\mathbf{F}$  is called a *lattice* if the following two axioms are satisfied.

1. If  $S \in \mathbf{F}$  and if  $T \in \mathbf{F}$ , then  $S \cup T \in \mathbf{F}$ .
2. If  $S \in \mathbf{F}$  and if  $T \in \mathbf{F}$ , then  $S \cap T \in \mathbf{F}$ .

Lattices have been widely studied. See for example, Birkhoff [8].

We next show that the collection of closures in a directed network  $G = (N, A)$  is a lattice. That is, we will show that if  $S \subseteq N$  is a closed set and if  $T \subseteq N$  is a closed set, then so are  $S \cap T$  and  $S \cup T$ . Suppose that  $S \in \mathbf{F}$ ,  $T \in \mathbf{F}$ , and  $i \in S \cup T$ . If  $(i, j) \in A$ , then either  $i \in S$ , (in which case  $j \in S$ ) or else  $i \in T$  (in which case  $j \in T$ ). In both cases,  $j \in S \cup T$ , and so we have shown that  $S \cup T$  is a closed set.

Suppose next that  $S \in \mathbf{F}$ ,  $T \in \mathbf{F}$ , and  $i \in S \cap T$ . If  $(i, j) \in A$ , then  $j \in S$  and  $j \in T$ , and so  $j \in S \cap T$ . We have thus shown that  $S \cap T$  is a closed set. So the set of closures in a directed network is a lattice.

### The Generalized Preprocessing Minimum Cost Element of a Lattice.

INSTANCE. An instance  $\mathbf{I} = \min(c(S) : S \in \mathbf{F})$ , where  $\mathbf{F}$  is a lattice over  $J$ , a subset  $J' \subseteq J$ , and an integer  $K$ .

QUESTION. If there a subset  $S \in \mathbf{F}$  and a cost vector  $d$  such that

- i.  $J' \subseteq S$ ;
- ii.  $\|w(d - c)\|_\infty \leq K$ ; and

iii.  $d(S) \leq d(S')$  for all  $S' \in \mathbf{F}$ .

**Algorithm 2. Generalized Preprocessing Minimum Cost Element of a Lattice Problem.**

**begin**

$d'_i := c_i - K/w_i$  for all  $i \in N$ ;

find an element  $S' \in \mathbf{F}$  such that  $d'(S')$  is minimum (in case of alternate optima, choose a minimum cost element  $S'$  of maximum cardinality);

**if**  $J' \subseteq S'$ ; **then** the answer to the generalized preprocessing minimum cost closure problem is yes;

**else**, there is no assignment  $d$  of weights so that a minimum cost closure of  $\mathbf{I}(d)$  contains  $J'$ ;

**end**

**Theorem 9.** *Algorithm 2 solves Generalized Preprocessing Minimum Cost Element of a Lattice Problem.*

**Proof.** Let  $S'$  be the minimum cost element of  $\mathbf{F}$  with respect to costs  $d'$  as determined in Algorithm 2. If  $J' \subseteq S'$ , the answer is clearly yes for the Generalized Preprocessing Problem.

Suppose conversely that the answer is yes for the Generalized Preprocessing Problem, and let  $(S, d)$  be a certificate such that that (i)-(iii) above are satisfied. We will next show that  $d'(S \cup S') \leq d'(S')$ , which by our selection of  $S'$  in Algorithm 2 implies that  $(S \cup S') = S'$ , and so the algorithm will produce an answer of yes as well. We claim that

$$d'(S \cup S') = d'(S') + d'(S \setminus S') \leq d'(S') + d(S \setminus S') \leq d'(S').$$

The first inequality is true because  $d' \leq d$ . The second inequality is true because  $d(S \setminus S') = d(S) - d(S \cap S')$ , and  $d(S) \leq d(S \cap S')$  because  $S \cap S' \in \mathbf{F}$ , and by condition (iii)  $S$  is optimal with respect to  $d$ . ♦

We note that Algorithm 2 solves the Generalized Preprocessing Minimum Cost Closure Problem in the time it takes to solve the Minimum Cost Closure Problem, which is polynomial, as implied by Johnson [1968], who modeled the Minimum Cost Closure Problem as a network flow problem.

## 7. SUMMARY AND CONCLUSIONS

We have shown that the preprocessing problem is often much more difficult than inverse optimization problems. We have shown the NP-completeness of the Preprocessing Shortest Path Problem

on acyclic networks, the Preprocessing Assignment Problem, the Preprocessing Minimum Cut Problem, and the Preprocessing Minimum Cost Node Disjoint (and Arc Disjoint) Cycle Problem. We have shown that the Preprocessing Problem reduces to the original problem, whenever the feasible sets forms a matroid or greedoid.

## Acknowledgments

The first author gratefully acknowledges support from the Fulbright Foundation. The second author gratefully acknowledges partial support from ONR grant ONR N00014-98-1-0317, and from NSF grant DMI-0217123.

## References.

- [1] R. K. Ahuja, T.L. Magnanti, and J.B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, New-Jersey, 1993.
- [2] R. K. Ahuja and J. B. Orlin, Combinatorial algorithms for inverse network flow problems. Working Paper, Sloan School of Management, MIT, Cambridge, MA., 1998.
- [3] R. K. Ahuja and J. B. Orlin, 2001 A Faster Algorithm for the Inverse Spanning Tree Problem. *Journal of Algorithms* **34** (2001) 177-193.
- [4] R. K. Ahuja and J. B. Orlin, Inverse Optimization, Part I: Linear Programming and General Problem. *Operations Research* **35** (2001) 771-783.
- [5] R. K. Ahuja, J. B. Orlin, and P. T. Soddalingam, Solving inverse spanning tree problems through network flow techniques. *Operations Research* **47** (1999) 291-300.
- [6] D. Beil and L. Wein, An Inverse-optimization-based auction mechanism to support a multi-attribute RFQ Process, to appear in *Management Science*, MIT Technical Report, 2002.
- [7] D. P. Bertsimas and J.N. Tsitsiklis, *Introduction to Linear Optimization*, Athena Scientific, Belmont, MA. 1997
- [8] G. D. Birkhoff. *Lattice Theory*, AMS Colloq. Public. Vol. XXV, 3rd ed. 1967,
- [9] Burton, D., and Ph. L. Toint, On an instance of the inverse shortest paths problem. *Mathematical Programming* **53** (1992) 45-61.
- [10] Burton, D., and Ph. L. Toint, On the use of an inverse shortest paths algorithm for recovering linearly correlated costs. *Mathematical Programming* **63** (1994) 1-22.
- [11] M.Dell'Amico, F. Maffioli and F. Malucelli, The base-matroid and inverse combinatorial optimization problems. *Discrete Applied Mathematics* **128** (2003) 337-353.
- [12] R. B. Dial, Minimal-revenue congesting pricing part I: a fast algorithm for the single origin case. *Transportation Res. Part B* **33** (1999) 189-202.
- [13] R. B. Dial, Minimal-revenue congesting pricing part II: an efficient algorithm for the general case. *Transportation Res. Part B* **34** (2000) 645-665.

- [14] J. Edmonds, Matroids and the Greedy Algorithm, *Mathematical Programming* **1** (1971) 127-136.
- [15] S. Fortune, J. Hopcroft, and J. Wyllie, The directed subgraph homeomorphism problem. *Theoretical Computer Science* **10** (1980), 111-121.
- [16] M. S. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, 1979.
- [17] S. Gentry, Partial Inverse Linear Programming. Technical Report. Massachusetts Institute of Technology, 2001.
- [18] S. Gentry, V. Saligrama, E. Feron, Dynamic Inverse Optimization. *Proceedings of the American Control Conference* (2001) Arlington, VA.
- [19] P. E. Green and V. Srinivasan, Conjoint Analysis in Consumer Research: Issues and Outlook, *Journal of Consumer Research* **5** (1978) 103-123.
- [20] C. Heuberger, Inverse Combinatorial Optimization: A Survey on Problems, Methods, and Results. Technical Report. University of Graz. 2002.
- [21] T. B. Johnson, Optimum pit mine production scheduling. Technical Report. University of CA at Berkeley. 1968
- [22] T.-C. Lai and J.B. Orlin, Identifying parts of an optimal solution before seeing the costs: NP-Completeness and general results. Working paper, Sloan School of Management, MIT. 1999.
- [23] T.-C. Lai and Y N Sotskov, Sequencing with uncertain numerical data for makespan minimization. *Journal of the Operational Research Society* **50** (1999) 230-243.
- [24] Lai, T-C, Y. N. Sotskov, N.Y. Sotskova, and F. Werner, Optimal makespan scheduling with given bounds of processing times. *Math Comput Modelling* **26** (1997) 67-86.
- [25] E. L. Lawler, *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart, and Winston, New York. 1976.
- [26] V. Srinivasan, A general procedure for estimating consumer preference distributions, *Journal of Marketing Research*, **12** (1975) 377-389.
- [27] Q. Wei, J. Zhang, and X. Zhang, An inverse DEA model for inputs/outputs estimate, *Eur. J. Oper. Res.* **121** (2000) 151-163.
- [28] S. Xu and J. Zhang, An inverse problem of the weighted shortest path problem. *Japanese Journal of Industrial and Applied Mathematics* **12** (1995) 47-59.
- [29] X. Yang, The Complexity of Partial Inverse Assignment Problem and Partial Inverse Cut Problem. *RAIRO Operations Research* **35** (2001) 117-126.
- [30] C. Yang and J. Zhang, Inverse maximum capacity path with upper bound constraints. *OR Spektrum* **20** (1998) 97-100.
- [31] Yang, C., J. Zhang, and Z. Ma. Inverse maximum flow and minimum cut problem. *Optimization* **40** (1997) 147-170.
- [32] Zhang, J., and M. Cai. Inverse problem of minimum cuts. *ZOR Mathematical Methods of Operations Research* **48** (1998) 51-58.
- [33] Zhang, J., and Z. Liu. Calculating some inverse linear programming problem. *Journal of Computational and Applied Mathematics* **72** (1996) 261-273.

