Theoretical and Computational Advances for Network Diversion

Christopher A. Cullenbine
Division of Economics and Business, Colorado School of Mines, Golden, Colorado 80401

R. Kevin Wood
Operations Research Department, Naval Postgraduate School, Monterey, California 93943

Alexandra M. Newman
Division of Economics and Business, Colorado School of Mines, Golden, Colorado 80401

The network-diversion problem (ND) is defined on a directed or undirected graph $G = (V, E)$ having nonnegative edge weights, a source vertex s, a sink vertex t, and a "diversion edge" e'. This problem, with intelligence-gathering and war-fighting applications, seeks a minimum-weight, minimal s-t cut $E_C \subseteq E$ in G such that $e' \in E_C$. We present (a) a new NP-completeness proof for ND on directed graphs, (b) the first polynomial-time solution algorithm for a special graph topology, (c) an improved mixed-integer programming formulation (MIP), and (d) useful valid inequalities for that MIP. The proof strengthens known results by showing, for instance, that ND is strongly NP-complete on a directed graph even when e' is incident from s or into t, but not both, and even when G is acyclic; a corollary shows the NP-completeness of a vertex-deletion version of ND on undirected graphs. The polynomial-time algorithm solves ND on s-t planar graphs. Compared to a MIP from the literature, the new MIP, coupled with valid inequalities, reduces the average duality gap by 10–50% on certain classes of test problems. It can also reduce solution times by an order of magnitude. We successfully solve unweighted problems with roughly 90,000 vertices and 360,000 edges and weighted problems with roughly 10,000 vertices and 40,000 edges. **NETWORKS, Vol. 000(00), 000–000 2013**

Keywords: network diversion; minimum cut; complexity; mixed-integer programming; valid inequality

Received July 2012; accepted June 2013
Correspondence to: R.K.Wood; e-mail: kwood@nps.edu
Contract grant sponsors: Office of Naval Research, Air Force Office of Scientific Research, and Defense Threat Reduction Agency (R.K.W.)
DOI 10.1002/net.21514
Published online in Wiley Online Library (wileyonlinelibrary.com).

1. INTRODUCTION

1.1. The Network Diversion Problem

The network-diversion problem (ND) arises in the context of war-fighting and intelligence-gathering (Curet [12], Cintron-Arias et al. [9]). An "interdictor" seeks to manipulate the routing choices of an adversary, the "network user". The network user wishes to traverse (or communicate across) an s-t path in a directed or undirected network $G = (V, E)$. The interdictor understands the network user's goal and seeks to force the network user to traverse (or communicate across) a special diversion edge $e' = (i', j') \in E$, where the network user is especially susceptible to physical attack (or where his communications can be intercepted). To accomplish this, the interdictor can attack and destroy, that is, "interdict," a subset of edges $E_D \not\supset \{e'\}$, which prohibits the use of those edges by the network user. But, interdicting an edge requires effort, and the interdictor prefers to use minimum total effort to accomplish his task. To this end, he defines edge weights $w_e > 0$ for all $e \in E \backslash \{e'\}$, which correspond to interdiction effort. The interdictor then wishes to solve ND: find $E_D \subset E$ such that (a) $G - E_D \equiv (V, E \backslash E_D)$ contains at least one s-t path, (b) every s-t path in $G - E_D$ contains e', and (c) $\sum_{e \in E_D} w_e$ is minimized. We call E_D a diverting (edge) set with respect to e' if it satisfies (a) and (b). Thus, ND defines the problem of finding a minimum-weight diverting set in G, given $\{s, t\} \subseteq V$ and diversion edge e'. This paper presents new theoretical and computational results for ND. "DND" ("UND") denotes ND defined on a directed (undirected) graph.

ND has an important, alternative characterization: if $G = (V, E)$, $\{s, t\} \subseteq V$, $e' \in E$, and $w_e > 0$ for all $e \in E$, then E_D is a minimum-weight diverting set for G with respect to e' if and only if $E_C = E_D \cup \{e'\}$ is a minimum-weight, minimal s-t cut that contains e'. "Minimal" implies that s and t are disconnected in $G - E_C$, but are connected in $G - E'_C$ for any $E'_C \subset E_C$. Because of this characterization, ND may

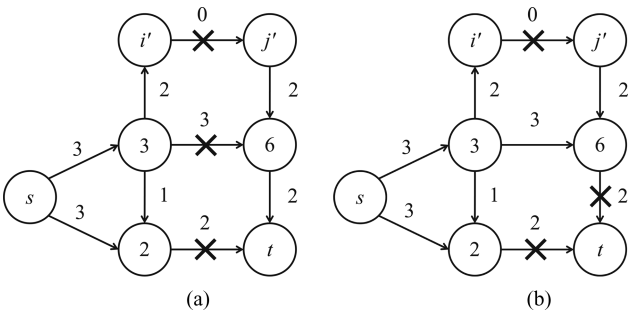


FIG. 1. Minimum-weight, minimal (a) and non-minimal (b) s-t cuts for a directed graph G that contain a specific edge $e' = (i', j')$. The number next to each edge is its weight, and edges marked by "×" define the cut. Figure 1a shows a minimum-weight, minimal s-t cut E_C^a that contains $e' = (i', j')$. This cut solves the network-diversion problem (ND) in this example. Figure 1b shows a minimum-weight (non-minimal) s-t cut E_C^b that contains e'. E_C^b has less total weight than E_C^a, but E_C^b does not solve ND because $G - (E_C^b \backslash \{e'\})$ contains no s-t paths.

appear at first glance to be a simple variant of the standard minimum-cut problem (e.g., Ahuja et al. [1, p. 167]), but ND turns out to be an interesting and difficult combinatorial-optimization problem in its own right. The difficulty arises because a minimum-weight s-t cut $E_C \supseteq \{e'\}$ may not be a minimal cut, and interdicting $E_D = E_C \backslash \{e'\}$ would destroy all s-t paths (see Fig. 1). This is not, of course, the interdictor's intent.

Unfortunately, a non-minimal cut is likely to appear when trying to solve ND using what may seem like an obvious approach: apply the standard minimum-cut model (e.g., Papadimitriou and Steiglitz [35, p. 119]), but fix the appropriate variable to force e' into the "restricted minimum-cut solution." Upon reflection, however, it is easy to see that this cannot work in general, because it is equivalent to (a) letting w_e represent a standard flow capacity for each edge $e \in E$, (b) defining $w_{e'} = 0$, (c) solving for a maximum s-t flow subject to the given edge capacities, and (d) hoping that e' appears in a corresponding minimum-capacity cut. ND demands more sophisticated techniques for its solution.

1.2. Applications

In one application of ND, a military commander wishes to force an enemy supply convoy to traverse a specific bridge e' as the convoy travels through a road network from a rear-echelon depot to a forward-operating base. The convoy is especially vulnerable to attack when it crosses e'. The military commander is the interdictor here; the convoy or its commander is the network user; the bridge is the diversion edge; road segments or bridges other than e' will be interdicted to effect the diversion; and an edge's weight reflects the number of aerial attacks that must be carried out on the edge to ensure its destruction. In another example, the interdictor represents an intelligence-gathering organization that wishes to attack links in a communications network so that all messages sent by the network user between points s and t must pass through a wiretapped link e' where they can be intercepted. Our results also cover a vertex-interdiction variant of ND, which would apply if attacks, and possibly wiretapping, must occur at communication hubs, that is, at vertices in the network.

The literature does not describe any real-world applications of mathematically optimized diversion, but we find numerous potential applications. For example, Krause [31] describes a "herding process" in which an interdictor pressures and removes options to steer "the enemy decision maker into a position in which the adversary leaders can neither exploit options nor gain an advantage without suffering unacceptable costs." Finelli [17] discusses the use of airpower in "herding targets." Jackson et al. [28] describe security measures designed to identify terrorists and "funnel them inward, where they can be isolated and engaged in a place and time of advantage to the authorities." In the context of counterdrug operations, Laughton [32] discusses the use of direct or indirect military actions and the way "they 'shape' the battlefield to 'funnel' the enemy for a future decisive engagement." Finally, Gayash et al. [22] describe a "man-in-the-middle" attack that "funnels communication" to a cyber criminal (interdictor).

1.3. Complexity

The original literature on ND [9, 12] addresses a more general version of DND, denoted here as "DNDm," with "m" implying multiple diversion edges. Specifically, DNDm defines a set of diversion edges E' and seeks a minimum-weight, minimal s-t cut E_C such that $|E' \cap E_C| \geq 1$. We show in Section 3 that DNDm can be solved through $|E'|$ solutions of DND, so no theoretical generality is lost in pursuing our simpler model. (The analogous result also holds for undirected networks.) Furthermore, if we take Curet's limit of $|E'| \leq 10$ in his computational tests as a practical limit [12], no generality is lost in practice either. If E' were to comprise a substantial proportion of E, however, additional research might be warranted.

Although previous computational work on ND covers both DND and UND (Curet [12], Cintron-Arias et al. [9], Erken [15], Yang and Park [45]), the only available result on theoretical complexity applies exclusively to DND and DNDm: DND is strongly NP-complete for general directed graphs, which is a fact that derives from the NP-completeness of the directed subgraph homeomorphism problem (Fortune et al. [19]). The NP-completeness of DNDm then follows by restricting m to 1.

Because of the limited results available, our paper explores more deeply the complexity of DND and UND in a variety of contexts, and provides a number of new results. For example, we show that DND is NP-complete even when the diversion edge is incident from s or into t, but not both, and we show that certain planar instances of DND and UND can be solved in polynomial time. The latter result could be important because ND on a planar graph can model the diversion of enemy troops traveling through a (planar) road network, and especially efficient computational methods would apply.

The computational complexity of ND remains open for some potentially interesting variants of this problem, however. For instance, we show that a vertex-deletion version of UND is NP-complete, but the complexity of the

nominal, edge-deletion version remains open. A key difference between UND and DND is that the feasibility problem for a general instance of DND is NP-complete, whereas the feasibility problem for UND is solvable in polynomial time.

As a final point on complexity, we note that two graph measures, inclusive vertex connectivity and inclusive edge connectivity, may seem to be related to network diversion; see Cribb [11] and the references therein. For example, inclusive vertex connectivity for a vertex v in an undirected graph measures the minimum number of vertices whose deletion causes v to become a cut vertex. This may be viewed as a vertex-deletion version of UND, without specification of s and t. We discuss both of these graph measures in Section 4.3 and explain why they do not help solve network-diversion problem.

1.4. Existing Solution Methods

Curet [12] and Cintron-Arias et al. [9] apply integer-programming (IP) methods to solve ND. Subsequently, Erken [15] develops a combinatorial algorithm, which is based on enumerating near-minimum-weight s-t cuts. His algorithm may warrant further investigation, but our computational experiments favor a mathematical-programming approach. For example, given 48 h of computation time, Erken's algorithm fails to find a single feasible solution to DND on one 25×25 star-mesh network—we describe this topology in Section 6.1—while mathematical-programming methods solve this problem instance in just a few minutes.

Yang and Park [45] apply a tabu-search heuristic (e.g., Glover [23]) to instances of DND with as many as 100 vertices, 600 edges, and three diversion edges. This heuristic typically runs more quickly than does Curet's IP model, but some of the heuristic solutions have optimality gaps that exceed 11%. By contrast, our computational results identify solutions in networks that are orders of magnitude larger, with optimality gaps of at most 1%.

Cho [8] combines an IP and solution-elimination constraints (SECs) in a decomposition algorithm to solve instances of DND with up to 2,550 vertices, 9,900 edges, and three diversion edges. (See Brown et al. [7] for a discussion of SECs. The SECs used by Cho may be interpreted as "combinatorial Benders cuts" as defined by Codato and Fischetti [10], and Cho uses the latter phrase.) The algorithm iteratively solves a minimum-weight s-t cut problem for a minimal cut E_C, and checks if $e' \in E_C$. If not, an SEC is appended to the IP to eliminate E_C as a solution, and the IP is re-solved. Assuming the problem is feasible, the process repeats until $E_C \supseteq \{e'\}$ and the cut is therefore optimal. The algorithm is finite, but its scalability is questionable, at least for some problem classes. For example, the algorithm adds at most 458 SECs in any of Cho's test problems, but that algorithm might require millions of such constraints to solve one of our test problems. In particular, the 25×25 star-mesh network mentioned above contains more than 5×10^8 minimal cuts $E_C \not\supset \{e'\}$, each of which has a weight less than that of an optimal solution to ND. (We established this fact by running Erken's algorithm [15] on the specified network for several days.)

Curet [12] (see also [9]) solves DND, approximately, by applying Lagrangian relaxation to a weaker variant of the mixed-integer programming (MIP) "P_1," which we present in Section 5.1. (Curet defines a 0-1 IP, but could have allowed some variables to be continuous, thereby yielding a MIP.) Computational results will show that our improved formulations dramatically outperform P_1 when solved by standard branch and bound. We do not pursue Lagrangian relaxation, but note that our formulations' tighter relaxations could be useful for that purpose.

The remainder of the paper is organized as follows. Section 2 provides technical definitions for DND and UND, and establishes the polynomial equivalence of a model with a single diversion edge and a model with multiple diversion edges. Section 3 formalizes the known NP-completeness result on DND, provides a new, stronger proof, and then extends that proof to cover other variations of ND. Section 4 extends some of the NP-completeness results from the previous section to undirected networks and describes a polynomial-time solution method for ND on certain planar graphs. Section 5 describes our specialized implementation of Curet's IP and develops our stronger MIP in two variants. Section 6 presents computational results for those models. Section 7 concludes the paper.

2. PROBLEM DEFINITION

We consider initially a directed graph $G = (V, E)$ with vertex set V and edge set $E \subset V \times V \backslash \{(i,i) | i \in V\}$. Source vertex $s \in V$ and sink vertex $t \in V$, $t \neq s$, are also defined, along with a diversion edge $e' \in E$. If e'' denotes an edge that may or may not exist in G, and E'' denotes a subset of edges that may or may not exist in G, the notation $G + e''$, $G - e''$, $G + E''$, and $G - E''$ signify $(V, E \cup \{e''\})$, $(V, E \backslash \{e''\})$, $(V, E \cup E'')$, and $(V, E \backslash E'')$, respectively.

Each edge $e \in E \backslash \{e'\}$ has weight $w_e \in \mathcal{Z}^+$, which represents the cost to interdict or "delete" that edge. For simplicity, and because e' is never actually interdicted, $w_{e'} \equiv 0$. The total weight of $E' \subseteq E$ is denoted $w(E') = \sum_{e \in E'} w_e$.

A (directed) s-t path in G is a set of edges of the form $E_{st} = \{(s, i_1), (i_1, i_2), \ldots, (i_{n-2}, i_{n-1}), (i_{n-1}, t)\}$. A path is simple if no vertices are repeated. Throughout, we assume that the original graph G, with no interdicted edges, contains at least one s-t path. A simple s-t path $E_{st} \supset \{e'\}$ is a diversion path with respect to $\{s, t\}$ and e'.

A (directed) cycle in G is a set of edges of the form $E_L = \{(i_0, i_1), (i_1, i_2), \ldots, (i_{n-2}, i_{n-1}), (i_{n-1}, i_0)\}$. The cycle is simple if no vertices except i_0 are repeated.

The set E_D is an s-t diverting set for G with respect to $e' \in E$ if (a) $G - E_D$ contains an s-t path, and (b) all s-t paths in $G - E_D$ contain e' (or, equivalently, all s-t paths in $G - E_D$ are diversion paths). The directed network-diversion problem (DND) defined on G, $\{s, t\}$ and e', and given edges weights w_e for all $e \in E$, seeks a diverting set E_D such that $w(E_D)$ is minimum among all such sets.

DND may be characterized in a useful, alternative fashion. An s-t cut E_C is any disconnecting set of edges with respect

to s and t. That is, $G - E_C$ allows no s-t paths. The s-t cut E_C is minimal if no proper subset is also an s-t cut. Through basic definitions, it is clear that DND is equivalent to finding a minimum-weight, minimal s-t cut E_C in G such that $e' \in E_C$.

UND is defined analogously to DND, with obvious modifications for an undirected graph. (We do specify the direction in which we wish the network user to traverse the diversion edge in UND, although another definition of the problem might not.)

Recall from Section 1.3 that DNDm defines a set of diversion edges E' and seeks a minimum-weight, minimal s-t cut E_C such that $|E' \cap E_C| \geq 1$. Proposition 1, below, justifies our focus on DND rather than on DNDm, as studied in [12] and elsewhere; the analogous result holds for UND and the multiple-diversion-edge version of UND, "UNDm." We require this definition: problem P_1 is polynomially reducible to problem P_2 if (a) a polynomial number of calls to an algorithm for solving P_2 will solve P_1, and (b) any conversions of input data for P_1 to input data for P_2 can be performed in polynomial time (Korte and Vygen [30, p. 368]). P_1 is not fundamentally more difficult than P_2 if P_1 is polynomially reducible to P_2.

Proposition 1. *DNDm is polynomially reducible to DND.*

Proof. Let DND(e') denote DND with diversion edge $e' \in E'$ explicitly identified and, similarly, let DNDm(E') denote DNDm with diversion-edge set E' identified. Let $E_1^*(e')$ denote an optimal solution to DND(e') (i.e., a minimum-weight, minimal s-t cut containing e'), and let $z_1^*(e')$ denote that solution's objective value. For simplicity, but without loss of generality, assume that DND(e') is feasible for each $e' \in E'$. Also, let $E_2^*(E')$ denote an optimal solution to DNDm(E') with objective-function value denoted $z_2^*(E')$. Now, for any $e'' \in E'$, $E_1^*(e'')$ is feasible for DNDm(E'), and thus $z_2^*(E') \leq \min_{e'' \in E'} z_1^*(e'')$. Conversely, for $e' \in E_2^*(E') \cap E'$, $E_2^*(E')$ is feasible for DND(e'), implying that $z_2^*(E') \geq z_1^*(e') \geq \min_{e'' \in E'} z_1^*(e'')$. Therefore, $z_2^*(E') = \min_{e'' \in E'} z_1^*(e'')$ and DND(e'') solves DNDm(E') for some $e'' \in E'$. Polynomial reducibility of DNDm to DND then follows, because DNDm(E') can be solved by making $|E'|$ calls to an algorithm for solving DND(e') using input data that is a subset of that required by DNDm(E'). ∎

Nothing in the above proof depends on the directed nature of the graph, so the following corollary is immediate:

Corollary 1. *UNDm is polynomially reducible to UND.*

Proposition 1 and Corollary 1 suffice to justify this paper's focus on DND and UND rather than on DNDm and UNDm, but the following result is also clear:

Corollary 2. *DND (UND) is polynomially equivalent to DNDm (UNDm).*

Proof. Polynomial equivalence requires that each problem be polynomially reducible to the other (Booth [5]). Proposition 1 proves that DNDm is polynomially reducible to DND. For the directed case, the other half of the proof follows from the fact that DNDm(E') is equivalent to DND(e') when $E' = \{e'\}$. The undirected version of this corollary follows similarly. ∎

3. COMPUTATIONAL COMPLEXITY FOR DND

This section and the next investigate the theoretical complexity of the decision problems associated with network diversion. Until further notice, $\bar{G} = (\bar{V}, \bar{E})$ denotes an undirected graph, and square brackets denote an undirected edge, for example, $[i,j]$. Also, "DND" and "UND" now refer to decision-problem variants of the corresponding optimization problems defined in Section 2. For instance, DND now corresponds to the following:

DIRECTED NETWORK DIVERSION (DND)
Given: Directed graph $G = (V, E)$, $\{s, t\} \subseteq V$, diversion edge $e' \in E$, $w_e \in \mathcal{Z}^+$ for all $e \in E \backslash \{e'\}$, $w_{e'} = 0$, and threshold $W \in \mathcal{Z}^+$.
Question: Does G contain a minimal s-t cut E_C such that $e' \in E_C$ and $w(E_C) \leq W$? ∎

3.1. Existing Complexity Results for DND

Curet [12] makes an informal claim that DND is NP-complete based on a reduction from the directed subgraph homeomorphism problem ("DSH"; see [19]). Yang and Park [45] formalize that argument, but their proof is available only in Korean. For later reference, and to clarify the contributions of the current paper, we provide here a brief proof following Yang and Park. The essence of the argument is that the existence of any minimal diverting set for diversion edge e' implies the existence of a simple directed s-t path that contains e'. Determining whether or not such a path exists is NP-complete, and thus DND must be NP-complete as well. We begin with two formal definitions.

DIRECTED NETWORK DIVERSION FEASIBILITY (DNDF)
Given: Directed graph $G = (V, E)$, vertex set $\{s, t\} \subseteq V$, and diversion edge $e' \in E$.
Question: Does G contain a minimal s-t cut E_C such that $e' \in E_C$? ∎

DIRECTED EDGE-RESTRICTED s-t PATH (DERPATH)
Given: Directed graph $G = (V, E)$, $\{s, t\} \subseteq V$, and "required edge" $e' \in E$.
Question: Does G contain a simple, directed s-t path E_{st} such that $e' \in E_{st}$? ∎

Fortune et al. [19] establish the following proposition as a corollary of an NP-completeness theorem for DSH (see Theorem 2 and Lemma 3 in that paper):

Proposition 2. *DERPATH is strongly NP-complete.*

Following [45] then, we can formally establish this theorem:

Theorem 1. *DND is strongly NP-complete.*

Proof. DND is a member of NP because, given a potential solution E_C, we can check in linear time whether or not the following holds: (a) $e' \in E_C$, (b) $w(E_C) \leq W$, (c) no s-t path exists in $G - E_C$, but (d) for each edge $e'' \in E_C$, an s-t path does exist in $G - E_C \backslash \{e''\}$. (Items (c) and (d) can be checked using a single call to a simple variant on breadth-first or depth-first search along with $|E_C|$ constant-time operations.)

To prove the theorem, it suffices to show that DNDF is NP-complete, because DND is a member of NP and the following restrictions convert DNDF to DND: define $w_e = 1$ for all $e \in E\backslash\{e'\}$, $w_{e'} = 0$, and $W = |E|$. Now we show that an instance of DNDF, with inputs G, $\{s,t\}$ and e', is feasible if and only if DERPATH with the same inputs is feasible. Strong NP-completeness will follow because no edge weights other than 0 or 1 are used in the proof.

Suppose DNDF is feasible, as demonstrated by a minimal s-t cut $E_C \supseteq \{e'\}$. No s-t path can exist in $G - E_C$ by the definition of an s-t cut, but such a path must exist in $(G - E_C) + e'$ by the definition of minimal. This implies the existence of a simple s-i' path in $G - E_C$, say $E_{si'}$, and a non-intersecting, simple j'-t path, say $E_{j't}$. It is easy to identify such paths using, say, breadth-first search, so assume this has been done. (These paths were difficult to find in G, but become easy to find in $G - E_C$, which is created from G knowing a solution to DNDF.) Then, $E_{st} \equiv E_{si'} \cup \{e'\} \cup E_{j't}$ is a feasible solution to DERPATH.

Conversely, suppose DERPATH is feasible, as demonstrated by a simple s-t path $E_{st} \supseteq \{e'\}$. Let $\hat{G} = (V, \hat{E})$, where $\hat{E} = E_{st}\backslash\{e'\}$: we know that vertex s is disconnected from vertex t in $\hat{G}$, but is connected in $\hat{G} + e'$. Now, in any fixed order, check each edge $e \in E\backslash\hat{E}$, and add e to $\hat{E}$ if and only if s and t remain disconnected in $\hat{G} + e$. In the final instance of $\hat{G} = (V, \hat{E})$ created in this fashion, no edge can be added to $\hat{E}$ without reconnecting s and t, and we know that $e' \in E_C \equiv E\backslash\hat{E}$. Consequently, E_C is a minimal s-t cut such that $e' \in E_C$, and DNDF is therefore feasible. ∎

We note that the proof above does not translate into a complexity proof for UND, because the undirected version of DERPATH is solvable in polynomial time (Shiloach [41], Verdieren and Schrijver [13]).

3.2. Stronger Complexity Results for DND

We have shown that DND is NP-complete in a proof based on the complexity of finding a simple, directed s-t path that contains a particular edge. This approach leaves open the possibility of stronger proofs that specify special conditions under which DND remains NP-complete. For instance, we know that DND is trivially solvable when $e' = (s,t)$, but does it remain that easy if $e' = (i,t)$ or if $e' = (s,j)$ for $i \neq s$ and $j \neq t$? For these cases, DERPATH is solvable efficiently as a shortest-path problem, so the complexity of DND remains unclear. (In fact, we will see that DND is NP-complete in these cases.)

The rest of this section provides a stronger proof and addresses special cases for DND. The proof is based on a transformation from the following well-known, NP-complete problem.

VERTEX COVER (Karp [29])
Given: Undirected graph $\bar{G} = (\bar{V}, \bar{E})$ and positive integer $K \leq |\bar{V}|$.
Question: Does there exist $\bar{V}_C \subseteq \bar{V}$ with $|\bar{V}_C| \leq K$ such that $\bar{V}_C$ is incident to each $e \in \bar{E}$? ∎

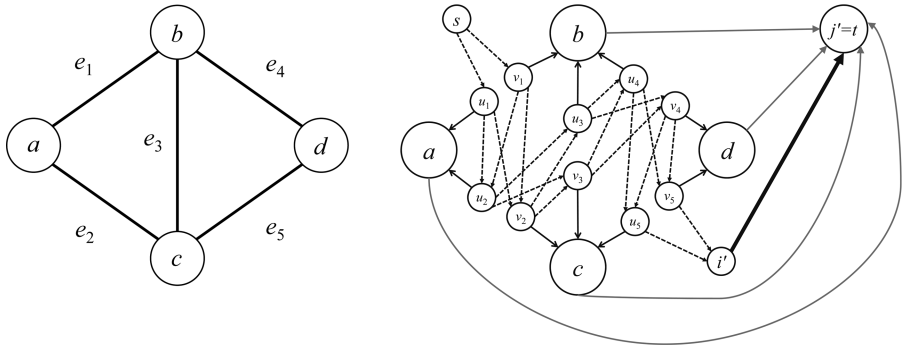
Theorem 2. *DND is NP-complete.*

Proof. The proof of Theorem 1 already shows that DND is a member of NP. To complete the current proof then, we demonstrate a polynomial transformation of VERTEX COVER to DND. For simplicity, the demonstration characterizes DND through a diversion edge e' and a corresponding diverting set E_D, rather than through a minimal cut. The transformation shows that an undirected graph $\bar{G} = (\bar{V}, \bar{E})$ has a vertex cover of size $K \leq |\bar{V}|$ if and only if an instance of DND, constructed on directed graph $G = (V, E)$, together with designated vertex set $\{s,t\} \subset V$ and designated edge $e' \in E$, has diverting set E_D with respect to e' such that $w(E_D) \leq K(3|\bar{E}| + 1) + 3|\bar{E}| - 1$.

Given an instance of VERTEX COVER defined on undirected graph $\bar{G} = (\bar{V}, \bar{E})$, we construct an instance of DND on directed graph $G = (V, E) \equiv (V_0 \cup V_1 \cup V_2, E_0 \cup E_1 \cup E_2 \cup E_3)$ using the following algorithm (see Fig. 2):

1. **Begin:** Define threshold $W \leftarrow K(3|\bar{E}| + 1) + 3|\bar{E}| - 1$ and define weights $M_1 \leftarrow 3|\bar{E}|+1$, $M_2 \leftarrow W + 1$, and $M_3 \leftarrow 1$.
2. Create miscellaneous vertices $V_0 \leftarrow \{s, t, i'\}$ and "original vertices" $V_1 \leftarrow \bar{V}$.
3. Define diversion edge $e' \equiv (i', t)$ and let $E_0 \leftarrow \{e'\}$.
4. Create "main edges" $E_1 \leftarrow \cup_{i \in V}\{(i,t)\}$, and let $w_e \leftarrow M_1$ for all $e \in E_1$.
5. Arbitrarily order all $e \in \bar{E}$ as e_ℓ, $\ell = 1, \ldots, |\bar{E}|$.
6. Initialize $V_2 \leftarrow \emptyset$ for "split vertices" and $E_2 \leftarrow \emptyset$ for "split edges."
   For $\ell = 1, \ldots, |\bar{E}|$, split undirected edge $e_\ell = [i,j]$ into two directed edges, (u_ℓ, i) and (v_ℓ, j), and let $V_2 \leftarrow V_2 \cup \{u_\ell, v_\ell\}$ and $E_2 \leftarrow E_2 \cup \{(u_\ell, i), (v_\ell, j)\}$.
   Let $w_e \leftarrow M_2$ for all $e \in E_2$.
7. Initialize $E_3 \leftarrow \emptyset$ for "potential-diversion-path edges."
   Connect adjacent split vertices with such edges:
   For $\ell = 1, \ldots, |\bar{E}| - 1$, $E_3 \leftarrow E_3 \cup \{(u_\ell, u_{\ell+1}), (u_\ell, v_{\ell+1}), (v_\ell, u_{\ell+1}), (v_\ell, v_{\ell+1})\}$.
   Add potential-diversion-path edges from s and into i':
   $E_3 \leftarrow E_3 \cup \{(s, u_1), (s, v_1), (u_{|\bar{E}|}, i'), (v_{|\bar{E}|}, i')\}$;
   Let $w_e \leftarrow M_3$ for all $e \in E_3$. **End.**

Note that e' cannot be deleted by definition, and every edge $e \in E_2$ is essentially "undeletable" because the weight of each exceeds the threshold W. Thus, in the following, we



(a) $\bar{G}$ (b) G

FIG. 2. Transformation from undirected graph $\bar{G}$ (Figure 2a) to directed graph G (Figure 2b) used in the proof of Theorem 2. In G, the thick black edge is the diversion edge and defines $E_0 = \{e'\}$; gray "main edges" define E_1; thin black "split edges" define E_2; and dashed "potential-diversion-path edges" define E_3. The set of "original vertices" is $V_1 = \{a, b, c, d\}$ and the set of "split vertices" is $V_2 = \cup_{\ell=1}^{5}\{u_\ell, v_\ell\}$.

may disregard all edges $e \in E_0 \cup E_2 = \{e'\} \cup E_2$, while creating a diverting set E_D with acceptable weight.

Now, suppose $\bar{G}$ has a vertex cover $\bar{V}_C$ with $|\bar{V}_C| = K$. Let E_1' be the main edges in G corresponding to that cover, that is, $E_1' = \{(v,t)|v \in \bar{V}_C\}$. Then, consider any split-vertex pair $\{u_\ell, v_\ell\}$ in $G - E_1'$. Since $\bar{V}_C$ is a cover in $\bar{G}$, for each $\ell = 1, \ldots, |\bar{E}|$, either the u_ℓ-t path along a main edge or the v_ℓ-t path along a main edge, or both, are eliminated in $G - E_1'$. Thus, we can construct an s-i' path $E_{si'} \subset E_3$ in $G - E_1'$ that never visits a split vertex that is connected to t through a main edge. It follows that $G - E_1' - E_3 \backslash E_{si'}$ includes an s-i' path, but no path to t through a main edge. Thus, $G - E_1' - E_3 \backslash E_{si'}$ contains the s-t path $E_{si'} \cup \{e'\}$, but no other. Consequently, $E_D \equiv E_1' \cup E_3 \backslash E_{si'}$ is a diverting set with respect to e'. Because (a) $|E_1'| = K$ and $w_e = M_1 = 3|\bar{E}| + 1$ for all $e \in E_1'$, and (b) $|E_3 \backslash E_{si'}| = 3|\bar{E}| - 1$ and $w_e = M_3 = 1$ for all $e \in E_3 \backslash E_{si'}$, $w(E_D) = K(3|\bar{E}| + 1) + 3|\bar{E}| - 1$, as required.

Conversely, suppose E_D solves DND with $w(E_D) \leq W \equiv K(3|\bar{E}| + 1) + 3|\bar{E}| - 1$ for some $K \in \mathcal{Z}^+$, $K \leq |\bar{V}|$. The set of main edges $E_D \cap E_1$ must correspond to a (possibly non-minimal) vertex cover $\bar{V}_C$ because (a) $G - E_D$ must maintain at least one s-i' path, (b) any such path visits at least one split vertex associated with each undirected edge $e \in \bar{E}$, and (c) each main edge in E_1 along an s-t path that bypasses the diversion edge must be deleted for each split vertex along an s-i' path; that is, at least one deleted main edge of E_D must "cover" every undirected edge $e \in \bar{E}$. Now, if $|E_D \cap E_1| > K$, then $w(E_D) \geq (K+1) \cdot 3(|\bar{E}| + 1) > W$, which is a contradiction. Thus, the cover $\bar{V}_C$ corresponding to E_D must have $|\bar{V}_C| \leq K$. ∎

Note that the proof of Theorem 2 modifies trivially if we wish to force a path through a given "diversion vertex" instead of a diversion edge: simply modify G by splitting the diversion edge (i', t) into undeletable edges (i', i'') and (i'', t) and by declaring i'' to be the diversion vertex.

More importantly, the proof of Theorem 2 also extends to a version in which all edges except e' have weight 1 by replacing edges in E_1 and E_2 with M_1 and M_2 parallel edges, respectively. Since M_1 and M_2 are polynomially bounded in $|\bar{V}|$ and $|\bar{E}|$, this corollary follows:

Corollary 3. *DND is strongly NP-complete.*

Finally, we present a simple, but important, corollary.

Corollary 4. *Assuming $e' \neq (s,t)$, DND is strongly NP-complete (a) when G is acyclic, and (b) even when the diversion edge is incident into t, or (c) incident from s.*

Proof. The proof of Theorem 2 uses an acyclic graph, so that covers case (a). Case (b) follows because e' is incident into t in the proof of Theorem 2. Case (c) follows because edge directions, and the identities of s and t, could be reversed in that proof. ∎

4. COMPUTATIONAL COMPLEXITY FOR UND

Theorem 1 presented in Section 3.1 does not lead to a complexity proof for UND because its proof is based on DERPATH, which happens to be solvable efficiently on undirected graphs. Unfortunately, Theorem 2, presented above, does not extend to a simple complexity proof for UND either. We might try replacing each directed edge in G with an undirected edge in the proof of Theorem 2, but undirected edges between split vertices and original vertices in G permit an s-t path $E_{st} \supset \{e'\}$ that may bypass original vertices and, consequently, E_{st} may not correspond to a solution to VERTEX COVER. The proof of Theorem 2 does lead to an NP-completeness proof for a vertex-deletion version of UND, however. This section presents that proof, shows how to solve UND in polynomial time for s-t planar graphs, and then summarizes all complexity results in this paper.

4.1. Vertex Deletion

Let $\bar{G} - \bar{V}_C$ denote the subgraph induced from undirected graph $\bar{G} = (\bar{V}, \bar{E})$ by deleting $\bar{V}_C$ from $\bar{G}$, along with all

edges incident to $\bar{V}_C$. Also, let $\bar{V}_C \subseteq \bar{V}\backslash\{s,t\}$ be a minimal s-t disconnecting vertex set for $\bar{G}$ if s and t are disconnected in $\bar{G} - \bar{V}_C$ but are connected in $\bar{G} - \bar{V}'_C$ for any $\bar{V}'_C \subset \bar{V}_C$.

VERTEX DELETION FOR UNDIRECTED NETWORK DIVERSION (VUND)
Given: Undirected graph $\bar{G} = (\bar{V}, \bar{E})$, $\{s,t\} \subset \bar{V}$, diversion vertex $v' \in \bar{V}\backslash\{s,t\}$, vertex weights $w_v \in \mathcal{Z}^+$ for all $v \in \bar{V}\backslash\{v'\}$, and $w_{v'} = 0$, and threshold $W \in \mathcal{Z}^+$.
Question: Does there exist a minimal s-t disconnecting vertex set $\bar{V}_C \subset \bar{V}\backslash\{s,t\}$ in $\bar{G} = (\bar{V}, \bar{E})$ such that $v' \in \bar{V}_C$ and $\sum_{v\in\bar{V}_C} w_v \leq W$? ▪

Note that the complexity of VUND would not change whether we defined it with respect to a diversion edge or diversion vertex, and we use the latter just for the sake of maintaining symmetric definitions. VUND models an interdictor trying to funnel an enemy through a diversion vertex v' by interdicting vertices rather than edges. In a communications network, for instance, it might be the hubs, that is, vertices, that are susceptible to interdiction rather than the links, that is, edges.

To see that VUND is NP-complete, consider the directed graph $G = (V,E)$ constructed in the proof of Theorem 2. First, replace the undeletable diversion edge (i',j') with undeletable, undirected split edges $[i',v']$ and $[v',j']$. Now, the requirement that "all s-t paths in the interdicted graph must pass through e'" becomes "... must pass through v'." Next, suppose that, for any original vertex $u \in V$, we replace the option "delete directed main edge $e = (u,t) \in E$ at cost w_e" with "delete original vertex $u \in V$ at cost $w_v = w_e$ where $e = (u,t)$." The proof of Theorem 2 holds then, after defining "deletions" to cover both edges and vertices in a restricted fashion. Furthermore, because deletion of original vertices prohibits the bypasses mentioned earlier, each remaining (directed) edge (i,j) of G may be replaced by the undirected edge $[i,j]$, keeping its status as "deletable" or "undeletable." Then, as we did with main edges, split each deletable edge $e = [i,j]$ into undeletable edges $[i,v]$ and $[v,j]$, where v is a deletable vertex with $w_v = w_e$. Denote the newly constructed undirected graph as $\bar{G}$. The proof of Theorem 2 still holds, except that vertex deletion in $\bar{G}$ replaces edge deletion in G and "diversion vertex" in $\bar{G}$ replaces "diversion edge" in G. We could have begun the procedure just described by first replacing each edge $(i,j) \in E - e'$ with w_e parallel edges, so Corollary 3 also applies here. Thus, we have proven:

Corollary 5. *VUND is strongly NP-complete.*

In the remainder of the paper, we return to standard notation for both undirected and directed graphs. That is, $G = (V,E)$ denotes a graph, $(i,j) \in E$ denotes an edge, and "undirected" or "directed" will be clear from the context.

4.2. UND on Planar Graphs

The definitions and facts stated here are well known, but the reader may wish to refer to Ford and Fulkerson [18] and/or Deo [pp. 90–95]. An undirected graph $G = (V,E)$ is planar if it can be embedded in a plane without edges crossing. Assuming G has no parallel edges, a *face* is a region in some embedding of G that is incident to three or more edges. The planar graph together with $\{s,t\} \subseteq V$ is said to be s-t planar if vertices s and t border a common face. An s-t planar graph can always be embedded in the plane such that s and t appear on the outer face. Indeed, s-t planar graphs are typically presented in such fashion. (Replacing every directed edge in Figure 3 of Section 6.1 with an undirected edge results in an s-t planar graph, although the definition of "face" must then be extended to incorporate parallel edges.) This section shows the polynomial solvability of UND defined on an s-t planar graph.

The class of s-t planar graphs is relevant to network diversion because one of these graphs might reasonably model a planar transportation network with a source and sink located on the periphery. For example, Harris and Ross [25] illustrate a network-interdiction problem on a railway network of the western Soviet Union and eastern Europe and, by deleting one irrelevant edge and adding super-source and super-sink constructs, that network becomes s-t planar.

The planar graph $G^* = (V^*, E^*)$, which is the planar dual to G, is constructed as follows:

1. Embed G in the plane.
2. Place a dual vertex $i^* \in V^*$ in each face of the primal graph, including the exterior face.
3. Connect each dual vertex pair $\{i^*, j^*\}$ in adjacent primal faces with a dual edge $e^* = (i^*, j^*)$. Let E^* denote the set of all dual edges, and note that a one-to-one correspondence has been created between crossing primal and dual edges.
4. For the purposes of studying network diversion, let $e'^* = (i'^*, j'^*)$ denote the dual edge that crosses the diversion edge. Also, define $w_{e^*} = w_e$ for all $e^* \in E^*$, where e^* and e denote corresponding (crossing) dual and primal edges, respectively.

The key connection between dual and primal graphs is that $E^*_S \subset E^*$ is a simple cycle in G^* if and only if the corresponding primal edge set $E_S \subset E$ defines a minimal separating set in G. Invoking the Jordan Curve Theorem, Phillips [36] shows that a minimum-weight s-t cut in G (i.e., a minimum-weight, minimal set that separates s from t) can be found by (a) defining a simple s-t path in G, which we shall call the reference path, and (b) by then finding a minimum-weight cycle E^*_C in G^* with odd parity, that is, which crosses the reference path an odd number of times. For UND, the following proposition is then obvious:

Proposition 3. *A solution to UND on a planar graph G corresponds directly to a minimum-weight, simple, odd-parity cycle $E^*_C \supset \{e'^*\}$, where parity is measured with respect to a simple s-t reference path in G that contains e'.*

It is easy to identify a reference path as required by the proposition if one exists. And then, using shortest-path

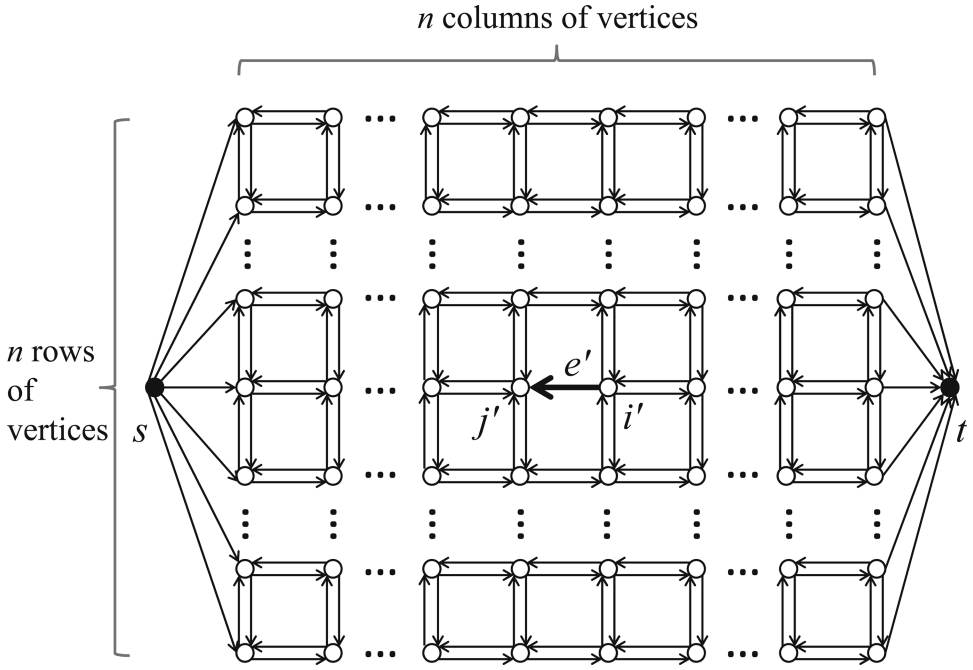


FIG. 3. An $n \times n$ directed grid network with a difficult orientation for $e' = (i', j')$. Edges of the form (s, i) and (i, t) are undeletable or, equivalently, have infinite interdiction weights. Other edge weights are as specified in the text.

techniques, it is easy to find a minimum-weight odd-parity cycle E_S^* in G^* such that $e'^* \in E_S^*$ (Letchford and Pearson [33]). Unfortunately, such an approach does not lead to a general, efficient method for solving UND on planar graphs, because E_S^* may not be simple, and because minimality of the corresponding *s-t* cut demands a simple cycle. The following special case of a planar graph does admit an efficient solution, however.

Theorem 3. *UND is solvable in polynomial time whenever G is s-t planar.*

Proof. (a) Embed G in the plane so that s and t are on the outer face, (b) split the outer face by adding edge (s, t) such that it preserves the planarity of G, (c) create dual graph G^* from G in the standard fashion, and identify one of the dual vertices in the "split face" as s^*, and the other as t^*, (d) delete the dual edge (s^*, t^*), and (e) find a simple path from s^* to t^* passing through e'^* (which can be accomplished efficiently using, say, network-flow techniques). Let E_{st} be the primal edges crossed by this path. It is well-known that E_{st} created in this fashion is a minimum-weight, minimal *s-t* cut in G (e.g., [1, p. 263–264]). By construction, $e' \in E_{st}$. ∎

We note that the proof applies if we start with a directed graph and replace the undirected dual graph with a directed one; see Schrijver [38] for general background on directed planar graphs and their duals; see [36] for an application to an interdiction problem; and note that step (e) in the proof can be carried out in polynomial time in a directed graph as long as that graph is planar [38]. Thus, the following corollary is immediate.

Corollary 6. *DND is solvable in polynomial time whenever G is s-t planar.*

We include the following simple result for the sake of completeness.

Corollary 7. *DND is solvable in polynomial time whenever G is planar and acyclic.*

Proof. Because $G = (V, E)$ is acyclic, any vertex that precedes s or follows t in an acyclic ordering of V is unreachable given that the network user must start at s and end at t. Any such vertices may be deleted to create an *s-t* planar graph, and Theorem 3 then applies. ∎

4.3. Inclusive Vertex and Edge Connectivity

To complete the discussion on the complexity of ND, we note that two graph measures seemingly related to network diversion arise in the context of social networks, namely, "inclusive vertex connectivity of a vertex v'" and "inclusive edge connectivity of an edge e'" (e.g., Boland and Ringeisen [4]). (Other versions of inclusive connectivity have been defined, but we limit discussion to these two for the sake of brevity. Note also that early papers, e.g., Ringeisen and Lipman [37], use the term "cohesion" rather than "inclusive connectivity.")

Given a connected, undirected graph $G = (V, E)$, and specified vertex $v' \in V$, the inclusive vertex connectivity of v', denoted $\kappa(v')$ here, is the minimum number of vertices that must be deleted from G to cause v' to become a cut vertex. Thus, it appears that computing $\kappa(v')$ might be related to solving VUND. But, $\kappa(v')$ can be computed using the following, polynomial-time procedure [37]:

1. **Begin:** Set all vertex capacities in G to 1, set all edge capacities to ∞, and define $N(v') = \cup\{i | (i, v') \in E\}$.
2. For each $\{i, j\} \subseteq N(v')$, identify a minimum-capacity i-j cut in G with respect to vertex capacities and let V_{ij} denote the vertices of that cut.
3. Define $\kappa(v') = \min_{\{i,j\} \subseteq N(v')} |V_{ij}|$. **End.**

Given the polynomial computability of $\kappa(v')$ and the NP-completeness of VUND, we see only one useful connection between the two concepts, a connection that is apparent in the above procedure: VUND can be solved in polynomial time whenever s and t are both adjacent to v'.

The inclusive edge connectivity of e', denoted $\lambda(e')$ here, is the minimum number of edges that must be deleted from G to cause $e' = (i', j')$ to become a bridge. Thus, it appears that computing this measure might be related to solving UND. But, $\lambda(e')$ is easily computed by setting edge capacities in G to 1, defining vertices as uncapacitated, and by solving for a (minimal) minimum-capacity i'-j' cut in G. Computing $\lambda(e')$ involves a substantial relaxation of the requirements for solving UND, specifically, that a solution define a minimal i'-j' cut and a minimal s-t cut. Computation of inclusive edge connectivity does point to an easy solution of UND when $e' = (s, t)$, but this special-case solution is obvious.

4.4. Summary of Complexity Results

Table 1 summarizes our computational-complexity results and lists important problem variants whose complexity remains open. We lack an NP-completeness proof for UND because neither DERPATH nor Theorem 2 applies to undirected graphs.

For several reasons, we also lack complexity results for DND and UND on general planar graphs. On the one hand, showing NP-completeness for these problems seems difficult: (a) DERPATH does not apply to UND at all, because DERPATH is solvable in polynomial time on an undirected graph, (b) DERPATH does not apply to DND on planar graphs, because the planar version of DERPATH is solvable in polynomial time [38], and (c) although VERTEX COVER is NP-complete on planar graphs (Garey and Johnson [21, p. 190]), the transformation from VERTEX COVER used in Theorem 2 does not maintain planarity for DND. On the other hand, attempting to find polynomial-time algorithms for these algorithms also seems difficult. Phillips [36] solves a maximum-flow network-interdiction problem in polynomial or pseudopolynomial time on general, directed, and undirected planar graphs, not just on s-t planar ones. Her algorithm involves finding an odd-parity simple cycle in G^*, but that cycle need not contain any particular edge such as e'^*. This restriction seems to increase, in a significant manner, the difficulty of solving UND and DND on general planar graphs.

TABLE 1. Summary of complexity results for ND. We provide four new theoretical complexity results (∗) and one stronger result (∗∗).

Type of ND			Complexity
Directed	Planar	Acyclic	Polynomial*
		General	Open
	Nonplanar	Acyclic	Strongly NP-complete*
		General	Strongly NP-complete**
Undirected	s-t Planar		Polynomial*
	General planar		Open
	Nonplanar		Open
	Vertex deletion		Strongly NP-complete*

"Open" indicates a variant for which the computational complexity remains unknown.

In the next section, "DND" and "UND" again refer to optimization problems. The section describes mathematical-programming formulations for solving general instances of DND and, through an appendix, general instances of UND.

5. MATHEMATICAL-PROGRAMMING FORMULATIONS FOR DND

This section first describes Curet's original, "single-commodity IP" for DND [12]. It then presents a new "two-commodity MIP" for that problem and, finally, proposes valid inequalities for that MIP. We prove that the new MIP is at least as strong as the original IP, and show later, empirically, that it can be strictly stronger. "DND" now refers to the optimization problem defined in Section 2.

The models in this section use the following definitions and facts. An s-t cutset $C = [S, T]$ is a partition of V into S and $T = V - S$, such that $s \in S$ and $t \in T$. The set of edges $E_C \equiv \{(i, j) \in E \mid i \in S, j \in T\}$ is clearly an s-t cut, although it may not be minimal. E_C defines the forward edges of the corresponding cutset.

5.1. A Basic Single-Commodity IP Formulation for DND, P_1

Curet [12] defines a set of diversion edges $E' \subset E$ in directed graph $G = (V, E)$, and formulates DNDm as the problem of finding a minimum-weight s-t cut E_C and a diversion path E_{st} such that $E_C \cap E_{st} \subseteq E'$. The following model specializes Curet's formulation to a single diversion edge e' (i.e., $E' = \{e'\} = \{(i', j')\}$), and strengthens that formulation by applying these facts that the specialization permits: (a) an optimal solution must have $i' \in S$ and $j' \in T$, and (b) any diversion path must include e'.

Indices and Index Sets: Previous definitions apply here.

Parameters:

- w_{ij} interdiction cost, $w_{ij} \in \mathcal{Z}^+$ for all $(i,j) \in E$, except $w_{i'j'} \equiv 0$
- ε a small penalty on the length of a diversion path, $\varepsilon > 0$
- d_i supply and demand values for flow: $d_s = 1, d_t = -1$ and $d_i = 0$ for $i \in V \backslash \{s,t\}$

Variables:

- α_i 1 if vertex $i \in T$ of cutset $C = [S,T]$, 0 otherwise
- β_{ij} 1 if edge (i,j) is a forward edge of the cutset C (i.e., if $(i,j) \in E_C$), 0 otherwise
- y_{ij} > 0 if edge (i,j) is part of a diversion path from s to t, 0 otherwise
- Note: In the subsequent text, bold letters denote vectors of the corresponding variables.

Formulation P_1:

$$\min \sum_{(i,j)\in E} w_{ij}\beta_{ij} + \varepsilon \sum_{(i,j)\in E} y_{ij} \tag{1}$$

$$\text{s.t.} \quad \alpha_i - \alpha_j + \beta_{ij} \geq 0 \ \ \forall\, (i,j) \in E \tag{2}$$

$$\alpha_s = 0, \ \ \alpha_t = 1 \tag{3}$$

$$\alpha_{i'} = 0, \ \ \alpha_{j'} = 1 \tag{4}$$

$$\beta_{i'j'} = 1 \tag{5}$$

$$\sum_{j|(i,j)\in E} y_{ij} - \sum_{j|(j,i)\in E} y_{ji} = d_i \ \ \forall\, i \in V \tag{6}$$

$$y_{i'j'} = 1 \tag{7}$$

$$\beta_{ij} + \beta_{ji} + y_{ij} + y_{ji} \leq 1 \ \ \forall\, (i,j) \in E\backslash\{e'\},\ i < j \text{ and } (j,i) \in E \tag{8}$$

$$\beta_{ij} + y_{ij} \leq 1 \ \ \forall\, (i,j) \in E\backslash\{e'\},\ (j,i) \notin E \tag{9}$$

$$y_{ij} \geq 0 \ \ \forall\, (i,j) \in E \tag{10}$$

$$\alpha_i \in \{0,1\} \ \ \forall\, i \in V, \ \ \beta_{ij} \in \{0,1\} \ \ \forall\, (i,j) \in E \tag{11}$$

The objective function (1) minimizes the total interdiction cost for edges in the identified cut E_C, plus a small penalty for path length. The path-penalty term may be viewed as a symmetry-breaking device (e.g., Sherali and Smith [39]). Intuitively, it helps avoid enumerating certain solutions that are equivalent from the interdictor's point of view. (Specifying $\varepsilon < 1/|V|$ implies that the total penalty in any optimal solution is less than 1. Given that all relevant edge weights are positive integers, an optimal solution to this model identifies a minimal, minimum-weight cut and an associated diversion path with minimum cardinality.)

Constraints (2) and (3) identify an s-t cut E_C with cutset $[S,T]$ (e.g., [35, pp. 118–119]). Constraints (4) and (5) force inclusion of the diversion edge $e' = (i',j')$ in the cut E_C. Constraints (6) and (7) require that one unit of flow be sent from s to t through e', provided that other constraints eliminate the possibility of flow traversing a cycle containing e'. Constraints (8) and (9) allow an edge $(i,j) \neq e'$ to be an element of E_C or to form part of a diversion path, but not both. Of course, e' must be included in both.

The fixed variables in constraints (4), (5), and (7) apply because we assume a single diversion edge; they have no counterparts in Curet's original formulation. Constraints (8) are strengthened versions of constraints (9) that take advantage of antiparallel edges. Curet's original formulation does not account for potential antiparallel edges, but our new formulations P_2 and P_2^+ do, so we account for them in P_1, also.

We also note that Curet requires all variables to be binary, yet only $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ need be so. Furthermore, given a fixed, feasible $\boldsymbol{\alpha} \in \{0,1\}^{|V|}$, extreme-point solutions of the (restricted) linear-programming (LP) relaxation of P_1 automatically yield $\boldsymbol{\beta} \in \{0,1\}^{|E|}$. Thus, we can simply require $\boldsymbol{\alpha} \in \{0,1\}^{|V|}$, $\boldsymbol{\beta} \geq \mathbf{0}$ and $\mathbf{y} \geq \mathbf{0}$. Extensive testing shows that this combination of binary and continuous variables solves most efficiently.

5.2. A Two-Commodity MIP Formulation for DND, P_2

The LP relaxation of P_1 allows a fractional solution that is caused, at least in part, by flow around a cycle that contains e'. This difficulty can be eliminated by replacing modeling constructs for the single-diversion path by constructs for two separate "diversion subpaths," one from s to i', and one from j' to t. Indeed, no flow is then possible around a cycle that includes e' and, in effect, we have strengthened a single-commodity MIP by reformulating it for two commodities, one for each subpath. (See Wolsey [43] for similar reformulation techniques.) The improved formulation P_2 follows.

Additional Variables:

- y_{ij}^S > 0 if edge (i,j) is part of a diversion subpath from s to i', 0 otherwise
- y_{ij}^T > 0 if edge (i,j) is part of a diversion subpath from j' to t, 0 otherwise

Additional Parameters:

- d_i^S supply and demand for "commodity S": $d_s^S = 1, d_{i'}^S = -1$ and $d_i^S = 0$ for $i \in V\backslash\{s,i'\}$
- d_i^T supply and demand for "commodity T": $d_{j'}^T = 1, d_t^T = -1$ and $d_i^T = 0$ for $i \in V\backslash\{j',t\}$

Formulation P_2:

$$\min \sum_{(i,j)\in E} w_{ij}\beta_{ij} + \varepsilon \sum_{(i,j)\in E} \left(y_{ij}^S + y_{ij}^T\right) \tag{12}$$

s.t. constraints (2)–(5), (11)

$$\sum_{j|(i,j)\in E} y_{ij}^S - \sum_{j|(j,i)\in E} y_{ji}^S = d_i^S \ \ \forall\, i \in V \tag{13}$$

$$\sum_{j|(i,j)\in E} y_{ij}^T - \sum_{j|(j,i)\in E} y_{ji}^T = d_i^T \ \ \forall\, i \in V \tag{14}$$

$$\beta_{ij} + \beta_{ji} + y_{ij}^S + y_{ji}^S + y_{ij}^T + y_{ji}^T \leq 1 \quad \forall\, (i,j) \in E\backslash\{e'\} \mid (j,i) \in E \text{ and } i < j \tag{15}$$

$$\beta_{ij} + y^S_{ij} + y^T_{ij} \leq 1 \ \ \forall\, (i,j) \in E \backslash \{e'\} \mid (j,i) \notin E \tag{16}$$

$$y^S_{i'j'} \ = \ 0, \ \ y^T_{i'j'} = 0 \tag{17}$$

$$y^S_{ij}, y^T_{ij} \geq 0 \ \ \forall\, (i,j) \in E \tag{18}$$

The formulation for P_2 is easy to understand given the preceding explanation for P_1, so we omit a description.

The following proposition hints that P_2 has a general advantage over P_1. Let F_1 and F_2 denote the feasible sets for the LP relaxations of P_1 and P_2, respectively. Then, we can say that P_2 is at least as strong as P_1 if, after converting F_1 and F_2 into commensurate terms, we can show that $F_2 \subseteq F_1$ (Bertsimas and Weismantel [3, p. 12]).

Proposition 4. *P_2 is at least as strong as P_1.*

Proof. Consider an arbitrary feasible solution $(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}, \hat{\mathbf{y}}^S, \hat{\mathbf{y}}^T)$ to F_2. Construct a new vector $\hat{\mathbf{y}} = \hat{\mathbf{y}}^S + \hat{\mathbf{y}}^T + \mathbf{e}_{i'j'}$, where $\mathbf{e}_{i'j'}$ is the unit vector of length $|E|$ with a 1 in the position associated with $e' = (i', j')$. It is easy to verify that $(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}, \hat{\mathbf{y}})$ defines a feasible solution to the LP relaxation of P_1. Because we begin with an arbitrary feasible solution from F_2 in this construction, it is clear that $F_2 \subseteq F_1$. ∎

5.3. An Improved Two-Commodity Formulation for ND, P_2^+

P_2 splits the single diversion path of P_1 into two separate subpaths. Let $[S, T]$ denote the *s*-*t* cutset associated with a feasible solution for P_2, and let $\mathbf{y}^S$ and $\mathbf{y}^T$ be feasible for that cutset. Then, the following properties must hold: (a) $y^S_{ij} > 0 \Rightarrow \{i,j\} \subset S$, and (b) $y^T_{ij} > 0 \Rightarrow \{i,j\} \subset T$. These properties lead to the following valid inequalities for P_2:

$$\sum_{j|(i,j)\in E} y^S_{ij} + \alpha_i \leq 1 \ \ \forall\, i \in V \backslash \{t\} \tag{19}$$

$$\sum_{j|(i,j)\in E} y^T_{ij} - \alpha_i \leq 0 \ \ \forall\, i \in V \backslash \{s\} \tag{20}$$

"P_2^+" denotes P_2 with these valid inequalities added.

Because P_2^+ simply adds valid inequalities (19) and (20) to P_2, P_2^+ is at least as strong as the former model (and P_1). Empirical results below show strict improvement is possible for P_2^+ over P_2 and for both P_2^+ and P_2 over P_1.

6. COMPUTATIONAL RESULTS

This section demonstrates the computational advantages and empirical strengths of the new network-diversion formulations P_2 and P_2^+ compared to the original formulation P_1. Wolsey [43] defines "percentage of the duality gap closed" as a reasonable measure of the strength of a reformulated MIP P' compared to a "baseline MIP" P:

$$\zeta(P, P') = 100\% \cdot \frac{z_{LP}(P') - z_{LP}(P)}{z^* - z_{LP}(P)}, \tag{21}$$

where $z_{LP}(P)$ and $z_{LP}(P')$ denote the optimal objective-function values for the LP relaxations of P and P', respectively, and where z^* denotes the optimal objective value for problem P. For a wide variety of network topologies and sizes, we report $\zeta(P_1, P_2)$ and $\zeta(P_1, P_2^+)$ to compare strengths of the three formulations, and we report computational times to compare practical efficiencies.

Computational tests cover four network types, two artificially constructed "structured networks" and two extracted from real-world data. The structured "grid" and "star-mesh" networks have been used by other authors for testing other solution methods for ND, so they provide useful, direct comparisons to those earlier methods. Tests on road networks and communications networks will indicate whether or not our new methods can help solve practical applications.

The network of primary and secondary roads in California defines the largest real-world network tested here, comprising 10,770 vertices and 11,975 undirected edges. The smallest real-world network tested is a fiber-optic communications network in the Seattle metropolitan region: it comprises 221 vertices and 441 undirected edges. We ensure that tests on structured networks more than cover that range of network sizes. Specifically, the smallest structured networks have about 100 vertices and 400 directed edges, and the largest about 90,000 vertices and 360,000 directed edges.

We carry out all computational tests on a 64-bit workstation with 12 GB of RAM and four 2.27-GHz processors running under version 2.6.32 of the Linux operating system. AMPL 12.1 ([20, pp. 203-217], [26]) generates the models and CPLEX 12.1 [27] solves them. A relative optimality tolerance of 1% applies, solution time is limited to 1 h for each problem instance, and $\varepsilon \equiv 1/100,000$. (As required, $\varepsilon \leq 1/|V|$ for all test problems.) Solver parameters are set separately for each formulation based on guidance from CPLEX's automated "tuning tool" [27, pp. 295–311]. Instructed to minimize heuristically the maximum solution time across a set of tuning-problem instances, this tool produces parameter settings that favor the solution of as many test-problem instances as possible given the per-instance computational limit of one hour. The tool identifies these non-default CPLEX parameter settings: "heuristicfreq −1," "cutpass 1," "probe −1" and "varselect 4" for P_1, and "heuristicfreq −1" and "varselect 4" for both P_2 and P_2^+; see [26, pp. 53–74] for descriptions of these parameters.

6.1. Structured Test Networks

This section describes tests on weighted and unweighted grid and star-mesh networks. Curet [12], Cintron-Arias et al. [9], and Yang and Park [45] test only grid networks while Erken [15] and Cho [8] test both types. Edge weights are drawn from a discrete uniform distribution on [1,5] as in [15]. (Curet [12] uses a discrete uniform distribution on [1,10]. Solution times using this distribution have more variability, but the conclusions reached regarding one formulation solving faster than another do not change.) We generate ten instances for each chosen size of grid and star-mesh network

TABLE 2. Solution statistics for P_1, P_2, and P_2^+ on weighted and unweighted, grid, and star-mesh networks.

			P_1 soln. stats			P_2 soln. stats			P_2^+ soln. stats				
Weighted?/ Topology	$\|V\|$	$\|E\|$	Avg. (s)	S.d. (s)	No. solved	Avg. (s)	S.d. (s)	No. solved	Avg. (s)	S.d. (s)	No. solved	$\zeta(P_1,P_2)$	$\zeta(P_1,P_2^+)$
Unweighted													
Grid 10	102	380	0.0	0.0	10	0.0	0.0	10	0.0	0.0	10	None	None
Grid 20	402	1,560	0.1	0.1	10	0.1	0.1	10	0.1	0.1	10	None	None
Grid 30	902	3,540	1.4	1.4	10	0.3	0.1	10	0.4	0.2	10	None	None
Grid 50	2,502	9,900	33.7	93.0	10	0.9	0.5	10	1.0	0.4	10	None	None
Grid 100	10,002	39,800	308.9	643.3	10	6.2	3.3	10	8.5	4.1	10	None	None
Grid 200	40,002	159,600	–	–	9	56.3	12.9	10	93.8	32.3	10	None	None
Grid 300	90,002	359,400	–	–	8	387.8	129.8	10	648.1	212.4	10	None	None
Weighted													
Grid 10	102	380	0.2	0.1	10	0.1	0.1	10	0.1	0.1	10	49.7%	49.7%
Grid 20	402	1,560	17.8	27.8	10	0.9	1.3	10	0.4	0.2	10	24.2%	25.7%
Grid 30	902	3,540	–	–	7	2.7	1.7	10	1.2	0.9	10	18.2%	18.8%
Grid 50	2,502	9,900	–	–	3	115.1	302.0	10	4.6	2.4	10	11.9%	12.3%
Grid 100	10,002	39,800	–	–	0	200.1	137.4	10	50.1	39.6	10	16.2%	18.5%
Grid 200	40,002	159,600	–	–	0	–	–	4	–	–	8	–	–
Grid 300	90,002	359,400	–	–	0	–	–	1	–	–	4	–	–
Unweighted													
SM 10	101	400	0.4	0.4	10	0.3	0.3	10	0.2	0.1	10	30.9%	30.9%
SM 15	226	900	26.7	39.5	10	3.4	4.3	10	2.0	2.6	10	20.6%	28.7%
SM 20	401	1,600	–	–	6	405.4	496.7	10	39.2	49.5	10	25.9%	32.3%
SM 25	626	2,500	–	–	2	–	–	3	719.4	654.5	10	12.3%	22.7%
SM 30	901	3,600	–	–	2	–	–	3	–	–	3	–	–
Weighted													
SM 10	101	400	0.6	0.2	10	0.6	0.5	10	0.2	0.1	10	18.4%	23.1%
SM 15	226	900	84.0	117.3	10	5.9	6.1	10	1.5	1.3	10	13.7%	26.1%
SM 20	401	1,600	–	–	5	388.1	504.3	10	12.5	9.5	10	11.4%	22.2%
SM 25	626	2,500	–	–	3	–	–	7	78.9	96.1	10	12.6%	23.3%
SM 30	901	3,600	–	–	0	–	–	1	–	–	8	–	–

Each row represents 10 randomly generated problem instances of the indicated indicated topology: "Grid n" and "SM n" correspond to an $n \times n$ grid network and an $n \times n$ star-mesh network, respectively; E includes the edge that is antiparallel to e'. Each row displays average solution time in seconds ("avg."), the standard deviation of those times ("s.d."), and the number of problems solved within the 1-h time limit ("no. solved"). Columns 13 and 14 show the average duality gap closed by P_2 and P_2^+, respectively, relative to P_1; see Equation (21). P_1 has no duality gap for unweighted grid networks, and "none" signifies this. [Equation (21) is undefined in this case.]

using the programs developed by Balcioglu and Wood [2] and by Erken [15], respectively.

The grid networks (see Fig. 3) approximate road networks with many intersections and potential routes from one side of the network to the other (Xie and Levinson [44]). Their design corresponds to that used in [12] and [15], although neither of these papers specifies placement and orientation of diversion edges. For a weighted grid network with n rows and n columns of vertices—rows and columns are ordered here from bottom to top and left to right, respectively—we place the diversion edge $e' = (i',j')$ such that i' is in row $r = \lceil \frac{n}{2} \rceil$ and column $c = \lfloor \frac{n}{2} \rfloor$, and j' is in row r and column $c - 1$. The diversion edge (i',j') is chosen randomly for unweighted grid networks, except that $i' = t$ and $j' = s$ are disallowed.

In the weighted grid networks, the orientation for the diversion edge, "toward s," opposes the general orientation of a diversion path, and this tends to make the problems more difficult to solve: typically, this orientation increases the number of edges in an optimal s-t cut, and empirical results show that number to be positively correlated with solution difficulty. For instance, when we reverse the orientation of the diversion edge for the 100×100 weighted grid networks covered in row 7 of Table 2, solutions average about three times faster for P_2 and about 2.5 times faster for P_2^+. (Independent of the diversion edge's orientation, however, P_1 cannot solve any of these instances within the 1-h time limit.)

We test star-mesh networks similar in design to those tested in [15] and [8]; see Figure 4 for an example, and see Miller [34] for a detailed description of this network type. A version of this topology has been proposed for wireless sensor networks [6], but practical sensor networks would probably be much smaller than the star-mesh networks tested here. For each weighted or unweighted problem instance, we disallow $i' = t$ and $j' = s$, but otherwise randomly select e' from among ray edges directed toward s. (Neither [15] nor [8] specify how e' is chosen.)

For each network type and size, Table 2 presents solution statistics, including the average duality gap closed by P_2 and P_2^+ relative to P_1. Table 3 provides detailed results on the ten 100×100 weighted-grid-network test instances to show

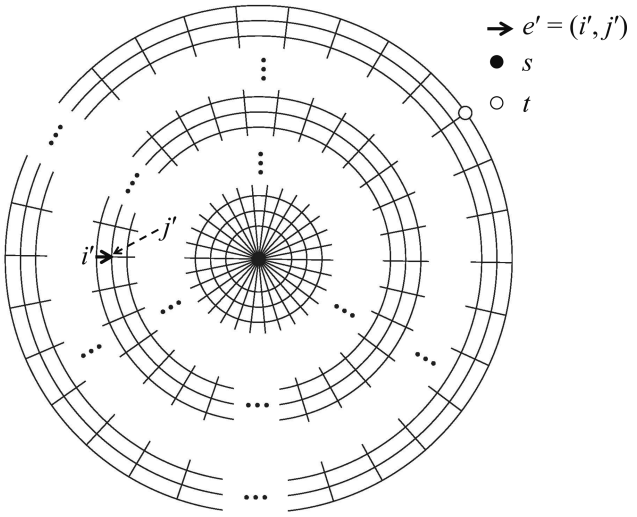


FIG. 4. An $n \times n$ star-mesh network (with n rays and n rings). Crossing line segments represent a vertex. Except for the diversion edge, each line segment between two vertices represents two, directed, antiparallel edges.

the ranges for solution times, objective-function values, and duality-gap improvements.

We note that the LP relaxation for each model can be solved, so duality gaps and their improvements can be computed as long as at least one MIP formulation identifies z^*. In fact, given the nonzero optimality tolerance, the best objective value found, z_{MIP}, may not equal z^*. Because the tolerance at 1% is small, however, little error results in replacing z^* with z_{MIP} in duality-gap calculations, so we make that replacement.

Results may be summarized as follows. P_2 and P_2^+ close the average duality gap relative to P_1 by 20 and 27% on unweighted star-mesh networks, and on weighted star-mesh networks by 19 and 24%, respectively. P_1 exhibits no duality gap on unweighted grid networks. Given the 1-h time limit for each problem instance, P_1 solves 155 of 240 instances (65%), P_2 solves 199 (83%) and P_2^+ solves 223 (93%).

For a few, small, test instances of unweighted grid networks—these have no duality gap—P_1 can actually solve more quickly than P_2 or P_2^+. On larger unweighted grid networks, however, both P_2 and P_2^+ solve more quickly, because a near-optimal solution is found at or near the root node for those formulations, while P_1 often requires a great deal of enumeration to find such a solution. (For this class of problems only, P_2 is faster than P_2^+ because, apparently, CPLEX enumerates about the same number of nodes in the respective branch-and-bound trees, but the per-node computation time is higher for P_2^+ because of its extra constraints, i.e., valid inequalities.) In general, P_2 is faster than P_1, and P_2^+ is faster yet. Indeed, P_2^+ is often an order of magnitude faster than P_1 on these problems.

The reader may have noted that the grid networks have an s-t-planar topology, and could actually be solved in polynomial time. On the other hand, the star-mesh networks are planar, but not s-t-planar, and thus their theoretical complexity remains open. Clearly, DND on a star-mesh network seems much more difficult to solve than on a grid network of comparable size. Perhaps this points to a difference in theoretical complexity.

All tests described above use the AMPL/CPLEX option "heuristicfreq −1," meaning that CPLEX does not search for feasible solutions using its internal heuristic (because parameter-tuning shows it to be unproductive). For the topologies tested, however, a simple, fast, problem-specific heuristic might provide a feasible starting solution that could reduce solution times. We have carried out only preliminary computations with heuristics, however, because it is clear that no heuristic can eliminate the advantage provided by the tighter LP relaxations of P_2 and P_2^+ compared to P_1. For instance, in one test on a 25×25 unweighted star-mesh network, even when initialized with an optimal solution which a heuristic might just produce, P_1 fails to prove that solution

TABLE 3. Solution details on 100×100 weighted grid networks.

	Soln. time (s)				z_{LP}				
Instance	P_1	P_2	P_2^+	z_{MIP}	P_1	P_2	P_2^+	$\zeta(P_1, P_2)$	$\zeta(P_1, P_2^+)$
1	[68%]	193	133	242	228.0	230.4	230.8	16.9%	20.0%
2	[69%]	60	18	240	233.0	233.3	233.3	4.7%	4.7%
3	[3%]	223	18	228	222.0	224.5	225.1	41.7%	51.8%
4	[63%]	399	98	243	222.0	224.5	225.0	11.9%	14.3%
5	[35%]	184	37	244	230.0	230.7	230.7	4.8%	4.8%
6	[75%]	307	50	241	232.0	232.2	232.4	2.0%	4.4%
7	‡	416	32	237	223.0	224.0	224.0	7.1%	7.1%
8	[4%]	50	19	242	231.0	234.7	234.7	33.3%	33.3%
9	[44%]	121	74	243	229.0	231.3	231.3	16.7%	16.7%
10	[6%]	48	22	231	226.0	227.2	227.4	23.3%	27.7%
Average	–	200	50	NA	NA	NA	NA	16.2%	18.5%

The results illustrate the range of solution statistics for the 10 problem instances summarized in row 12 (Weighted, Grid 100) of the data in Table 2. P_1 fails to solve any of the 10 instances within the 1-h time limit: in column 2, a percentage in square brackets indicates the relative optimality gap remaining after 1 h of computation, and "‡" indicates no feasible solution was obtained in that time. "z_{MIP}" is the the best objective value found, ignoring the contribution from ε; this objective value is within 1% of the true optimum. "z_{LP}" is the optimal objective value of the respective model's LP relaxation.

TABLE 4. Input-data statistics for tests on road-network models for the U.S. states of Rhode Island, Colorado, and California.

	Original network		Reduced networks			
State	$\|V\|$	$\|E\|$	Min $\|V\|$	Min $\|E\|$	Max $\|V\|$	Max $\|E\|$
Rhode Island	1,364	1,838	622	2,128	628	2,146
Colorado	4,045	4,678	860	2,934	867	2,952
California	10,770	11,975	2,104	7,178	2,113	7,202

"Original network" represents data for what we view as an undirected network as extracted from U.S. Census data files. "Reduced networks" represents the corresponding data after applying topological reductions. The size of a reduced network depends on the choice of s, t, i', and j', so we report minimum and maximum sizes.

to be within 1% of optimality in over 4 h of computation. By contrast, P_2^+ solves this problem instance in about five minutes without the benefit of any initial solution.

Above, we have presented "parallel" computational results on two classes of structured networks. This is easy to do because similar techniques generate all of these networks. Our real-world test networks have completely different origins, however, and we adjust methods on the two classes differently. Consequently, below, we describe computational tests on these network classes separately.

6.2. Road Networks

Table 4 presents statistics for network models of the primary and secondary roads in U.S. states of Rhode Island, Colorado, and California, extracted from U.S. Census Bureau data [42]. These states range from the smallest measured by land area in the United States (Rhode Island) to the second largest by that measure in the contiguous 48 states (California); the Rhode Island and Colorado data might therefore represent networks in geographically constrained regions of a larger country, whereas the California data might represent the complete road network of a fairly large country. Appendix 1 describes how these data are created and manipulated, including how standard (e.g., series) topological reductions apply.

Note that $|E|$ specifies the number of undirected edges: we assume that traffic can flow in both directions on each road segment, and interdiction of any segment would stop traffic in both directions. Thus, tests here reflect solutions of UND. Appendix 2 provides a formulation of P_1 that is converted to handle undirected edges. Similar conversions of P_2 and P_2^+ apply, also.

Most of the problem instances here require that we first apply certain topological reductions to make the instances solvable. Series reductions appear to be crucial because (a) the original networks include many series segments, (b) this structure leads to an enormous number of near-optimal solutions, and (c) empirically, we find that many of these solutions must be explored during a branch-and-bound enumeration.

Table 5 presents computational results for these problems using the previously specified CPLEX parameters. The results parallel those seen above: in general, P_2 solves more quickly than P_1, and P_2^+ solves more quickly than P_2. The results also indicate that, in fact, P_2^+ can handle fairly large, real-world networks successfully, although one of the 30 unweighted California test instances does elude solution.

6.3. Communications Networks

To compare the network-diversion formulations on communications networks, we imagine the need to intercept communications in a fiber-optic network that covers a metropolitan area in some foreign country. For simplicity and reproducibility, we test on networks in the United States extracted from a fiber-optic network owned and operated by Zayo Group LLC [46]. Specifically, we create five basic test models by extracting a subset of Zayo's "Metro Z" network for the metropolitan area surrounding each of the cities of Seattle, Minneapolis, Denver, Phoenix, and Indianapolis. The network data are extracted from the compressed KML file "Zayo-US-Network-EXTERNAL-11-1-2012.kmz" obtained at the Zayo Group's website [47].

TABLE 5. Computational results for UND on topologically reduced road-network data for Rhode Island, Colorado, and California.

	P_1 soln. stats			P_2 soln. stats			P_2^+ soln. stats				
Weighted?/ Topology	Avg. (s)	S.d. (s)	No. solved	Avg. (s)	S.d. (s)	No. solved	Avg. (s)	S.d. (s)	No. solved	$\zeta(P_1,P_2)$	$\zeta(P_1,P_2^+)$
Unweighted											
Rhode Island	–	–	28	–	–	28	4.9	11.1	30	10.4%	12.0%
Colorado	–	–	22	9.3	20.7	30	3.5	2.7	30	19.3%	21.5%
California	–	–	26	–	–	28	–	–	29	–	–
Weighted											
Rhode Island	–	–	28	–	–	29	2.6	3.7	30	17.9%	23.5%
Colorado	–	–	22	9.2	20.4	30	3.5	2.7	30	19.3%	21.5%
California	–	–	27	–	–	28	23.7	99.3	30	30.3%	32.2%

Each row gives results averaged over 30 instances, for which s, t, and e' are chosen randomly. (Appendix 1 describes general data preparation and the rules used to choose these entities.) Table 2's caption defines the column entries.

TABLE 6. Input-data statistics on fiber-optic test networks, listed in order of increasing $|V|$.

City	Min lat. (N) (degrees)	Max lat. (N) (degrees)	Min long. (W) (degrees)	Max long. (W) (degrees)	$\|V\|$	$\|E\|$
Seattle, WA	47.50	47.75	122.25	122.40	221	241
Minneapolis, MN	44.90	45.10	93.00	93.40	562	668
Denver, CO	39.60	39.80	104.90	105.10	666	941
Phoenix, AZ	33.20	33.50	111.70	112.30	2,088	3,148
Indianapolis, IN	39.40	39.60	86.00	86.30	2,502	3,092

The network data are extracted from compressed KML data (KMZ format) as described in the text. An extracted network comprises vertices whose latitudes and longitudes fall within the specified region, and all edges with both endpoints in the region, with one exception: a vertex is deleted if it has no incident edges using this construction scheme.

(See [24] for background on the Keyhole Markup Language, and on the KML and KMZ file formats for that language.) For each city, Table 6 lists the latitude and longitude limits that define the metropolitan area for our purposes, and the resulting values for $|V|$ and $|E|$ for each extracted network. Note that our metropolitian areas are unofficial, but the roughly rectangular regions more than cover the relevant cities' boundaries.

Table 7 provides computational results for UND on the five networks for each of the three formulations. The results correspond to thirty feasible instances, each with a randomly selected source, sink, and diversion edge. Because P_2^+ successfully solves all instances of all five test problems, we have not applied any topological reductions in these tests. The results follow the same pattern as seen in other tests: P_2^+ is the best formulation, P_1 is the worst, and P_2 falls in between.

7. CONCLUSIONS

The network diversion problem (ND) specifies a "diversion edge" $e' = (i', j')$ in a graph $G = (V, E)$, along with a source vertex s, sink vertex t, and non-negative edge weights. It then seeks a minimum-weight, minimal s-t cut E_C in G such that $e' \in E_C$. "DND" and "UND" correspond to ND on directed and undirected graphs, respectively.

DND was known to be strongly NP-complete, but we present a new proof that provides more information on special topologies. For instance, the proof implies that DND is NP-complete even when the diversion edge is incident from s or into t (but not both), and even when G is acyclic. Additionally, we describe polynomial-time algorithms for DND and UND on s-t planar graphs, and show that a vertex-deletion version of UND is strongly NP-complete. The complexity of the nominal edge-deletion version of UND remains open, however.

This paper also presents a new MIP formulation for DND. "P_1" denotes the only previously known formulation, "P_2" denotes our new, basic formulation, and "P_2^+" denotes a strengthened variant. P_1 identifies (a) a minimum-weight cut E_C containing e', and (b) a "diversion path" from s to t that intersects the cut only at e'. The path, modeled using standard flow-balance constraints, ensures the minimality of E_C. The continuous relaxation of P_1 allows flow around a cycle that includes $e' = (i', j')$, which P_2 avoids by separating the diversion-path flow into two, mutually exclusive "commodities," namely, flow from s to i' and flow from j' to t. P_2^+ adds valid inequalities to P_2 that the two-commodity formulation enables.

We test all formulations on (a) artificially generated grid networks with up to 90,002 vertices and 359,400 directed edges, (b) artificially generated "star-mesh networks," (c) road networks from the states of Rhode Island, Colorado, and California, and (d) fiber-optic communications networks from five metropolitan areas in the United States. We successfully solve unweighted instances of the largest grid-network problems, although some some smaller real-world networks elude consistent solutions. For example, the California road-network problems have roughly 11,000 vertices and 12,000 undirected edges; these values reduce to about

TABLE 7. Computational results for UND on fiber-optic networks in five metropolitan areas of the United States.

	P_1 soln. stats			P_2 soln. stats			P_2^+ soln. stats				
City	Avg. (s)	S.d. (s)	No. solved	Avg. (s)	S.d. (s)	No. solved	Avg. (s)	S.d. (s)	No. solved	$\zeta(P_1,P_2)$	$\zeta(P_1,P_2^+)$
Seattle, WA	0.05	0.06	30	0.03	0.05	30	0.02	0.02	30	20.0%	27.7%
Minneapolis, MN	2.13	8.58	30	2.28	7.05	30	0.26	0.17	30	6.7%	21.4%
Denver, CO	95.10	318.99	30	122.24	438.15	30	0.70	0.94	30	10.4%	27.1%
Phoenix, AZ	–	–	29	–	–	29	3.12	9.78	30	18.7%	28.1%
Indianapolis, IN	–	–	24	–	–	26	2.06	1.82	30	21.4%	37.2%

Each row gives results averaged over 30 feasible instances, for which s, t, and e' are chosen randomly. Table 2's caption defines the column entries.

2,100 and 7,200, respectively, after topological reductions; all 30 weighted instances solve in about 30 s on average; but one of the unweighted instances cannot be solved in 1 h.

In brief, we find that the new formulation P_2^+ often solves an order of magnitude faster than does P_1, P_2 is usually faster than P_1, but rarely faster than P_2^+. Improved efficiency for P_2^+ can be attributed to its tighter linear-programming relaxation: typically, P_2^+ reduces the duality gap with respect to P_1 by 10–50%. We also demonstrate that P_2^+ can solve problems that cannot be solved by exact methods based on combinatorial Benders decomposition or enumeration of near-minimum-weight cuts. Thus, except for special cases or until improved formulations appear, P_2^+ should be viewed as the standard for solving network-interdiction problems. We further note that topological (e.g., series) reductions can prove key to solving certain sparse-network problems.

APPENDIX 1: ROAD-NETWORK TEST DATA

This appendix describes data preparation for the road-network tests described in Section 6.2.

Data for each state are taken from census data [42] in "shapefile" format. Using ArcGIS software [16], each file is converted to a "coverage" (Zeiler [48, pp. 4–5]), which, in turn, is converted to an "e00-format" text file (Sherman [40, pp. 148–149]). Nodes are identified by latitudes and longitudes taken out to three decimal places. This implies an accuracy of 100–200 m, so some short road segments are merged into single vertices; all resulting self-loops are deleted.

Once the basic conversion above is made for a given state, the algorithm below creates 30 different problem instances. The algorithm uses this definition: two undirected edges $e_1 = (i,j)$ and $e_2 = (j,k)$, $i \neq k$, are series edges if degree(j) = 2, that is, if no other edges are incident to j.

1. **Begin.** Select s randomly from among vertices in the northern third of the state, with the north-south range being measured in terms of minimum and maximum latitudes.
2. Choose t similarly, but from the southern third of vertices in the state.
3. View the original network as undirected and merge any parallel edges to create $G = (V,E)$. (We assume that interdiction of undirected edge (i,j) would halt traffic flowing on both underlying directed edges (i,j) and (j,i). Appendix B shows how to modify P_1 to handle undirected edges; similar modifications apply to P_2 and P_2^+.)
4. Choose $e' = (i',j') \in E$ randomly such that both i' and j' lie in the middle third of the latitude range for the state.
5. Using a network-flow model, test feasibility of UND and return to Step 3 if the problem is infeasible.
6. For all $e \in E\backslash\{w_{e'}\}$, if the model is unweighted, set $w_e = 1$, and otherwise choose w_e randomly as a uniformly distributed integer in $[1,5]$.
7. Recursively delete any vertex $i \in V\backslash\{s,t\}$ with degree(i) = 1, along with incident edges.
8. While any pair of series edges $e_1 = (i,j)$ and $e_2 = (j,k)$ remains in E, replace e_1 and e_2 with $e_3 = (i,k)$, define $w_{e_3} \equiv \min\{w_{e_1}, w_{e_2}\}$, and delete j. (Note: If two edges are in series, an optimal solution to UND would interdict at most one, and it would be the one with the smaller weight.)
9. Merge any new parallel edges $e_1 = (i,j)$ and $e_2 = (i,j)$ into a single edge $e_3 = (i,j)$ with weight $w_3 = w_1 + w_2$. (Note: Unlike initial parallel edges identified in Step 3, we assume that interdiction of parallel edges here would require separate attacks. This assumption seems reasonable as parallel edges at this step can result from reductions on edges that are distant except at intersections. By contrast, we find that most initially parallel edges lie in close proximity to each other. For instance, such edges can arise from a freeway segment and its adjacent segment of frontage road.)
10. Repeat Steps 8 and 9 recursively until no series or parallel reductions can be made.
11. The resulting network becomes one instance of a "reduced network" as listed in Table 5. **End.**

APPENDIX 2: FORMULATION P1 MODIFIED FOR AN UNDIRECTED GRAPH

This formulation converts P_1 from Section 5 to handle an undirected graph $G = (V,E)$, that is, to solve UND. The road-network tests in Section 6.2 use this formulation and analogous formulations for P_2 and P_2^+. Note that the formulation does specify that the diversion path should traverse from s to i', from i' across e' to j', and then from j' to t.

$$\min \sum_{(i,j)\in E} w_{ij}\beta_{ij} + \varepsilon \sum_{(i,j)\in E} (y_{ij} + y_{ji}) \tag{22}$$

s.t.

$$\alpha_i - \alpha_j + \beta_{ij} \geq 0 \ \ \forall\, (i,j) \in E \tag{23}$$

$$\alpha_j - \alpha_i + \beta_{ij} \geq 0 \ \ \forall\, (i,j) \in E \tag{24}$$

$$\alpha_s = 0, \ \ \alpha_t = 1 \tag{25}$$

$$\alpha_{i'} = 0, \ \ \alpha_{j'} = 1 \tag{26}$$

$$\beta_{i'j'} = 1 \tag{27}$$

$$\sum_{j \mid (i,j)\in E \text{ or } (j,i)\in E} (y_{ij} - y_{ji}) = d_i \ \ \forall\, i \in V \tag{28}$$

$$y_{i'j'} = 1, \ \ y_{j'i'} = 0 \tag{29}$$

$$\beta_{ij} + y_{ij} + y_{ji} \leq 1 \ \ \forall\, (i,j) \in E\backslash\{e'\} \tag{30}$$

$$y_{ij} \geq 0 \ \ \forall\, (i,j) \in E \tag{31}$$

$$\alpha_i \in \{0,1\} \ \ \forall\, i \in V, \ \ \beta_{ij} \in \{0,1\} \ \ \forall\, (i,j) \in E \tag{32}$$

Acknowledgments

The authors thank two anonymous referees for their helpful comments on the first draft of the paper, and thank Professor Douglas Shier for pointing out the work by Ringeisen and others on inclusive connectivity.

REFERENCES

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, Network flows: Theory, algorithms and applications, Prentice Hall, 1993.

[2] A. Balcioglu and R.K. Wood, "Enumerating near-min *s-t* cuts," Network interdiction and stochastic integer programming, D. Woodruff (Editor), Kluwer Academic Publishers, Boston, 2003, pp. 21–49.

[3] D. Bertsimas and R. Weismantel, Optimization over integers, Dynamic Ideas, Belmont, Massachusetts, 2005.

[4] J. Boland and R. Ringeisen, On super i-connected graphs, Networks 24 (1994), 225–232.

[5] K.S. Booth, Isomorphism testing for graphs, semigroups, and finite automata are polynomially equivalent problems, SIAM J Comput 7 (1978), 273–279.

[6] N. Boudriga and M. Obaidat, Mobility, sensing, and security management in wireless ad hoc sensor systems, Comput Elect Eng 32 (2006), 266–276.

[7] G.G. Brown, W.M. Carlyle, R.C. Harney, E.M. Skroch, and R.K. Wood, Interdicting a nuclear-weapons project, Oper Res 57 (2009), 866–877.

[8] D.H. Cho, An optimization algorithm for the network diversion problem using combinatorial Benders' cut, Master's Thesis, Department of Industrial Engineering, Korea Advanced Institute of Science and Technology, Daejon, Korea, 2009.

[9] A. Cintron-Arias, N. Curet, L. Denogean, R. Ellis, C. Gonzalez, S. Oruganti, and P. Quillen, A network diversion vulnerability problem, Technical report 1752, Institute for Mathematics and its Applications: Mathematical Modeling in Industry Summer 2000 Program for Graduate Students, Minneapolis, Minnesota, February 2001.

[10] G. Codato and M. Fischetti, Combinatorial Benders' cuts for mixed-integer linear programming, Oper Res 54 (2006), 756–766.

[11] D. Cribb, Stability properties of inclusive connectivity for graphs, Ph.D. Thesis, Clemson University, Clemson, South Carolina, 1993.

[12] N.D. Curet, The network diversion problem, Mil Oper Res 6 (2001), 35–44.

[13] de Verdière, É.C. and A. Schrijver, "Shortest vertex-disjoint two-face paths in planar graphs," 25th Int Symp Theoret Aspects Comput Sci (STACS), Dagstuhl, Germany, Vol. 1, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2008, pp. 181–192.

[14] N. Deo, Graph theory with applications to engineering and computer science, Prentice-Hall, Inc., Upper Saddle River, New Jersey, USA, 1974.

[15] O. Erken, A branch and bound algorithm for the network diversion problem, Master's Thesis, Naval Postgraduate School, Monterey, California, December 2002.

[16] ESRI, ArcGIS 9, what is ArcGIS?, ESRI, Redlands, California, 2004.

[17] F. Finelli, Transforming aerospace power, Airpower J 12 (1999), 4–14.

[18] L. Ford and D. Fulkerson, Maximal flow through a network, Can J Math 8 (1956), 399–404.

[19] S. Fortune, J. Hopcroft, and J. Wyllie, The directed subgraph homeomorphism problem, Theoret Comput Sci 10 (1980), 111–121.

[20] R. Fourer, D. Gay, and B.W. Kernighan, AMPL: A modeling language for mathematical programming, 2nd edition, Duxbury Press, Pacific Grove, California, 2003.

[21] M.R. Garey and D.S. Johnson, Computers and intractability: A guide to the theory of NP-completeness, W. H. Freeman & Co., New York, New York, 1979.

[22] A. Gayash, V. Viswanathan, and D. Padmanabhan, SQUARE-lite: Case study on VADSoft project, Special report CMU/SEI-2008-SR-017, Carnegie Mellon Software Engineering Institute, Pittsburgh, Pennsylvania, 2008.

[23] F. Glover, Tabu search – Part I, ORSA J Comput 1 (1989), 190–206.

[24] Google Developers KML tutorial, webpage. Available at: https://developers.google.com/kml/documentation/kml_tut. Accessed 21 January 2013.

[25] T. Harris and F. Ross, Fundamentals of a method for evaluating rail net capacities, Research memorandum RM-1573, The RAND Corporation, 1955.

[26] IBM, IBM ILOG AMPL Version 12.1 User's Guide, International Business Machines Corporation, 2009.

[27] IBM, IBM ILOG CPLEX V12.1 User's Manual for CPLEX, International Business Machines Corporation, 2009.

[28] B.A. Jackson, P. Chalk, K. Cragin, B. Newsome, J.V. Parachini, W. Rosenau, E.M. Simpson, M. Sisson, and D. Temple, Breaching the fortress wall: Understanding terrorist efforts to overcome defensive technologies, RAND Corporation, Santa Monica, California, 2007.

[29] R. Karp, "Reducibility among combinatorial problems," Complexity of computer computations, R. Miller and J. Thatcher (Editors), Plenum Press, New York, New York, 1972, pp. 85–103.

[30] B. Korte and J. Vygen, Combinatorial optimization: Theory and algorithms, 3rd edition, Springer-Verlag, Berlin-Heidelberg, Germany, 2006.

[31] M.E. Krause, Decision dominance: Exploiting transformational asymmetries, Defense Horizons 23 (2003), 1–8.

[32] M.S. Laughton, "Operational fires in support of counterdrug campaigns," Working paper, Department of Joint Military Operations, Naval War College, Newport, Rhode Island, 1996.

[33] A.N. Letchford and N.A. Pearson, A fast algorithm for minimum weight odd circuits and cuts in planar graphs, Oper Res Lett 33 (2005), 625–628.

[34] L.E. Miller, Catalog of network connectivity models, webpage. Available at: http://w3.antd.nist.gov/wctg/netanal/netanal_netmodels.html. Accessed 14 July 2011.

[35] C.H. Papadimitriou and K. Steiglitz, Combinatorial optimization: Algorithms and complexity, Dover Publications, Mineola, New York, 1998.

[36] C.A. Phillips, "The network inhibition problem," Proc Twenty-Fifth Ann ACM Symp Theory Comput, San Diego, California, 1993, pp. 776–785.

[37] R. Ringeisen and M. Lipman, Cohesion and stability in graphs, Discr Math 46 (1983), 191–198.

[38] A. Schrijver, Finding *k* disjoint paths in a directed planar graph, SIAM J Comput 23 (1994), 780–788.

[39] H.D. Sherali and J.C. Smith, Improving discrete model representations via symmetry considerations, Manag Sci 47 (2001), 1396–1407.

[40] G. Sherman, The geospatial desktop, Locate Press, Williams Lake, British Columbia, Canada, 2012.

[41] Y. Shiloach, The two paths problem is polynomial, Technical report STAN-CS-78-654, Stanford University, Stanford, California, 1978.

[42] U.S. Census, 2011 TIGER/Line shapefiles, webpage. Available at: ftp://ftp2.census.gov/geo/tiger/TIGER2011/ROADS/. Accessed 10 March 2012.

[43] L. Wolsey, Strong formulations for mixed integer programming: A survey, Math Program 45 (1989), 173–191.

[44] F. Xie and D. Levinson, Topological evolution of surface transportation networks, Comput Environ Urban Syst 33 (2009), 211–223.

[45] H. Yang and S. Park, A tabu search algorithm for the network diversion problem, J Mil Oper Res Soc Korea 30 (2004), 30–47.

[46] Zayo Group, Form 10k (annual report), webpage. Available at: http://www.zayo.com/sites/default/files/Annual%20Report%20FY%202012.PDF. Accessed 18 January 2013.

[47] Zayo Group, U.S. network KMZ (11-01-12), webpage. Available at: http://www.zayo.com/sites/default/files/images/Zayo-US-Network-EXTERNAL-11-1-2012.kmz. Accessed 18 January 2013.

[48] M. Zeiler, Modeling our world: The ESRI guide to geodatabase design, Environmental Systems Research Institute, Inc., Redlands, California, 2000.