UCRL-TR-201959

Optimal Resource Allocation in Electrical Network Defense

Y. Yao and T. Edmunds
Lawrence Livermore National Laboratory

Dimitri Papageorgiou
University of Texas at Austin

Rogelio Alvarez
Naval Post Graduate School, Monterey, California

December 2003

U.S. Department of Energy
Lawrence Livermore National Laboratory

DISCLAIMER

This work was performed under the auspices of the U. S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.

Table of Contents

TABLES

1 INTRODUCTION

Infrastructure networks supplying electricity, natural gas, water, and other commodities are at risk of disruption due to well-engineered and coordinated terrorist attacks. Countermeasures such as hardening targets, acquisition of spare critical components, and surveillance can be undertaken to detect and deter these attacks. Allocation of available countermeasures resources to sites or activities in a manner that maximizes their effectiveness is a challenging problem. This allocation must take into account the adversary's response after the countermeasure assets are in place and consequence mitigation measures the infrastructure operation can undertake after the attack. The adversary may simply switch strategies to avoid countermeasures when executing the attack. Stockpiling spares of critical energy infrastructure components has been identified as a key element of a grid infrastructure defense strategy in a recent National Academy of Sciences report [1].

Consider a scenario where an attacker attempts to interrupt the service of an electrical network by disabling some of its facilities while a defender wants to prevent or minimize the effectiveness of any attack. The interaction between the attacker and the defender can be described in three stages:

1) The defender deploys countermeasures,
2) The attacker disrupts the network, and
3) The defender responds to the attack by rerouting power to maintain service while trying to repair damage.

In the first stage, the defender considers all possible attack scenarios and deploys countermeasures to defend against the worst scenarios. Countermeasures can include hardening targets, acquiring spare critical components, and installing surveillance devices. In the second stage, the attacker, with full knowledge of the deployed countermeasures, attempts to disable some nodes or links in the network to inflict the greatest loss on the defender. In the third stage, the defender re-dispatches power and restores disabled nodes or links to minimize the loss. The loss can be measured in costs, including the costs of using more expensive generators and the economic losses that can be attributed to loss of load.

The defender's goal is to minimize the loss while the attacker wants to maximize it. Assuming some level of budget constraint, each side can only defend or attack a limited number of network elements. When an element is attacked, it is assumed that it will be totally disabled. It is assumed that when an element is defended it cannot be disabled, which may mean that it will be restored in a very short time after being attacked.

The rest of the paper is organized as follows. Section 2 will briefly review literature related to multilevel programming and network defense. Section 3 presents a mathematical formulation of the electrical network defense problem. Section 4 describes the solution algorithms. Section 5 discusses computational results. Finally, Sec. 6 explores future research directions.

2 LITERATURE REVIEW

The infrastructure network defense problem can be formulated as a tri-level optimization model, which is an extension of the bi-level program or Stackelberg game [2, 3]. In the two-stage game, a leader acts first to deploy countermeasures; then a follower in the game observes the countermeasure deployment and chooses a strategy with maximal system impact.

Algorithms have been developed to address instances of the bi-level programming problem [4-14]. Bard, Moore, and Edmunds replace the bi-level program by an equivalent single-level nonlinear program using Kuhn-Tucker optimality conditions [4, 5, 12]. The equivalent nonlinear optimization problem is then solved with a branch and bound scheme, where each branch corresponds to a complementary slackness condition. Israeli and Wood use Bender's decomposition and set covering methods to solve bi-level linear programs in which only the upper bound of the follower's objective function is affected by the leader's decisions [13, 14]. Israeli and Wood have extended their solution algorithm to tri-level linear programs with special properties.

Salmeron, Wood, and Baldick [15] formulate the electrical network interdiction problem (the attacker's problem) as a bi-level program. Their formulation differs from Israeli and Wood's in that the leader's decision will affect not only the upper bounds but also other constraints in the follower's problem. They develop heuristics to generate good solutions to the bi-level program by approximating the leader's objective with a penalty function generated from the solution to the follower's problem. The heuristic does not guarantee an optimal solution.

This paper extends Salmeron, Wood, and Baldick's bi-level model to a tri-level one in formulating the electrical network defense problem. The algorithm generates an optimal solution to the tri-level programming problem.

3 PROBLEM FORMULATION

As shown in Section 1, the electrical network defense problem consists of three stages or levels. This section will present the mathematical formation for this problem level by level, starting with the last one.

3.1 Optimal Power Flow

To mitigate the impact of an attack, the defender needs to produce and distribute power with the undamaged generators and transmission facilities. The objective is to minimize the generation cost and the cost due to unmet demands. This is an AC optimal power flow (OPF) problem, which is nonlinear. It is common practice in electric utility management to use a linear approximation of the AC OPF problem for reliability and planning purposes [19]. That is the approach taken in this paper.

The objective function, which reflects costs of operation and unmet demands, can be expressed as

$$f = \sum_{i \in G,D} c_i z_i \tag{1}$$

where

z_i is a variable representing power flow from a generator or along a transmission line, the phase angle at a bus, or the unmet amount of a demand,
c_i denotes the unit cost associated with z_i,
G is the set of indices of all variables for power from generators, and
D is the set of indices of all variables for unmet demands.

There are several sets of constraints. The first one describes the power-flow and phase angle relationship:

$$z_i - B_i(z_{\theta o(i)} - z_{\theta t(i)}) = 0 \quad \forall i \in L \tag{2}$$

where L is the set of indices of all variables for line power flows,
B_i is susceptance for line i,
θo (i) is the variable index for the phase angle at the bus where line i originates, and
θt (i) is the variable index for the phase angle at the bus where line i terminates.

The second set of constraints provides lower and upper bounds for the variables:

$$l_i \le z_i \le u_i \quad \forall i \in A, D, G, L \tag{3}$$

where A is the set of indices of all phase angles variables.

More specifically, variables in different sets are limited by the following lower and upper bounds.

$$\begin{aligned} l_i &= -\pi, u_i = \pi, \quad \forall i \in A \\ l_i &= 0, u_i = d_i, \quad \forall i \in D \\ l_i &= 0, u_i = \overline{p}_i, \quad \forall i \in G \\ l_i &= -\overline{p}_i, u_i = \overline{p}_i, \quad \forall i \in L \end{aligned}$$

where d_i is the power requirement at a demand point and $\overline{p}_i$ represents the capacity of a generator or a transmission line.

Finally, there are constraints for conservation of flows:

$$\sum_{i \in G(b)} z_i + \sum_{i \in L_t(b)} z_i - \sum_{i \in L_o(b)} z_i + \sum_{i \in D(b)} z_i = \sum_{i \in D(b)} d_i \quad \forall b \in B \tag{4}$$

where B includes all buses,
G(b) is the index set for all generators at bus b,

$D(b)$ is the index set for all demands at bus b,
$L_o(b)$ is the index set for all lines originating at bus b,
$L_t(b)$ is the index set for all lines terminating at bus b, and
d_i is the demand for real power at bus i.

In summary, the defender's response model is an OPF problem.

(OPF)

$$\min_{z} f(z)$$
subject to: (2) to (4)

3.2 Attacker's Problem

The attacker will decide which network element to attack. Let $y_i = 1$ if element i is attacked and $y_i = 0$ if it is not. The attacker's decision will impact the optimal power flow (OPF) model. Two sets of equations in problem OPF need to be modified. Equations (2) will become

$$\prod_{j \in J(i)} (1 - y_j) z_i - B_i (z_{\theta o(i)} - z_{\theta t(i)}) = 0 \qquad \forall i \in L \tag{5}$$

where $J(i)$ is the set of indices of y_i's that can make $z_i = 0$.

Equations (3) will become

$$\prod_{j \in J(i)} (1 - y_j) l_i \le z_i \le \prod_{j \in J(i)} (1 - y_j) u_i \qquad \forall i \in A, D, G, L \tag{6}$$

For convenience, (6) has included phase angles $(z_i, \forall i \in A)$ and unmet demands $(z_i, \forall i \in D)$. These variables cannot be directly affected by the attacker; however (6) will still be true if the corresponding y_i's are fixed at zero.

The parameterized optimal power flow model is as follows.

$$\text{OPF}(y) = \min_{z} f(z)$$
subject to: (4) to (6)

The attacker is limited by a budget.

$$\sum_{\forall j \in J} p_j y_j \le c \tag{7}$$

where J is the set of indices of all variables representing an attackable element,

p_j is the amount of resources needed to attack element j, and
c is total amount of available resources.

There are a number of logical relationships between attack variables that are reasonable to presume [15]. The first is that an attacker could disable all lines on a given tower by destroying the tower, and therefore in an optimal solution only one y_j needs to be one in a set of y_j's for a group of parallel lines.

$$\sum_{j \in L^y \cap paralell} y_j \leq 1 \quad \text{for all groups of parallel lines} \tag{8}$$

where L^y represents all attackable elements in the set L. In general, the superscript y will denote a subset for all attackable elements from a corresponding set in the optimal flow model.

The second set of constrains is

$$y_g + y_b \leq 1 \quad \forall g \in G^y(b), \ \forall b \in B^y \tag{9}$$

Equations (9) impose the logic that y_g and y_b cannot both be 1 in an optimal solution since attacking a bus will disable all generators attached to it. Similar constraints follow.

$$y_l + y_b \leq 1 \quad \forall l \in Lo^y(b), Lt^y(b), \forall b \in B^y \tag{10}$$

Equations in (10) make sure that y_l and y_b cannot both be 1 in an optimal solution since attacking a bus will disable its lines as well.

$$y_b + y_s \leq 1 \quad \forall b \in B^y, \forall s \in S^y \tag{11}$$

where S^y represents all attackable substations. Equations (11) reflect the fact that y_s and y_b cannot both be 1 in an optimal solution since attacking a substation will disable all of its buses.

The attacker's problem can be formulated as follows:

$$\max_{y} \text{OPF(y)}$$
$$\text{subject to:} \quad (7)\text{-}(11)$$

Expanding OPF(y), we have an explicit bi-level model for the attacker's problem.

(AP)

$$\max_{y} \text{OPF(y)}$$
$$\text{subject to} \quad (7)\text{-}(11)$$
$$\text{OPF}(y) = \min_{z} f(z)$$
$$\text{subject to: (4) to (6)}$$

Note that the problem is nonlinear due to the multiplicative terms in (5) and (6) of OPF(y).

3.3 Defender's Problem

The decision whether or not to defend a network element can be represented by a binary variable, x_k, where $x_k = 1$ if the element is defended, and $x_k = 0$ otherwise. The defender's decision impacts the attacker through the following set of constraints:

$$y_j \le \prod_{k \in K(j)} (1 - x_k) \forall j \in J \qquad (12)$$

where $K(j)$ is the set of all elements required for the attackable element j to function.

The parameterized attacker's problem is as follows:

AP(x) = max *OPF(y)*
y
subject to: (7) to (12)

Like the attacker, the defender also has a budget constraint:

$$\sum_{\forall k \in K} q_k x_k \le b \qquad (13)$$

where K is the set of indices of all variables representing a defendable element,
q_k is the amount of resources needed to defend element k, and
b is total amount of available resources.

There are two sets of logical constraints for the defender's problem. The following one is similar to (8).

$$\sum_{k \in L^x \cap parallel} x_k \le 1 \qquad \text{for all group of parallel lines} \qquad (14)$$

where L^x represents all defendable elements in the set L. In general, the superscript x will denote a subset for all defendable elements from a corresponding set in the optimal flow model.

The next constraint set is similar to (11).

$$x_b + x_s \le 1 \quad \forall b \in B^x, \forall s \in S^x \qquad (15)$$

where S^x represents all defendable substations. Equations (15) impose the logic that x_s and x_b cannot both be 1 in an optimal solution since defending a substation implicitly means that all of its buses will be defended as well.

The complete model for the defender's problem is

$$\begin{aligned} &\underset{x}{\text{Min}} \quad AP(x) \\ &\text{subject to:} \quad (13) \text{ to } (15) \end{aligned}$$

Expanding AP(x), we have an explicit tri-level formulation of the defender's problem.

(DP)

$$\begin{aligned} &\underset{x}{\text{Min}} \quad AP(x) \\ &\text{subject to:} \quad (13) \text{ to } (15) \\ &\qquad AP(x) = \max_{y} OPF(y) \\ &\qquad\qquad \text{subject to:} \quad (7) \text{ to } (12) \\ &\qquad\qquad\qquad OPF(y) = \min_{z} f(z) \\ &\qquad\qquad\qquad\qquad \text{subject to: } (4) \text{ to } (6) \end{aligned}$$

4 SOLUTION METHODS

The tri-level optimization model for electrical network defense can be viewed as a nested bi-level optimization model. Each of the bi-level problems is solved with a set packing/covering approach that is similar to the set covering based scheme in [13]. This approach is valid under the following assumption.

Assumption 1: The optimal power flow (OPF) model is always feasible for any feasible defense/attack plan.

This assumption is not an issue for our purpose since the OPF model can always be made feasible by introducing appropriate variables to represent unmet demands. In game theory terms, Assumption 1 guarantees that the inducible region is nonempty.

In this section, we will describe a set packing/covering solution, followed by enhancements in several algorithmic steps. First consider the interaction between the attacker and the defender. Without any defense ($\boldsymbol{x^0 = 0}$), the attacker would find the best attack plan, $\boldsymbol{y^0(x^0)}$ that inflicts the greatest possible loss on the defender. The defender can avoid the maximum loss by "covering" the attacker's plan, $\boldsymbol{y^0(x^0)}$. This can be accomplished the following constraint or cut:

$$\sum_{\forall k \in K1(y^0)} x_k \geq 1 \tag{16}$$

where $K1(\boldsymbol{y^0})$ contains all the indices of $\boldsymbol{x^0}$ with their corresponding components of $\boldsymbol{y^0}$ equal to one. The defender must set at least one the x_k's in (16) to 1 in the next defense plan, $\boldsymbol{x^1}$, the attacker will derive the new attack plan $\boldsymbol{y^1(x^1)}$, and a new cut will be created to cover $\boldsymbol{y^1(x^1)}$:

$$\sum_{\forall k \in K1(y^1)} x_k \geq 1 \tag{17}$$

Now the defender's problem has to satisfy both (16) and (17), in additional to (13), (14), and (15). The procedure will continue until not all cuts can be satisfied along with the other constraints; i.e. the defender cannot cover all attack plans proposed due to the defender's budget constraint. At this point, an optimal solution is found. This solution procedure will converge to an optimal solution in a finite number of steps since there are only a limited number of possible attack plans to cover.

In subsection 4.1, we provide an example to demonstrate the solution method. Subsection 4.2 describes the solution algorithm. The last two subsections will discuss special algorithms used in each of the steps in details.

4.1 A Simple Example

Consider an electric power grid with only three attackable components: a substation and two buses of equal value (denoted bus 1 and bus 2). Note that the substation is independent of the two buses, i.e. defending the substation will not simultaneously protect the buses. The defender has 3 units of resource to protect the grid while the attacker has 4 units of resource to attack it. Defending the substation requires three resources versus two resources for a single bus. The damage costs are five units for losing units of the substation and three units for losing an individual bus.

Table 1 Example Parameters

	available resources
Defender	3 units
Attacker	4 units

Component	**Damage**	**Attack/Defend Resources**
Substation	5	3
Bus 1	3	2
Bus 2	3	2

Below, we describe how the algorithm would proceed on this simple example.

Round 1

Attacker

Initially, the attacker seeks to maximize the amount of damage he inflicts. In so doing, he chooses to use all of his resources to attack bus 1 and bus 2, inflicting damage of six units.

Defender

The defender must then defend the network to minimize this maximum attack strategy. The defender chooses to use two of his resources to defend bus 1. Alternatively, he could have chosen to defend bus 2 since they are independent and virtually identical. Thus, the total damage has been reduced from six units to three units.

Round 2

Attacker

In the second iteration, the attacker again wishes to maximize damage. Since the defender chose to defend one of the buses, the attacker alters his strategy and attacks the substation, inflicting five units of damage at a cost of three resources.

Defender

In response, the defender attempts to obviate this attack and attempts to protect the substation at a cost of three resources. However, this defense is impossible, as the defender would exceed his budget or resource constraint.

Thus, the game terminates since the defender's problem has become infeasible. From Round 1, the defender will protect bus 1 (or bus 2, but not both) to prevent six units of damage. However, the attacker will then choose to attack the substation to inflict five units of damage. The whole process is summarized in the following table.

Table 2 Attack/Defense Game Summary

	Attacker			Defender		
	Strategy	Damage inflicted	Resources used	Strategy	Damage inflicted	Resources used
Round 1	Attack bus 1 & bus 2	6	4	Defend bus 1	3	2
Round 2	Attack substation	5	3	(Defend bus 1 & sub)	(0)	(2+3)

*Note that the defender's move in Round 2 is in parenthesis to denote that this move is infeasible.

It might seem surprising that the defender would choose to protect a single bus and "waste" a resource rather than defend the substation and prevent five units of damage. However, it is important to remember that the defender's goal is to interdict the attacker's optimal strategy, which in this case would be to attack both buses resulting in six units of damage.

This simple example also provides some insight into possible sensitivity analyses. For instance, we might like to know how the solution would have changed if the defender had had more resources. By evaluating the reduction in damage, we could find a shadow price for the defender's resources. Such an analysis might prove useful in real-world applications.

4.2 Solution algorithms

The algorithmic steps of the solution procedure are as follows.

Algorithm DP(DP, *x, *f**)**
Input: defender's problem, DP
Output: optimal defender's plan, ***x**** with an associated objective function value f*

Initialization: Construct relaxed defender's problem RDP(**x**): find x that satisfies (13) to (15)

Set the optimal defense objective value, $f^* = \infty$.
Set s = 0; $\boldsymbol{x}^0 = 0$

Step 1: Solve the attacker's problem AP($\boldsymbol{x}^s$) with Algorithm AP(AP($\boldsymbol{x}^s$), $\boldsymbol{y}^s$, f^s).
Step 2: If $f^s < f^*$, set $\boldsymbol{x}^* = \boldsymbol{x}^s$ and $f^* = f^s$.
Step 3: Add the following cut to RDP($\boldsymbol{x}$):

$$\sum_{\forall k \in K1(yt)} x_k \geq 1 \qquad (18)$$

Step 4: Set s = s+1.
Step 5: Solve RDP($\boldsymbol{x}$). If it is feasible, let the solution be $\boldsymbol{x}^s$ and go to Step 1.
Step 6: An optimal solution has been found; $\boldsymbol{x}^*$ is an optimal defense plan with objective value of f^*; and $\boldsymbol{y}^*$ is the optimal attack plan associated with $\boldsymbol{x}^*$.

Algorithm AP is used in step 1 to solve the attacker's problem:

Algorithm AP(AP(x), y^*, f^*)
Input: AP(x), the attacker's problem with a given defense plan $\boldsymbol{x}$.
Output: optimal attack plan $\boldsymbol{y}^*$ with an associated objective function value of value f^*.

Initialization: Construct relaxed attacker's problem
RAP($\boldsymbol{y}$): find $\boldsymbol{y}$ that satisfies (7) to (12)
Set the optimal attack objective value $f^* = -\infty.$,
Step : Set t = 0; $\boldsymbol{y}^0 = 0$

Step 1: Solve OPF($\boldsymbol{y}^t$) as a linear program and let the objective function value be f^t
Step 2: If $f^t > f^*$, set $y^* = \boldsymbol{y}^t$ and $f^* = f^t$
Step 3: Add the following cut to RAP($\boldsymbol{y}$):

$$\sum_{\forall k \in K1(zt)} y_k \geq 1 \qquad (19)$$

Step 4: Set t = t+1.
Step 5: Solve RAP($\boldsymbol{y}$). If it is feasible, let the solution be $\boldsymbol{y}^t$ and go to Step 1.
Step 6: (An optimal solution, y^*, has been found.) Return $\boldsymbol{y}^*$ and f^*.

In Algorithms DA and AP, most computation involves the solutions of three models, OPF in step 1 of AD, and RDP and RAP in step 5 of both DA and AD. The OPF model is a linear program, and both RDP and RAP models are integer programs. All of the models have special properties we exploit to improve computational efficiency.

4.3 Solution of Optimal Power Flow Model

Iteration t of Algorithm AP produces a parameterized OPF problem, OPF($\mathbf{y}^t$). If $\prod_{j \in J(i)} (1 - y_j) = 0$ in (5), z_i is free, and the corresponding constraint can be removed; otherwise the constraint will remain in the problem. If $\prod_{j \in J(i)} (1 - y_j) = 0$ in (6), z_i is fixed at 0; otherwise, it can change within the lower and upper bounds. Obviously, OPF($\mathbf{y}^t$) differs from OPF($\mathbf{y}^{t-1}$) only in the number of constraints of type (5) and in the bounds of the z_i's. To reduce computational effort, we used the dual simplex method to solve OPF($\mathbf{y}^t$) starting from the solution to OPF($\mathbf{y}^{t-1}$) except for t = 0.

To speed up Algorithm AP, we also attempted to use the following theorem [14].

Suppose, $\mathbf{z}_b$ is a feasible basic solution to OPF, then the following constraint can be added to RAP without changing the optimal solution as long as f($\mathbf{z}_b$) <= f* (the best optimal objective function value found so far).

$$\sum_{\forall k \in K1(zb)} y_k \geq 1 \tag{19}$$

An attempt was made to generate as many cuts as possible while solving OPF($\mathbf{y}$). However, almost all the cuts obtained this way were the same because all $\mathbf{z}_b$'s have the same non-zero components. This can be attributed to two factors: (1) the incremental dual simplex method tends to go through only a few of the feasible extreme points in a neighborhood of the previous optimal solution and they have the same non-zero decision variables, and (2) the LP solver we used, COIN CLP [17] makes the LP problem feasible close to the optimal solution. As a result, the attempt to generate many cuts in one solution of OPF was abandoned.

4.4 Solution of Set Packing/Covering Problem

Both the relaxed attacker's problem (RAP) and the relaxed defender's problem (RDP) are integer programs with special characteristics. Besides resource constraints (7) and (13) and lower and upper bounds, they have two types of constraints: set packing constraints (8)-(11) and (14)-(15) and set covering constraints (18) and (19). In this section we present algorithms to solve these Set Packing/Covering (SPC) problems.

The SPC problem can be solved as an integer program (IP) with a branch-and-bound algorithm. However, a generic IP algorithm does not exploit the special structure of the SPC problem. As new set covering constraints are added in the solution procedure as described in Sec. 4.1, more time is required to solve the SPC problem. Tests show that the IP algorithm approach is not fast enough for even small problems.

Israeli and Wood outline another approach [14]. In this approach, the SCP problem is first attacked by a simple, greedy heuristic; if it fails to find a feasible solution, the problem is solved with a generic IP algorithm (Greedy-then-IP). We used a greedy algorithm that is a modified version of an approximation algorithm for set covering [16]. The modification was made to

handle the set packing constraints. Testing showed that the Greedy-then-IP approach was more than three times faster than the generic IP algorithm.

To take advantage of the special properties of the SPC problem, we have developed an enumerative algorithm (ENUM-SPC) to search for a feasible solution or to prove its infeasibility. The search space is pruned by the budgetary and packing constraints. We describe the ENUM-SPC algorithm below.

Algorithm ENUM-SPC(SPC, x*)
Input: a set packing/covering (SPC) problem.
Output: a feasible solution to SPC, **x*** or an indicator (**x*** = 0) that the problem is infeasible.

Initialization: Arrange x_j's in non-decreasing order of their resource requirements (i.e., if $k > j$, then $c_k >= c_j$) and ignore all x_j's with $c_j > b$, the total amount of available resources. Assume that there are n variables remaining after the arrangement.

Compute the maximum search depth, $D = \left\lfloor \frac{b}{c_1} \right\rfloor$

where c_1 is the first and minimum of the c_j's.

The algorithm will try to find a non-zero variable for each level, up to level D. Let *NL* be a list of the indices of the non-zero variables.

Given non-zero variables for all of the previous levels, a level will have a max index, m, such that x_j must be 0 for all j > m because of the resource constraint. Let *ML* be a list of the max indices.

Set current search level, $d = 1$;
$NL(1) = 1$; $ML(1) = n$.

Step 1: status = Forward(*D*, *d*, *NL*, *ML*).
Step 2: If status = "solution found," go to 6
Step 3: Backward(*d*, *NL*, *ML*).
Step 4: If $d > 0$, go to step 1.
Step 5: The problem is infeasible; $\boldsymbol{x}^* = 0$.
Step 6: For all j, set $x_j = 1$ if $j \in NL$, otherwise, $x_j = 0$; $\boldsymbol{x}^* = \boldsymbol{x}$.

Algorithm Forward(*D*, *d*, *NL*, *ML*)
Input: *D* – max search depth
d – current search level
NL – list of indices of non-zero variables for levels up to *d*
ML – list of max indices for levels up to *d*

Output: status (indicator on whether a feasible solution has been found) and updated *d*, *NL* and *ML*.

Compute resources used on levels up to *d*, $r = \sum_{j=1}^{d} c_j$

Reduce the size of *NL* and *ML* to *d*.

//Find a level where a variable can be set to one.
For level = *d* + 1 to D
$i = \min(j \mid (1 \le j \le n) \cap j \notin NL \cap (c_j + r <= b) \cap ((j \cup NL) \in SP)$
(SP – convex set defined the packing constraints)
If i exists
NL(level) = i
$l = \max(j \mid (1 \le j \le n) \cap j \notin NL \cap (c_j + r <= b) \cap ((j \cup NL) \in SP)$
ML(level) = *l*
max_level = level
Else
max_level = level-1
End-If-Else
End-For
d = max_level

//Find a variable and set it to one while maintaining feasibility.
$i = \min(j \mid (1 \le j \le ML(d)) \cap j \notin NL \cap ((j \cap NL) \in SP) \cap ((j \cup NL) \in SC)$
(SC – convex set defined by the set covering constraints)
If i exists
NL(d) = i
status = "solution found"
Else
status = "solution not found"

End-If-Else
Return status

Algorithm Backward(*d*, *NL*, *ML*)
Input: *d* – current depth
NL – list of indices of non-zero variables for levels up to *d*
ML – list of max indices for levels up to *d*

Output: Updated d, and NL. (If the whole feasible region has been searched, d is set to zero.)

For level = d-1 to 0 decrement level
If level = 0
d = 0

Else

$i = \min(j \mid ((NL(level)+1) \le j \le ML(level)) \cap (j \notin NL[<level]) \cap ((j \cup NL[<level]) \in SP)$

(where $NL[<level]$ is the first (level-1) elements of NL.)

If i exits

NL(level) = i

d = level

level = -1

End-If

End-If-Else

End-For

Algorithm ENUM-SPC will be invoked as many times as the number of cuts generated in the solution of the attacker's or defender's problem. ENUM-SPC will skip the initialization step except during the first call and when it fails to find a feasible solution by searching from the previous solution. Testing has showed that ENUM-SPC was about 10 times faster than the general IP approach.

5 IMPLEMENTATION AND COMPUATIONAL RESULTS

We implemented the algorithms described in Section 4 in Visual C++, while utilizing the open source code Common Optimization Interface for OR (COIN-OR) [17]. CLP (COIN's linear program solver) was used to solve the OPF problem repeatedly and incrementally. The integer program solver was constructed using COIN-OR's SBB (Simple Branch and Bound) package.

All test runs in this section were performed on the IEEE Reliability Test System (RTS) One Area Network [15, 18]. The RTS One-Area Network consists of 2 substations, 24 buses, 33 generators and 38 lines. Power demands on a bus are divided into groups: one for residential users and the other for commercial users.

The resources required for the attacker to disable a link, bus, and substation are 1, 2, and 3, respectively. One unit of resource may include a combination of manpower, equipment and money. To defend the same network elements, the defender will need the same number of units of resources. The defender's objective is to minimize the amount of loss due to attacks.

Test runs were conducted on a Windows 2000 machine (2.4GHz speed and 1GB memory), and the code was not optimized for performance. The set packing/covering problem was solved with the ENUM-SPC algorithm.

One parameter that affects the performance a great deal is the budget constraint for both the attacker and the defender. As more resources are available, the number of feasible attack/defense strategies grows exponentially. In one test case, the attacker's budget is fixed at 2 and defender's budget varies. The computational results are summarized in Table 3.

Table 3 Summary Results for Attack Budget of Two Units

Defense Budget	Solution Time (min)	Optimal Obj Value ($)
0	3	346678
1	3	341075
2	3	236467
3	3	236467
4	4	233359
5	5	233359
6	6	232170
7	7	232170
8	25	232170
9	62	219859
10	1322	219859

Table 3 reveals that a defense budget of two achieves a cost-effective level of performance. Any defense budget above two would result in marginal decrease in system loss. In practical terms, this means that it is very important to protect a few strategic elements if their failure would inflict a great loss on the network. Table 3 also shows an exponential growth in solution time as the defense budget increases past a value of seven.

6 FUTURE RESEARCH

The limiting factor for the real world application of the solution method in this paper is slow solution time for large-sized problems. Most computational time was spent solving the attack and defense SPC problems. They have to be solved repeatedly, and each time the number of constraints will increase by one, eventually resulting in an exponential growth in demand for computation power. If the OPF solution process could generate many cuts (instead of one) in one iteration, the number of times that the attacker SPC problem has be solved would be greatly reduced. As mentioned in Sec. 4, CLP was not suited for generating many cuts. To this end, we would need a specialized LP solver that uses a 2-phase approach (CLP uses a 1-phase approach) to arrive at feasibility and then moves slowly to optimality by traversing as many basic feasible points as possible.

Another approach for generating many cuts is to use an LP solver that can find all (alternative) basic optima. Each optimal point would be used to create one cut for the attacker's SPC problem. Even though finding all basic optima is a tough combinatorial problem, the effort may well offset the computation burden of solving many SPC problems.

Solution of the SPC problem itself also needs improvement. The ENUM-SPC algorithm will not solve a large-size model. More intelligent bounding rules are needed in the search process. Parallel algorithms could be explored. For example, many different branches of the branch and bound tree could be explored simultaneously. A higher level of parallelism permits many attack/defense plans to be evaluated concurrently.

The modeling and solution methods of this paper can be readily applied to other types of infrastructure systems such as oil and gas pipelines, transportation, water distribution and

telecommunication networks. While the defender's response model will differ for these types of networks, the defense and attack decision models will be very similar. The solution method in this paper can be applied to many types of networks.

7 REFERENCES

1 National Research Council (NRC), *Making the Nation Safer: The Role of Science and Technology in Countering Terrorism*, National Academy Press (2002).

2 Bard, Jonathan F., *Practical Bilevel Optimization: Algorithms and Applications*, Kluwer (1998).

3 Shimizu, Kiyotaka, Yo Ishizuka, and Jonathan F. Bard, *Nondifferentiable and Two-Level Mathematical Programming*, Kluwer (1997).

4 Edmunds, Thomas A. and Jonathan F. Bard, "Algorithms for Nonlinear Bilevel Mathematical Programs," *IEEE Transactions on Systems, Man, and Cybernetics, Vol. 21, No. 1*, pp. 83-89 (1991).

5 Edmunds, T. A. and J. F. Bard, "An Algorithm for the Mixed-Integer Nonlinear Bilevel Programming Problem," *Annals of Operations Research, Vol. 32*, pp. 149-162 (1992). Brotcorne, Luce, Martine Labbe, Patrice Marcotte, and Gilles Savard, "A Bilevel Model and Solution Algorithm for A Freight Tariff-Setting Problem," *Transportation Science, Vol. 34, No. 3* (August 2000).

6 Brotcorne, Luce, Martine Labbe, Patrice Marcotte, and Gilles Savard, "A Bilevel Model for Toll Optimization on a Multicommodity Transportation Network," *Transportation Science, Vol. 35, No. 4* (November 2001).

7 Edmunds, Thomas A. and R. Scott Strait, "Evaluating Arms Control Treaty Verification Regimes: A Risk Analysis Approach," *Probabilistic Safety Assessment and Management*, G. Apostolakis, ed., Elsevier, New York (1991).

8 Labbe, Martine, Patrice Marcotte, and Gilles Savard, "A Bilevel Model of Taxation and Its Application to Optimal Highway Pricing," *Management Science, Vol. 44, No. 12* (December 1998).Patriksson, Michael and R. Tyrrell Rockafellar, "A Mathematical Model and Descent Algorithm for Bilevel Traffic Management," *Transportation Science, Vol. 36, No. 3* (August 2002).

10 Kennington, Jeffery and Mark Lewis, The Path Restoration Version of the Spare Capacity Allocation Problem with Modularity Restrictions: Models, Algorithms, and an Empirical Analysis, *INFORMS Journal on Computing, Vol. 13, No. 3*, pp. 181-190 (Summer 2001).

11 Bard, J., and J. Moore, A Branch and Bound Algorithm for the Bilevel Programming Problem, *SIAM Journal on Scientific and Statistical Computing*, 11:281-292 (1990).

12 Israeli, E., and R. Kevin Wood, "System Interdiction and Defense," Rough Draft, Operations Research Dept., Naval Postgraduate School, Monterey, CA, February 2002.

13 Israeli, E., and R. Kevin Wood, "Shortest-Path Network Interdiction," *NETWORKS*, Vol. 40(2), 97–111 (2002).

14 Salmeron, J, K. Wood and R Baldick, "Optimizing Electric Grid Design Under Asymetric Threat," Technical Report NPS-OR-03-002, Naval Postgraduate School, Monterey, California (February 2003).

15 Cormen, T. H., C. E. Leiserson and R. L. Rivest, *Inroduction to Algorithms* (p975), The MIT Press, 1990.

16 **CO**mputational **IN**frastructure for **O**perations **R**esearch, http://www-124.ibm.com/developerworks/opensource/coin/index.html, 2003.

17 IEEE, "IEEE Reliability Test System-1996," IEEE Transactions on Power Systems, Vol. 14, No 3 (1999).

18 Shimizu, Kiyotaka, Yo Ishizuka, and Jonathan F. Bard, *Nondifferentiable and Two-Level Mathematical Programming*, Kluwer (1997).

University of California
Lawrence Livermore National Laboratory
Technical Information Department
Livermore, CA 94551