# A PATH GENERATION and initial solution strategy to the Multi-arc dISRuPTION with Maximum Impact on Network Flow

Mohammad Najarian

In a network flow problem, critical arcs can be defined as arcs in which a failure occurrence results in maximum impact on the network flow. The problem of multi-arc disruption with maximum impact (MADP) aims to find multiple critical arcs whose simultaneous disruptions result in the maximum flow disruption of the network. This problem can be formulated as a mixed-integer programming (MIP) model. The MIP problem is computationally expensive; hence, this paper aims to reduce computational time. The contributions are a) a new problem formulation is presented following a pattern generation approach to provide a near optimal solution, and b) a fast heuristic is developed to find a good initial feasible solution for warm-start MIP. For this heuristic, a new centrality measure is developed. Our numerical results show that the initialization heuristic reduces the computational time for MIP drastically (between 10% to 90% percent). Moreover, the proposed pattern generation reduces the CPU time considerably while the gap between optimal objective value and the pattern generation approach is small and in in 70% of the cases the gap is zero.

## Introduction

Disrupting an adversarial network is of interest in many domains. The adversarial network can be an enemy infrastructure, such as the transportation [1] of a drug, weapon, or human trafficking network [2]; or an information network [1]. From a game theoretical perspective, an interdictor (attacker) has limited resources to disrupt an adversarial network while an adversary (defender) aims to maximize the flow through the disrupted network (residual network) [2]. In many cases, the resource limit can be translated into the number of arcs in the network to be disrupted (. For example, resources can be a limited number of missiles available to strike enemy’s transportation system. Another example is the number of checkpoints in the supply chain of precursor chemicals for drug production [3]. In this case, the attacker is interested in selecting arcs whose disruption will have the highest impact on the defender’s network [4], i.e., it minimizes the maximum flow in the residual network. Another application of the problem is to predict a terrorist’s potential targets and reduce the impact caused by the terrorist attack on the network [5]. For example, a terrorist sets fire in a place and tries to maximize the shortest path of fire trucks to prevent them from reaching to the fire on time. The goal of the terrorist is to find the most vital arcs whose failure will result in the greatest increase in the length of the shortest path [6]. Therefore, one might be interested in protecting those arcs from such attacks to mitigate the resulting impact.

Different approaches can be used to find arcs to be disrupted. One approach can be based on some network measures. If the attacker has enough resources and the goal is to shut down the whole network, node and arc connectivity measures can be utilized. The *node connectivity* is the smallest number of nodes whose removal results in a disconnected or single-node graph. It is the smallest number of node-distinct paths between any two nodes [7]. Similarly, the *arc connectivity* is the smallest number of links whose removal results in a disconnected graph. Another group of metrics include *centrality measures,* which are commonly described as indices of prestige, prominence, importance, and power [8]. The shortest-path-betweenness centrality is a measure of the extent to which a component is located on the paths between two other components [9]. It indicates how much a component has control over the flow between the components. Hence, their removal can have a negative impact on the network. The *eigenvector centrality* measures the influence of a node in a network and it depends on both the centrality of the component and the centrality of its neighbors [10]. Google’s PageRank algorithm is a variant of eigenvector-centrality and is used to rank web pages in the search engine results. Eigenvector-centrality is Katz centrality and it measures the relative degree of influence of a node in a social network [11]. Freeman and Borgatti [8] introduced a *flow-betweenness centrality* () that incorporates the strength of the linkage between nodes into the metric. Flow can be physical (e.g. used goods and money) or it can be non-physical (e.g. gossip, attitude). In an *s-t* network flow [12], arcs can be ranked based on their relative contribution to the flow in the network, i.e., based on their value. This metric can be useful when we are interested in a single vital arc; if the single arc with the highest is disrupted, the decrease in the network output will be high. However, for the problem of multiple-arc failure with maximum impact on network flow, these metrics may not provide the best solution, as they do not consider the effect of simultaneous disruption of arcs on the network flow.

An approach to solve MADP is to formulate the problem as a bilevel (attacker-defender) model, where in the upper level the attacker selects arcs to disrupt and in the lower level, the defender maximizes the flow through the residual network. The upper level objective is to minimize the maximum flow in the lower level. When the attacker has enough resource to disrupt one arc (, the problem of finding a single vital arc in a network flow can be solved using the Newton algorithm [13]. The bilevel problem of finding arcs is NP-hard [2]. Some efforts were made in the literature to reduce the computational cost [2] by using a binary programming model that maximizes the sum of the arc capacities under attack; however, solving this model is still computationally expensive [3]. A Lagrangian-relaxation based heuristic has also been explored to provide a local optimum [3].

The goal of this study is to present methods to reduce the computational burden for solving the MADP. The contributions are twofold. First, a mathematical programming model is presented, and a pattern generation algorithm is proposed to solve it. Second, a new centrality measure and a heuristic are introduced to provide a good initial solution for the MIP formulation. The heuristic utilizes the path-based network flow problem. This initial solution is also used in the proposed pattern generation. The organization of the rest of this paper is as follows. Chapter 2 describes the path-based mathematical programming model for MADP. Chapter 3 presents the solution methodology, which includes two algorithms. The first algorithm provides an initial solution generation strategy and the second algorithm is a pattern generation algorithm used to solve the MADP. Section 4 provides the numerical results. The conclusion is drawn in Chapter 5.

## Multiple-Arc Failure Maximum Impact (MADP) Mathematical Formulation

Let be a directed graph or digraph where is a set of vertices (nodes) and is a set of arcs. Let have two special nodes *s* and *t* (called the source and the sink, respectively). Each arc is an ordered pair . A path ( between nodes and is a sequence of adjacent nodes , , such that . In this study, the networks are assumed to be directed acyclic graph (DAG), or in other words, a directed graph with no cycles. Suppose an arc has a capacity of . A flow is a function which assigns the amount of flow to arc .

10

6

6

2

3

3

9

Figure 1: MADP in a network flow

The MADP can be formulated as a bilevel programming model [14]. Prior to a disruption, an adversary (defender) flows the maximum commodity or information through a network from a source vertex () to a terminal vertex () (Figure 1). In the upper level, the attacker disrupts a predetermined number () of arcs in the adversarial network (e.g. arcs and in Figure 1 with ). The defender notices the damages to the network and acts accordingly, i.e., in the lower level problem he changes its operations to maximize the flow through the residual network (the network excluding the disrupted arcs). The attacker is aware of the defender’s goal and tries to choose arcs in a way that the adversary’s maximum flow is minimized. Let Yuv be a binary variable that shows the attacker’s decision, where

shows the attacker’s decision. In the upper level, the attacker selects arcs (Equation (1-2)), whose disruption will drop the flow through those arcs to 0 (Equation (1-4)). The flow through each arc is limited by its capacity (Equation (1-4)). Moreover, the flow is conserved (Equation (1-5)), and it does not flow into the source or out of the sink (Equation (1-6)). After disruption, the defender maximizes the flow in the residual network. The goal of the attacker is to minimize the maximum flow of the defender (Equation (1-1).

|  |  |  |
| --- | --- | --- |
|  | | (1-1) |
| s.t. |  | (1-2) |
|  | | (1-3) |
| s.t. |  | (1-4) |
|  | | (1-5) |
|  | | (1-6) |
|  | | (1-7) |

## A path-based model formulation

The path-based formulation [15], [16] of Problem (1) is used in the literature to reduce the problem size. Let be a set of all distinct paths that contain at least one unique arc. For brevity, instead of denoting an arc with , it is shown as , where is a unique number assigned to an arc. Hence, a path-arc incidence matrix (P) can be defined as

|  |  |  |
| --- | --- | --- |
|  |  | Arcs |
|  | Paths |  |

where

One should note that this matrix does not show the sequence of incidences. Moreover, if arc is incident to more than one path, then the aggregation of flows in all paths passing through should be less than the arc’s capacity (i.e., ). Let be the vector of the defender decision variables and be the vector of the attacker decisions. Equations (3-1) through (3-5) construct the path-based bilevel MADP. In the upper level (Equations (3-1) and (3-2)), the attacker decides which arcs to attack. These arcs must meet the resource limitation (Equation (3-2)). In the lower level (Equations (3-3) through (3-5)), the defender maximizes the network flow in the residual network.

|  |  |  |
| --- | --- | --- |
|  | | (3-1) |
| s.t. |  | (3-2) |
|  | | (3-3) |
|  | | (3-4) |
|  | | (3-5) |

## Solution Approach

To reduce the computational burden for solving Problem (3), we provide two different solution approaches. The first one is a warm-start approach for the MIP model in which a heuristic method is used to initialize values of so that the MIP model can be solved much faster than the MIP model with default values provided by a commercial solver. The second approach is based on a model reformulation, followed by a pattern generation algorithm used to solve it.

## MIP formulation

Problem (3) is reformulated as an MIP model through the following process. After the attacker has made the decision on which arcs () to disturb the remains constant for the defender in the lower level, in a way that the Equations (3-3) through (3-5) construct a linear programming. Hence, it can be replaced by its dual problem.

Let be the dual variable corresponding to constraints (3-4). The dual of the subproblem for a fixed is

|  |  |  |
| --- | --- | --- |
|  | | (4-1) |
| s.t. |  | (4-2) |
|  | | (4-3) |

Problem 4 is an LP formulation of the minimum cut problem [17]. By applying these changes in the subproblem of Problem (3), the MADP becomes as:

|  |  |  |
| --- | --- | --- |
|  | | (5-1) |
| s.t. |  | (5-2) |
|  | | (5-3) |
|  | | (5-4) |
|  | | (5-5) |

By replacing z in Equation (5-1) with Equation (5-3), the problem becomes

|  |  |  |
| --- | --- | --- |
|  | | (6-1) |
| s.t. |  | (6-2) |
|  | | (6-3) |
|  | | (6-4) |

The objective (6-1) then can be changed to

|  |  |
| --- | --- |
|  | (7) |

The objective function in Equation (7) is non-convex, and the optimality is not guaranteed. Therefore, we linearize it by setting and exploiting a sufficiently large number. As a result, the new formulation is:

|  |  |
| --- | --- |
|  | (8-1) |
|  | (8-2) |
|  | (8-3) |
|  | (8-4) |
|  | (8-5) |
|  | (8-6) |

Equation (8-1) and the variable linearization equations (Equations (8-2) and (8-3)) minimize the minimum cut capacity. This is equivalent to minimizing the maximum flow in the network. Equation (8-4) is the dual constraint in the path based minimum cut problem [17]. Equation (8-5) sets the limit on the number of arcs to be disrupted.

## Initialization heuristic

This section introduces an initialization heuristic for variable vector to solve the MIP model faster. A new metric, min-flow-betweenness centrality (), is developed, which is an integral part of the heuristic. The maximum-flow-betweenness (MFB) centrality introduced in [8] evaluates the maximum possible flow that can pass through an arc. The MFB of an arc is less than or equal to the arc capacity. For example, in Figure 1, the arc () with capacity 6 has a an MFB equal to 4 which implies that the network set up doesn’t allow it to utilize more than 4 units of its capacity. However, if an arc with high MFB is disrupted, other paths may have the capacity to carry a large portion of that flow (i.e., there can be an arc with high MFB but with low impact). Instead, our metric considers the minimum flow that can pass through arc and does not decrease the maximum flow. The network in Figure 1 has four paths, delivered as , , , and Assume that in the optimal solution, the flow along p1, p2, p3, and p4 are 6, 2, 1, and 4, respectively; these values are combined to create a maximum flow of 15. The maximum flow that can pass through the arc () is 2. However, if we choose to flow 3 units through , the flow in will drop to 0 while the maximum flow of the network remains 15. The value of for the arc () is 0.

Let and represent the new indices such that . When the problem refers to finding a single arc with the highest impact on the maximum flow, the solution is to select arc . However, in the case of components, removing may not result in the highest impact. Consider the simple network in Figure 1. In this network , and , and the network has a maximum flow of 7. For , the solution to MADP is either or , and the residual network maximum flow will be 2. If and we choose the two components with highest values of (i.e., and ), then two units can still flow through the rest of the network. However, if we choose and , then no flow can pass through the network.



Figure 2: A simple network.

The numbers on the arcs are arc capacities.

In neuroscience, the functional connectivity is a matrix that captures the change in the level of activation regions of the brain in response to specific experimental conditions [18]. We define the functional connectivity of arcs similarly.

**Definition:** For two arcs and in , *functional connectivity* is defined as the possible amount of change in the *MFB* of if the functionality of is disrupted.

The vector constructs the rows of the functional connectivity matrix . Using the functional connectivity, we propose an algorithm that determines arcs (i.e. vector ) that serves as an initial solution. Let be the index of the arc with the largest . Algorithm 1 begins by selecting arc which reduces the maximum flow most significantly, i.e., by . The value of for the residual network after removing arc will be decreased by ( .Then, we choose the arc with largest as the second arc. In this algorithm, matrix is collocated with the paths’ capacity in the last column, in the last row, and two, the number of arcs of to be disrupted in the last entry of the matrix. The tableau matrix will look like the Figure 2:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Arcs |  |
| M= |  |  | |
|  |

Figure 3: Collocated matrix , Q, , and .

This algorithm also removes the redundant paths, thus drastically reducing the problem size at each iteration.

**Algorithm 1**

**Input**: A set of paths

**Output**::= list of arcs to be disrupted

1. Set counter =1; .
2. Calculate/update .
3. Select the arcs with highest .
4. Delete the rows in which column entry is 1.
5. Delete column j and add j to the set .
6. If counter is equal to k or the maximum flow is 0 then

go to Step 6

else

go to Step 2

1. Set counter=counter +1.
2. Output the

## A pattern generation approach

The number of variables in Problem (3) and Problem (8) grow exponentially as the number of arcs grows. To address this challenge, a pattern generation (PG) is proposed by reformulating Problem (3) to Problem (9). In PG, a master problem and corresponding subproblems are constructed [19]. The PG process starts with an initial pattern , finds the dual solution of the master problem, and uses them in the pricing subproblem to generate a new pattern. A difference between PG and the column generation [20] is that in column generation, the constructed pattern in the subproblem is appended to the technical coefficients, whereas in PG the current pattern is replaced with the new pattern. We propose a new formulation for Problem (3) and solve it using the pattern generation. A pattern has exactly entries where value 1 (i.e., indicates which arc is disrupted and as a result the flow in all the paths passing through them is stalled (Equation (9-2)).

|  |  |  |
| --- | --- | --- |
|  | | (9-1) |
| s.t. |  | (9-2) |
|  | | (9-3) |

In each iteration, the subproblem generates a new pattern that provides a better solution to Problem (9). Consider the general maximization problem in the form of

|  |  |
| --- | --- |
|  | (10-1) |
|  | (10-2) |
|  | (10-3) |

where is the cost coefficient, is the non-negative decision variable, is the resource constraint, and is the technological coefficient. Equation (8)

|  |  |
| --- | --- |
|  | (11) |

shows the current objective expression [21] for a feasible solution. Variables are divided into two groups: basic (V) and non-basic (NBV) variables. Matrix consists of the columns of corresponding to BVs, and is the th column of . If a variable with negative enters the basis, then the objective value will be increased; if a variable with positive value enters, then the objective value will be decreased. We are interested in the latter. The subproblem for the PG algorithm is

|  |  |  |
| --- | --- | --- |
|  | | (12-1) |
| s.t |  | (12-2) |
|  | | (12-4) |

in which equals . Problem (12) attempts to find a pattern that can decrease the current maximum flow. The pattern must satisfy the constraint in Equation (6-2), i.e., this pattern has exactly non-zero variables. If the object value of Problem (12) is positive and is different from the previous solution, then the new pattern will replace the current pattern. Algorithm 2 summarizes the PG process for MADP. It starts with generating an initial solution using Algorithm 1; then, it finds the dual values of the master problem. Using the dual values, Problem (12) finds a new pattern, and the process continues until the stopping criteria are met.

|  |
| --- |
| **Algorithm 2** Pattern Generation for MADP |
| Input: the network  Output: The optimal solution to MADP   1. Create an initial plan using Algorithm 1 2. Set the current pattern to the initial pattern*.* 3. Run the master problem (Problem 6) for the current pattern. 4. Use the optimal dual solution of the master problem and solve the subproblem    1. If the optimal objective value of the subproblem is positive and the current pattern is different from the previous pattern, set the current pattern to the pattern generated in the subproblem and go to 3.    2. Else go to 5 5. Output the current pattern as optimal. |

## Numerical Results

We start the numerical example with a simple network presented by Freeman [8] (Figure 4); we name it Net 1 for future reference. This network has four paths; the first path () begins at node , passes through node 3, and ends at node . Equivalently, path passes through arcs and (shown as ). Similarly, the other paths are :, , and . Figure 5 shows the incident matrix associated with these four paths. Suppose that attacker has sufficient resources to attack two arcs (i.e., ). The attacker then has options to choose from (Figure 5). Step 1 in Algorithm 1 starts with an empty set and sets the counter to 1. In Step 2, vector is calculated (Figure 6). The arc with the highest value of (i.e., with for Net 1) is selected in Step 3. Removing will disrupt the flows from paths , and . Hence, in Step 4, these paths are removed from the incidence matrix. Then, the column is removed, and added to set (i.e., ). Sine the counter is 1, being less than , we go to Step 2 for the second iteration. We can easily find the functional connectivity for , which is the updated after the removal of ; updating the functional connectivity from results in:

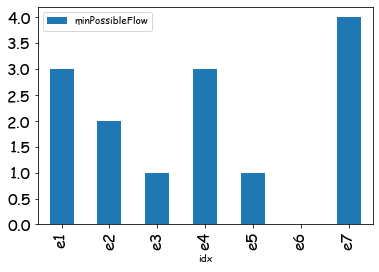
|  |  |
| --- | --- |
|  | (10) |

Based on the , two options and will provide the same result. We select arbitrarily and update . The resulting initial solution is   
. Removing these two arcs will drop the maximum flow to 0. In this case, we do not need to run the MIP model.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1  3  3  5  2  2  2  Figure 4: Network of Net 1  [8] | |  |  |  | | --- | --- | --- | |  |  | Arc | |  |  |  |   Figure 5: Path arc incidence matrix |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 |
|  | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
|  | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
|  | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
|  | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |

Figure 6: Possible options (patterns) to select two arcs from Net 1



Arc

Figure 7: The minimum-flow betweenness for Net 1

The PG process begins with the initial pattern obtained using Algorithm 1. The master problem is solved after fixing to the initial pattern. The duals of the constraints of the master problem then are used in the pricing subproblem to generate a new pattern . To validate the PG approach, 200 randomly generated graphs are used for numerical experiments (see Appendix I). Figure 7 shows the histogram of the number of arcs for the generated networks , detailing a near-normal distribution with a mean of 100 and a standard deviation of 47 arcs. For each network, the optimal solution of the MADP problem using the PG is compared to the exact solution obtained from the MIP model. The gap between the two objective values resulting from the two approaches is measured as

|  |  |
| --- | --- |
|  | (11) |

Figure 9 shows the histogram of the gap in the generated networks. The MADP is solved for each network for . This histogram shows that the MADP found an optimal solution (i.e., gap = 0) in 71% of the problems, while the gap ranged between 0 and 0.15 for 20% of the cases. Although the PG does not guarantee optimality, it finds a good solution at a small fraction of time to solve the optimization model. Hence, it can be a decent way to generate an initial starting solution for the MIP model from which it can start its Branch-and-Bound process to speed up the convergence.

|  |  |
| --- | --- |
| C:\Users\User\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\234EA1D8.tmp  Number of arcs  Frequency  Figure 8: Histogram of number of arcs | C:\Users\User\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\606BDDA6.tmp  gap  Frequency  Figure 9: Histogram of the objective gap between optimal PG and objectives |

To illustrate the effectiveness of the proposed solution approach, a subset of networks is randomly selected from the previously generated networks; they are labeled as Net 2, …, Net 5. The network in Figure 3 with 5 nodes and 7 arcs is also included and is named Net 1. All computational experiments are made on a personal computer with Core i5-3470S 2.9 GHz CPU and 8 Gigabyte of RAM. To account for the effect of the performance of the PG on the CPU time, each experiment is run 10 times, and the average time is used in the analysis. Table 1 summarizes the effect of the initialization on the CPU time for the selected networks and for values for . For Net 1 and , the initialization heuristic provides a feasible solution to the MIP problem with a flow of two in 0.0020 seconds. This solution is used as the initial solution (i.e., a warm-start) to the MIP model. After 0.0130 seconds, the MIP model yielded an optimal objective value of 2. In this case, the initialization heuristic happened to find the optimal solution. Similar results for some other cases are marked with a “\*” in column ( of Table 1. For Net 1 and , the initialization provided a solution with a flow of zero in 0.0030 seconds. The flow of 0 means that the disrupted arcs have produced the maximum impact on the network that disrupting more arcs will not disturb the network flow. Hence, the initial solution is optimal, and the algorithm stops. Similar results were observed for Net 3 and , in which the initialization heuristic resulted in an objective value of 0. Thus, there is no need to proceed to solving the MIP model.

Table : The effect of a warm-start strategy on the MIP model

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID**  **(*N*, *E*)** |  | **Objective Value** | | **Solution Time (Seconds)** | |
| Initialization Heuristic | MIP with Warm-Start | Initialization Heuristic | MIP with Warm-Start |
| **Net 1**  **(5, 7)** | **1\*** | 2 | 2 | 0.0020 | 0.013 |
| **2** | 0 | - | 0.0030 | 0.000 |
| **Net 2**  **(14, 32)** | **1\*** | 17 | 17 | 0.0040 | 0.175 |
| **2** | 12 | 10 | 0.0060 | 0.160 |
| **3** | 9 | 5 | 0.0060 | 0.237 |
| **Net 3**  **(25, 89)** | **1\*** | 21 | 21 | 0.003 | 3.909 |
| **2** | 16 | 14 | 0.007 | 3.815 |
| **3** | 11 | 9 | 0.010 | 3.751 |
| **4** | 4 | 4 | 0.011 | 4.540 |
| **5\*** | 2 | 2 | 0.018 | 4.779 |
| **6** | 0 | - | 0.014 | 0.000 |
| **Net 4**  **(32, 166)** | **1\*** | 26 | 26 | 0.007 | 81.28 |
| **2** | 22 | 18 | 0.011 | 85.59 |
| **3** | 21 | 14 | 0.023 | 114.08 |
| **4** | 13 | 10 | 0.026 | 115.02 |
| **5** | 11 | 7 | 0.031 | 128.40 |
| **6** | 11 | 5 | 0.030 | 206.31 |
| **Net 5**  **(38, 216)** | 1\* | 65 | 65 | 0.0424 | 165.42 |
| 2 | 56 | 55 | 0.0749 | 175.92 |
| 3 | 55 | 46 | 0.0514 | 191.43 |
| 4 | 48 | 39 | 0.0938 | 190.27 |
| 5 | 46 | 32 | 0.1408 | 198.28 |
| 6 | 39 | 25 | 0.1338 | 193.35 |

Now we examine the computational performance of three solution approaches: a) MIP formulation without warm-start, where the initial solution is provided by the solver; b) MIP formulation with warm-start, in which the initial solution is provided by Algorithm 1; and c) PG. The computational time of finding a solution to the MADP for the test networks are summarized in .

Table : Computational performance comparison of three solution approaches

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **CPU time**  **MIP** | **MIP with warm-start** | | **PG** | | |
| **CPU time** | **Reduction in CPU time due to initialization** | **CPU time** | **Gap** | **Reduction in CPU time compared to MIP** |
| **Net 1**  **(5,7)** | 1 | 0.025 | 0.015 | 40% | 0.02 | 0% | 20% |
| 2 | 0.018 | 0.003 | 83% | 0.017 | 0% | 6% |
| **Net 2**  **(14, 32)** | 1 | 0.200 | 0.179 | 10% | 0.129 | 0% | 36% |
| 2 | 0.184 | 0.166 | 10% | 0.143 | 0% | 22% |
| 3 | 0.263 | 0.243 | 8% | 0.117 | 0% | 56% |
| **Net 3**  **(25, 89)** | 1 | 4.19 | 3.91 | 7% | 2.88 | 0% | 31% |
| 2 | 4.96 | 3.82 | 23% | 4.73 | 0% | 5% |
| 3 | 4.00 | 3.76 | 6% | 3.9 | 0% | 3% |
| 4 | 5.62 | 4.55 | 19% | 2.78 | 0% | 51% |
| 5 | 5.20 | 4.80 | 8% | 5.01 | 0% | 4% |
| 6 | 3.84 | 0.01 | 99% | 2.77 | 0% | 28% |
| **Net 4**  **(32, 166)** | 1 | 93.05 | 81.28 | 13% | 68.95 | 0% | 26% |
| 2 | 97.87 | 85.60 | 13% | 89.58 | 0% | 8% |
| 3 | 163.43 | 114.10 | 30% | 139.16 | 0% | 15% |
| 4 | 137.63 | 115.04 | 16% | 132.12 | 0% | 4% |
| 5 | 149.36 | 128.42 | 14% | 130.92 | 0% | 12% |
| 6 | 230.64 | 206.34 | 11% | 111.82 | 0% | 52% |
| **Net 5**  **(38, 216)** | 1 | 217.64 | 165.46 | 24% | 133.92 | 0% | 38% |
| 2 | 232.75 | 176.00 | 24% | 136.63 | 0% | 41% |
| 3 | 258.37 | 191.48 | 26% | 128.61 | 0% | 50% |
| 4 | 260.94 | 190.37 | 27% | 131.81 | 0% | 49% |
| 5 | 269.33 | 198.42 | 26% | 127.76 | 0% | 53% |
| 6 | 375.60 | 193.48 | 48% | 124.96 | 4% | 67% |

These results show that the MIP with the proposed initialization heuristic reduced the computational time by a range of 6% to 99% when compared to the MIP without the warm-start. Especially, the reduction in CPU time was over 24% for Net 5 with 216 arcs. The PG also improved the computational time over the MIP without the warm-start. Moreover, the outperformance of the PG was more pronounced for larger-sized problems (e.g., Net 5). For example, for Net 5 and , the initialization reduced the CPU time of MIP by 48%, while PG reduces it by 67%.

## Conclusion

In this study, we have developed solution approaches to reduce the computational time of the multi-arc disruption with maximum impact on network flow problem (MADP). Our approaches include a pattern generation (PG) process and a warm-start heuristic to solve the MADP. The subproblem of the PG is designed to find a new pattern for an arc failure that can result in the highest reduction of the maximum flow in the network. The PG performed very well and provided the optimal solution in 71% of the test networks attempted in this paper. A minimum-flow-betweenness centrality metric is introduced for the warm-start heuristic. The warm-start strategy reduced the number of iterations as well as the computational time. In several cases, the initial solution was found to be the optimal solution. The performance comparison on the CPU time for solving the MIP model with-and-without the warm-start heuristic revealed that the initialization approach significantly reduced the computational time by a range of 6% to 99%.

Our approach in this paper for the initialization heuristic and the pattern generation relies on the network paths. Finding network paths is computationally extensive; many researchers try to develop path generation algorithms that generate a portion of paths which solving the problem using those paths can provide an objective value close to the the objective value of the original problem (i.e., the problem in which all the paths are included). Future work to consider includes developing an enhanced functional connectivity for the initialization heuristic. Functional connectivity is used to show the effect of one edge disruption on the other edges and on the network paths. A good functional connectivity can result in a better initial solution for warm-start strategy.

References

[1] Joint Force Development, “Joint Interdiction,” Federation of American Scientists, Washington, DC, Sep. 2016.

[2] R. K. Wood, “Deterministic network interdiction,” *Mathematical and Computer Modelling*, vol. 17, no. 2, pp. 1–18, Jan. 1993, doi: 10.1016/0895-7177(93)90236-R.

[3] L. Bingol, “A Lagrangian Heuristic for solving a network interdiction problem,” p. 57.

[4] J. O. Royset and R. K. Wood, “Solving the Bi-Objective Maximum-Flow Network-Interdiction Problem,” *INFORMS Journal on Computing*, vol. 19, no. 2, pp. 175–184, May 2007, doi: 10.1287/ijoc.1060.0191.

[5] R. L. Church, M. P. Scaparra, and R. S. Middleton, “Identifying Critical Infrastructure: The Median and Covering Facility Interdiction Problems,” *Annals of the Association of American Geographers*, vol. 94, no. 3, pp. 491–502, Sep. 2004, doi: 10.1111/j.1467-8306.2004.00410.x.

[6] K. Malik, A. K. Mittal, and S. K. Gupta, “The k Most Vital Arcs in the Shortest Path Problem,” *Operations Research Letters*, vol. 8, no. 4, p. 5, 1989.

[7] A. H. Dekker and B. D. Colbert, “Network Robustness and Graph Topology,” in *ACSC*, 2004.

[8] L. C. Freeman, S. P. Borgatti, and D. R. White, “Centrality in valued graphs: A measure of betweenness based on network flow,” *Social Networks*, vol. 13, no. 2, pp. 141–154, Jun. 1991, doi: 10.1016/0378-8733(91)90017-N.

[9] M. E. J. Newman, “A measure of betweenness centrality based on random walks,” *Social Networks*, vol. 27, no. 1, pp. 39–54, Jan. 2005, doi: 10.1016/j.socnet.2004.11.009.

[10] S. P. Borgatti and M. G. Everett, “A Graph-theoretic perspective on centrality,” *Social Networks*, vol. 28, no. 4, pp. 466–484, Oct. 2006, doi: 10.1016/j.socnet.2005.11.005.

[11] M. J. Park, O. M. Kwon, and J. H. Ryu, “A Katz-centrality-based protocol design for leader-following formation of discrete-time multi-agent systems with communication delays,” *Journal of the Franklin Institute*, vol. 355, no. 13, pp. 6111–6131, Sep. 2018, doi: 10.1016/j.jfranklin.2018.06.022.

[12] Y. P. Aneja, R. Chandrasekaran, and K. P. K. Nair, “Maximizing residual flow under an arc destruction,” *Networks*, vol. 38, no. 4, pp. 194–198, Dec. 2001, doi: 10.1002/net.10001.

[13] S. H. Lubore, H. D. Ratliff, and G. T. Sicilia, “Determining the most vital link in a flow network,” *Naval Research Logistics Quarterly*, vol. 18, no. 4, pp. 497–502, 1971, doi: 10.1002/nav.3800180408.

[14] R. K. Wood, “Bilevel Network Interdiction Models: Formulations and Solutions,” in *Wiley Encyclopedia of Operations Research and Management Science*, American Cancer Society, 2011.

[15] D. Mount, “CMSC 451: Lecture 14 Network Flows: Basic Deﬁnitions,” p. 4, 2017.

[16] M. Rungta, G. Lim, and M. Baharnemati, “Optimal egress time calculation and path generation for large evacuation networks,” *Annals of Operations Research*, vol. 201, no. 1, pp. 403–421, 2012.

[17] J.-P. W. Kappmeier, “Generalizations of Flows over Time with Applications in Evacuation Optimization,” 2015, doi: 10.14279/depositonce-4358.

[18] B. P. Rogers, V. L. Morgan, A. T. Newton, and J. C. Gore, “Assessing Functional Connectivity in the Human Brain by FMRI,” *Magn Reson Imaging*, vol. 25, no. 10, pp. 1347–1357, Dec. 2007, doi: 10.1016/j.mri.2007.03.007.

[19] M. E. Lübbecke and J. Desrosiers, “Selected Topics in Column Generation,” *Operations Research*, vol. 53, no. 6, pp. 1007–1023, Dec. 2005, doi: 10.1287/opre.1050.0234.

[20] G. B. Dantzig and P. Wolfe, “Decomposition Principle for Linear Programs,” *Operations Research*, vol. 8, no. 1, pp. 101–111, Feb. 1960, doi: 10.1287/opre.8.1.101.

[21] Mokhtar S. Bazaraa, John J. Jarvis, Hanif D. Sherali, *Linear Programming and Network Flows*, 4th ed. Wiley, 2011.

[22] Ted Lewis, *Critical Infrastructure Protection in Homeland Security: Defending a Networked Nation*. Wiley-Interscience, 2006.

# Appendix 1: Generating Random Graphs

The random networks are generated as follows. First, the number of nodes is determined for each graph to be generated. Then, for every pair of nodes, a uniform random number is generated. An arc between two nodes is added to the network only if the generated random probability is greater than a probability threshold. The pseudo code for network generation is presented below.

**Algorithm 1**

**Input**: Number of nodes () and probability threshold ()

**Output**:A random network

Initialize a network with n nodes and no arcs

For do:

For do:

= random uniform probability between 0 and 1

If then:

Add arc with random capacity to the network.

Output the network

Most infrastructure and adversarial networks are sparse [22], which means not all nodes are connected to each other. Therefore, this sparsity must be taken into consideration when generating random graphs. Selecting high probability threshold values (e.g., 0.9) mean that the probability generated for an arc should be larger than 0.9 so that the arc is added to the network. Hence, selecting a higher threshold will result in a sparser network.