SARNET optimal protection strategies: a modeling approach

Stojan Trajanovski
University of Amsterdam, The Netherlands

Presented by dr. Paola Grosso

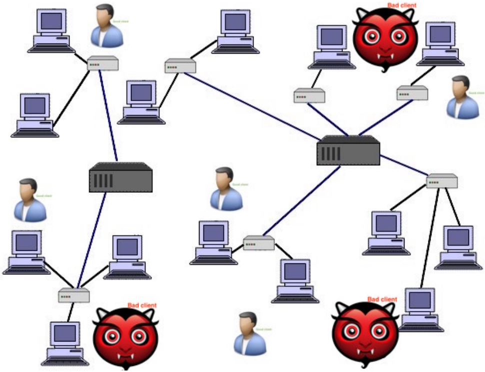
Industry workshop
5th October 2016

Outline

Motivation

Modeling DDoS attacks

1. A given topology with nodes and links
2. Nodes = {good clients, bad clients, service nodes, other "routers"}
3. Boost the "good traffic" (good clients $\longrightarrow$ service nodes)
4. Shrink the "bad traffic" (bad clients $\longrightarrow$ service nodes)



Motivation

How to react? **- Finding optimal response**

1. current topology (nodes, links & interconnections)
2. permitted links (that can be made on or off)
3. current values and box constrains on the link bandwidths
4. filtering a certain flow
5. already determined good clients and bad/attackers
6. already determined service nodes

Multiobjective nature:

1. maximize the flow from good clients to the service nodes
2. minimize the flow from attackers to the service nodes
3. under the given constrains

Outline

Model

Representing the network as a graph: (given inputs)

- directed graph $G = (\mathcal{N}, \mathcal{L})$ with
  a set of nodes $\mathcal{N}$ and a set of links $\mathcal{L} = \{(i, j) | i, j \in \mathcal{L}\}$
- $k_{ij} \in \{0, 1\}$ represents the initial link (i, j) presence
- c_{ij} and $c_{ij}^{\max}$ are the starting and the maximum allowed capacities of link (i, j), respectively
- $\mathcal{C} \subset \mathcal{N}$ is the set of clients, $\mathcal{A} \subset \mathcal{N}$ is the set of attackers, and $\mathcal{S} \subset \mathcal{N}$ are service nodes

The aim is to **maximize** the successful flow from the nodes in $\mathcal{C}$ to $\mathcal{S}$ to **minimize/protect** from the flow from $\mathcal{A}$ to $\mathcal{S}$ with a minimum cost

Permitted actions:

1. link delete or add (not for all pairs of nodes)
2. bandwidth up or down
3. flow filtering

Outline

Problem definition

Decision variables:

$$f_{ijkm} \in \mathbb{R} \text{ is a part of the flow on link } (i,j) \text{ carrying a traffic from } k \text{ to } m$$
$$l_{ij}^{+} \in \{0,1\} \text{ is 1 if link } (i,j) \text{ has been added and 0 otherwise}$$
$$l_{ij}^{-} \in \{0,1\} \text{ is 1 if link } (i,j) \text{ has been removed and 0 otherwise}$$
$$l_{ij} \in \{0,1\} \text{ is 1 if link } (i,j) \text{ is present in the network and 0 otherwise}$$
$$z_{ij} \in \mathbb{R} \text{ is the increase/decrease of the bandwidth on the link } (i,j)$$

Maximize the flow from $\mathcal{C}$ to $\mathcal{S}$ and **minimize** the flow from $\mathcal{A}$ to $\mathcal{S}$

We formulated **Mixed Bi-linear Integer Programming (MBIP)** optimization problem. MBIP are usually hard to solve.

Problem definition

Objective function:

$$\max \quad \alpha \sum_{c\in\mathcal{C},i\in\mathcal{N},k\in\mathcal{N},m\in\mathcal{N}} f_{cikm} - \beta \sum_{a\in\mathcal{A},i\in\mathcal{N},k\in\mathcal{N},m\in\mathcal{N}} f_{aikm} \qquad (1)$$

Constrains:

$$\forall i \in \mathcal{N}, k \in \mathcal{N}, m \in \mathcal{N}, \quad \sum_{j:(i,j)\in\mathcal{L}} f_{ijkm} - \sum_{j:(j,i)\in\mathcal{L}} f_{jikm} = 0, \qquad (2)$$

$$\forall (i,j) \in \mathcal{L}, \quad \sum_{k\in\mathcal{N},m\in\mathcal{N}} f_{ijkm} \leq c_{ij}^{\max} \qquad (3)$$

$$\forall (i,j) \in \mathcal{L}, \quad \sum_{k\in\mathcal{N},m\in\mathcal{N}} f_{ijkm} - z_{ij} = c_{ij} \qquad (4)$$

Problem definition

$$\forall (i,j) \in \mathcal{L}, k \in \mathcal{N}, m \in \mathcal{N}, \quad f_{ijkm} \geq 0 \tag{5}$$

$$\sum_{(i,j)\in\mathcal{L}} l^{+}_{ij} \leq C_{\text{add}} \tag{6}$$

$$\sum_{(i,j)\in\mathcal{L}} l^{-}_{ij} \leq C_{\text{rem}} \tag{7}$$

$$\sum_{(i,j)\in\mathcal{L}, k\in\mathcal{N}, m\in\mathcal{N}} a_{ijkm} \leq C_{\text{filter}} \tag{8}$$

$$\forall (i,j) \in \mathcal{L}, \quad (1 - l_{ij}) \sum_{k\in\mathcal{N}, m\in\mathcal{N}} f_{ijkm} = 0 \tag{9}$$

$$\forall (i,j) \in \mathcal{L}, k \in \mathcal{N}, m \in \mathcal{N}, \quad a_{ijkm} f_{ijkm} = 0 \tag{10}$$

$$\forall (i,j) \in \mathcal{L}, k \in \mathcal{N}, m \in \mathcal{N}, \quad a_{ijkm} - l_{ij} \leq 0 \tag{11}$$

$$\forall (i,j) \in \mathcal{L}, \quad l_{ij} + l^{-}_{ij} \leq 1 \tag{12}$$

$$\forall (i,j) \in \mathcal{L}, \quad l^{+}_{ij} - l_{ij} \leq 0 \tag{13}$$

$$\forall (i,j) \in \mathcal{L}, \quad (1 - l_{ij}) f_{ij} = 0 \tag{14}$$

$$\forall (i,j) \in \mathcal{L}, \quad (l_{ij} + l^{-}_{ij} - k_{ij})(l_{ij} - l^{+}_{ij} - k_{ij}) = 0 \tag{15}$$

Outline

Algorithms

Two algorithms:

1. Close-to-exact algorithm (branch & bound)
2. Dedicated heuristic
3. Performance and running time analysis between both

Close-to-exact algorithm (1)

Concept of the algorithm

- non-linear and non-convex constrains, hence hard to solve
  1. there are known special case instances that are NP-hard!
  2. formal proof for more would be a contribution
- it can still be found close-to-optimal solution!
  1. based on the MBIP formulation
  2. non-polynomial algorithm
  3. branch & bound techniques
- using **yalmip** in Matlab (that unites several optimization packages)
  - CPLEX (IBM)
  - MOSEK
  - GUROBI
  - SeDuMi

  under academic license

Dedicated heuristic (2)

Concept of the algorithm

- it is heuristic, but polynomial time!
- based on the links "centralities" regarding the flows
- greedy in nature
- Overview:
  1. for each potential link, if added in the network
     1.1 calculate all pairs max flow for each source and destination (*)
     1.2 compute the weighted objective sum/function (**)
     1.3 sort the weighted sums list in descending order (***)
  2. for each existing link, if removed from the network
     2.1 do (*), (**) and (***) from above
  3. try adding links from the sorted list in 2. until:
     (i) there is an improvement in the weighted sum/objective flow
     (ii) there are no more links than the given maximum C_{add}
  4. try removing links from the sorted list in 3. until:
     (i) there is an improvement in the weighted sum/objective flow
     (ii) there are no more links than the given maximum C_{rem}
  5. calculate the weighted sum/objective flow with the obtained topology (no link addition/removal constrains)

Dedicated heuristic (2): 1/out of 3

Pseudo code

```
addedLinks ⟵ [];
removedLinks ⟵ [];
for each l ∈ ℒ do
    tempG ⟵ G;
    totalFlow ⟵ 0;
    if l does not exist in G then
        for r ∈ requests do
            maxFlow ⟵ maxFlow(tempG,r);
            currentFlow ⟵ flow(start(r),end(r));
            if start(r) ∈ "Good clients" then
                totalFlow ⟵ totalFlow + α currentFlow;
            else if start(r) ∈ "Bad clients" then
                totalFlow ⟵ totalFlow - β currentFlow;
        end
        add (l, totalFlow) in addedLinks;
    else
        /* similar code for removedLinks */
    end
end
```

Dedicated heuristic (2): 2/out of 3 (cont.)

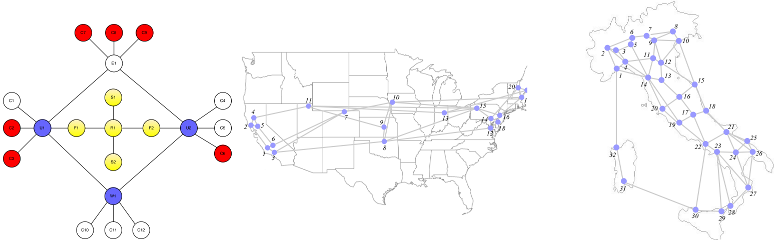
```
takeDescendingLinks(addedLinks, C_add); /*the highest traffic C_add links*/
takeDescendingLinks(removedLinks, C_rem); /*the high. traffic C_rem links*/
currentFlow ⟵ weightedObjectivemaxFlow(G); tempG ⟵ G;
for each entry ∈ addedLinks do
    totalFlow ⟵ 0; tempG ⟵ G + entry.link;
    for r ∈ requests do
        maxFlow ⟵ maxFlow(tempG,r);
        currentFlow ⟵ flow(start(r),end(r));
        if start(r) ∈ "Good clients" then
            totalFlow ⟵ totalFlow + α currentFlow;
        else if start(r) ∈ "Bad clients" then
            totalFlow ⟵ totalFlow - β currentFlow;
    end
    if totalFlow>currentFlow then
        currentFlow ⟵ totalFlow; G ⟵ tempG;
    else
        break;
    end
end
```

Dedicated heuristic (2): 3/out of 3 (cont.)

```
tempG ⟵ G;
for each entry ∈ removedLinks do
  | /* similar consecutive removal as the addition in the previous slide */
end
currentFlow ⟵ weightedObjectivemaxFlow(G);
return G, addedLinks, removedLinks, currentFlow;
```

Outline

Used topologies



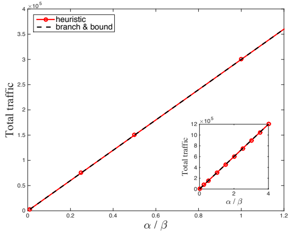
(a) SARNET topology. (b) ARPANET. (c) Italian backbone network.

Figure: Used topologies.

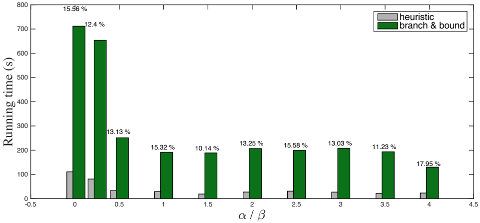
Table: Real networks used in the evaluation.

Networks	N	L	Description
SARNET	21	22	the project topology
ARPANET	20	32	first packet switching network
ITALY	32	62	main fiber connections in Italy

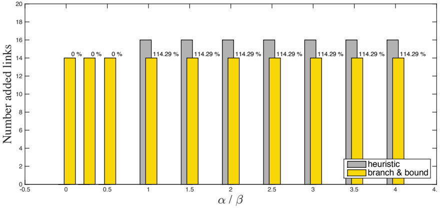
Results (SARNET, dense topology)



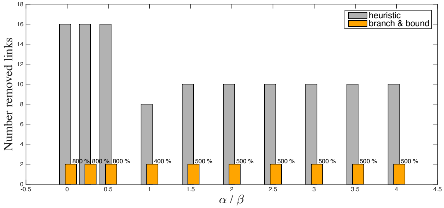
(a) Weighted flow sum.



(b) Running time.



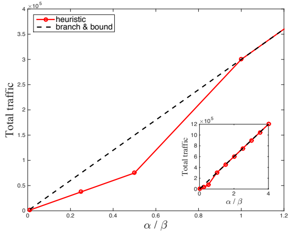
(c) Number of added links.



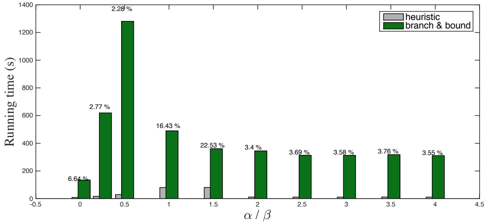
(d) Number of removed links.

Figure: SARNET comparison (dense topology).

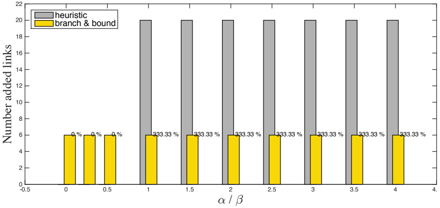
Results (SARNET, sparse topology)



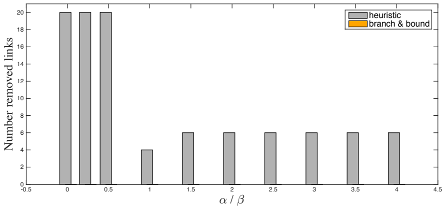
(a) Weighted flow sum.



(b) Running time.



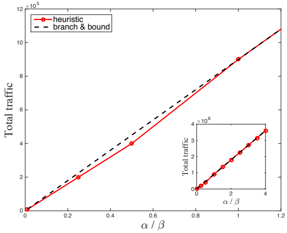
(c) Number of added links.



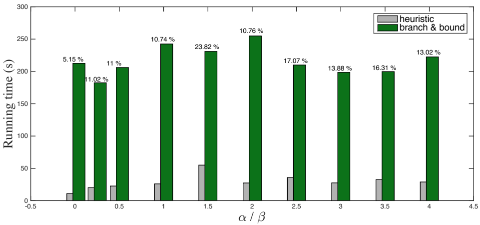
(d) Number of removed links.

Figure: SARNET comparison (sparse topology).

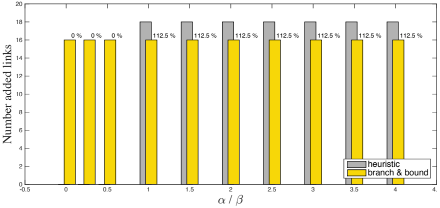
Results (ARPANET, dense topology)



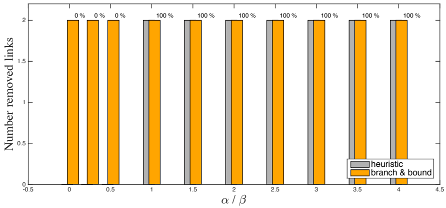
(a) Weighted flow sum.



(b) Running time.



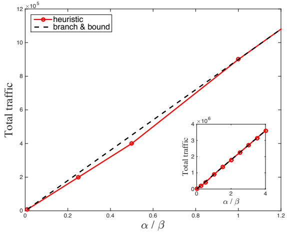
(c) Number of added links.



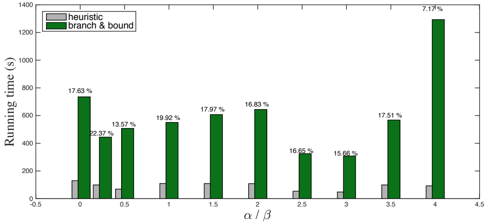
(d) Number of removed links.

Figure: ARPANET comparison (dense topology).

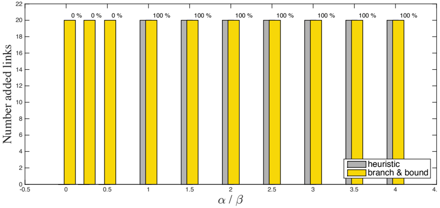
Results (ARPANET, sparse topology)



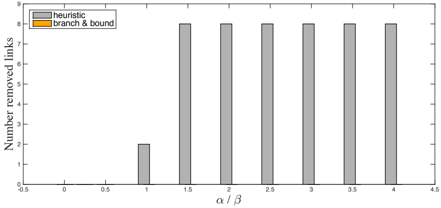
(a) Weighted flow sum.



(b) Running time.



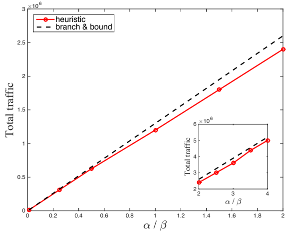
(c) Number of added links.



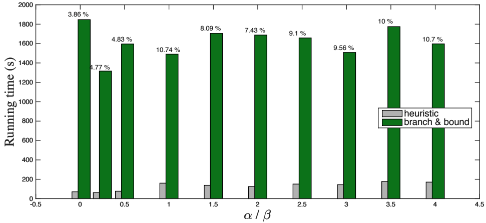
(d) Number of removed links.

Figure: ARPANET comparison (sparse topology).

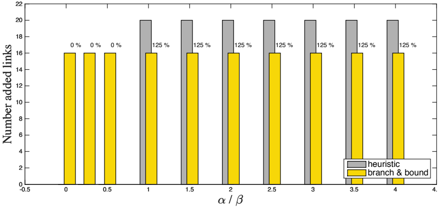
Results (Italy, dense topology)



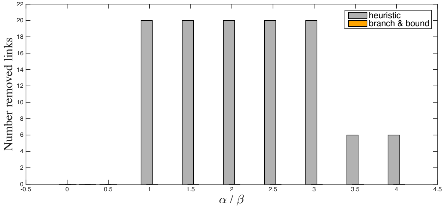
(a) Weighted flow sum.



(b) Running time.



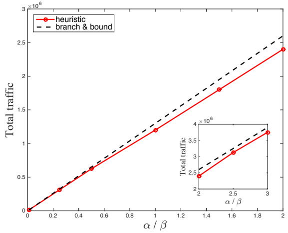
(c) Number of added links.



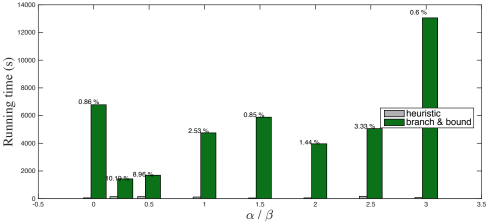
(d) Number of removed links.

Figure: Italy comparison (dense topology).

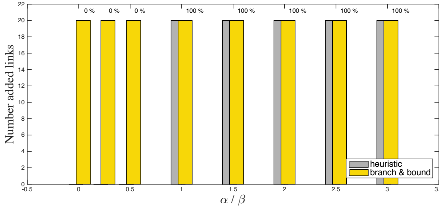
Results (italy, sparse topology)



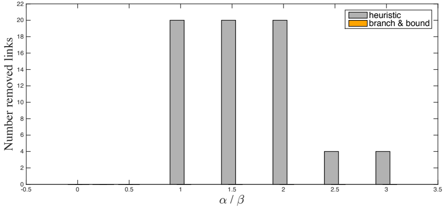
(a) Weighted flow sum.



(b) Running time.



(c) Number of added links.



(d) Number of removed links.

Figure: Italy comparison (sparse topology).

Outline

Conclusions

Contributions

- flow network models have been proposed
- two algorithms for solving the problem:
  1. Close-to-exact algorithm (branch & bound, bi-linear mixed programming)
  2. Dedicated greedy heuristic
- the greedy heuristic shows **surprisingly** good performance
  1. the two algorithms are **closed** in objective function performance (especially for $\alpha \geq \beta$)
  2. the heuristic is **significantly faster** than the MBIP - by factors 5 - 10
  3. 2 algorithms give different solutions:
     (i) numbers of added links similar for $\alpha \geq \beta$
     (ii) MBIP tends to not added as many links as the heuristic!
     reason the removal appears after the addition, hence "most of the job has been done" - perhaps trying variants

Conclusions

Possible future steps

- complexity of the problem
  1. known to be NP-hard for the general case
  2. proving the NP-hardness on some particular cases (only link addition or removal ...)
- integration with the SC demo and the real response of the strategies
- modeling the inter-domain issues
- modeling the virtualization

Questions?

email: s.trajanovski@uva.nl