Ph.D thesis Review

This thesis is about Patrolling Security Games, and the uncertainty models and problem formuations (spatial uncertainty and missed detection). Furthermore, the author explored more about multiple resources for defender and attacker and learning behaviour profiles of attacker. The author also provides mathematical complexity functions for each algorithm.

1 Patrolling Security Games

The game any of the two following events. The first one is when D patrols a target t that is under attack by A from less than $d(t)$ turns. In such case the attack is prevented and A is captured. The second one is when target t is attacked and D does not patrol t during the $d(t)$ turns that follow the beginning of the attack. In such case the attack is successful and A escapes without being captured. When A is captured, D receives a utility of 1 and A receives a utility of 0. When an attack to t is successful, D receives $1 - \pi(t)$ and A receives $\pi(t)$. The game may not conclude if A waits for every observed position of D, and never attacks. In such case, D receives 1 and A receives 0. The game is constant sum and therefore it is equivalent to a zero-sum game through a positive affine transformation. Here, D is the leader and A is the follower. In zero-sum games, the leader's strategy at the leader-follower equilibrium is its maxmin strategy and it can be found by employing linear mathematical programming, which requires polynomial time in the number of actions available to the players.

2 Uncertainty- Alarm System with only spatial uncertainty

The alarm system uses a number of sensors spread over the environment to gather information about possible attacks and raises an alarm signal at any time an attack occurs. The alarm system detects an attack but it is uncertain about the target under attack. Two examples of alarm systems for patrolling setting are provided.

1. Low-accuracy alarm system that generates the same signal anytime a target is under attack, not spatial information
2. System with a more accurate info about the location of the attack when t_i is attacked the alarm system generates s_i with high probability

When alarms goes off, D finds the best strategy starting from vertex v to responds that signal, using signal response game. $SRG - v$. An $SRG - v$ is essentially a two-level game in which A decides the target to attack and D decides the sequence of moves on the graph.

To find best strategy, the thesis presents algorithms for the exact and approximate equilibrium of the patrolling game. Here, it is finding the set of best routes (covering route) called covering set. Definitions are given in 4.1 to 4.9. According to inter set dominance, Find the maximal covering set, MAX_COV_SET, this will cost exponential time to compute.

Covering sets can be computed with:

1. Dynamic Programming
2. Branch and bound

For Dynamic Programming , DP-ComputeCovSets (exact algorithm, Algorithm 5), to compute the strategies available to D, but this cannot approximate maximal covering route. This will just give the set of covering routes. MonotonicLongestRoute (Approximate Algorithm, Algorithm 6) to find the maximal covering route.

For Branch And bound- When penetration time are relax and good heuristics will make depth first search find the maximal covering route faster than breath-first search. Exact Algorithm (Algorithm 7) The algorithm is a tree search using two global variables mainly CL_{min} and CL_{max} The covering routes will be returned in CL_{max} while CL_{min} is used for pruning. Algorithm 8, Tree-search is used for traversing the tree. If it is an open branch, it continues to call Tree-search recursively until the r is fully expanded. Pruning is done with Close()(Algorithm 9). Where CL_{min} minimal set of close routes is used. General idea: a closed route r belongs to CL_{min} only if CL_{min} does not already contain another $r_0 \subseteq r$.

CL_{max} maintains a set of the generated maximal closed routes. This will be returned as final solution.

The thesis presented Expand algorithm for route expansion which partitioned the targets $T(s)$ into T_{tight} and T_{large} where $d(t) < \delta * w * *_{v,t}$ is T_{tight} and T_{large} otherwise.

The insertion of target for expanded nodes works with the following rules:

- insertion of a target belonging to T_{tight} is always preferred to the insertion of a target belonging to T_{large}, independently of the insertion position;
- insertions of t 2 T_{tight} are ranked according to h considering first the insertion position and then the target;
- insertions of t 2 T_{large} are ranked according to h considering first the target and then the insertion position.

This has exponential computational complexity since it builds full tree of covering routes.

2.1 Approximate Algorithm

p denote completeness and $p < 1$ in favour of a less computational effort

When p is $k/|T(s)|$, the complexity becomes polynomial in size of input. Proof in Theorem 4.9.

The linear program in Page 84 will give the optimal signal response strategy for D with $U(r,t) = \pi(t)$ if $t \notin r$ and 0 otherwise. The complexity is polynomial since it has $|T| + |S|$ constraints and $O(|V||S|max_{v,s}|R_{v,s}|)$ variables. And it only depends on $max_{v,s}|R_{v,s}|$ which only depends on $|T(s)|$

But Theorem 4.10 says that in Patrolling Games without false positives or false negatives(missed detection), the best patrolling strategy is to find the best placement in v^*.

When no false positives and no missed detection are present, the optimal Defender strategy is to stay in a fixed location, wait for a signal, and respond to it at best. This strategy keeps being optimal even when non-negligible missed detection rates are allowed. Thus, in conclusion, for the best equilibrium, it is reduced to problem of finding best placement v^*. (SolveSRG(v) Algorithm 11). The complexity is linear in $|V|$.

2.2 Experiments

The simulations for real life case (Expo) show that the optimal patrolling strategy coincides with such fixed placement even under false negatives rates of at least approximately equals to 0.3.

3 Alarm Systems with Missed Detection

3.1 Problem formulation

$p(s|t) =$ the probability of generating a signal s given that target t has been attacked
$s_0 =$ null signal corresponding to the absence of alarms
$s_i(i > 0) =$ an alarm signal caused by some attack
Thus $p(s0|t) = \alpha$ for every target t where α is missed detection rate.

The game being constant sum, the best leader's strategy is its maxmin/minmax strategy.

There are two parts defender strategy ($sigma_p^D$ and $sigma_a^D$) $sigma_p^D$ is patrolling strategy is adopted when no alarm signal is received (s_0 is received). $sigma_a^D$ is signal response strategy when alarm signal is received.

Here, a special type of patrolling strategy called covering cycle is proposed. Where $\alpha = 0$, the optimal patrolling strategy is placement strategy. If $\alpha > 0$, any placement-based strategy will fail to capture attacks in the occurrence of a false negative.

Finding equilibrium (minmax strategy) can be formulate with a non-linear mathematical programming formulation with $l = 1$ as shown in Pg.110. (Question & Comments 5)

Unlike, the algorithms without missing detection, this problem cannot be separated into two independent programs for best $sigma_p^D$ and best $sigma_a^D$: in fact, doing so would inevitably provide sub-optimal patrolling and signal response strategies.

Resolution Approach is to use support graph of optimal patrolling strategy. Since the alarm system only protects targets via signal responses. By using support graph, the algorithm performance improves to $O(2^|T|)$ instead of $O(2^|V|)$. And for each support subset, the algorithm guess the remaining vertices composing the support by querying a patrolling strategy oracle.

3.2 Patrolling Strategy Oracle

The thesis proposed two patrolling strategy oracles: Covering cycle oracle and Random Oracle
Covering cycle oracle, tries to compute a covering cycle protecting a given subset of targets T_0. If found, no better patrolling strategy can be given for any support containing the same targets since it always guarantees capture on them, eve in presence of missed detections.

The CCO must solve a PSPACE-hard problem (Question & Comments 1), so no exact efficient method can be designed. For the case where algorithm is provided with correctness guarantees, it limits the solution length to k—V— and applies a backtracking search. The algorithm is still exponential in the worst case (even for a fixed k), but has an acceptable performance on realistic settings, even with large numbers of targets. The CCO is the leading oracle: when a covering cycle is found it is always the preferred choice for the considered subset of targets.

The Random oracle (RO) computes the optimal patrolling strategy. This problem can be done with non-linear programming techniques.

The RO is meant to work as a backup and is run in parallel to the CCO (as shown in Figure 5.3). When the CCO terminates with no answer or reaches a timeout, check if RO was able to terminate and, in the positive case, the patrolling strategy from RO can be used to continue with the search.

3.3 Signal Response Oracle (SRO)

The SRO takes as input a patrolling strategy (returned by CCO or RO) and computes the optimal signal response from every vertex in the support induced by that strategy.

If is a covering cycle, then each target in the support will be fully protected upon a missed detection. (Question & Comments 5). The original covering cycle SR strategy will be linear program if $P_{capture}$ is always 1 and $sigma_p^D$ is fully protected from missed detection. I_v is 1 and $sigma_p^D$ can be removed. Implementation of SRO adopts DP-ComputeCovSets and MonotonicLongestRoute .

3.4 Target Selection Heuristics

The first one follows a dynamic approach where the ranking of possible solutions to explore changes as new subsets are explored.

Dynamic Method (greedy search):
This method progressively includes in the support targets that are the most strategically appealing to the Attacker.

Two static approaches where the order of preference between the candidate solutions is fully determined by the problem instance and does not vary during the search process. By ranking targets according to their value or by ranking targets according to target distance.

4 Multiple resources for defense

The Defender can control an arbitrary number of resources, denoted by m.
Steps:

1. Minimizing number of defensive resources
   - Computing the minimum covering placement is in log-APX. (Theorem 6.2, 6.3 with proofs)
   - But for tree and cycle graphs – it is solvable by polynomial time with (Algorithm 12 - Tree Placements)
2. Covering routes are computed (same as single resource)
3. Signal Response strategy
   - Three signal response oracles

4.1 Fully coordinated SRO

The game can be solved computing the maxmin strategy by linear programming and it is NP-hard.

4.2 Partially Coordinated SRO

Non-Linear Program (NLP) whose size is linear in the number of resources compressing exponentially the size of the game. And solved by means of global-optimization techniques. It is able to find the global maximum within a given accuracy and, in the case the posed time limit is expired, able to return the quality of the best solution found w.r.t. the tightest upper bound found.

4.3 No coordinated SRO

NC-SRO does not any coordination among multiple resources it's just multiple instances of single SRO agents

Resolution Approach can be seen in Figure 6.1. First find the minimum-size resource placement by solving the associated SET-COVER formulation, if not solved in a short time, adopts greedy algorithm followed by local search. Compute covering placement, covering routes and finally run SRO.

In the Experiments to find the best covering placement with multi-defender setting the algorithm is implemented with NC-SRO to get the minimum computation time. The other is to benchmark quality of SROs, shown that FC-SRO perform better than PC-SRO even though time to time, it cannot finish due to the timeout. For the utility trend experiment, FC-SRO and PC-SRO utilities increase, but FC-SRO faster in finding good solutions than PC-SRO.

5 Multiple resources for attacks

5.1 Model Formulation

The utilities of D and A are $(-\sum_{i=1}^{k} \gamma_i \pi(t_i), \sum_{i=1}^{k} \gamma_i \pi(t_i))$, where $\gamma_i = 0$ if A attacks target t_i at τ and the patroller traverses t_i by $d(t)$ turns after τ , catching the attacking resource, otherwise, if A completes the attack on target t_i without being detected, $\gamma i = 1$. Since the game is in perfect information, the suitable solution concept is the Subgame Perfect Equilibrium (SPE), which requires a strategy profile to be a Nash Equilibrium for each subgame of the game tree. And there is a polynomial time algorithm to find the equilibrium path of SPE.

5.2 Simultaneous attacks

Denote k as number of resources for the attacker, if k = 1, the placement strategy is the best and it can be solved in polynomial time of $|V|$ For multiple resources k, the attacker's combinations on target $|T|$ is $\binom{T}{k} \approx |T|^k$ The space of actions is exponential to both D and A. The complexity is NP-hard.

5.3 Sequential attacks

Simultaneous attacks ¡ sequential attacks for attacker gain. (Proposition 7.3)

5.3.1 For attacks with 2 resources

For equilibrium path of SPE, it suggests PathFinder and AttackPrediction algorithms (Algorithm 13 and 14). The goal is to figure out if it can cover t' within its deadline and what route should be followed. $M(i;j) = (r_{si}; r^c_{ij}; u_{ij})$ consists of the route r_{si} from v_s to v_i, the best covering route r^c_{ij} from vertex v_i at time instant j to cover the targets under attack and the utility u_{ij} for D associated to such route. The algorithm will return M with highest utility for defender.

5.3.2 For Attacks with k resources

A third dimension l to M considering all the combinations with repetitions of the $k-1$ targets under attack. Since a target can be visited even after its deadline, each element of M contains also the set of covered targets.

AttackPrediction has a cost equal to $O(2^k k^5)$ and the cost of invoking the extended version of PathFinder is $O((|V| + k)^k 2^k k^6 |V|^2)$

5.4 Resolution approach with coordinated defense

1. Robustness to the wrong guess
    - Overestimating and underestimating (both has loss for defender)
    - Competitive factor (worst case ratio v/v*) for defender
        - Defender gets competitive factor of 0 for the wrong guess (both underestimating and overestimating).
2. Deploy online algorithms?
    - For deterministic, the best competitive factor is 1/k-1 where k is attacker resources
    - Sometimes randomization will give better competitive factor than deterministic algorithms.

6 Learning attacker's profile

Attacker's Behavior Identification in Security Games (Theorem 8.1)

6.1 Analyzed attacker profiles

1. Stochastic attacker
2. Strategy aware attacker - Stackelberg Attacker
3. Strategy aware attacker - SUQR Attacker (Question & Comments 4)

If the defense system is targeted for stochastic attacker and the attacker became stackelberg attacker there is a loss for defender (vice versa) For the loss, loss of $stretagy\_aware\_algo > stochastic\_algo$

6.2 Identifying the attacker

1. Follow belief (Algorithm 15)
2. Follow regret (Algorithm 16,17)

6.3 Experiment Results

- Follow belief is more efficient
- The FB algorithm presents an upper bound over the pseudo-regret that is strictly better than that of MAB algorithms, i.e., a constant regret $O(1)$ in N vs. a logarithmic one $O(lnN)$.
- In both FB and FR, when the behavioral profile of the opponent can only be either Sta or Sto, both algorithms are twice more efficient than in cases in which SUQR adversaries are introduced.