Improvements to the linear optimization models of patrol scheduling for mobile targets

Canadian AI 2015/Submission #38

No Institute Given

Abstract. Recent work in the field of security for the generation of patrol schedules has provided many solutions to real-world situations by using Stackelberg models and solving them with linear programming software. More specifically, some approaches have addressed the difficulties of defining patrol schedules for moving targets such as ferries to minimize their vulnerabilities to terrorist threats. However, one important aspect of these types of problems that hasn't been explored yet concerns the concept of time-windows. In this work, we show the relevance of considering time-windows when generating solutions such as to attend to a broader class of problems and generate sound solutions. We propose some improvements to the model for the generation of patrol schedules for attacks on mobile targets with adjustable time durations while keeping the constraints linear to take advantage of linear programming solvers. To address the scalability issues raised by this new model, we propose a general column-generation approach composed of a master and slave problem, which can also be used on the original problem of patrol generation without time-windows. Finally, we discuss and propose a new two-phase equilibrium refinement approach to improve the robustness of the solutions found.

1 Introduction

In the last few years, there has been a growing interest in the field of security to improve and automate some of the complex tasks which would normally need to be done by a field expert. The tasks we are referring to relate to the definition of optimal patrol schedules to protect specific locations or targets. Traditionally, these tasks have been done by hand using expert knowledge of the problem to find the optimal schedule which would minimize the chances for a hostile agent to successfully perform an attack or other form of single action.

Without specific metrics to objectively quantify the quality of the schedules generated, a great risk is always present that a small window of vulnerability may have been overlooked, leaving some targets exposed to potential attacks.

A natural approach to solve these types of problems is to model them using game theory. The problem is formulated in such a way that the goal is to find an optimal set of schedules that minimize the probability of success for hostile agents. More specifically, Stackelberg models allow for a representation of this class of problems such that they can often be solved using standard linear

programming software very efficiently. Various flavors of security problems have been explored in [1], [2], [3], [4] where aspects such as the number of patrollers present, the topology of the graphs or the type of actions of potential attackers. Many real-life examples can be also be found in [5] describing how many of these challenges have been confronted.

The specific problem addressed in this article relates to the problem of moving targets, previously studied in [6], for which a linear program model was proposed with discretized values for time and distances to generate the patrol schedules. The solution provided was very clever but had some scalability issues for larger problems (as studied for the discrete case in [7]).

In this work, we provide a scalable model for the continuous version of this problem. Our contribution consists in adding the notion of a minimal time window required by the attacker to successfully complete an attack. In order to do so, we work on a full representation of the problem rather than a compact one. This different representation allows for a column generation approach to be used where a master problem will work on a subset of all the paths possible in the problem, and a slave problem will generate new paths for the master problem by solving a shortest path problem.

We then propose a two-phase equilibrium refinement approach where some of the constraints of the problems are relaxed in a second phase of the algorithm to minimize the utilities subject to a fixed bound. Finally we present some results showing a great improvement for the utility values for the refinement approach.

2 Domain definition

For the aforementioned application, a schedule is defined as a piecewise linear function normalized between 0 and 1 (i.e., a ferry travelling between point A and point B). $F_q : q = 1..Q$ is a set of Q mobile targets with corresponding schedules. The utility gained by an attacker is also defined as a piecewise linear function with a value between 0 and 10. The time is discretized in M points, or $M-1$ intervals and the distance is discretized in N points (also $N-1$ intervals), both normalized between 0 and 1. A compact representation of the transition graph is used in the first model, which helps to reduce the number of variables needed to represent the problem in this formulation. Instead of associating probabilities to a list of a full trajectories, probabilities will be associated to each $f(i, j, k)$ where a patrol goes from node i to node j at time-step k (see Figure 1). To keep the graph consistent, and set a starting point for the initial generation of the trajectories, some constraints on the probabilities are added as $p(i, k)$. $p(i, k)$ is the probability that a patrol is at node i at time-step k. From these probabilities, a full trajectory can be extracted as a Markov strategy when needed.

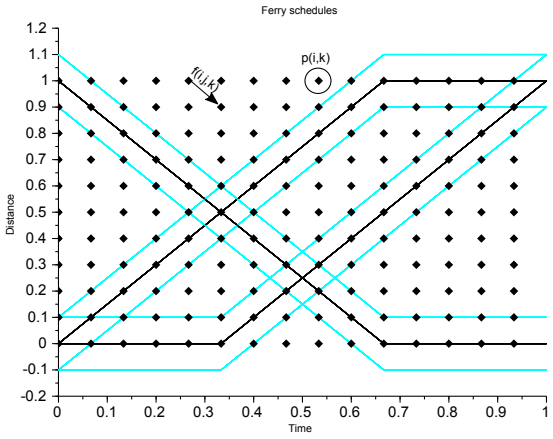

Fig. 1. Discretized transition graph in compact representation.

3 Minimum Time-windows

The problem of patrol schedule assignment to protect moving targets travelling in a straight line between two points can be resumed to an assignment of position and time to a patrol, in regards to maximum speed constraints.

The ideal formulation of the problem would take into account the continuous aspects of the patrols' trajectories in time. However, such a formulation is not possible given the complexity it would entail to find a solution which generates continuous trajectories maintaining a Stackelberg equilibrium.

An easier and faster way is to approach this problem through a linear programming model to take advantage of the many very efficient and fast existing solvers. The solution proposed in [6] was to discretize the time and distance variables to reduce the problem to a linear one. What we propose in this article is related to that model but with the added complexity of minimum time-windows for attacks to occur.

Using a model without time-windows as done in [6] can be very efficient but lacks credibility for some applications. It is not unrealistic to expect an attack on a moving target to occur with a certain duration. The hypothesis we make for this work is that an attacker's utility will be greatly decreased if a patrol covers the target at least once during this interval, diminishing chances of a successful attack. Figure 2 shows the utilities for an example scenario where patrol schedules were generated for 2 patrols and 3 targets over a 30 minutes time-span (normalized between 0 and 1). The black line shows the utility as computed for each time-steps and intersections. It is possible to see that some spikes in the utility only occur at a very small time-interval. More precisely, the utility at time 0.26 is equal to 3.57. However, this spike has a very short duration, and if we were to suppose that an attack has a minimum time-window of the size

of one time-step, the real utility value would be reduced to 1.6, which is showed by the yellow curve in the graph. The yellow curve was generated by iteratively going through all time-steps and intersections, and taking the minimum value of the utility for the size of the time-window (1/16).

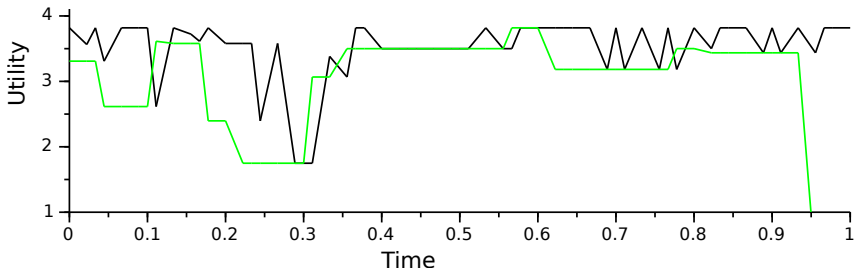

Fig. 2. Shows in green the reduced attacker's utility when taking into account a time window of width $\frac{1}{16}$ as compared to the pointwize utility in black

Although we could intuitively use such a heuristic (taking the max of the values computed over an interval) to compute the utility in regards to a time-window for a solution generated using the previous model, the resulting values will not be accurate if the values from a compact representation are used for reasons detailed in the next section.

3.1 Model

To properly take into account minimum time-windows for the attacker utility the following model needs to be solved:

$$\min_{f(i,j,k),p(i,k)} z \tag{1}$$

$$f(i,j,k) \in [0,1] \qquad \forall i,j,k \tag{2}$$

$$f(i,j,k) = 0 \; \forall i,j,k : |d_j - d_i| > v_m \delta t \tag{3}$$

$$p(i,k) = \sum_{j=1}^{N} f(j,i,k-1) \qquad \forall i,k > 1 \tag{4}$$

$$p(i,k) = \sum_{j=1}^{N} f(i,j,k) \qquad \forall i,k < M \tag{5}$$

$$\sum_{i=1}^{N} p(i,k) = 1 \qquad \forall k \tag{6}$$

$$z \geq AttEU(F_q,t) \quad \forall q,t : (t + \alpha_t) < t_M \tag{7}$$

The constraints (2),(4),(5),(6) are defined such as to keep graph consistency. More precisely, to insure that probabilities inherent to a patrol's schedule remains possible such that there are as many patrols that enter a node as there are that leave it. Constraint (3) insures that a ship's maximum speed is taken into account when verifying which nodes can be reached in-between two time intervals. For future reference, we define the graph related constraints as

$$G = \{f(i,j,k), p(i,k) : (2),(3),(4),(5),(6) \text{ are satisfied}\} \tag{8}$$

The utility of an attacker, defined as $AttEU()$ and used on line (7), is where lies the main difficulty for representing this problem. It is dependent on the utility specified for the problem as a piecewise linear function, on a probability of successful detection of attacker by the patrol, and more importantly, on the probability that a patrol's planned trajectory will intersect at least once with the target it is protecting. The computation of this utility relies on the hypothesis that we can directly extract the chances of a successful attack from the current graph representation. However, using a compact graph representation, crucial information is lost in regards to the probabilities of a successful attack over a time-window.

3.2 Probability of a Successful Attack

One of the difficulties in implementing a minimum time-window for potential attacks relates to the computation of the probabilities of success of the attack. Using a compact model with discretized time-steps as previously described has the advantage of significantly reducing the number of variables necessary to represent the problem from $O(N^M)$ to $O(MN^2)$. The problem with this representation, however, is that for attacks with time windows some of the information necessary to compute the probabilities of an attack being detected is lost.

To evaluate the probabilities of an attacker being detected in a time-window, it becomes necessary to evaluate the full trajectory of patrols in this time window. This can be seen in Figure 3 where in a model without time-windows and compact representation, probabilities are associated to edges $E_{1..6}$ time-steps t_k and intersections θ. In this figure, to compute the probability that a patroller will cross in the window of attack, a naïve and wrong approach would be to add the probabilities of all edges E crossing with the attack window. Doing so will result in possibly overestimating the probabilities. In the example (Figure 3), four trajectories may be followed by a patroller. E_1-E_6, E_2-E_4, E_2-E_5 and E_3-E_5. If all intersections are added together, then one would add the probabilities of taking edges E_2, E_5, E_6, even though a patrol taking the path E_2-E_5 should only count once. The correct way of computing the probability that a patrol crosses at least once with the attack-window corresponds to $E_1 * E_6 + E_2 * E_5 + E_3 * E_5$, which introduces a nonlinear expression to the problem in the compact form.

One way of representing the probability of detection of an attack for a time-window combined with the utility gained by the attacker is to use a full representation in the following way:

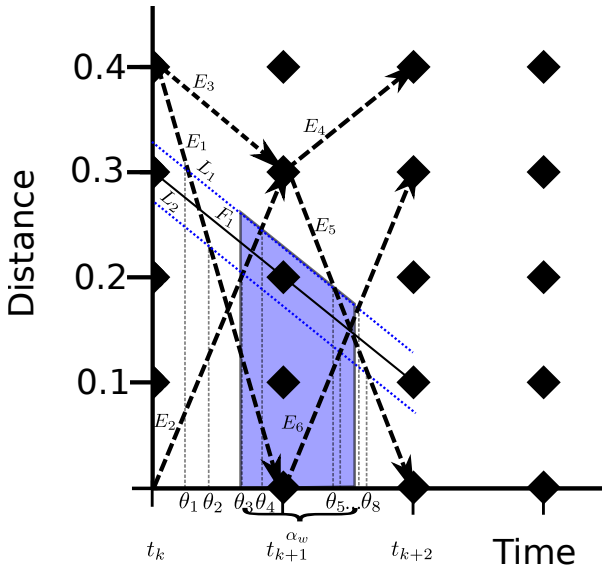

Fig. 3. L_1 and L_2 represent the distance from which the target F_1 is considered as covered by the patrol. $E_{1..6}$ is the compact representation of trajectories to be followed by the patrols. The darkened area shows a minimum attack time-window (of size α_n).

$$\begin{aligned} \min \quad & z \\ \text{s.t.} \quad & \\ & \sum_{p \in P} \lambda_p = 1 \\ & z \geq (1 - \sum_{p \in P} I_p^{qt} * \lambda_p) * U(t) \qquad \forall q, t \\ & \lambda_p \geq 0 \end{aligned} \tag{9}$$

where P is the set of all possible paths in the graph and λ_p is the variable associated with the probability of following path p. Using the same trick as in [6] to take into account the continuity of the problem, I_p^{qt} is a vector indicating whether path p intersects at least once with ferry q between t and $t + \alpha_w$.

$U(t)$ is the utility gained by the attacker on average between time t and time $t + \alpha_w$. It is assumed that $t \in T$ is the set of all time steps as well as time-intersections shown as θ in Figure 3.

3.3 Column Generation for Mobile Targets

To solve the optimization model with paths instead of arcs, it becomes necessary to find a way to work on a subset of paths rather than all of them, since the problem becomes too large as stated earlier.

A classic approach in linear programming is to use column generation to work around this problem.

Rewriting the problem without time-windows as a list of arcs $x_{ij} \in A$, and adding a source (0) and target (M+1) node to get rid of the constraints $p(i,k)$ we get:

$$
\begin{array}{ll}
\min_x z & \\
s.t. & \sum_{j:(0,j)\in A} x_{ij} = 1 \\
& \sum_{j:(i,M+1)\in A} x_{iM+1} = 1 \\
& \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = 0 \qquad i > 1, i < M+1 \\
& z + \sum_{ij} I_{ij}^{qt} x_{ij} * U(q,t)) \geq U(q,t) \ \forall q,t \\
& x_{ij} \in [0,1] \quad \forall i,j
\end{array}
\tag{10}
$$

where I_{ij}^{qt} is a pre-calculated vector for all patrols intersecting with F_q at time t equal to 1 (0 otherwise).

$$
I_{ij}^{qt} = \underbrace{\left[\begin{array}{ccc} I_{1,1}^{1} & \cdots & I_{2,2}^{1} \\ \hdashline \vdots & & \\ \hdashline I_{1,1}^{q*t} & \cdots & I_{2,2}^{q*t} \end{array}\right]}_{ij} \Bigg\} q*t
\qquad
X = \underbrace{\left[\begin{array}{ccc} x_{1,1,1} & \cdots & x_{1,1,3} \\ \hdashline \vdots & & \\ \hdashline x_{2,2,1} & \cdots & x_{2,2,3} \end{array}\right]}_{p} \Bigg\} ij
$$

We can replace the x_{ij} by $\sum_{p\in P} x_{ijp}\lambda_p$ where the constant $x_{ijp} \in \{0,1\}$ represents path p and λ_p the flow associated with this path, and divide the problem in a master and slave problem. The master problem will work on a subset of paths $P_s \in P$, while the slave problem will generate new paths to be added to the master problem.

We get the problem:

$$
\begin{array}{lll}
\min & z & \\
\text{s.t.} & & \\
& \sum_{p\in P_s} \lambda_p = 1 & \\
& -z - \sum_{ij} I_{ij}^{qt} \sum_{p\in P} x_{ijp}\lambda_p * U(q,t)) \leq -U(q,t) \ \forall q,t & \\
& \lambda_p \geq 0 &
\end{array}
\tag{11}
$$

The reduced cost can be computed in relation to the earlier representation, which we will call the Master Problem (MP):

$$
\begin{array}{ll}
\min_x & z \\
\text{s.t.} & \\
 & \sum_{p \in P} \lambda_p = 1 \qquad\qquad\qquad\qquad\qquad : \pi_0 \\
 & z + \sum_{p \in P} (\sum_{ij} I_{ij}^{qt} x_{ijp}) * \lambda_p \geq \sum_{ij} I_{ij}^{qt} \qquad \forall q,t \qquad : \pi_1 \\
 & \lambda_p \geq 0 \qquad\qquad \forall p
\end{array}
\tag{12}
$$

Taking π_0 and π_1 as the dual variables, the reduced cost of one column (path) can be computed as:

$$c_p = 1 - (1 + \sum_{ij} I_{ij}^{qt} x_{ij}) * \pi_1 + \pi_0 \qquad \forall q,t$$

.

The column-generation sub-problem, which we will define as Slave Problem (SP), will be to find a path which minimizes the reduced cost:

$$
\begin{array}{ll}
\min\limits_{x_{ij}} & 1 - \pi_1 + \sum\limits_{ij} I_{ij}^{qt} x_{ij} * \pi_1 + \pi_0 \\
s.t. & \sum\limits_{j:(0,j) \in A} x_{ij} = 1 \\
 & \sum\limits_{j:(i,M+1) \in A} x_{iM+1} = 1 \\
 & \sum\limits_{j:(i,j) \in A} x_{ij} - \sum\limits_{j:(j,i) \in A} x_{ji} = 0 \qquad i > 1, i < M+1 \\
 & x_{ij} \in [0,1] \; \forall i,j
\end{array}
\tag{13}
$$

One issue of this model concerns the computation of the full matrix I for all possible intersections. It is impossible to only generate the intersections for the relevant paths in the master problem since when generating new columns (paths) to add in the slave problems, the reduced cost has to take into account the constraints in the master problem. In the case of a single resource, this means there will be as many constraints as there are variables. However when more resources are available, the number of variables will expand exponentially, making this approach much more interesting in terms of scalability.

4 Equilibrium refinement

The notion of equilibrium refinement is not a new one and has been explored in articles such as [6], [8] and [9]. For a problem similar to the one presented here, the approach proposed generates solutions under the assumption that a defender will want to minimize his maximum potential lost (attacker utility) at any single point in time.

Operating under such an assumption is logical for this security problem since the attacker has no reason to attack a target for which his gain (utility) will be lower than if he attacks at another time.

However, as was pointed out by [6], this may leave other points of attack needlessly vulnerable if the attacker doesn't attack to the critical point (we consider a critical point as a maximum value which cannot be further decreased). This weakness in the model becomes most obvious when using a LP solver based on the simplex method. The solutions which are generated by this method will normally be on extreme points, which will often mean most of the utilities at all time steps will be at the maximum value.

If we relax slightly the initial hypothesis on the rationality of the attacker, the problem may now shift significantly and have an important impact on the results. If, for example, the attacker doesn't possess all of the information concerning the schedules of the patrols, or is unable to recreate the model used in this paper to generate the probabilities associated with each patrol, he may decide to attack at a sub-optimal point in time. There is no reason why we shouldn't take advantage of this.

The problem can be seen as a desire to minimize the time as well as the utility to be gained for performing an attack, or more precisely, the integral of the function defining the attacker's utility.

Obviously, the approach we propose here is just one of many possible tools which should allow an expert to better select a solution approach, and its value should be defined in regards to the type of problem to be resolved.

We define a critical time-step c_t as a point in time for which the utility of the attacker will be the highest, given a selection of discretized time and distance points. In the following model, these critical time-steps will be a subset of the intersections of the position of the patrols in time with the targets they are covering. Recall the definition of the graph constraints (8).

Phase 1:

$$\min_{f(i,j,k),p(i,k)} z \tag{14}$$

$$f(i,j,k), p(i,k) \in G \tag{15}$$

$$z \geq AttEU(F_q, t_k)\ \forall q, k \tag{16}$$

Phase 2:

$$\min_{f(i,j,k),p(i,k)} \sum_{i=1}^{P} v_i \tag{17}$$

$$f(i,j,k), p(i,k) \in G \tag{18}$$

$$v_i \geq d_k * AttEU(F_q, t_k)\ \forall q, k, i \tag{19}$$

$$v_i \leq z \quad \forall i \tag{20}$$

In $Phase1$, a solution will be generated for the initial problem to compute the solution to the Stackelberg model. Once this solution is available, another model will be solved in a second phase where the objective function will now

be modified to be the summation of the values at each critical time-points, rather than the max as was done in phase 1. By setting an upper bound on the values of each critical time-point, we insure that the solution stays valid as a Strong Stackelberg Equilibrium, but we also improve the other time-points where possible attacks could occur, in the case of a sub-optimal choice by the attacker.

Figure 4 shows the trajectories of three ferries, for which two patrols are to be assigned to protect. Using similar parameters as in [6], we can see in Figure 5 an initial patrol schedule generated for the problem. It is possible to see that the patrols sometime actually take an arc that doesn't protect any of the ferries while a better choice would be possible. This sort of behavior is a result of doing a min-max approach. On Figure 6 we can see a net improvement over the trajectories that are generated, meaning that less unnecessary moves will be done by the patrol when they could be monitoring a ferry. Figures 7 and 8 show the utilities which have been computed at each phase, and we can see the difference between those two sets of utilities in Figure 9. It is obvious from this last figure that the approach we propose has a great impact on the improvement of the final solution for the patrol schedules since only at one point does the value slightly worsens (but is still bounded by the maximum utility computed in the first phase).

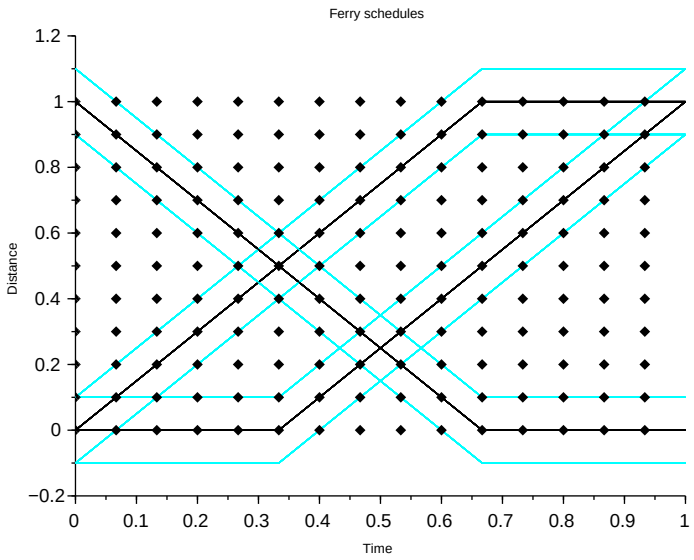

Fig. 4. Ferry trajectories with vulnerability corridors.

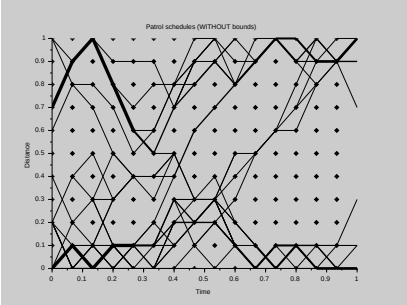

Fig. 5. Trajectories generated in phase 1 of base model.

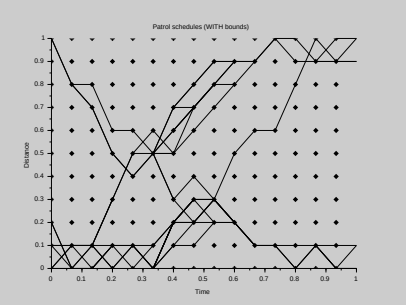

Fig. 6. Trajectories generated in phase 2 with bounded utility value for all time steps.

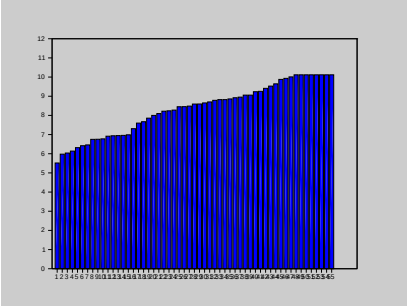

Fig. 7. Utilities ordered by magnitude with base model.

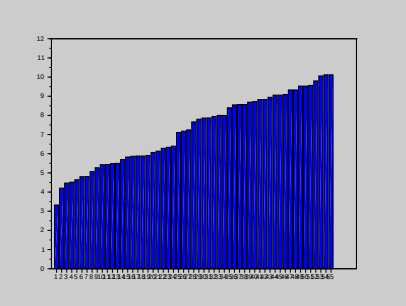

Fig. 8. Utilities ordered by magnitude after second pass.

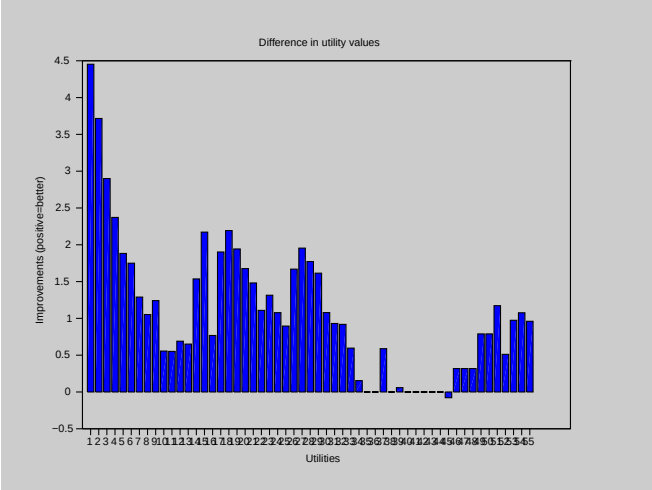

Fig. 9. Difference of utility values between first and second pass

5 Conclusion

We proposed a model to take into account the minimal duration of an attack. This increases the realism but generate much larger linear programs for which scalability issues are unavoidable.

We believe that the column generation approach presented in this article will be of great use to tackle problems where it is necessary to take into account time-windows. The way the column generation approach is implemented will obviously have a great impact on the final result, and it allows for a great flexibility between solution quality and memory/cpu usage. Many refinement can also be done to the model, possibly for the generation of the paths for the slave problem, by using heuristics such as to generate promising trajectories first and possibly converge more rapidly to a solution.

Equilibrium refinement is an heuristic mean to reduce the time windows of high vulnerability. Our two phases approach yields very good improvements.

References

1. Basilico, N., Gatti, N., Amigoni, F.: Patrolling security games: Definition and algorithms for solving large instances with single patroller and single intruder. Artificial Intelligence **184–185** (2012) 78 – 123
2. Basilico, N., Gatti, N., Amigoni, F.: Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1. AAMAS '09, Richland, SC, International Foundation for Autonomous Agents and Multiagent Systems (2009) 57–64
3. Bošanský, B., Lisý, V., Jakob, M., Pěchouček, M.: Computing time-dependent policies for patrolling games with mobile targets. In: The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 3. AAMAS '11, Richland, SC, International Foundation for Autonomous Agents and Multiagent Systems (2011) 989–996
4. Lisý, V., Píbil, R., Stiborek, J., Bosanský, B., Pechoucek, M.: Game-theoretic approach to adversarial plan recognition. In Raedt, L.D., Bessière, C., Dubois, D., Doherty, P., Frasconi, P., Heintz, F., Lucas, P.J.F., eds.: ECAI. Volume 242 of Frontiers in Artificial Intelligence and Applications., IOS Press (2012) 546–551
5. Tambe, M.: Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned. 1st edn. Cambridge University Press, New York, NY, USA (2011)
6. Fang, F., Jiang, A., Tambe, M.: Protecting Moving Targets with Multiple Mobile Resources. Journal of Artificial Intelligence **48** (2013) 583–634
7. Xu, H., Fang, F., Jiang, A.X., Conitzer, V., Dughmi, S., Tambe, M.: Solving zero-sum security games in discretized spatio-temporal domains. In: Proceedings of the 28th Conference on Artificial Intelligence (AAAI 2014), Québec, Canada. (2014)
8. Miltersen, P., Sørensen, T.: Computing proper equilibria of zero-sum games. In van den Herik, H., Ciancarini, P., Donkers, H., eds.: Computers and Games. Volume 4630 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2007) 200–211
9. Fudenberg, D., Tirole, J.: Game Theory. 1 edn. The MIT Press, Cambridge, MA (1991)