Scalable Randomized Patrolling for Securing Rapid Transit Networks

Pradeep Varakantham, Hoong Chuin Lau, Zhi Yuan
School of Information Systems
Singapore Management University
{pradeepv,hclau,zhiyuan}@smu.edu.sg

Abstract

Mass Rapid Transit using rail is a popular mode of transport employed by millions of people in many urban cities across the world. Typically, these networks are massive, used by many and thus, can be a soft target for criminals. In this paper, we consider the problem of scheduling randomised patrols for improving security of such rail networks. Similar to existing work in randomised patrols for protecting critical infrastructure, we also employ Stackelberg Games to represent the problem. In solving the Stackelberg games for massive rail networks, we make two key contributions. Firstly, we provide an approach called RaPtoR for computing randomized strategies in patrol teams, which guarantees (i) Strong Stackelberg equilibrium (SSE); and (ii) Optimality in terms of distance travelled by the patrol teams for specific constraints on schedules. Secondly, we demonstrate RaPtoR on a real world data set corresponding to the rail network in Singapore. Furthermore, we also show that the algorithm scales easily to large rail networks while providing SSE randomized strategies.

Introduction

Massive rail networks have reduced the transportation time significantly in many of the major cities across the world including Beijing, Singapore, Paris, New York, Hong Kong, London. However, due to the number of people they serve every hour of every day, these networks can be soft targets for various types of criminals (terrorists, pick pocketers, fare offenders, etc.) without proper security. In this paper, we consider this problem of scheduling randomised patrols for massive rail networks, where it is very difficult to have enough patrols to secure all the rail stations all the time.

Recent research (Tambe 2012) has emphasized and demonstrated the use of game theory in protecting large scale critical infrastructure such as airports, train stations, ports, forests etc. More specifically, in environments where there is a scarcity of resources, Stackelberg games have been used to represent the problem of resource allocation to protect potential targets in adversarial environments. Equilibrium strategies for the Stackelberg games are randomized and hence provide an ideal foil to deterring intelligent adversaries planning to attack. However, existing approaches cannot yet handle resource allocation in massive rail networks.

To that end, we make the following contributions in this paper: (a) Firstly, we provide a scalable two phase approach called RaPtoR (Randomized Patrolling for Rail networks) that computes coverage probabilities for various targets in the first phase and then computes an execution plan at runtime by connecting samples from the randomized coverage probabilities; (b) Secondly, we demonstrate the scalability of our approach by generating randomised schedules for patrol teams patrolling the Singapore rail network that has 108 stations, 30 patrol teams and up to 8 hour shifts for patrol teams each day. The Stackelberg game models are constructed from the real world data set containing the utilisation of rail stations by travellers throughout the day.

Related Work

(Chu and Chan 1998) and (Elizondo et al. 2010) have studied the problem of Crew rostering in public transportation systems. (Stern 1986) and (Sharma 2007) have studied and proposed algorithms for scheduling security guards/police officers/patrol cars in various distinctive settings. However, these crew rostering and scheduling algorithms have not modelled adversaries while computing the schedules.

The line of research that has the most similarities to our work is the application of game theory to patrol scheduling. (Tsai et al. 2009) and (Ordonez et al. 2012) have modeled the strategic security allocation problem as a Strackelberg game and developed the Intelligent Randomization In Scheduling (IRIS) system. This is a tool for scheduling Federal Air Marshals (FAMs) that provide law enforcement aboard U.S. commercial flights. However, unlike the FAMS problem in which a patrol consists of very limited number of flights (usually 2 or 3), we allow for more complex patrols that have to account for distance between targets and also longer planning horizons for each of the patrol teams. Our problem is closely related to the one addressed by (Yin et al. 2012) and we will provide a detailed comparison in a later section.

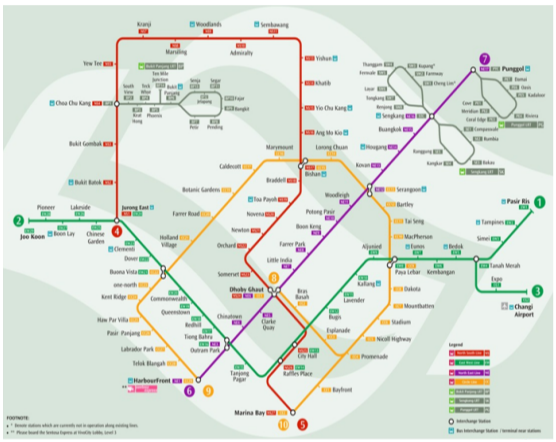

Figure 1: Singapore MRT Map

Application: Security of Mass Rapid Transit (MRT) Stations

We address the problem of generating randomized schedules for police patrols to secure Mass Rapid Transit (MRT) systems for trains. We are specifically interested in the Singapore MRT system shown in Figure 1. There are a total of 108 stations or targets for adversaries (represented by $\mathcal{T}$ henceforth) spread along seven different lines and are patrolled by a set of patrol teams $\mathcal{P}$. The number of people visiting the stations at various times during the day changes and consequently, the risk of not patrolling a train station also changes. Each patrolling team has a shift of $\mathcal{H}$ hours visiting one station per hour. An example patrol schedule of shift length 5 is provided below:

9:00AM	10:00AM	11:00AM	12:00PM	1:00PM
Changi Airport	Expo	Lavender	Bugis	City Hall

While the problem setting described here is similar to the one in (Yin et al. 2012), there are major differences:
Scale: For optimality of coverage, we have to account for transfers between lines and hence cannot assume independent lines. This increases the scale of the problem considerably. In addition, the number of train stations are higher (108 compared to a maximum of 22), the number of patrol teams are higher and the aspect of computing $\mathcal{H}$ ($>= 8$) length patrols increases the complexity. In our problem, there is also a need for schedules to reduce the distance travelled by patrol teams.
Breaks and other constraints: Another key distinction is the presence of breaks for the patrol teams. In addition, there are other constraints such as no one team can visit the same station twice during a shift and no two teams can take the break in the same station at the same time. These make the randomisation aspects difficult.
Objective: In our problem, the objective for the patrol teams is to secure the stations against criminal attacks. However, the main objective for the defenders in Yin *et al.*'s work is to catch fare offenders. Due to this, the underlying optimization is vastly different.

While our experimental results are based on the Singapore rail transport network, our research is applicable and can scale to rail transport networks in many cities, including but not limited to London, Beijing and Paris.

Deterministic Model

Previously, (Lau and Gunawan 2012) have addressed the problem of patrolling MRT stations in Singapore, while considering a known adversary model, which launches an attack according to a probability distribution. This probability distribution is typically constructed based on geographical density, criminology analysis, or intelligence sources. Here we extended the model in (Lau and Gunawan 2012) to improve its applicability and scalability. The key points of this approach are described as follows:
(a) Akin to previous work, we employ an integer linear program (ILP) to solve the overall optimisation problem. By employing novel valid inequalities, we calculate tighter linear program (LP) relaxation bounds, thereby improving the computation performance.
(b) The objective is twofold: maximize the coverage of most populous stations at the most populous hours, and minimize the total distance traveled by patrol teams.
(c) Starting and break times of patrol teams are determined automatically and optimally by solving the ILP model.

This approach was able to compute patrolling schedules for each of the individual lines (maximum of 31 stations) in less than 30 seconds. When transfers between lines were allowed, we solved a problem with 3 lines (67 stations) in slightly above 20 minutes. However, we were unable to solve problems of larger scale due to memory problems.

There are two drawbacks with this approach. Firstly, the optimal schedule generated is deterministic. Secondly, the scalability of the model is limited: only instances of up to 67 stations or with 15 patrol teams can be solved to date. These drawbacks motivated us to employ the Stackelberg representation and our new approach, RaPtoR, which is the main contribution of this paper.

Stackelberg Representation

Games are a popular model to represent strategic interactions among multiple players. The goal for each player is to maximize his/her own utility function. Hence, the solution concept for a game is an equilibrium solution, i.e. a joint strategy, where no agent has an incentive to deviate.

In the recent past, Game theory and more specifically Stackelberg Games have been used to represent the computation of patrol schedules for security of critical infrastructure (Tambe 2012) including airports, planes (federal air marshals), ports, forests. In these problem domains, the defenders (security agencies) set their patrol strategy first and hence are the leaders, whereas the attackers (terrorists, drug pedlars, and other criminals) are the followers who observe the patrol strategy over time and attack. It has been shown that the randomised strategies corresponding

to Strong Stackelberg Equilibrium provide better coverage than existing methods of computing deterministic and randomised strategies (Tsai et al. 2009; Jain et al. 2010)

Akin to previous work, we also employ a Stackelberg representation for the train security problem mentioned in the previous section. In fact, we employ the compact security game model introduced by (Kiekintveld et al. 2009). $\mathcal{T} = \{t_1, t_2, \cdots\}$ is the set of targets (MRT stations) that may be attacked and the defender has a set of patrol teams, $\mathcal{P} = \{p_1, p_2, \cdots\}$ to cover the targets. There is no distinction in the individual patrol teams and can be assigned to any specific target. Key additions to this compact representation to capture the MRT problem are the time horizon, $\mathcal{H}$ and a distance value between targets, d_{t_1,t_2}.

The following four utilities are defined for each of the targets at each decision epoch e ($\leq \mathcal{H}$) : (a) Utility to the defender for covering a target: $U^c_{\theta,e}(t)$; (b) Utility to the defender for an uncovered attack on a target: $U^u_{\theta,e}(t)$; (c) Utility to the attacker for attacking a covered target: $U^c_{\psi,e}(t)$; (d) Utility to the attacker for attacking an uncovered target: $U^u_{\psi,e}(t)$. In addition to covering targets, patrol teams have to spend as little time as possible in travelling to targets.

	Covered	Uncovered
Defender	5	-10
Attacker	-5	10

Assuming rational behaviour and Stackelberg representation of the interactions, the attacker and defender would execute strategies that are in Strong Stackelberg Equilibrium (SSE). In a SSE, the defender leads using a coverage vector, C=$(C^1, C^2, \cdots, C^{\mathcal{H}})$, which represents a randomized allocation of patrol teams to targets over all the decision epochs. Coverage vector at an epoch is defined as $C^e = (c^e_1, \cdots, c^e_t, \cdots)$, where c^e_t denotes the probability of coverage for target t at epoch e. With respect to the attacker, we can consider either of the following problems: (a) An attack can happen every epoch; **OR** (b) An attack can happen only once across all epochs. We focus on (a), because it is the more generic of the two. However, as we show in our later sections, minor modifications to the solution approach for (a) solves (b).

Let $\mathcal{A}$ denote the set of targets that if attacked would provide the highest utility for the attacker at each decision epoch given the coverage vector. In SSE, the opponent is assumed to pick one target from $\mathcal{A}^e$ at each decision epoch e that provides the highest utility for the defender. Given a coverage vector C^e for a decision epoch, the utility for the defender when attacker attacks a target t is :

$$U_{\theta,e}(t, C^e) = c^e_t U^c_{\theta,e}(t) + (1 - c^e_t) U^u_{\theta,e}(t) \qquad (1)$$

While the coverage vector represents the solution for the Stackelberg representation, the final outcome required of solving the MRT problem is a daily plan for each patrolling group over the span of their horizon, $\mathcal{H}$. The crucial objective in deriving this daily schedule is to keep the overall distance travelled by the patrolling groups to a minimum, so that most of the time is spent in securing the train stations and not in travelling.

Approach: RaPtoR

We now describe the RaPtoR (Randomized Patrols for Rail networks) algorithm. RaPtoR is divided into two phases:
Phase 1: compute the Strong Stackelberg Equilibrium (SSE) coverage for all the targets over all decision epochs by solving the Stackelberg representation of the problem.
Phase 2: compute execution plans for individual patrolling groups over the entire horizon based on the coverage samples generated from the coverage distributions in phase 1.

Connecting the equilibrium coverage strategies of each decision epoch at runtime for generating execution plans is one of the main contributions of this paper and as we will show in our experimental section, this allows for a significant increase in scalability.

Computing SSE coverage for targets

Algorithm 1 GetCoverage1 ($\mathcal{T}$, $\mathcal{P}$, $\mathcal{H}$, U_θ, U_ψ)

```
e ← H
for all e ≥ 1 do
    c^e ← SolveLP1 (T, P, e, U_{θ,e}, U_{ψ,e})
    e ← e − 1
end for
return c
```

Initially, we focus on the problem where an attacker can potentially attack at every decision epoch. Due to independence of attacks, we can compute the coverage for each decision epoch just based on the utility vectors for that epoch and independent of utility or coverage vectors at other epochs. Algorithm 1 calls the SolveLP1() function for each decision epoch to compute the corresponding coverage vectors.

SolveLP1($\mathcal{T}$, $\mathcal{P}$, e, $U_{\theta,e}$, $U_{\psi,e}$):

$$\max \quad d$$

$$a^e_t \in \{0,1\} \qquad \forall t \in \mathcal{T} \qquad (2)$$

$$\sum_{t \in T} a^e_t = 1 \qquad (3)$$

$$c^e_t \in [0,1] \qquad \forall t \in \mathcal{T} \qquad (4)$$

$$\sum_{t \in T} c^e_t \leq |\mathcal{P}| \qquad (5)$$

$$d - U_{\theta,e}(t, C^e) \leq (1 - a^e_t) \cdot Z \qquad \forall t \in \mathcal{T} \qquad (6)$$

$$0 \leq k - U_{\psi,e}(t, C^e) \leq (1 - a^e_t) \cdot Z \qquad \forall t \in \mathcal{T} \qquad (7)$$

The function SolveLP1 computes the Strong Stackelberg Equilibrium for a given decision epoch e, given the attacker ($U_{\psi,e}$) and defender ($U_{\theta,e}$) utility functions for that epoch. The objective of this LP is to compute a coverage vector that will maximise the utility obtained by the defender, where in the attacker will choose a target that will maximize his/her utility given the coverage vector. This optimization problem was previously introduced by (Kiekintveld et al. 2009). a^e_t is a binary variable, that if set to 1 indicates that the attacker will attack target t at this decision epoch e. c^e_t is a continuous variable, which indicates the desired coverage required a target t at decision epoch e. Z is a large positive constant. d and k are intermediary variables.

(2) and (3) enforce the condition that only one target can be attacked at an epoch. (4) constrains coverage for a target at an epoch to less than 1 and (5) ensures that the number of targets covered is bounded by the number of patrol teams available. Utility for the defender is determined by the attack vector selected by the attacker according to (6) and (7) ensures that the attacker plays a best response deterministic strategy to the coverage vector employed by the defender. Finally, the objective is to maximize the utility obtained by the defender.

Now, we explain the minor modifications required to account for the fact that an attack can happen only once over the entire horizon: (a) Instead of one optimisation problem to be solved for each decision epoch, we now have one optimisation problem that yields the coverage vectors for all decision epochs. (b) An attack can occur only once over the entire time horizon and on only one target and therefore the constraint (3) is modified as follows: $\sum_{t\in\mathcal{T}}\sum_{e\leq\mathcal{H}} a_t^e = 1$. (c) Constraints (4), (6), (7) have to not only iterate over targets ($\forall t \in \mathcal{T}$), but also have to iterate over decision epoch ($\forall e \leq \mathcal{H}$).

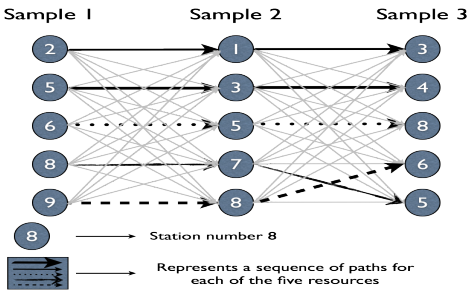

Figure 2: Example problem

Computing Execution Plans

The second phase of the algorithm computes an execution plan for each patrol team given the coverage samples obtained from coverage vectors at each decision epoch. In rail networks, typically, all stations are reachable from all other stations given enough time. Due to this reachability, unlike previous work (Jain et al. 2010), we can always find paths connecting coverage samples derived from coverage vectors.

A coverage sample, S^e represents a set of targets to be covered (by all patrol teams) at a decision epoch and is obtained by sampling the coverage vector for that decision epoch, C^e. Formally, an execution plan for a patrol team is represented as a sequence of targets, $(t^1, t^2, ...t^{\mathcal{H}})$, where t^e is a target within the coverage sample S^e .To maximise the time spent in patrolling stations, a desired property of this algorithm is to minimize the maximum time taken to travel between targets by any patrol team.

Figure 2 provides an illustration of the problem once the coverage samples are generated from the coverage vectors at each decision epoch. In this example, we have three coverage samples and the goal is to compute execution plans for five patrol teams. The number of sequences are exponential in the number of epochs and in the general case, an exact algorithm which minimizes the overall distance covered would have exponential complexity. Dark black lines indicate one set of execution plans.

Algorithm 2 GetExecutionPlan $(\mathcal{H}, C)$

$D^{\mathcal{H}} \leftarrow \mathbf{0}$
$\Pi^{\mathcal{H}} \leftarrow \emptyset$
$e \leftarrow \mathcal{H}$
for all $e \geq 2$ **do**
  $S^e \leftarrow$ GETSAMPLE(C^e)
  $S^{e-1} \leftarrow$ GETSAMPLE(C^{e-1})
  $(D^{e-1}, \Pi^{e-1}, \mathbf{x}^e) \leftarrow$ STITCHSAMPLES $(S^{e-1}, S^e, D^e, \Pi^e)$
end for
return x

To account for this complexity, we provide a dynamic programming algorithm that stitches plans – computed for adjacent decision epochs – together over the entire time horizon. In the example of Figure 2, we first compute the execution plans corresponding to Sample 2 and Sample 3 given the distance function between all targets (represented as $dist_{a,b}$). Then, we compute the solution for Sample 1 and Sample 2, using the execution plans computed for the Sample 2 and Sample 3 combination. This is a highly scalable algorithm whose complexity increases linearly with time horizon, $\mathcal{H}$.

Algorithm 2 provides the execution plan over the entire horizon for all patrol teams and to that end, employs the STITCHSAMPLES() function to compute the execution plans between coverage samples at two adjacent decision epochs. To capture dependencies among targets to be visited across non-adjacent epochs, the solution for the future epochs (distance - D^e, plan - Π^e) is passed as an argument to the STITCHSAMPLES() function. "A station cannot be visited by the same patrol team within the same shift" is an example of a dependency/constraint that spans non-adjacent epochs.

We now describe the mixed integer optimization employed by the STITCHSAMPLES() function. Let $S^e(k)$ denote the id of the kth target in the eth sample. $x^e_{i,j} = 1$ indicates that the resource used for patrolling $S^e(i)$ is also used for patrolling $S^{e+1}(j)$. $\Pi^e(i)$ denotes the set of targets covered by one patrol team which has to patrol the target $S^e(i)$ from epoch $e+1$ to $\mathcal{H}$. $D^e(j)$ denotes the distance travelled from $S^e(j)$ at decision epoch e to horizon $\mathcal{H}$.

STITCHSAMPLES (S^e, S^{e+1}, D^{e+1}, Π^{e+1}):

$$\min \quad d$$

$$\sum_i x^e_{i,j} = 1, \forall j \tag{8}$$

$$\sum_j x^e_{i,j} = 1, \forall i \tag{9}$$

$$d \geq x^e_{i,j} \cdot dist_{s^e(i),s^{e+1}(j)} + D^{e+1}(j), \forall i, j \tag{10}$$

$$x^e_{i,j} \in \{0, 1\}, \forall i, j \tag{11}$$

$$x^e_{i,j} = 0, \textbf{if } S^e(i) = S^{e+1}(j) || s^e(i) \in \Pi^{e+1}(j), \forall i, j, e \tag{12}$$

The objective of the optimization model is to minimize the maximum distance travelled by any team. (8) and (9) ensure that one target at e is connected to only one target at $e+1$ and vice versa. (10) computes the maximum of the distances travelled by the patrol teams. (12) ensures that no target is visited multiple times by the same patrol team.

Accounting for Breaks Another practical consideration in patrol scheduling is to include breaks for every patrol team. In the case of Singapore MRT patrols, the constraint is to schedule two breaks for every patrol team, while considering the adversary and keeping the schedule randomized. Accounting for breaks requires two minor changes to the phase 2 of RaPtoR:
(1) Add extra nodes to each coverage sample. These nodes correspond to the nodes where a patrol team takes a break and the total number of extra nodes is equal to the maximum number of teams that can take a break at any epoch (typically an input parameter decided by the security agency).
(2) Modify the constraints in the STITCHSAMPLES() optimisation problem to represent nodes not having incoming or outgoing links (unlike the example in Figure 2). Furthermore, we also need to include constraints corresponding to the number of breaks and their occurrence, such as there have to be two breaks and they cannot happen in adjacent decision epochs.

Theoretical Results As shown in (Kiekintveld et al. 2009), the first phase of our approach already provides SSE coverage for all the targets. Here, we provide a discussion on the guarantees provided by the second phase of our approach that computes execution plans.

Proposition 1 *If there are no constraints that span multiple non-adjacent epochs, then phase 2 of RaPtoR computes optimal execution plans with respect to the distance objective.*

Proof Sketch We prove this by contradiction. Let us assume there exists an optimal set of execution plans that are different and better. We can then construct execution plans which will perform better by using the execution plans generated by our approach. ■

Evaluation

We show the performance of our algorithms in the context of the Singapore rail network for which we have the real data set that provides the usage of the rail stations.

Data Set

Figure 1 provides the map of the rail stations in the network. This map contains 108 stations spread over 7 lines covering most areas in Singapore. We also refer to the 78 problem in our results, which correspond to the stations on the 4 primary lines. There are transfers allowed between various lines at 11 stations. In 2012, there are almost 2 million rides in a single weekday[1], and hence the rail network operations and safety are of prime importance.

[1]http://www.smrt.com.sg/AboutSMRT/InvestorRelations/KeyOperatingMatrix/Trains.aspx

Our data set provides the usage statistics for all the 108 stations during 19 hours[2] on each day. This data was collected by a card system that records the arrival and departure of passengers as they enter and exit various stations. We consider the volume of passenger flow at each station as the potential loss for not patrolling that station at that specific time and use that in the construction of the game theoretic model. We assume a zero sum model, i.e. whatever is lost by the defender is gained by attackers and vice versa.

In addition, the following constraints must be satisfied regarding the schedule of patrol teams:
(a) No patrol team should visit the same station twice during their shift of 8 hours.
(b) Each team should have at most two breaks and the breaks should not be adjacent to each other.
(c) The total amount of time spent travelling to the next target should be less than 15 minutes.

Results

Firstly, we demonstrate the scalability of the two phases of the RaPtoR algorithm. Figures 3(a)-(d) provide the run-time performance of the algorithm. We primarily focus on the case where an attack can happen at every decision epoch.

In Figures 3(a)-(b), we present the run-time results[3] on the Singapore MRT network with 78 stations. We also experimented with 108 station problem and very similar results were obtained [4]. On X-axis we have the number of resources, while on the Y-axis, we have the total run-time. The runtime is for phase 1 in Figure 3 (a) and for phase 2 in 3(b). Each line in the graphs represents runtime for a different horizon (ranging between 8 to 12) of the execution plans. Here are the main observations and conclusions from the results in Figures 3(a) and (b):

- The key result is that even with a horizon of 12 and number of resources equal to 35, we obtain solutions in less than 4 seconds.
- Even though the difference in run-times is minor as we increase the number of resources and horizons, we see that the difference is always there and this difference is statistically significant.

In the second set of runtime results provided in Figures 3(c)-(d), we deal with additional random problems with many more stations to demonstrate the overall scalability of the algorithm. We plot the time taken by phase 1 and phase 2, as the number of resources are varied from 15-35 with the number of stations taken from the set $\{78, 108, 120, 150, 180\}$. As shown in the figures, the overall time taken in both phases is less than 10 seconds.

Figure 3 (e) provides the planned total time spent in traveling by each of the patrol teams. While we do not enforce the constraint of 15 minutes in travel time between targets at adjacent epochs, we can obtain travel times less than 15

[2]MRT stations closed from 12:00 - 5:00 AM.

[3]RaPtoR was run on a 1.8 GHz Intel Core i5 with 8 GB RAM to obtain the runtime results.

[4]We show results with 78 stations, to show a clear coverage graph for Figure 4 and be consistent across all results

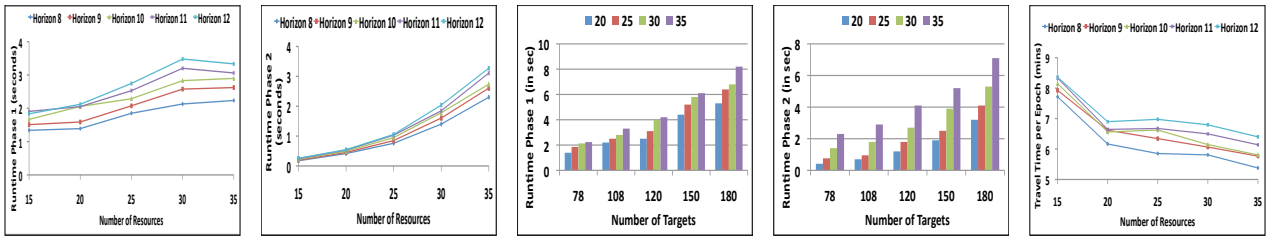

Figure 3: (a)-(d): Runtime performance of RaPtoR phase 1 and phase 2; (e) Travel time performance of RaPtoR phase 2

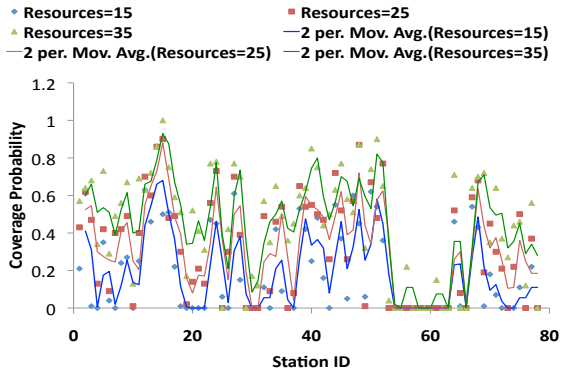

Figure 4: Sampled coverage vs Number of Resources

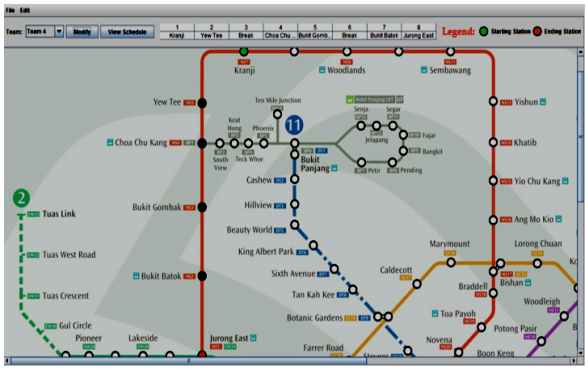

Figure 5: Screenshot of the Patrol Scheduler

minutes with our phase 2 approach. On X-axis, we have the number of resources[5] and on the Y-axis, we have the planned travel time in minutes. We provide the results for horizons 8-12 and on problem with 78 stations. The phase 2 approach is not optimal when there are constraints spanning multiple non-adjacent epochs, which in fact, are present in the Singapore data set. Even though the approach is not optimal, it retains a nice property of the optimal algorithm, i.e., as more resources are provided, our phase 2 algorithm is able to find better execution plans with respect to distance travelled.

In Figure 4, we show the sampled coverage obtained from the coverage vectors generated by the phase 1 of RaPtoR. On the X-axis, we have the individual targets and on the Y-axis, we have the coverage probability over 100 samples at a specific decision epoch. This result was observed at all decision epochs. We compare the coverage obtained when we increase the resources from 15 to 25 to 35 and we plot the moving average of the coverage. As we see from the graphs, the coverage obtained with more resources is higher than the coverage with fewer number of resources even with only 100 samples on a problem with 78 targets.

Along with this work, we developed a decision support system for possible deployment in the future. Figure 5 displays a screenshot of our patrol scheduler.

We have conducted some initial experiments for the case where attacks can happen only once over the entire horizon. The phase 1 of the modified approach solves the problems of 78 and 108 stations with 35 resources in less than 5 seconds. It should be noted that phase 2 results are independent of the assumptions on attacks.

[5]We use resources and patrolling teams synonymously

References

Chu, S. C., and Chan, E. C. 1998. Crew scheduling of light rail transit in hong kong: from modeling to implementation. *Computers and Operations Research* 25(11):887 – 894.

Elizondo, R.; Parada, V.; Pradenas, L.; and Artigues, C. 2010. An evolutionary and constructive approach to a crew scheduling problem in underground passenger transport. *Journal of Heuristics* 16(4):575–591.

Jain, M.; Kardes, E.; Kiekintveld, C.; Tambe, M.; and Ordonez, F. 2010. Security games with arbitrary schedules: A branch and price approach. In *AAAI*.

Kiekintveld, C.; Jain, M.; Tsai, J.; Pita, J.; Ordez, F.; and Tambe, M. 2009. Computing optimal randomized resource allocations for massive security games. In *AAMAS*.

Lau, H. C., and Gunawan, A. 2012. The patrol scheduling problem. In *Practice and Theory of Automated Timetabling (PATAT)*.

Ordonez, F.; Tambe, M.; Jara, J. F.; Jain, M.; Kiekintveld, C.; and Tsai, J. 2012. Deployed security games for patrol planning. In *Handbook on Operations Research for Homeland Security*.

Sharma, D.K. Ghosh, D. G. A. 2007. Lexicographic goal programming model for police patrol cars deployment in

metropolitan cities. *International Journal of Information and Management Sciences*.

Stern, Z.S., . T. Y. 1986. Multi-objective scheduling plans for security guards. *Journal of the Operational Research Society*.

Tambe, M. 2012. Security and game theory. Cambridge University Press.

Tsai, J.; Rathi, S.; Kiekintveld, C.; Ordonez, F.; and Tambe, M. 2009. Iris - a tool for strategic security allocation in transportation networks. In *AAMAS*.

Yin, Z.; Jiang, A.; Johnson, M.; Tambe, M.; Kiekintveld, C.; Leyton-Brown, K.; Sandholm, T.; and Sullivan, J. 2012. Trusts: Scheduling randomized patrols for fare inspection in transit systems. In *IAAI*.