GRAND-VISION: An Intelligent System for Optimized Deployment Scheduling of Law Enforcement Agents

Jonathan Chase, Tran Phong, Kang Long, Tony Le, Hoong Chuin Lau*
Singapore Management University
{jdchase, bptran, tonyle, hclau}@smu.edu.sg, long.kang.2018@mitb.smu.edu.sg

Abstract

Law enforcement agencies in dense urban environments, faced with a wide range of incidents to handle and limited manpower, are turning to data-driven AI to inform their policing strategy. In this paper we present a patrol scheduling system called GRAND-VISION: Ground Response Allocation and Deployment - Visualization, Simulation, and Optimization. The system employs deep learning to generate incident sets that are used to train a patrol schedule that can accommodate varying manpower, break times, manual pre-allocations, and a variety of spatio-temporal demand features. The complexity of the scenario results in a system with real world applicability, which we demonstrate through simulation on historical data obtained from a large urban law enforcement agency.

Introduction

Public security organizations around the world are focusing on law enforcement based on the concept of ”Reactive to Proactive”. Some measurement concepts, such as “Visible Security”, which focuses on prediction (i.e. shortening arrival time to crime scene) and prevention (i.e. smart patrol) have been explored and conceptualized. With the aging of society and increasingly limited human resources, the solution to this challenge revolves around the use of technology to establish an efficient crime prediction, prevention and response schedule.

In this paper, we present a data-driven AI planning system, currently on trial with a local law enforcement agency (LEA) called GRAND-VISION that performs daily deployment scheduling of law enforcement agents. GRAND-VISION is an abbreviated name for Ground Response Allocation and Deployment - Visualization, Simulation, and Optimization. It harnesses historical incident data and other factors such as demographics and public holidays to predict the occurrence of incidents over time and space with high accuracy on a daily basis. Based on such prediction, as well as the daily supply of agent resources and other inputs, the system generates hourly deployment schedules that provide the best response to incidents, taking breaks and other constraints into consideration.

Executing a deployment schedule on a daily basis allows law enforcement to be tailored to the demand characteristics of a given day, and makes it harder for criminals to anticipate where law enforcement agents will be. When special events occur, a daily deployment can accommodate manual pre-allocation of resources to areas determined by a commander, likewise, if break times need to be changed, or, manpower is short on a given day, it can be accommodated intelligently. This last point is particularly pertinent during the COVID-19 pandemic, when law enforcement manpower may be stretched thin due to the need to self-isolate after exposure, or to assist in contact tracing and enforcement of quarantine measures.

This paper builds upon the work in (Chase et al. 2019), but improves it in a number of significant ways. The contributions of this paper are as follows. Firstly, the agent allocation optimization model is revised to make the allocation of an agent to a location more intuitive while enforcing the ‘greedy dispatch’ principle and reducing the size of the model, removing the need for the Iterated Local Search heuristic. Secondly, we introduce a shift scheduling model that incorporates the very important aspect of break times while preserving the manpower needs determined by the agent allocation. Break times introduce significant complexity since we need to ensure that break times are always covered. Thirdly, we propose an improved incident prediction model that employs deep learning to break down patrol region boundaries, allowing for more accurately predicted incident counts, and takes a holistic approach to incident generation rather than requiring multiple models for predicting additional parameters. Lastly, we generate experimental results via dispatch simulation using up-to-date historical data from a real LEA to demonstrate the superiority of our proposed approach over current practice and its ability to adapt to reduced manpower supply.

Related Work

There is a large body of work on incident prediction for the purpose of effective law enforcement. Traditional methods include simple aggregation of historical demand (e.g. (Malleson and Andresen 2015)), spatial (but not temporal) hot spot identification (such as (Levine 2017)), and risk terrain modeling (e.g. (Caplan, Kennedy, and Miller 2011)). More recently, (Mukhopadhyay et al. 2016) learns demand

*Corresponding Author

on a continuous time spatial grid, but only burglary incidents are modeled, with generated incidents used to design police deployment. Neural networks (deep learning) have been applied to forecast crimes. In (Wang et al. 2017) for example, the authors adapted ST-ResNet (Zhang, Zheng, and Qi 2017) to collectively predict crime distribution over the Los Angeles area. They pre-process the raw crime data via regularization in both space and time to enhance predictable signals and then apply hierarchical structures of residual convolutional units to train multi-factor crime prediction models.

Optimization has been applied to law enforcement resource deployment, such as deploying security officers to patrol rail stations (Lau, Yuan, and Gunawan 2016) and police cars to respond to crime incidents (Mukhopadhyay et al. 2016). In (Saisubramanian, Varakantham, and Lau 2015), a risk-based approach was used for deploying ambulances that improved response times. In (Wang et al. 2020), the authors applied integer programming to solve a police patrol problem with the objective of maximizing the police visibility rate to improve public safety and the additional constraint of response time guarantee. Many of these problems may also be modeled as a multi-agent task allocation problem. For example, (Amador, Okamoto, and Zivan 2014) allocates security agents to incidents as they occur.

Overall Approach - Data-Driven Optimized Deployment Scheduling

In the problem setting we consider, illustrated in Fig. 1, emergency incidents occur and are reported to a central dispatcher. The dispatcher notes the start time of each incident and assigns the nearest available agent to attend. Incidents may be classified as either urgent or non-urgent, with different response time targets associated with each. Some incidents require more than one agent to attend, and they must remain in attendance until the service time of the incident is complete, whereupon they return to the region they have been assigned to patrol. To ensure that as many incidents are attended within the target time as possible, we propose a system that designs a deployment schedule that assigns agents to patrol locations throughout their shift.

In this section we outline our overall approach that generates deployment schedules by combining incident generation, allocation, scheduling, and simulation. First, the Incident Generator produces sets of synthetic incidents, with each set corresponding to one scenario of the shift being planned for. Second, the Optimizer uses these incidents to identify the sectors that the available agents must be deployed to, with the output being the number of agents required in each sector on a 2-hourly basis. Third, the number of agents required in each sector is mapped to the set of actual agents by the Scheduler on an hourly basis, which aims to minimize the amount of movement agents are required to do while accounting for breaks. Finally a set of generated test incidents is used to evaluate the scheduled deployment using a dispatch Simulator. The description of each of the four components, as illustrated in Fig. 2, is detailed in the following sections.

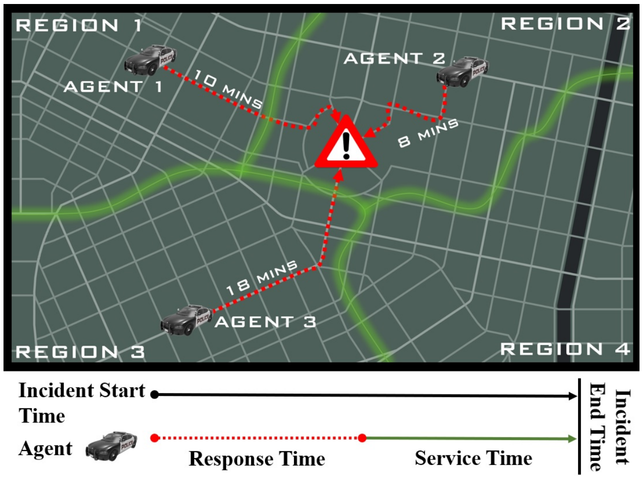


Figure 1: Agents patrol their assigned patrol regions. When an incident occurs, the nearest agent is assigned to attend, with the aim of meeting the response time QoS requirements.

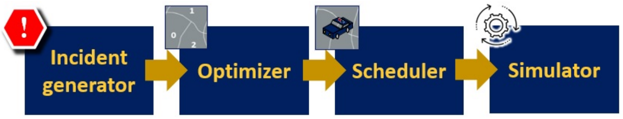


Figure 2: The GRAND-VISION deployment generation process.

Deep Learning-driven Incident Generation

Our partner LEA has provided us with a large set of incident data, containing details of incident timing, urgency, location (in lat-long coordinates), duration, and type, as well as the nature of the response, including the responding agents, and their response times. We combine this with other data such as demographics and land use features. However, one particular challenge we face compared to other works, is the sparsity of the dataset, particularly for some serious, but rare, incident types. Thus, for precise daily prediction, accounting for random 'noise' in the incident occurrence is a high priority.

To generate an effective deployment plan, we develop an incident generation method that can generate sets of realistic emergency incidents. A generative method was introduced in (Chase et al. 2019) but we address a shortcoming with the previous method through the application of Deep Learning. The prior work used statistical methods, namely, a Gaussian Process (GP) model, to predict incident counts by patrol region and time period. The predicted counts were rounded to the nearest integer and additional parameter distributions were learned for values such as priority and agent demand. However, the GP requires that each region be considered as an independent distribution, but in reality, they are part of a larger area and the region boundaries introduce significant noise. Suppose we have a uniform probability distribution over the area of a square. The square produces one incident every hour. Now suppose we cut the square up into 4 parts so

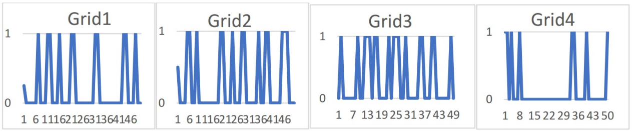
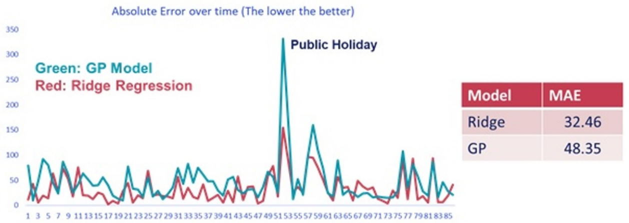


Figure 3: Example of the consequences of predicting by regions atomically.



Model	MAE
Ridge	32.46
GP	48.35

Figure 4: Comparing the error over 12 weeks of data, the noise introduced by region boundaries results in inferior performance predicting overall counts for the GP model.

that the incident can happen anywhere in the square. If we simulate this over a few timestamps, we will get something like the individual grid graphs in Fig. 3.

Each grid represents a quarter on the square and can look like a time series from a purely random process. This is analogous to the counts in each region and fitting a machine learning model on this would be only learning the noise rather than the true distribution. Even if it were possible to accurately model the noise, the average counts in each grid would be around 0.25, which would be rounded down to 0, and instead of 1 incident being generated between the 4 grids, 0 would be generated.

Incident Count Prediction

Rather than attempting to predict the counts at the micro level and assigning parameters accordingly, we break down the region boundaries and apply Ridge Regression to learn the incident counts at the macro level. Comparing the predicted total count from the regression to the aggregated counts produced by the GP model, we find an improvement in the comparative Mean Absolute Error (MAE) metric, as shown in Fig. 4.

A Generative Deep Learning Model

The previous generation method in (Chase et al. 2019) applied separate models to predict the additional parameters of service time, agent demand, start time (in minutes), and priority, given the patrol region and start hour of an incident predicted by the GP. However, by using a Generative Adversarial neural Network (GAN), we can output all variables at once. This eases maintenance as only a single model must be generated with new data, and can train on multiple years of data, something that is not possible with the computationally expensive GP method. Specifically, we employ a Wasserstein-Conditional GAN (WCGAN) that is able to produce distributions conditioned on types of day (of which we identify 5 types: weekdays, weekends, and 3 categories of public holiday). Unlike the GP model, the WCGAN predicts the latitude and longitude coordinates of incidents, which allows us to place incidents more precisely. For the purposes of incident generation, the Ridge Regression predicts the total number of incidents for the entire geographic area, and each incident to be generated is sampled from the WCGAN model. To produce an input for the deployment optimization, the lat-long of the generated incidents are assigned to the region whose centroid point is closest to them. Thus, we achieve region-level incident generation without the noise issues of region-level prediction. To take full advantage of the temporal data available, we consider a version of the WCGAN model that incorporates additional longer-term information, such as day of the year, week of the year, and the cyclic nature of these time periods.

Prediction Performance

The fine-grained spatial generation requires a novel metric to properly evaluate its performance. We employ a matching method, where an incident is considered correctly predicted if it occurs within a specified distance and time of an unmatched incident in the test set. We define recall as the proportion of actual incidents that are correctly matched, and precision as the proportion of predicted incidents that match an actual incident. Combining the two, we evaluate the model's performance using the F1 score metric. To compare with the GP model, we assign a lat-long value to each GP-generated incident by distributing incidents uniformly within their region. For each day of test data we generated multiple incident samples and evaluated the average F1 score across all samples. Using a matching distance of 2km and 12 minutes (guaranteeing a successful response to an urgent incident), we compare the GP, GAN, GAN with additional temporal factors (labelled 'GAN time'), and a random baseline generation algorithm. We plot the average performance for different sample sizes in Fig. 5 and find that both GAN models outperform the GP method, with the 'GAN time' model achieving slightly better performance due to its additional temporal knowledge. Fig. 6 compares the 'GAN time' model with the GP, showing the box plots for each sample size. In addition to a better average performance, the GAN scores have a much smaller range, indicating a greater consistency in performance over the GP.

Optimization of Agent Allocation with Variable Manpower

The Optimizer phase of the deployment algorithm performs an allocation of on-duty agents to locations to minimize the risk of failing an incident response, and we extract the number of agents required per location to serve as input to the Scheduler. Agents are allocated to patrol sectors and are assigned to respond to sets of training incidents according to the principle that the nearest available agent is dispatched ('greedy dispatch'). Given that the problem is a mixed integer problem (MIP), we aim to achieve the responsiveness necessary for an algorithm run daily, by splitting the input incidents into 2-hour blocks. The optimization problem is

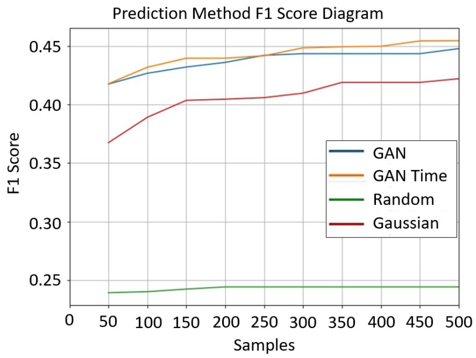


Figure 5: 4-way method comparison between GP, two GAN variants, and a baseline random algorithm.

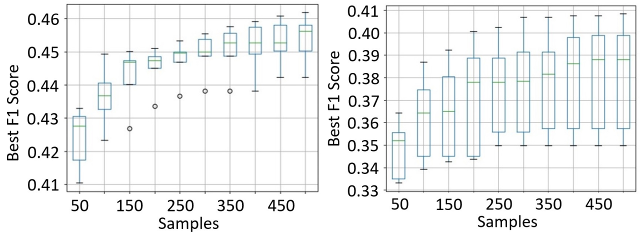


Figure 6: Box plots of GAN (left) vs GP (right) matching performance. GAN achieves better average performance with a narrower spread of values.

solved for each 2-hour block independently, giving the allocation required in a faster time. The response time minimization model is based on Sample Average Approximation (SAA), which produces a single solution that yields the best expected risk across all generated incident scenarios.

The notation used in the formulation is defined in Table 1. The objective function, given in equation (1) minimizes response time failures, and a constraint is introduced to fix the total number of agents to the supply required by the shift (2).

The full optimization model is given in (1)-(17). (2)-(4) control the mandatory limits on agent allocation to regions. (5) guarantees pre-allocation requests are honoured. (6) ensures that the demand of each incident is satisfied. (7)-(14) enforce greedy dispatch ensuring that the arrival time of an agent is the minimum of all available agents, using the first agent to become available if all are occupied at the incident start time. (15)-(16) establish the end time of an incident, in tandem with the established arrival times, and finally, (17) identifies if an incident is a success or failure.

$$\min \quad \frac{\sum_{r,s} z^r_s}{\sum_s |\mathcal{R}_s|} \tag{1}$$

s.t.

$$\sum_{i,l} y^l_i = Y_{max} \tag{2}$$

Indices:	
i	Agent index from set $\mathcal{I}$
q, r	Incident indices from set $\mathcal{R}_s$ and their location l^q,l^r
s	Scenario index from set $\mathcal{S}$
Decision Variables:	
z^r_s	Binary variable indicating if the response time target was met for request r, scenario s
y^l_i	Binary variable indicating if agent i is assigned to location l
$y^r_{i,s}$	Binary variable indicating if agent i serves incident r in scenario s
Derived Variables:	
δ^r_s	Time when incident r is first attended in scenario s
e^r_s	Ending time for incident r, scenario s
$\xi^r_{i,s}$	Latest e^r_s for all incidents attended by i before attending r
$\Delta^{q,r}_{i,s}$	Binary indicator if i completes q only after r starts
$\gamma^r_{i,s}$	Binary indicator if $\Delta^{q,r}_{i,s}$ holds for i for any q before r
Parameters:	
T_{l^i,l^r}	Travel time from location of agent i to location of incident r
d^r_s	Number of agents required by incident r, scenario s
c^r, T_c	Priority class of incident r and the response time QoS target for that priority
t^r_s	Start time of incident r, scenario s
g^r_s	Service time of incident r, scenario s
Y_{max}	The maximum number of agents to be allocated for each scenario
M	An arbitrary large value

Table 1: Key notations used in optimization model.

$$\sum_l y^l_i = 1 \quad \forall i \tag{3}$$

$$\sum_i y^l_i \le MaxPerSector \quad \forall l \tag{4}$$

$$\sum_i y^l_i \ge preallocation_l \quad \forall l \tag{5}$$

$$\sum_i y^r_{i,s} = d^r_s \quad \forall r, s \tag{6}$$

$$y^q_{i,s} + \frac{e^q_s - t^r_s}{M} - 1 \le \Delta^{q,r}_{i,s} \quad \forall q < r, r > 0, i, s \tag{7}$$

$$\frac{1}{r}\sum_{q=0}^{r-1} \Delta^{q,r}_{i,s} \le \gamma^r_{i,s} \quad \forall i, r > 0, s \tag{8}$$

$$\sum_l T_{l^i,l^r} \cdot y^l_i + t^r_s \ge \delta^r_s \quad r = 0, \forall i, s \tag{9}$$

$$\sum_l T_{l^i,l^r} \cdot y^l_i + t^r_s + M \cdot \gamma^r_{i,s} \ge \delta^r_s \quad \forall r > 0, i, s \tag{10}$$

$$\sum_l T_{l^i,l^r} \cdot y^l_i + \xi^r_{i,s} + M \cdot (1 - \Delta^{q,r}_{i,s}) \ge \delta^r_s \quad \forall r > 0, q < r, i, s \tag{11}$$

$$e^q_s - M \cdot (1 - y^q_{i,s}) \le \xi^r_{i,s} \quad \forall r > 0, q < r, i, s \tag{12}$$

$$\sum_{l} T_{l^i,l^r} \cdot y_i^l + t_s^r + M \cdot (y_{i,s}^r - 1) \leq \delta_s^r \quad \forall r, i, s \tag{13}$$

$$\sum_{l} T_{l^i,l^r} \cdot y_i^l + e_s^q + M \cdot (\Delta_{i,s}^{q,r} - 1) + M \cdot (y_{i,s}^r - 1) \leq \delta_s^r \quad \forall r > 0, q < r, i, s \tag{14}$$

$$e_s^r \leq \delta_s^r + \sum_{l} T_{l^i,l^r} \cdot y_i^l + g_s^r + M(1 - y_{i,s}^r) \quad \forall i, r, s \tag{15}$$

$$e_s^r \geq \delta_s^r + \sum_{l} T_{l^i,l^r} \cdot y_i^l + g_s^r + M(y_{i,s}^r - 1) \quad \forall i, r, s \tag{16}$$

$$z_s^r \geq \frac{(\delta_s^r - t_s^r) - T_c}{M} \quad \forall r, c^r = c, s \tag{17}$$

Handling Break Times at the Optimizer Phase

The mapping of break times to agents is performed at the scheduling stage. To facilitate this, at the Optimizer stage we perform the allocation using a modified agent supply calculated by determining the minimum number of agents that are on duty at any time during each 2-hour period (i.e. the maximum number of agents on break at any given time). Thus, if the value of Y_{max} is 15, but during a given 2-hour period, there are up to 3 agents whose break times overlap with each other at some point, we solve the model for that 2-hour period using $Y_{max} = 12$. This ensures that the demand forwarded to the Scheduler never exceeds the number of active agents available for allocation.

Smoothing the Solution by Re-Allocating Redundant Agents

If the incidents in a 2-hour period are sparse, not all agents may be allocated to respond to an incident and the allocation result for these agents may not be meaningful. Therefore, we only use the agent allocations from the optimization where the agent is actually deployed to an incident. For the remainder, we compare the solution from each 2-hour period to the one after in turn, and allocate agents to minimize the difference (using the Cosine distance) between each pair of solutions. This helps the Scheduler find a better quality solution. Where there are no differences, but spare agents remain due to disparity in supply caused by breaks, we assign the spare agents to unoccupied sectors to increase robustness to unanticipated random demand.

Agent Scheduling with Fine-Grained Break Time Support

The optimization phase result defines the number of agents required in each region at each 2-hour period. This is followed by the scheduling stage which maps this allocation to the list of agents on duty, taking into account their respective break times. The aim of the Scheduler is to ensure that all required sectors have agent coverage so that the demand is met. The Scheduler employs an MIP to minimize the total travel time between sectors at the end of each time period. Unlike the Optimizer, the Scheduler assigns agents on a 1-hour basis, with break times supported down to a granularity of 15 minute intervals. This allows break times to be adequately covered without agents being clustered together unnecessarily.

Variable	
i	Agent index
j	Location index
t	Time index, member of set $\mathcal{T}$, default length 1 hour
u	Time index, member of set $\mathcal{U}$, where $\lvert\mathcal{U}\rvert = 4 \cdot \lvert\mathcal{T}\rvert$, default length 15 minutes
$y_{i,j}^t$	Binary variable: indicates assignment of agent i to location j in time t
x_{i,j_1,j_2}^{t-1}	Binary variable: indicates i moves from location j_1 to j_2 from time $t-1$ to time t
T_{j_1,j_2}	Parameter: Time to travel from location j_1 to j_2
d_j^t	Parameter: number of agents required in location j in time t
b_u^i	Binary parameter: if agent i is on break during time period u, true, else false
L^{max}	Parameter: sets the default maximum number of agents per sector to improve balancing. Is overruled by d_j^t if it is larger

Table 2: Key notations used in scheduling model.

The reader may be concerned whether Constraint Programming (CP) could be used instead of an MIP. In our past attempts at solving the constrained scheduling problem with a CP model (using the CP Optimizer), the run time took much longer than expected compared with the following MIP model, and hence we chose the latter instead.

We summarize the notations used in our MIP model in Table 2. The MIP formulation for the scheduling problem is given in (18)-(23).

$$\min \quad \sum_{i} \sum_{j_1} \sum_{j_2 \neq j_1} \sum_{t>0} T_{j_1,j_2} x_{i,j_1,j_2}^{t-1} \tag{18}$$

s.t.

$$\sum_{i \mid b_u^i \neq 1} y_{i,j}^{(u/4)} \geq d_j^{(u/4)} \quad \forall j, u \tag{19}$$

$$\sum_{i \mid b_u^i \neq 1} y_{i,j}^{(u/4)} \leq \max(d_j^{(u/4)}, L^{max}) \quad \forall j, u \tag{20}$$

$$\sum_{j} y_{i,j}^t = 1 \quad \forall i, t \tag{21}$$

$$y_{i,j_1}^{t-1} + y_{i,j_2}^t \leq 1 + 2x_{i,j_1,j_2}^{t-1} \quad \forall i, j_1, j_2, t > 0 \tag{22}$$

$$y_{i,j_1}^{t-1} + y_{i,j_2}^t \geq 2x_{i,j_1,j_2}^{t-1} \quad \forall i, j_1, j_2, t > 0 \tag{23}$$

The objective function (18) minimizes the sum of time spent travelling between sectors by all agents across all time periods. The travel time for not changing sector is taken to be 0. (19) ensures that each location can meet its demands. Break times are met by using a more fine-grained set of

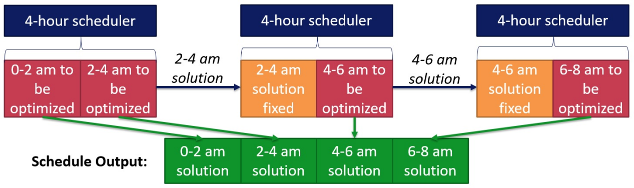


Figure 7: Partitioning heuristic for scalability. The Scheduler takes the allocation solution from the Optimizer and assigns actual agents to required locations.

time periods to determine if an agent is on break during that time. Agents are allocated on a 1-hourly basis, but are not counted towards meeting demand during the time they are on break. Formulating the constraint this way enables a 1-hour demand period to be covered by more than 1 agent as long as their non-overlapping break periods cover the full time window. (20) works in the same manner but sets an upper limit on the number of agents allocated to a sector to prevent clustering of any spare agents, using a default maximum value unless the required demand determined by the Optimizer is higher than the default. (21) ensures that each agent is assigned to exactly 1 location. (22) and (23) define x^{t-1}_{i,j_1,j_2}, which takes value 1 when y^{t-1}_{i,j_1} and y^{t}_{i,j_2} take value 1 (indicating that agent i goes from location j_1 to location j_2 at the crossover from time $t-1$ to time t). If only one of the assignment values is 1, the indicator is 0 and the travel time for that pair is not counted.

Ensuring Scalability with a Partitioning Heuristic

Like the Optimizer, the Scheduler is an MIP and suffers from scalability issues because, whilst the model appears simple in writing, constraints (22)-(23) can have a very large number of permutations. Strictly partitioning at the 2-hour level is not useful for the scheduling model due to the dependence of later solutions on earlier plans. Instead, we implement a partitioning heuristic that divides the set of time periods into a subset (e.g. of 2 1-hour periods) but takes the values of $y^{t_0}_{i,j}$ in each partition to be input parameters rather than decision variables. The value of these parameters is determined by the solution for that time period in the previous partition, with the only exception being for the first time period, which is solved in the normal fashion. The partition approach is illustrated in Fig. 7. Solving smaller problems in sequence is much faster than one large problem, and the partition can achieve a good travel time performance because earlier allocations are considered.

Additional Operational Considerations During COVID-19

The LEA conducting the trial for the GRAND-VISION system currently adopts a hierarchical system of command and control, with each agent operating within the wider geographic area reporting to a commander in charge of a smaller home cluster of 2-5 patrol locations. The goal of this work is to achieve better emergency response by creating a more fluid structure, with agents able to patrol multiple regions in the course of a shift. However, the current COVID-19 pandemic has resulted in a proliferation of safe distancing measures, with one consideration of our partner agency being to limit the amount of time agents spend outside of their home cluster, limiting interactions between team members where possible. To accommodate this temporary situation, we use a 'dummy' time period to manually allocate each agent to a region in their home cluster before the shift starts, thus each agent will start as close to home as possible. We then examine two alternative methods of discouraging deployments outside the home cluster, and compare them to the unrestricted scheduling model for travel time performance. We choose not to introduce a constraint on individual travel times, which could be added to the scheduling model in (18)-(23) as follows:

$$T_{j_1,j_2} x^{t-1}_{i,j_1,j_2} \leq T_{\max} \quad \forall i, j_1 \neq j_2, t > 0, \tag{24}$$

where $T_{\max}$ denotes the chosen upper time limit per journey. This method may not be effective in achieving the intended result while introducing the possibility of infeasibility, which is an undesirable outcome. Therefore, the first method we test is to quadratically penalise longer journeys by squaring the travel time value in the objective function. Thus (18) is re-written as:

$$\min \sum_{i} \sum_{j_1} \sum_{j_2 \neq j_1} \sum_{t>0} T^2_{j_1,j_2} x^{t-1}_{i,j_1,j_2}. \tag{25}$$

The drawback of this method is that it assumes a scenario in which the regions within a home cluster are closer than regions outside, which may not be true if regions are not uniform in size, as is the case in our scenario. Thus we consider the second method, which is to apply a penalty coefficient to the travel time, but only when the destination region is outside the home cluster for a particular agent. The objective function is re-formulated as follows:

$$\min \sum_{i} \sum_{j_1} \sum_{j_2 \neq j_1} \sum_{t>0} a^{j_2}_{i} T_{j_1,j_2} x^{t-1}_{i,j_1,j_2}. \tag{26}$$

$a^{j_2}_{i}$ is the penalty for travelling to a destination outside the home cluster, and follows the rule:

$$a^{j_2}_{i} = \begin{cases} 1, & \text{for } j_2 \in \mathcal{C}_i \\ 60, & \text{otherwise,} \end{cases}$$

where $\mathcal{C}_i$ denotes the set of patrol regions that constitute agent i's home cluster. We use a penalty of at least 60 minutes for this problem as it is comfortably larger than any individual journey, but the value should be calibrated for the timing values of the geographic area in question. A study of parameter tuning for the penalty coefficient is left as an extension for further work.

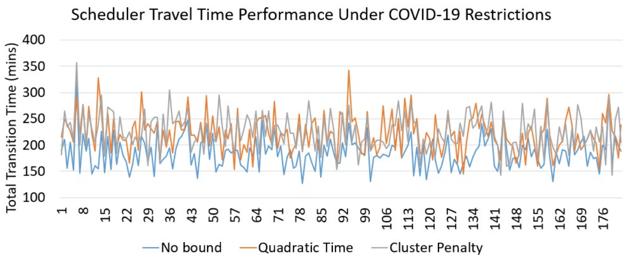


Figure 8: Travel time performance comparison of alternative methods of restricting travel due to COVID-19.

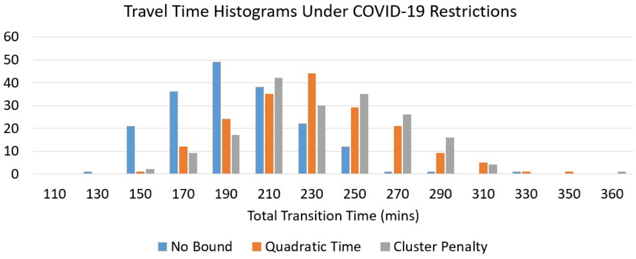
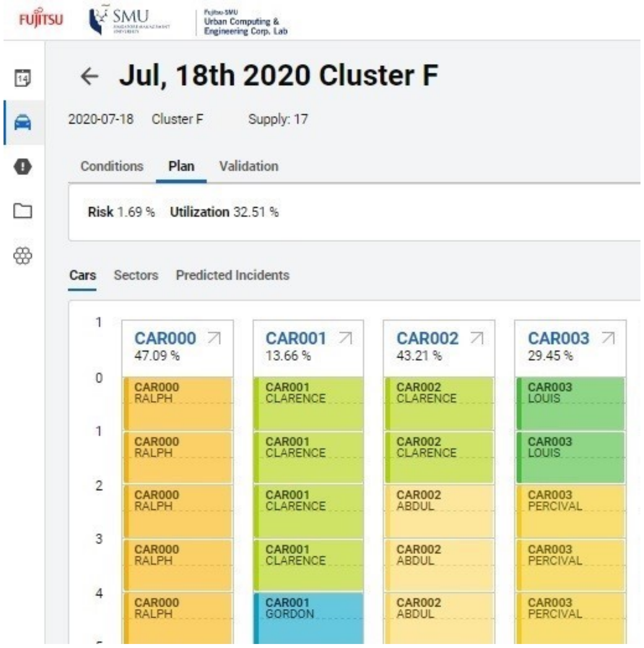


Figure 9: Histogram comparison of the distribution of travel times for alternative methods of travel restriction.



Fujitsu-SMU Urban Computing & Engineering Corp. Lab

← **Jul, 18th 2020 Cluster F**

2020-07-18 Cluster F Supply: 17

Conditions **Plan** Validation

Risk 1.69 % **Utilization** 32.51 %

Cars Sectors Predicted Incidents

1	CAR000 47.09 %	CAR001 13.66 %	CAR002 43.21 %	CAR003 29.45 %
0	CAR000 RALPH	CAR001 CLARENCE	CAR002 CLARENCE	CAR003 LOUIS
1	CAR000 RALPH	CAR001 CLARENCE	CAR002 CLARENCE	CAR003 LOUIS
2	CAR000 RALPH	CAR001 CLARENCE	CAR002 ABDUL	CAR003 PERCIVAL
3	CAR000 RALPH	CAR001 CLARENCE	CAR002 ABDUL	CAR003 PERCIVAL
4	CAR000 RALPH	CAR001 GORDON	CAR002 ABDUL	CAR003 PERCIVAL

Figure 10: Example schedule output screen with agent locations colour-coded on an hourly basis.

This last formulation incentivizes the deployment of agents to their own cluster, but does not prevent their deployment elsewhere when required. To evaluate the impact of each method on the travelling time required for the deployment schedule, we create a new deployment plan for each method over 180 days. For the quadratic time method and the penalty coefficient method, we calculate the actual travel time of the resulting schedule, to allow direct comparison with the unmodified method. We compare the time series performance of each method in Fig. 8 and plot a histogram of the travel times in Fig. 9. The unmodified solution serves as a baseline, as it will always provide the optimal performance (though when using generated incidents for training there will be day-to-day variation). The alternative methods do not have significantly inferior performance against the baseline. However, these results do not determine which method is best of themselves, as the resulting deployment plan for a less time-efficient method may be preferable from an operational perspective. Thus, they can be used by the partner agency to make an informed policy decision, leaving room for the experience of the field commanders, that can be implemented for the field trial.

Evaluation by Simulation

To produce our experimental results we train the deep learning model on a full year's real incident data, provided by our partner LEA. We execute all experiments through our deployment system, shown in Fig. 10. To find a robust solution that does not require a large scenario set for the SAA model, we generate 5 candidate plans with identical settings but different generated training incidents which, along with a simple greedy strategy (assign one agent per region), are evaluated on one month of real test data via dispatch simulation. For each 2-hour period in the 12-hour shifts (shifts are split at 8am and 8pm into day and night shifts), the deployment plan with the best average performance is used. Once the best deployment in each 2-hour period is found, the resulting combined solution is evaluated across a set of 7 months of test data (for confidentiality reasons we present test results using synthetic, rather than historical, incidents). We evaluate on a geographic area consisting of 13 patrol regions, with a default supply of 13 agents, a typical problem size for our LEA.

Dispatch Simulator

We use a dispatch simulator to evaluate the performance of deployment plans against test incidents. The simulation handles test incidents (synthetic or real) in chronological order. The behaviour of each agent is modelled according to the individual deployment determined by the shift plan. When an incident occurs, the available agent with the shortest response time is dispatched without reference to future incidents. If no agent is available, the first agent to become available is assigned. For urgent incidents we define success as the first responder arriving within 15 minutes, while the target is 30 minutes for non-urgent cases.

Performance Against Current Practice

We evaluate the performance of our solution against current practice. We calculate the monthly average failure rate ('risk') and compare it to the current practice performance, which is a static plan that does not vary by day. For day shifts we have a 1-hour training session and three 30-min breaks and for night shifts there are two 45-minute breaks

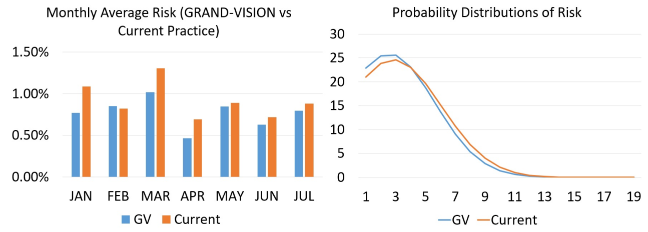


Figure 11: Monthly averages and daily distributions of failure rates on shifts and day types, GRAND-VISION vs current practice.

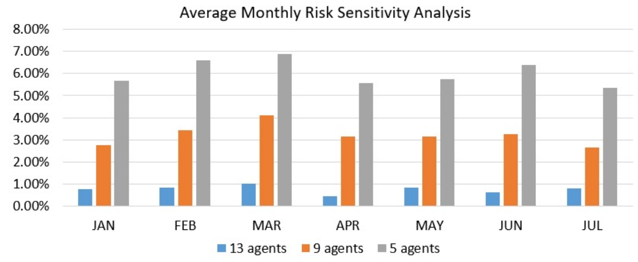


Figure 12: Failure rate sensitivity analysis giving insight into the potential to reassign manpower in emergencies without compromising performance.

and a 45-minute training session. In Fig. 11 we plot the failure rate performance on all day types for both day and night shifts, and show the distribution of daily risks. While the current practice can occasionally perform better on a given day, the monthly averages and distribution of daily failures is strongly in favour of the GRAND-VISION planner.

Performance with Reduced Agent Supply

The default agent supply is intended to provide good performance in most circumstances outside of the worst case. Given the global COVID-19 pandemic, law enforcement agents may be redeployed to assist in enforcement of stay-home notices, or to perform contact tracing, or may themselves be placed in quarantine due to the ubiquity of the virus. In eventualities such as this, an agency would benefit from insights into the level of manpower reduction they can tolerate without performance dropping to unacceptable levels. In Fig. 12 we perform a sensitivity analysis on the agent supply, simulating the monthly average performance for 13, 9, and 5 agents. We find that while a 5 agent supply yields an unacceptable increase in failure rates (even when the agents do not have break times), 9 agents can perform acceptably if required to, without losing time for training or breaks. Using the system in this way allows agencies to better utilize their manpower in an informed way, even when an unanticipated scenario like the pandemic occurs.

The GRAND-VISION System

The GRAND-VISION system is implemented in Python based on the Django framework for database integration, and linked to a web-based frontend app. A central backend server process handles requests received from the frontend. A planning worker process polls the database for pending deployment tasks. The planning worker executes the process given in Fig. 2, taking the incidents generated by the Incident Generator, which is a separate dedicated Flask app running TensorFlow. The predicted travel time from one location to another is provided by another dedicated service which is linked to a travel time prediction model not described in this paper. By adopting this modular service structure, new methods, such as enhanced incident prediction or travel time prediction, can be easily integrated for future system extension. The Optimizer and Scheduler modules use the Python API of the IBM CPLEX solver.

There are two types of users - Commander and Analyst. The Analyst user has access to all the system features, including the ability to generate deployment plans for all geographic areas under the agency, tune parameters such as the number of training scenarios for the Optimizer, and upload incident data for validation and updating the Incident Generator. The Commander can generate and view deployment plans for their own area of responsibility only.

Development and Trial Experience

A number of challenges and insights have come through working directly with our partner agency. Firstly and topically, the COVID-19 pandemic has presented difficulties, both in physically meeting with the client, and in executing the field trial, as the agency was not inclined to introduce experimental plans during such a disruptive time. However, it has presented some interesting research opportunities, such as the methods of movement restriction discussed earlier. It also allows us to investigate the influences of a pandemic on emergency incident behaviour, as the incident prediction model is based on pre-virus data. We will use the trial to assess its accuracy under unprecedented conditions.

Secondly, when developing solutions, there are a number of operational considerations to accommodate. At an early stage of the project, we considered altering the shifts, but the agency was concerned about the level of disruption to their logistics and their patrol agents, who structure their lives around the current shift pattern. This was a common refrain during the project development, as ideas had to be evaluated not just for their research merit, but for their operational feasibility and the ability to 'sell' the ideas to the commanders who are concerned with staff morale.

Thirdly, working with the agency from the start provided valuable insights into what factors must be specifically tailored compared to other use cases. Each LEA is different, and the system usability and functionality could be developed in line with the experience of the agents who would have to use it. In a real world problem, it is sometimes necessary to recognise the limitations of automated planning and scheduling, and know how to leave room for expert knowledge (such as through provision for pre-allocations).

Reflection on Ethical Considerations

The application of AI to predictive policing in the US has raised ethical concerns, as policing data can be influenced by officer prejudices and corrupt behaviour during its collection (Richardson, Schultz, and Crawford 2019). When used in a predictive system, this 'dirty' data amplifies discriminatory practices, resulting in the targeting of marginalized communities. Our system is deployed in a non-US country, which results in a different ethical landscape, which we discuss here. Firstly, the racial community segregation visible in US cities is not present in our partner country due to extensive public housing with allocation of flats via public ballot, constrained by a government requirement that the racial makeup of each estate must reflect the national demographics. Secondly, our data is limited to objective emergency call records, thus there is less scope for pollution by corrupt police practices. However, if the tool is applied in other countries, care must be taken to ensure that the priority classification and level of response is not biased by the location of an incident (e.g. in a deprived neighbourhood). The prediction data also cannot capture unreported matters; communities with low trust in law enforcement may be reluctant to place a call. Thirdly, the LEA does not respond exclusively to crimes per se, but also frequently to general incidents including public assistance and medical emergencies. This means the goal is to reduce response time rather than crime incidence, so there is limited incentive for data manipulation to 'improve' crime statistics. Again, however, the tool could be employed exclusively for criminal incidents, in which case proper oversight would need to be applied to ensure all incidents were properly responded to. This tool is not immune to improper use by unprincipled governments. Confidence in a lack of manipulation is supported by our partner country's international reputation for low levels of corruption, and the trust the public has in the LEA, which has an ethnically diverse staff. However, the fairness of the system is dependent on the data used to predict demand, and must be carefully selected, else oppressive regimes may target opposition for their own benefit.

Conclusion

This paper presents a practical system for generating agent deployments for a real LEA, using generated incidents to inform the creation of patrol schedules. The system is currently undergoing field trials with our partner agency, and future work will present the results of the trial. The trial results will also be used to guide the improvement of the deployment engine, particularly the Incident Generator, and the development of a fully-featured travel time prediction module. Other avenues for further work include parameter tuning of the COVID-19 Scheduler considerations and improvements to the scalability of the Optimizer module to support a more robust set of incident inputs.

Acknowledgements

This research is funded by the National Research Foundation Singapore under its Corp Lab @ University scheme and Fujitsu Limited as part of the A*STAR-Fujitsu-SMU Urban Computing and Engineering Centre of Excellence.

References

Amador, S.; Okamoto, S.; and Zivan, R. 2014. Dynamic Multi-agent Task Allocation with Spatial and Temporal Constraints. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '14, 1495–1496.

Caplan, J. M.; Kennedy, L. W.; and Miller, J. 2011. Risk Terrain Modeling: Brokering Criminological Theory and GIS Methods for Crime Forecasting. *Justice Quarterly* 28(2): 360–381.

Chase, J.; Nguyen, D. T.; Sun, H.; and Lau, H. C. 2019. Improving Law Enforcement Daily Deployment Through Machine Learning-Informed Optimization under Uncertainty. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, 5815–5821.

Lau, H. C.; Yuan, Z.; and Gunawan, A. 2016. Patrol scheduling in urban rail network. *Annals of Operations Research* 239(1): 317–342.

Levine, N. 2017. *CrimeStat: A Spatial Statistical Program for the Analysis of Crime Incidents*, 381–388. Cham: Springer International Publishing.

Malleson, N.; and Andresen, M. A. 2015. Spatio-temporal crime hotspots and the ambient population. *Crime Science* 4(1): 1–8.

Mukhopadhyay, A.; Zhang, C.; Vorobeychik, Y.; Tambe, M.; Pence, K.; and Speer, P. 2016. Optimal Allocation of Police Patrol Resources Using a Continuous-Time Crime Model. In *Decision and Game Theory for Security - 7th International Conference, GameSec 2016*, 139–158.

Richardson, R.; Schultz, J.; and Crawford, K. 2019. Dirty Data, Bad Predictions: How Civil Rights Violations Impact Police Data, Predictive Policing Systems, and Justice. *New York University Law Review Online* 94(192).

Saisubramanian, S.; Varakantham, P.; and Lau, H. C. 2015. Risk Based Optimization for Improving Emergency Medical Systems. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, 702–708.

Wang, B.; Zhang, D.; Zhang, D.; Brantingham, P. J.; and Bertozzi, A. L. 2017. Deep Learning for Real Time Crime Forecasting. *arXiv:* 1707.03340.

Wang, W.; Dong, Z.; An, B.; and Jiang, Y. 2020. Toward Efficient City-Scale Patrol Planning Using Decomposition and Grafting. *IEEE Transactions on Intelligent Transportation Systems* To appear. URL https://ieeexplore.ieee.org/document/8930604.

Zhang, J.; Zheng, Y.; and Qi, D. 2017. Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, 1655–1661.