Multi-Agent Planning with Cardinality: Towards Autonomous Enforcement of Spectrum Policies

Maqsood Ahamed Abdul Careem, Aveek Dutta and Weifu Wang
Department of Electrical and Computer Engineering
University at Albany SUNY, Albany, NY 12222 USA
{mabdulcareem, adutta, wwang8}@albany.edu

***Abstract*—The distributed nature of policy violations in spectrum sharing necessitate the use of mobile autonomous agents (*e.g.*, UAVs, self-driving cars, crowdsourcing) to implement cost-effective enforcement systems. We define this problem as Multi-agent Planning with Cardinality (MPC), where Cardinality represents multiple, unique agents visiting each infraction location to collectively improve the accuracy of the enforcement tasks. Designed as a practical and deployable system, our solution leverages crowdsourced information to determine the optimum Cardinality and provide a routing schedule for the agents to achieve the desired level of accuracy of detection and localization at minimum possible cost. We show that by estimating spatial orientation of the agents with single antenna, the accuracy is improved by 96% over crowdsourcing only. Using geographical maps as the basis, we solve the scheduling problem with a 3-approximation ratio in polynomial time that exhibits statistically similar performance under variety of urban locale across multiple continents. The longest path traversed by an agent on average is 1.2km per unit diagonal length of a rectangular geographic area, even when there are twice as many infractions as agents.**

I. Introduction

Enforcement of spectrum policies is complementary to the well-studied problem of Dynamic Spectrum Access (DSA). However, the distributed nature of these policy violations (defined as *"Targets"*) require accurate, cost-effective and mobile, autonomous entities[1](defined as *"Agents"*) to carry out enforcement tasks. These tasks can be generalized as various levels of signal measurement, waveform classification and localization in order to pin-point rogue sources with very high accuracy. The balance between cost and accuracy of such an enforcement system critically depend on the appropriate amount of resources (agents) mobilized to the right location in the shortest possible time. More so because wireless signal classification greatly benefits from proximity to the potential source, different sensing parameters (bandwidth, sample rate, battery constraints, etc) and aggregation of observations from multiple agents.

To this end, crowdsourced paradigm [1], [2], [3] has been shown as a viable apparatus. However, it suffers from many inefficiencies like lack of trust and efficient incentive mechanism that may not provide bounded guarantees of accuracy (*e.g.*, detection and location) and cost (*e.g.*, incentives, capital and operational costs). Instead, we envision a hybrid approach that leverage crowdsourced measurements (akin to eye-witness accounts) to deploy mobile, autonomous agents to the target sites depending on the veracity of these measurements. Our work builds on any crowdsourced paradigm, where the *wisdom of crowd* is simply used to assess the need for additional resources to achieve a desired level of accuracy and cost, thus avoiding unnecessary and restrictive burden on the *crowd* (like undesired mobility, prioritized sensing, low incentive, etc [4], [3]). This MPC system operates in two steps as shown in figure 1. The Fusion center collects the initial assessment of the target[2] and derives the *cardinality* necessary to collectively improve the accuracy at a bounded cost. This information is used to deploy mobile agents to perform the additional detection and localization tasks under the constraint of scheduling a fixed number of agents in minimum time.

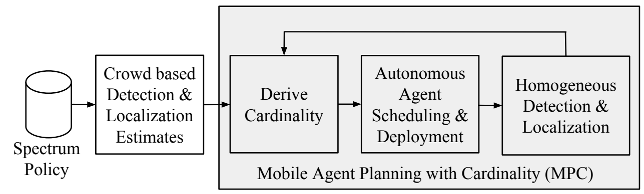
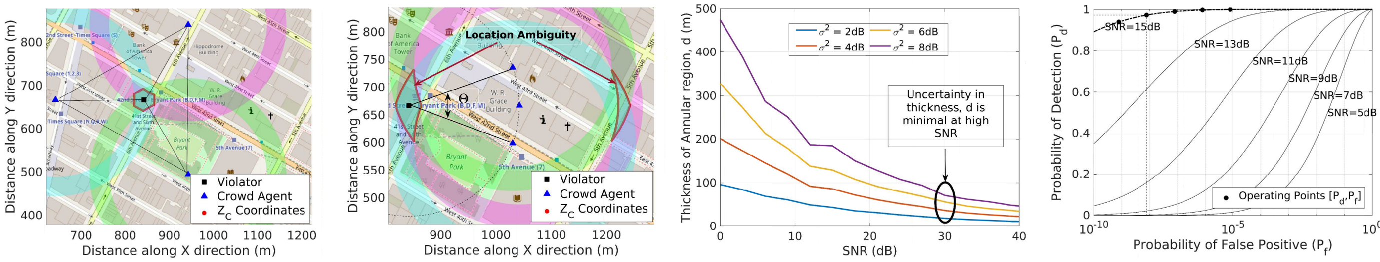


Fig. 1: MPC blocks: Crowdsourced measurements provide the basis for deploying mobile agents to detect and localize targets. The shaded blocks are the contribution of this work.

Cardinality, refers to the number of unique, mobile and homogeneous agents (not including the participants from the crowd) visiting targets, simultaneously or otherwise, to achieve a target accuracy of the enforcement tasks. Accuracy has two primary dimensions: a) Detection of a *bad* signal (often expressed as a confusion matrix [5] and b) Location estimate. Since the agents are homogeneous they can be directed (although at a cost) to a near-optimal orientation or detect signals with desirable operating points to independently maximize along both the dimensions. We adopt the widely used geometric trilateration [1], [6] as the basis to locate a target and calculate the optimum cardinality that minimizes the Geometric Dilution of Precision (GDOP). This is followed by routing a finite number of agents to multiple targets while fulfilling the cardinality determined in the previous step. The solution to this lies at the intersection of finding the shortest path between nodes in a graph and finding a schedule (or order) for the agents to visit a set of targets. However, in MPC, the additional requirement of fulfilling the cardinality for each target, makes the solution orthogonal to the existing literature [7], [8]. It is not necessary to route all the agents (as per the cardinality) to a target at the same time. To ensure a fast convergence of the scheduling algorithm the agents may start from any point and take any path as long as it covers all the targets in the least possible time.

[1]We do not impose any restriction on the type of autonomous agents as long as those use the road infrastructure to navigate. These agents can be radio nodes mounted on autonomous vehicles or low flying UAVs (for sensing ground based communication and avoid obstacles) or a combination of both.

[2]Crowdsourced agents may detect infractions with a wide variety of accuracy (false and true positives) due to heterogeneous hardware and their relative proximity to the target. There are many crowdsourced models [1], [3] but our work subsumes any such paradigm without loss of generality.



(a) Ideal arrangement of agents leads to low GDOP

(b) High GDOP using crowd-sourced measurements.

(c) Thickness of the annulus, d decreases with SNR

(d) Receiver operating Characteristic of a detector

Fig. 2: In trilateration, the location of a target is given by the intersection of the annular regions. The thickness of the annular regions reduces with SNR based on (1). However, GDOP depends on the relative positioning of the agents as well. Also, An ROC curve dictates the performance of any detector and assimilating results from various agents leads to higher accuracy.

In the final step the accuracy is iteratively improved until the target level is achieved. Trilateration with no GDOP results in a convex polygon that includes the target (§II). Each agent is initially routed to the centroid of the polygon and then visits each vertex to collect measurements and report to the Fusion center. This ensures, aggregation of multiple sensing results (using some form of weighted combination) at a very high SNR. It is to be noted that these tasks may involve deeper signal processing and possible indoor sensing as well, which is not in the scope of this work. Since, the cost incurred to conduct this localized sensing, is small compared to the overall cost of scheduling it can be safely ignored in the larger context of the cost of enforcement.

Collectively, these three parts constitute a solution to the MPC problem that operate in lock-step with any crowdsourced paradigm to achieve very high accuracy at a bounded cost that is also minimum under the above constraints.

II. Background and Known Results

Trilateration under noise: Although our solution is independent of the underlying crowdsourced paradigm, we adopt trilateration based localization [1] to derive the cardinality for the infractions. Trilateration [6] involves estimating the distance (also called *range*) of a receiver from a potential source based on the path loss incurred by a signal using an approximation of the wireless channel. For example, in the Hata-Urban [9] channel model, the distance from a transmitter, d is related to the path-loss, PL as,

$$PL{=}A{+}B\log(d){+}C \implies d{=}10^{\frac{PL-A-C}{B}} \tag{1}$$

$$\text{where, } A{=}69.55+26.16\log(f_c)-13.82\log(h_b) \\ -3.2(\log(11.75h_m))^2-4.97$$

$$B{=}44.9{-}6.55\log(h_b) \text{ and } C{=}0 \quad \text{(Large metropolitan areas)}$$

$$PL \text{ [dBm]}{=}P_t \text{ [dBm]} - \text{SNR [dB]} - P_N \text{ [dBm]}$$

P_t is the transmit power and SNR is the received signal to noise ratio at the agent. P_N denotes the average noise power in absence of any signal is assumed to be -96dBm.

In (1), uncertainty arise from the assumption about P_t, measurement noise in estimating the SNR and approximation of the channel model. These errors are collectively modeled as a random variable $X \sim \mathcal{N}(\mu, \sigma^2)$, with μ=0 and σ^2=2dB. This noise model leads to two limits, $[\text{SNR}{\pm}(X{=}x)]$dB that translates to two range values using (1): d_{outer} and d_{inner}, resulting in annular regions (instead of circles) of thickness $d = (d_{outer} - d_{inner})$ for each enforcer. Thus, geometric trilateration using these annular regions provides an *estimate* of the location of the violator. The overlapping area of the annular regions creates a *convex polygon* containing the violator and its area is a measure of accuracy of localization. Figure 2a shows an ideal scenario where the location of the violator is estimated by using the measurements reported by three closest (highest SNR) members of the crowd. It has a very low GDOP because the crowd agents are uniformly distributed on all sides of the violator providing an accurate estimate of the target. While, figure 2b shows such a scenario where the agents are located within a certain angle of the violation. This produces multiple convex polygons because of GDOP. This is precisely the drawback of any crowdsourced paradigm. It is to be noted that the GDOP can only be eliminated if there is a viable way of positioning the agents, which is not possible in a purely crowdsourced enforcement paradigm. The GDOP is used as the guiding metric to derive the cardinality of a target.

Accuracy and GDOP: Intuitively, it is desirable to choose crowd agents that are operating at high SNR (closer to target). The area of the convex polygon is a function of SNR and the noise model given by (1), which defines the thickness of annular regions. Figure 2c shows that higher the SNR, lower is the median width of the annular region, d and consequently lower is the uncertainty in the location of the targets Hence, it is desirable to position the agents as close to the target as possible. Therefore, one of the objectives is to deploy mobile agents to surround the initial crowdsourced estimate of the convex polygon in order to minimize GDOP. The number of agents required for this is also the cardinality of the target.

ROC of a signal detector: Signal detection and parameter estimation is a rich and well-studied area. The Receiver Operating Characteristic (ROC) curve (figure 2d) shows the ROC curve for Neyman-Pearson detector [10]) is universally used to define the performance of a classifier or an estimator. In this work, the agents rely on the ROC curve to choose an operating point based on the SNR of the received signal similar to [1], [11]. Directing crowd agents to *always* operate at a desirable operating point can be cost prohibitive but a group of homogeneous autonomous agents can be mandated to yield a high detection result, especially since the SNR is also very high at the vertices of the polygon as mentioned above. Therefore, our work is independent of any specific detection scheme and simply ensures that the agents are always delivering the highest possible accuracy, *e.g.*, aggregating the operating points chosen by the agents on the 15dB curve in figure 2d will always yield the *best* result for detection.

III. Multi-Agent Planning with Cardinality

In the context of the MPC problem, let the set of m targets be denoted by $T = \{T_1, ..., T_m\}$ located at coordinates specified by the set $\mathbb{t} = \{t_1, ..., t_m\}$. Let the crowdsourced estimates of the locations of the set of m targets be $\mathbb{t}_C = \{t_{C,1}, ..., t_{C,m}\}$. Let the set of n autonomous agents be denoted by $A = \{A_1, ..., A_n\}$, with coordinates $\mathbb{a} = \{a_1, ..., a_n\}$. Let the set of m convex polygons for targets T, as determined by the crowdsourced and autonomous agent based localization, be denoted by $\mathcal{Z}_C = \{\mathcal{Z}_{C,1}, ..., \mathcal{Z}_{C,m}\}$ and $\mathcal{Z}_A = \{\mathcal{Z}_{A,1}, ..., \mathcal{Z}_{A,m}\}$ respectively.

Algorithm 1: MPC Algorithm

1 **Function** *MPC(Map,* $\mathbb{a}$*,* $\mathcal{Z}_C$*)*
2 $\quad$ $\gamma_{th} = 10m^2$; $\mathbb{t}_C$ = *getCentroids*($\mathcal{Z}_C$);
3 $\quad$ **while** *True* **do**
4 $\quad\quad$ [$\mathcal{C}$,$\mathcal{Z}_A$] = *findCardinality*(Map, $\mathbb{t}_C$, Z_C);
5 $\quad\quad$ $\mathbb{t}$ = *getCentroids*($\mathcal{Z}_A$);
6 $\quad\quad$ $\mathcal{P}$ = *findAgentSchedule*(Map, $\mathbb{a}$, $\mathbb{t}$, $\mathcal{C}$);
$\quad\quad$ `// Take measurements & evaluate actual` $\mathbb{t}$
7 $\quad\quad$ **if** $\mathcal{Z}_A < \gamma_{th}$ **then** break; **else** $\mathcal{Z}_C = \mathcal{Z}_A$;$\mathbb{t}_C = \mathbb{t}$;
8 $\quad$ **end**
9 $\quad$ **return** $\mathcal{P}$;
10 **end**

Algorithm 1 shows the steps in solving the MPC problem. It is initialized with the starting locations of the autonomous agents, $\mathbb{a}$, and $\mathcal{Z}_C$, followed by updating the target location set, $\mathbb{t}_C$ with the geometric centroids of $\mathcal{Z}_C$ in line 2. The accuracy of localization is defined by the area of $\mathcal{Z}_A$, and the target value is chosen to be $10m^2$. Although a higher accuracy can be achieved in theoretical sense, in practice, the accuracy is limited by the feasibility of deploying agents to the vertices of $\mathcal{Z}_A$. In other words, if the vertices of $\mathcal{Z}_A$ fall over (or inside) any structure, then it requires additional resources to further improve the accuracy of localization. Algorithm 1 terminates under such infeasible conditions but provides the maximum accuracy in outdoor setting.

This algorithm has two key steps: A) Derivation of Cardinality, and B) Scheduling of autonomous agents. Step-A calculates the cardinality ($\mathcal{C}$) and the convex polygons ($\mathcal{Z}_A$) in $findCardinality$ (line 4), using the estimates from the crowdsourced phase, $\mathcal{Z}_C$ and $\mathbb{t}_C$. Figure 3a shows an example of a target with crowdsourced detection and localization. Figure 3b shows the cardinality for that target, the optimal orientation of the autonomous agents and the improvement in the accuracy of localization over crowdsourced localization by employing Algorithm 2 in §IV. Then the target location set, $\mathbb{t}$ is updated with the geometric centroids of $\mathcal{Z}_A$ in line 5.

Step-B uses the locations, $\mathbb{t}$ and cardinality $\mathcal{C}$ from Step-A to determine the paths, $\mathcal{P}$ for each agent by calling the subroutine $findAgentSchedule$ in line 6, outlined in §V. Figure 3c shows an example schedule in a major city in the US with a small set of agents and targets. Two properties are evident from the schedule: 1) Paths for different agents overlap but the same agent never visits a target more than once and 2) The agents can start and finish at any target as long as it minimizes the length of the longest traversing agent. These two properties collectively lead to Algorithm 3 in §V-A that iteratively prunes the paths as the cardinality for the targets are fulfilled, terminating with the quickest possible schedule for all agents.

Then the agents are deployed to each target and measurements are taken to validate the calculated $\mathcal{Z}_A$ and $[P_d, P_f]$ (true and false positives). If $\mathcal{Z}_A$ is greater than the threshold γ_{th}, then steps A and B are repeated by setting $\mathcal{Z}_C = \mathcal{Z}_A$ and $\mathbb{t}_C = \mathbb{t}$, until $\mathcal{Z}_A$ is less than the threshold γ_{th}. In other words at each round of enforcement we use the estimated convex polygon, $\mathcal{Z}_A$ and locations $\mathbb{t}$ as the inputs for the next round of enforcement. This procedure ensures that each violation is localized with target accuracy threshold with no ambiguity. The output of Algorithm 1 are the paths of all the agents ($\mathcal{P}$).

In practice, once an agent visits a target, it performs a single round of patrolling by visiting the vertices of the optimal polygon circumscribing $\mathcal{Z}_C$ as shown in figure 3b. At each vertex the agent collects measurements (SNR) and estimates the annular region based on the noise model mentioned in §II. Since, each target is visited by a number of agents equal to its cardinality the average of all the measurements minimizes the error in $\mathcal{Z}_A$ and $[P_d, P_f]$. This aggregation is independent of the MPC algorithm and can be designed to achieve other objectives like trust and fault tolerance. The cost of a single round of patrolling by the agents is negligible, since the area of $\mathcal{Z}_A$ are very small compared to the cost of scheduling the agents to the the target locations. Hence, this cost is ignored from the overall cost of scheduling.

IV. Step-A: Determination of Cardinality

Definition 1 (Cost of Localization) *The Cost of Localization for target $T_j \in T$, given $\mathcal{Z}_{C,j}$, i agents deployed to target T_j, and the convex polygon $\mathcal{Z}^i_{A,j}$ is defined as,*

$$Cost\ of\ Localization = \frac{\mathcal{Z}^i_{A,j}}{\mathcal{Z}_{C,j}} + \lambda i \tag{2}$$

where $\mathcal{Z}^i_{A,j}$ denotes the convex polygon with i agents on the vertices of the smallest polygon circumscribing $\mathcal{Z}_{C,j}$ and $\mathcal{Z}^i_{A,j}/\mathcal{Z}_{C,j}$ denotes the improvement in the accuracy of localization ($\mathcal{Z}^i_{A,j} \ll \mathcal{Z}_{C,j}$) over crowdsourcing after deploying i agents. The regularization parameter, λ is a non-negative value that trades off the localization accuracy with the cost of deploying more agents.

Definition 2 (Cardinality) *The Cardinality of a target $T_j \in T$, denoted by C_j is the total number of unique agents A_i, that are required to visit T_j. For each T_j $\exists$ $C_j \geq 1$.* The set of cardinality for the m targets is denoted by $\mathcal{C} = \{C_1, ..., C_m\}$.

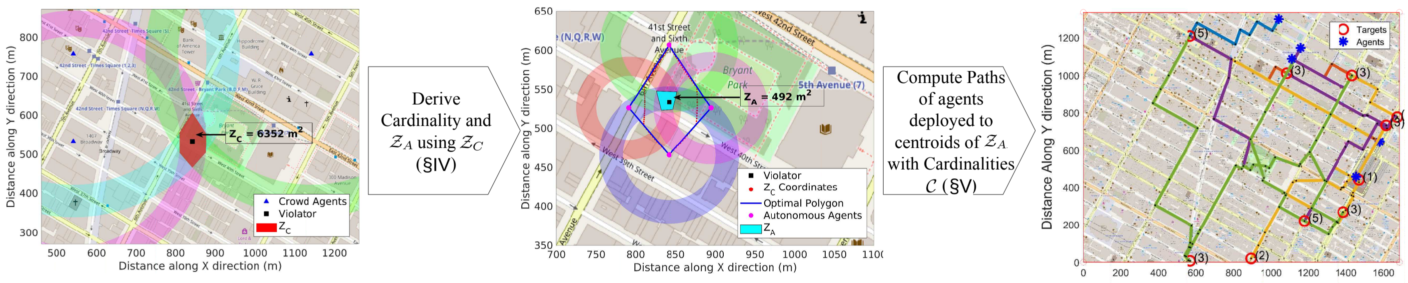
Thus, the desired cardinality, C_j for each target, $T_j \in T$, is the number of unique agents (i) for which the Cost of Localization for target T_j is minimum.

$$C_j = \arg\min_i (Cost\ of\ Localization) \tag{3}$$

A. Algorithm to determine Cardinality

The key idea here is to find an optimal polygon for each target T_j that circumscribes the convex polygon, $\mathcal{Z}_{C,j}$. By deploying the autonomous agents to the vertices of this optimal polygon we can ensure that the target is localized with low GDOP and high accuracy while choosing optimum operating points on the ROC for signal detection.

Initially the number of edges of $\mathcal{Z}_C$ is extracted in line 3. For each target T_j, the optimal polygon that circumscribes $\mathcal{Z}_{C,j}$ is determined. To do this we scan through the number of agents (i) starting from 3 agents to a maximum number that

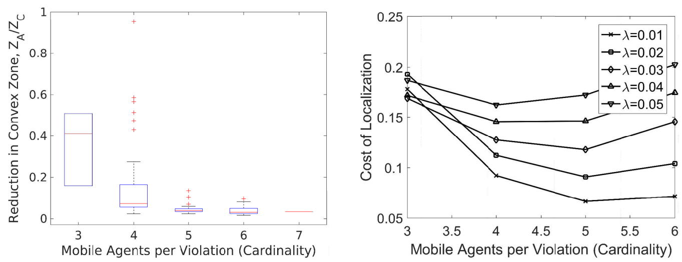


(a) Crowdsourced localization and $\mathcal{Z}_C$. Colored disks portray the annular regions of the crowd agents.

(b) Autonomous agent localization and $\mathcal{Z}_A$. Colored disks portray annular regions of autonomous agents.

(c) Example routing in NYC. Cost metric = $2.374 km/km$.

Fig. 3: (a), (b) The autonomous agent based localization achieves a 92.25% reduction in the area of the convex polygon containing the violation. (c) Example routes and the cost metric (details in §VI) for 5 agents and 10 targets in New York City.



(a) Reduction in the convex polygon ($\mathcal{Z}_A/\mathcal{Z}_C$) with Cardinality

(b) Cost of Localization vs Cardinality.

Fig. 4: Accuracy and cost of localization using Algorithm 2. For $\lambda = 0.01$, the optimal cardinality is 5 and the median reduction in the area of the convex polygon is 96%.

Algorithm 2: Algorithm to determine Cardinality

1 **Function** *findCardinality(Map,* $\mathbb{t}_C$, $\mathcal{Z}_C$*)*
2   **for** *j=1:size(*$\mathcal{Z}_C$*)* **do**
3   numEdges = $\mathcal{Z}_C$.Edges;
    `// Find optimal circumscribing polygon`
4   **for** *i=3:numEdges* **do**
5   *MinPoly*[i] = *findMinPoly*($\mathcal{Z}_C$,i);
6       $\bar{\mathcal{Z}}_A[i]$ = *findConvexPoly*(*MinPoly*[i], $\mathbb{t}_C$[j]);
7       $\overline{Cost}_{Loc}$[i]=$\frac{\bar{\mathcal{Z}}_A[i]}{\mathcal{Z}_C} + \lambda i$;
8   **end**
9   [$Cost_{Loc}$[j], $\mathcal{C}$[j]] = *min*($\overline{Cost}_{Loc}$);
10     $\mathcal{Z}_A[j] = \bar{\mathcal{Z}}_A[\mathcal{C}[j]]$; `// Convex Polygons`
11   **end**
12   **return** $\mathcal{C}$, $\mathcal{Z}_A$;
13 **end**

is equivalent to the number of edges of $\mathcal{Z}_{C,j}$. For each i we find the smallest polygon, $MinPoly$ with i number of sides that circumscribes $\mathcal{Z}_{C,j}$ (line 5). Line 6 calculates the convex regions $\bar{\mathcal{Z}}_A[i]$, when i agents are deployed to the vertices of $MinPoly$. This step involves, computing the annular regions and trilaterating as described in §II assuming that the target is at $\mathbb{t}_C$[j] and the agents are at the vertices of $MinPoly$. The overlapping area of these annular regions is the convex polygon $\bar{\mathcal{Z}}_A[i]$. Next, we determine the cost of localization for each i according to Definition 1. The optimal circumscribing polygon is the polygon that gives the minimum cost of localization. Thus, line 9 gives the Cost of Localization for target T_j and the cardinality of T_j is equal to the number of sides of the optimal polygon. The above steps are repeated, to determine the cardinality, $\mathcal{C}_j$ and $\mathcal{Z}_{A,j}$ for all targets, $T_j \in T$.

B. Impact on Localization

Figure 4 shows the fidelity of localization of the autonomous agents oriented as described in Algorithm 2. Figure 4a shows the reduction in the area of the convex polygons encompassing the targets over crowdsourced localization. It shows that the higher the number of agents deployed to the target, the smaller is $\mathcal{Z}_A$ and the higher is the accuracy of localization. Figure 4b shows the dependence of the optimal cardinality on the regularization parameter, λ. At larger values of λ, a lower cardinality provides a lower cost of localization.

C. Impact on Detection

The set homogeneous autonomous agents can be directed to operate at a desirable operating point on the ROC. Also, since Algorithm 2 positions the agents on the vertices of the optimal polygon (figure 3b) and the SNR is very high at these vertices they guarantee a near optimum detection result (F-score[12] ≈ 1) regardless of the chosen operating point, as shown in figure 2d. Such a guarantee cannot be made for crowd agents as mandating the crowd to operate at a fixed operating point may be cost prohibitive. Thus, it is guaranteed that autonomous agents provide better detection accuracy than crowd agents.

V. Step-B: Schedule Autonomous Agents

After ascertaining the improvement in localization based on the optimal cardinality, unique agents are routed to each target. The starting point of the scheduling phase is the construction of an undirected, weighted graph $G = (V, E)$ from the road network of the geographical area being enforced for spectrum policies, where the roads are mapped as edges E and the intersections as vertices V. A Path in $G = (V, E)$ is defined as a subgraph $P = (V_s, E_s)$, if V_s is a set of k vertices of its base graph G and $E_s = \{(x_1, x_2), (x_2, x_3), ..., (x_{k-1}, x_k)\} \subseteq E$ is the set of $k-1$ edges that connect those vertices. The length of a path depends on the number of its edges and their weights, $w(v_i, v_j) = w(v_j, v_i)$. In this paper, the weight associated with each edge is the geographical distance between the corresponding vertices.

The *Cost of Scheduling* n agents to visit m targets is the time it takes for all the agents to cover m targets while satisfying the cardinality for each target. This time is determined by the agent that takes the longest time to traverse its path. Since, all agents are assumed to travel at the same speed, the time taken by an agent is determined by the sum of the edge weights of E_s. Finding the costliest path, $P = (Vs, Es)$ is the central goal of this work. For practical purposes, $m \geq n$.

Definition 3 (Cost of Scheduling) *Given the path P_i of an agent a_i of length l_i, the Cost of Scheduling is the length of the path of the longest travelling agent.*

$$Cost\ of\ Scheduling = max_{\forall i} c(P_i) = max_{\forall i} l_i$$

Where, the cost $c(.)$ of a path of k vertices (targets) is the sum of its edge weights,

$$c(P) = \sum_{i=1}^{k} w(x_i, x_{i+1})$$

Definition 4 (Uniqueness) *Uniqueness is the necessary condition that requires distinct agents A_i to visit each target T_j in order to fulfill its cardinality C_j).*

For example, if $C_j = 2$, *Uniqueness* guarantees that even if agent A_i traverses multiple times through target T_j, it still needs another agent, other than A_i to visit T_j to fulfill the cardinality of 2. In practice, unique agents provide additional layers of information that can be assimilated for higher accuracy [1].

Definition 5 (Schedule) *Given a set of m targets $\{T_1, T_2, ..., T_m\}$ at $t_1, t_2, ..., t_m$, where $t_j \in V$, a set of n agents $\{A_1, A_2, ..., A_n\}$ at $a_1, a_2, ..., a_n$, where $a_i \in V$, $m \geq n$, and $max_{\forall j}(C_j) < n$, the Schedule is to find the paths $P_i, \forall A_i \in A$ to visit m targets in the shortest possible time with C_j unique agents visiting T_j, $\forall T_j \in T$.*

The solution is the set of paths, $\mathcal{P} = \{P_1, ..., P_n\}$ such that the the cost of scheduling (according to Definition 3) is minimum, while ensuring the number of unique agents visiting each target T_j is exactly equal to $C_j, \forall T_j \in T$ (Definition 2 and 4). The targets can be visited at any time during their traversal without any constraint of waiting time or synchronization, until the cardinality is satisfied for each target.

A. Algorithm for the Schedule

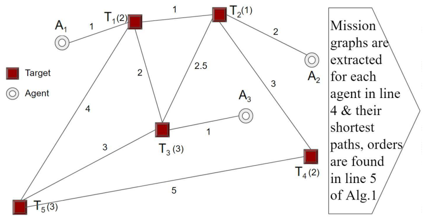
The algorithm is initialized with the locations of the agents, $\mathbb{a}$ and the targets, $\mathbb{t}$ with corresponding Cardinality $\mathcal{C}$, projected on to a graph $G = (V, E)$, extracted from open source map engines like OpenStreetMap [13]. For one round of enforcement activity, the locations of the targets are assumed to be constant while the agents follow a schedule to visit the targets. Line 2 in Algorithm 3 initializes these steps. It is assumed that the Dispatch, where the algorithm is executed has prior information about the initial conditions.

The goal of finding the shortest path between the points in a graph is accomplished by creating a *Mission-Graph* for every agent in $\mathbb{a}$ as shown in line 3 and the corresponding subroutine in lines 20 – 28. The Mission-Graph is defined as a complete graph $\bar{G}_i = \bar{V}_i, \bar{E}_i$, where, $\bar{V}_i = T \cup A_i$ The edge weights, $\bar{w}(p, q)$ is the length of the shortest route between nodes $p, q \in \bar{V}$, computed using Dijkstra's shortest path algorithm in lines 24 and 25. In other words, the Mission-Graph provides the best geographical route for each agent A_i to reach every target T_j and also the shortest route between any two targets in $\bar{V}$. Given, the shortest paths in the *Mission-Graph*, Line 4 calculates the schedule (order) for each agent to cover all the targets in $\bar{V}$ in the shortest time, which is also the sum of edge-weights $\bar{w}(p, q)$ in the path P_i. This is equivalent to solving the TSP for each agent and dropping the last edge of the TSP tour to obtain the path P_i. Considering the best achievable performance to solve the TSP for each agent, we utilize the approach in [14].

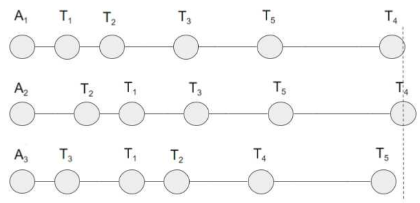
Algorithm 3: Path Pruning Algorithm

1 **Function** *findAgentSchedule(Map, $\mathbb{a}$, $\mathbb{t}$, $\mathcal{C}$)*
2 targets_assigned=[$\mathbb{t}$;...;$\mathbb{t}$];$\mathcal{X}$=[n,...,n]; `// Visits Count`
3 $\bar{G}$=*findMissionGraphs*(*Map*,$\mathbb{a}$, $\mathbb{t}$);
`// Order for all agents to cover all targets`
4 [$\mathcal{P}$,costs] = TSP($\bar{G}$, $\mathbb{a}$, $\mathbb{t}$);
`// Compute shortest path to find Schedule`
5 **while** $\mathcal{X} \neq \mathcal{C}$ **do**
6 i=0; k=getMax(costs);
`// Prune redundant edges`
7 **while** *True* **do**
8 l=$\mathcal{P}$[k][end-i];
9 **if** *$\mathcal{X}$[l] > $\mathcal{C}$[l]* **then**
10 targets_assigned[k][end-i]=[]; break;
11 **end**
12 i=i+1;
13 **end**
`// Reevaluate TSP for costliest agent`
14 $\bar{G}$[k]=graph(Dijkstra([$\mathbb{t}$,$\mathbb{a}(i)$],G[k]));
15 [$\mathcal{P}$[k], costs[k]] = TSP($\bar{G}$[k], $\mathbb{a}$[k], targets_assigned[k]);
16 $\mathcal{X}$[l]=$\mathcal{X}$[l]-1;
17 **end**
18 **return** $\mathcal{P}$, max(costs);
19 **end**
20 **Function** *findMissionGraphs(Map, $\mathbb{a}$, $\mathbb{t}$)*
21 City_Graph=graph(Map);`// Extract connectivity`
22 $\bar{G}$=[];`// Extract Mission Graph for each agent`
23 **for** $i = 0$ ***to*** *size(a)* **do**
24 DistanceMatrix=Dijkstra([$\mathbb{t}$,$\mathbb{a}(i)$],City_Graph)
25 $\bar{G}$[i]=graph([$\mathbb{t}$,$\mathbb{a}(i)$],DistanceMatrix)
26 **end**
27 **return** $\bar{G}$;
28 **end**

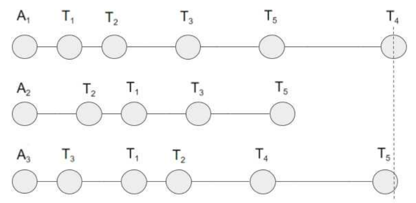
Pruning for least costly path: Given the objectives in Definitions 2 & 5, the algorithm iteratively prunes the path of the costliest agent obtained from line 4 (shortest tour on the *Mission-Graph*) to find a schedule with the minimum cost of scheduling. Central to the pruning step is the adherence to the cardinality C_j for each target. This is outlined in Lines 5–17. The pruning begins by selecting the costliest agent indexed by k (the agent that traverses the longest path) and choosing the farthest target (indexed by l) that the agent k visits, as indicated in lines 6 and 8 respectively. Line 9 checks for the condition if the cardinality of this target, $\mathcal{C}[l]$ has been fulfilled by other agents visiting it prior to the agent k. The variable $\mathcal{X}$ keeps track of the number of visits for each target, which is initialized in line 2 with the maximum number of visits possible for each agent, n ($max_{\forall j}(Cj) < n$ in Definition 5). It is decremented by one in line 16 every time a target is removed and the path is pruned to minimize the cost of scheduling. The intuition behind this approach is that by removing this redundant node (cardinality already fulfilled) from $\mathcal{P}[k]$, it produces a local minima for the overall time taken among all agents. The condition in line 9 is checked for each target in $\mathcal{P}[k]$ and after all the redundant paths are removed, the shortest route among the remaining targets in $\mathcal{P}[k]$ is computed again using Dijkstra's algorithm, followed by finding the shortest tour by solving the TSP [14] in lines 14 and 15 respectively and the visits count variable $\mathcal{X}$ is decremented. The reason for recomputing Dijkstra's algorithm and the shortest tour in



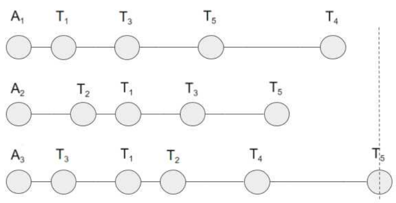
(a) City map with 3 agents, 5 targets with different cardinality and edge weights



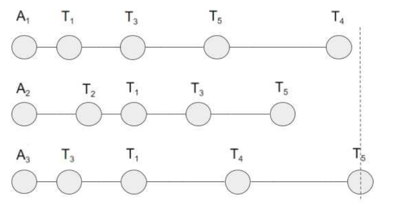
(b) Iter 1: Initial Path Estimate: A_2-costliest agent, T_4-farthest redundant target



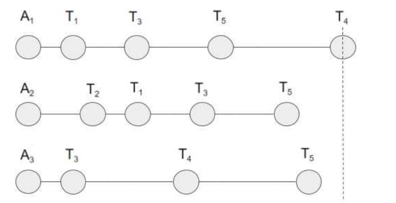
(c) Iter 2: Remove T_4 from A_2's path. A_1-costliest agent, T_2-farthest redundant target



(d) Iter 3: Remove T_2 from A_1's path. A_3-costliest agent, T_2-farthest redundant target



(e) Iter 4: Remove T_2 from A_3's path. A_3-costliest agent, T_1-farthest redundant target



(f) Iter 5: Remove T_1 from A_3's path, A_1-costliest agent with all cardinality fulfilled

Fig. 5: Example illustration of Algorithm 3 with 3 agents and 5 targets (T_1 - T_5 with cardinality $\{2, 1, 3, 2, 3\}$ respectively). In each step the costliest agent (longest travelling agent) is identified and the redundant target (cardinality already fulfilled) is removed. After 5 iterations, Agent A_1 is the costliest. This cost is normalized with the diameter of the graph used a cost metric for evaluation in §VI. The algorithm also provides the *best* schedule for the remaining agents.

lines 14 and 15 is to ensure that once a node is removed from the current TSP tour, the weight of the new edge between the nodes immediately prior and after the one removed *may not* be equal to the sum of the two edges prior to the removal. In other words, if $a \to b \to c$ is a TSP tour and edge b is removed in line 10 then $w(a, c) \neq [w(a, b)+w(b, c)]$. Although the inequality strictly depend on the graph G (city map), it cannot be ascertained a priori and hence line 14 and 15 ensures that the final schedule is always has the minimum cost. This process (lines 5-17) is repeated until the cardinality is fulfilled for all targets in $\mathbb{t}$ (line 5) and the last calculated shortest tour given by line 15 is the final schedule for the agents, returned as $\mathcal{P}$ along with the final cost of scheduling in line 18.

Example Illustration of Algorithm 3: Figure 5 shows an example of the iterative evolution of Algorithm 3. Figure 5a shows a simple city graph, G with edges representing the roads along with 5 targets and 3 agents located at the intersection of these roads. The cardinality of the targets $T_1 - T_5$ is $\{2, 1, 3, 2, 3\}$ respectively. After computing the shortest tour in the *Mission-Graph* for all agents in lines 3 and 4, the costs (length of the lines) and the resultant paths (order) for each agent are indicated in Figure 5b. In the first iteration, agent A_2 travels the longest to cover all the targets and is identified as the costliest agent. The farthest target in A_2's path is T_4. As T_4 has a cardinality of 2, requiring only 2 of the 3 agents to visit, it is considered to have a redundant visit in A_2's path. In other words, T_4 can be visited in shorter time by the other two agents and fulfill the cardinality of 2. Hence, removing this redundant and costly target T_4 from agent A_2's path (as per lines 9 – 11) reduces the cost of A_2's path while fulfilling the cardinality for all the targets. The new path for A_2 and its cost is determined from the new Mission-Graph (without T_4) as per line 14 and 15.

Figure 5c shows the paths and the costs of the agents at the beginning of Iteration 2. In the second iteration, A_1 is determined to be the costliest agent and T_4 as the farthest target. However, as T_4 has a cardinality of 2 and has two agents visiting it (including A_2). So, this is not considered as a redundant visit. So, the algorithm continues to look for the farthest target in A_1's path that has redundant visits, until it detects T_2 as the farthest redundant visit, which is removed from A_1's path. Note, T_5 and T_3 in A_1's path, both require all three agents to visit as they have cardinality of 3, hence those two nodes cannot be removed from A_1's path. The updated path and its corresponding cost is shown in figure 5d. Similarly, in the third iteration, A_3 is the costliest agent and T_2 is the farthest redundant target in its path (removing other nodes will not meet the cardinality for those). While the removal of T_2 does not improve the cost, as T_4 can only be reached via T_2, it should be noted that removal of any targets and the corresponding edges does not increase the cost (due to the triangular rule governing Euclidean graphs [14]). In Iteration 4, shown in figure 5e, A_3 is still the costliest agent, and T_1 is the farthest redundant target because its cardinality can be met in shorter time by the other two agents. Consequently, T_1 is removed from A_3's path and after recomputing the new schedule and the path cost the final schedule is obtained as shown in in figure 5f. The cost of scheduling for this graph is the total cost of the A_1's path because that is the minimum time required to visit all the targets while fulfilling the cardinality.

B. Analysis of Algorithm 3

We show that the Schedule is NP-hard, hence there is no optimal solution in polynomial time.

Claim 1. *The Schedule is NP-hard.*

Proof: Consider a subproblem of the Schedule, with 1 agent having to visit all the targets, with all the targets having a cardinality of 1. This is equivalent to solving the TSP for that agent. Since, the TSP is NP-hard and it is a special case of the Schedule, it is inferred to be at least NP hard. ■

TABLE I: Notations used in §V-D

Notation	Interpretation
i	Agent i, $i \in \{1, 2, ..., n\}$
t_k	Target j, $j \in \{1, 2, ..., m\}$
P_i	Path of Agent i returned by Algorithm 3
P_i^*	Path of Agent i returned by OPT
l_i	Cost of Agent i obtained by Algorithm 3
l_i^*	Cost of Agent i obtained by OPT
$l_i(t_k)$	Increase in cost of agent i by adding target t_k in Algorithm 3
$l_i^*(t_k)$	Increase in cost of agent i by adding target t_k in OPT
T_x^i	Set of targets visited by both P_i and P_i^*
T_y^i	Set of targets visited by P_i, but not by P_i^*
T_z^i	Set of targets visited by P_i^*, but not by P_i

C. Complexity of Algorithm 3

In absence of an optimal algorithm, Algorithm 3 yields a solution for the Schedule in polynomial time.

Lemma 1. *Algorithm 3 has complexity of $O(nm^4)$, where n is the number of agents and m is the number of targets in G.*

Proof: Since the cardinality of the targets is fixed, the number of iterations of Algorithm 3 is bounded by a fixed number. The algorithm is initiated with all agents visiting all targets ($\mathcal{X}$ in line 3) and executes until each target is visited by a number of agents equal to its cardinality. Each iteration removes one redundant visit from the costliest agent (as shown in Figure 5). So, for each target T_j, the algorithm executes $(n - C_j)$ times and therefore, the total iterations in Algorithm 3 for all the targets is,

$$(n - C_1) + (n - C_2) + ... + (n - C_m) = m.n - \sum_{i=1}^{m} C_i \quad (4)$$

Recall that, $n < m$ and $max_{\forall i \in m}(C_i) \leq n$. Hence,

$$\sum_{i=1}^{m} C_i \leq m.n \implies m.n - \sum_{i=1}^{m} C_i \geq 0 \quad (5)$$

The 3/2-approximation for TSP [14] used in Algorithm 3 has a complexity of $O(m^3)$. Since, the TSP is computed once in every iteration, the complexity of Algorithm 3 is,

$$=O(m.n - \sum_{i=1}^{m} \mathcal{C}_i).O(m^3) \quad = O((m.n - \sum_{i=1}^{m} \mathcal{C}_i).m^3)$$

$$=O(n.m^4 - m^3.\sum_{i=1}^{m} \mathcal{C}_i)$$

$=O(n.m^4)$ From (5)

Hence, Algorithm 3 has a complexity of $O(nm^4)$. ■

Note: The 2-approximate solution of TSP based on the Minimum Spanning Tree (MST) of the corresponding graph [15], has a complexity of $O(m.log(m))$. Using such an implementation in Algorithm 3, the complexity can be improved to $O(n.m^2 log(m))$. Using the MST has the added advantage of solving the Schedule for non-metric graphs, such as in the presence of traffic the costs of edges are no longer just a function of distance. This problem is out of scope of this work and will be investigated in future.

D. Approximation Ratio for Algorithm 3

In absence of a polynomial time, optimal solution for the Schedule, an approximation ratio is a bound, which guarantees that any solution from Algorithm 3 is always within a constant factor of the solution from an optimal algorithm. In other words, using the notations in Table I, if agent p is the costliest in Algorithm 3 and agent q is the costliest in the optimal algorithm, then $l_p \leq 3.l_q^*$ is provably correct.

Let, OPT be the optimal algorithm for the Schedule problem that returns the paths $P_i^*, \forall i \in A$, with minimum cost among all the possible paths that fulfills the cardinality of the targets. It is to be noted, that we do not make assertions on the design of OPT except to acknowledge that a Minimum Spanning Tree (MST) can be constructed from the targets in any optimal path $P_i^*, \forall i \in A$, similar to deriving a solution of the TSP problem (used in Algorithm 3) from a corresponding MST. Also, $P_i, \forall i \in A$, in Algorithm 3 (computed in lines 6–21) can be obtained using the round-trip MST of the Mission-Graphs ($\bar{G}_i$) instead of the 3/2-approximate TSP approach [14]. Under such implementation, we observe that if $T_y^i = 0$, i.e. targets in $P_i \subseteq$ targets in P_i^*, then by the construction of MST [16] we observe Property 1.

Property 1. *If $T_y^i = 0$, then l_i is <u>no worse</u> than twice the optimal cost l_i^*. i.e, $l_i \leq 2.l_i^*$.*

Furthermore, the following properties can be observed based on the design of Algorithm 3 and the definition of OPT.

Property 2. *Since, Algorithm 3 and OPT both return the costliest paths among all the agents (say l_p and l_q^*), the paths travelled by any other agent, <u>must not</u> be costlier than l_p or l_q^*. Thus, for any agent $i \in A$ we have, $l_i \leq l_p$ for Algorithm 3 and $l_i^* \leq l_q^*$ for OPT.*

Property 3. *In Algorithm 3 and OPT, all targets must be visited by the same number of agents (Definition 2 in §V).*

Property 4. *If a target t_k is removed from an agent i's path, it must have been the costliest path at some prior iteration of the algorithm (line 8–15). So, if agent p is the costliest agent at the end of the algorithm, the increase in agent i for visiting t_k <u>must be</u> such that $l_i + l_i(t_k) \geq l_p$.*

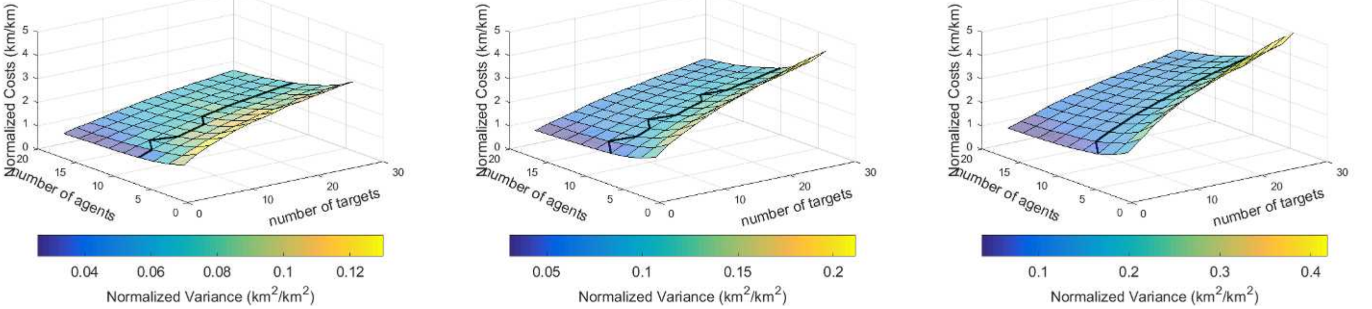
Property 5. *From Table I, we can express the costs l_i and l_i^* of agent i as,*

$$l_i = l_i(T_x^i) + l_i(T_y^i)$$
$$l_i^* = l_i^*(T_x^i) + l_i^*(T_z^i)$$

Theorem 1. *Algorithm 3 is 3-approximation for the Scheduling Problem.*

Proof Overview: Let the costliest paths returned by Algorithm 3 and OPT be l_p and l_q^* respectively. Our goal is to find a relationship between these two quantities, by first establishing an inequality between the costs of the same agent in Algorithm 3 and OPT, and then using the inequality and the properties to relate the costs of different agents in the two algorithms. This result is used to relate the costs of agents which have non-overlapping targets in the paths obtained from Algorithm 3 and OPT. We consider two cases: 1) The targets in $P_p \subseteq$ the targets in P_p^* and 2) The targets in $P_p \nsubseteq$ the targets in P_p^*.

Due to space constraint, the complete proof of the approximation ratio is provided as an anonymous external document at <u>www.dropbox.com/s/xa7yzce5z0s1m4q/Proof.pdf?dl=0</u> .



(a) Normalized cost metric in London (b) Normalized cost metric in NYC (c) Normalized cost metric in Paris

Fig. 6: Normalized cost metric for **Average Cardinality = 3** for (a) London (b) NYC and (c) Paris.The dark line highlights the points beyond which the cost variation is below 10%. The variance is indicated using the color scale.

VI. Performance Evaluation

For practicality, the algorithm was evaluated in three prominent cities: New York City (NYC), Paris and London, primarily to understand the performance on different graphs, with the roads mapped as edges, and the intersections to vertices. To compare the outcome of Algorithm 3 among different cities, the cost metric (Definition 1) is normalized by the diameter of the graph and its unit is represented as km/km. For a rectangular area, the diameter is simply the diagonal. Figure 3c shows the cost metric and paths for one instance in NYC where 5 agents are routed using Algorithm 3 among 10 targets with different cardinality. To further investigate the Scheduling performance, we perform a parameter space analysis on larger geographical area and more agents and targets.

A. Parameter Space Analysis

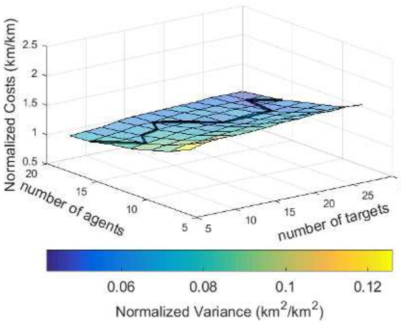
For each graph (city) the number of agents (n), the number of targets (m), the location of agents (a), the location of targets (t) and the cardinality of targets (C_j) were varied and the effect on the paths (P_i) and the costs (l_i) of the agents were recorded. The location of agents and targets were chosen randomly among the available nodes in the graph.The agents are varied from 4 to 20 and number of targets from 4 to 30 and executed 500 unique arrangements of agents and targets. The cardinality was varied from 1 to n ($1 \leq C_j \leq n$). However, the distribution of C_j is controlled in two ways: (1) Constant average cardinality and (2) Decreasing the average cardinality with increasing number of targets (by ensuring a Constant number of total visits across all the agents).

1) Constant Average Cardinality: The cardinality was distributed among the targets such that an average cardinality of 3 is maintained across all agents. This ensures that even with increasing number of agents, the average number of visits required for the targets is 3, such that for a fixed number of violations the cost reduces as we deploy more agents. Figure 6 shows the mean and variance of the normalized cost metric for the three cities. The fixed average cardinality justifies the drop in cost observed when more agents are deployed. The cost increases with increasing violations, since the total number of visits required also increases linearly as the average cardinality is fixed. However, it is observed that the rate of reduction in the cost metric drops with increasing number of agents, denoted by the dark line, which shows the 10% reduction in the cost metric for different number of targets. This line indicates a boundary for cost-effective enforcement as adding more agents does not lead to substantial reduction in the cost metric. Further, the variance of the cost metric (the color axis) increases with the number of targets and drops with the number of agents. This is influenced by the fact that, as there are more infractions, the potential of having diverse target distributions (such as clusters or well separated targets) increases resulting in a larger variation. Figure 6 also reveals that the algorithm performs statistically similar with respect to the mean and variance of the cost metric among the cities regardless of the attributes of the city maps. However, closer inspection indicates that Paris portrays a slightly larger cost followed by New York and London. This behaviour is attributed to the features of the road network in these cities. In Paris, the agents have to travel via the central hub to cover the targets. This feature contributes to a higher travel time. In comparison, NYC has highly connected, grid-like road systems with plenty of connectivity between the targets, resulting in a relatively lower cost metric compared to Paris. It is interesting to observe that in London and New York the 10% line is located between 10-12 agents and for Paris between 8-10, suggesting that in London and New York the costs can be further improved by increasing the number of agents compared to Paris.

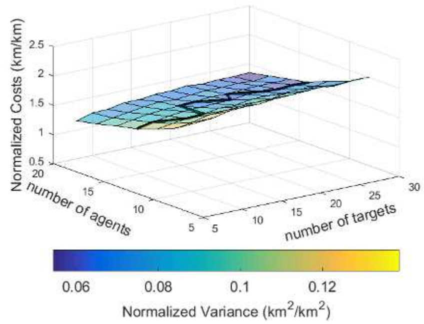
2) Constant Total Number of Visits: The total number of visits was fixed at 40 to ensure that even with increasing number of targets the total number of visits required by all the targets combined is limited to 40, such that the average cardinality reduces as the number of targets increases, limiting the growth in the cost. Since, the total number of visits is constant and the average cardinality reduces with the number of targets, it is observed that the change in cost with the number of targets was minimal compared to the previous case. Practical implications of limiting the total number of visits include situations where the cost must be maintained at a specific value even with increasing violations. This is a metric that is controlled by the Dispatch to conserve enforcement resources. The variation of the mean normalized cost metric in Figure 7 display a similar pattern with minuscule increase from London to Paris. However, it is observed that the variance decreases with increasing number of targets unlike in the previous case. This is because the lesser the targets the higher is the average cardinality introducing more variation in the paths of agents for lesser number of violations.

B. Overall System Performance

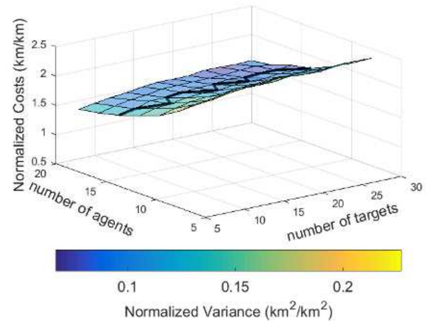
The overall performance metrics for Algorithm 3 is shown in Figure 8 for an average cardinality of 3. Figure 8a indicates



(a) Normalized cost spread in London



(b) Normalized cost spread in NYC

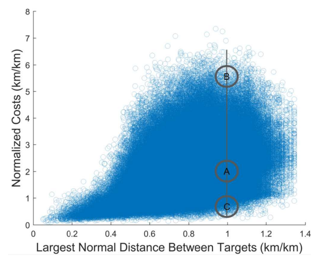


(c) Normalized cost spread in Paris

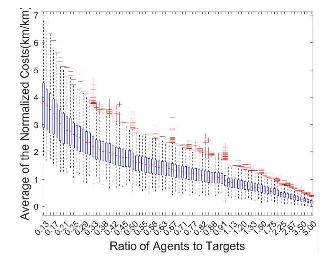
Fig. 7: Normalized cost metric for **Total Visits = 40**. The dark line highlights the points beyond which the cost variation is below 10%. The variance is indicated using the color scale.

increasing cost metric with increasing separation of targets, confirming its influence on the cost. The large variation of cost observed at the same target separation, is due to the dependence of the cost metric on the distribution of the targets. Among the regions highlighted on Figure 8a at a target separation of 1, region A exhibits expected behaviour, where the cost metric is comparable to the separation of the targets. Region B has an unusually high cost metric due to widely distributed and hostile violations (high cardinality) with few available agents. On the contrary, region C portrays a much lower cost than the separation, which occurs when there are more agents than targets that are clustered within a small area.

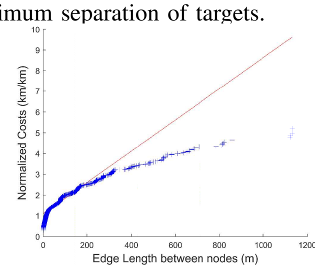
The variation of the average cost metric with the ratio of agents to targets shown in Figure 8b, confirms that deploying more agents for the same number of infractions, decreases the cost. The plot shows the ability of the algorithm to scale as the trend of the cost is similar even for larger systems of agents and targets. i.e if the hostility in an environment increases, increasing the number of agents by the same factor will ensure similar cost performance. The figure also shows the ability of the algorithm to perform load balancing, which is naturally guaranteed by the algorithm as it prunes the path of the costliest agents, improving the balance among agents, ensuring that no agent is overworked. The behaviour of the cost metric with the attributes of the city graph is depicted by the QQ plots. Figure 8c compares the distribution of the cost metric with that of the length of the edges in the city graph. The distribution of the cost and the average distance between the targets are highly correlated as shown in figure 8d. Results from London and Paris exhibits similar behaviour that follows the trends observed in figures 6 and 7.



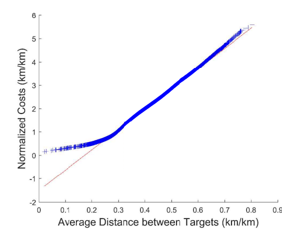
(a) Variation of Cost with maximum separation of targets.



(b) Variation of Cost with ratio of agents to targets



(c) Cost vs edge length



(d) Cost vs average inter-target distance

Fig. 8: Comparison of the distribution of Normalized Cost Metric for NYC with that of (a) Edge lengths and (b) Average Distance between Targets.

VII. Related Work

The Scheduling problem has its roots in Multiagent planning (MP) which is the NP-hard problem [17] of finding the shortest paths of agents with targets visited at least once. The MP problem has been approached using parallel graph algorithms [18], Integer-Linear programming (ILP) algorithms [19] and A* search algorithms with guarantees [20]. In general these methods do not enforce unique visits at the target. In the context of Multiagent Planning with uniqueness (MPU), an intriguing class of problems is the Multiple Traveling Salesmen Problem (MTSP) which finds closed tours for agents, while enforcing uniqueness. MTSP is challenging due to its combinatorial nature and NP-hardness [21]). Approaches to solve the MTSP include the transformation into an equivalent TSP and solving by exact or approximate algorithms [7], or solving the MTSP directly using Linear Programming or exact methods [22]. These methods do not solve the MPU directly as they yield tours and not paths for agents. In [23] a heuristic search method is proposed to solve the Multiagent Path Finding problem, which is similar to MPU, except that the endpoints of the tours are also fixed. [21] presents a Genetic Algorithm Inspired Descent (GAID) method for solving the MPU. For detecting and localizing infractions we require multiple agents visiting each infraction to address its heterogeneity (Cardinality). An exact method for heterogeneous MTSP (some targets can be visited only by a specific agent) is provided in [22]. However, the class of MTSP does not addresses the notion of multiple visits. In the general MP problem targets aren't restricted to a single visit, and may incur multiple visits, but no control is enforced on number of visits. An interesting class of problems here is the Vehicular Routing Problems (VRP/MDVRP) [8], which might facilitate multiple visits [24], however does not ensure uniqueness or constraints on number of visits. A class of problems in VRP deal with some notion of heterogeneity (some targets can be visited only by a specific agent), for which a 8-approximation algorithm is presented in [25]. However, these approaches are based on graph partitioning, where imposing cardinality constraint is challenging. Multiagent patrolling problems [26] enable targets to be visited multiple times, however by the same agent. Under the constraint of Cardinality, MP problem evolves to Multiagent Planning with Cardinality (MPC) problem. To the authors best knowledge Multiagent planning problem with Cardinality and the notion of using the crowd as eyewitnesses to efficiently deploy agents to improve the enforcement of spectrum policies, is unprecedented in literature.

VIII. Conclusion

In this paper, we architected and analyzed a solution for the MPC problem which consists of algorithms to derive the near-optimum cardinality of the targets and to compute schedule for all the agents to fulfill the cardinality of the targets in the shortest possible time. This contribution is a complementary and necessary precursor to advance signal processing for enforcing spectrum etiquette using mobile, autonomous agents. Through simulations and analysis, we draw four firm conclusions: 1) The autonomous agents are able to detect and localize targets with higher accuracy than a purely crowdsourced regime. 2) The scheduling algorithm is polynomial and provides the shortest paths for the agents while conforming to the cardinality requirement, 3) The scheduling algorithm has a provable bound of 3-approximation ratio. 4) The scheduling algorithm exhibits strong generality across different geographical regions, by producing statistically similar results for varying degree of violations. While we await practical system implementation, the encouraging results from this work lay the foundation towards adopting a real-time, autonomous enforcement system for spectrum policies.

References

[1] A. Dutta and M. Chiang, "See something, say something, crowdsourced enforcement of spectrum policies," *IEEE Transactions on Wireless Communications*, vol. 15, no. 1, pp. 67–80, Jan 2016.

[2] Z. Li, A. Nika, X. Zhang, Y. Zhu, Y. Yao, B. Y. Zhao, and H. Zheng, "Identifying value in crowdsourced wireless signal measurements," in *WWW*, 2017.

[3] A. Nika, Z. Zhang, X. Zhou, B. Y. Zhao, and H. Zheng, "Towards commoditized real-time spectrum monitoring," in *Proceedings of the 1st ACM Workshop on Hot Topics in Wireless*, ser. HotWireless '14. New York, NY, USA: ACM, 2014, pp. 25–30. [Online]. Available: http://doi.acm.org/10.1145/2643614.2643615

[4] D. Yang, G. Xue, X. Fang, and J. Tang, "Crowdsourcing to smartphones: Incentive mechanism design for mobile phone sensing," in *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, ser. Mobicom '12. New York, NY, USA: ACM, 2012, pp. 173–184. [Online]. Available: http://doi.acm.org/10.1145/2348543.2348567

[5] T. Fawcett, "An introduction to roc analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861 – 874, 2006, rOC Analysis in Pattern Recognition. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S016786550500303X

[6] P. Parkinson, Bradford W., Spilker, James J., Jr., Axelrad, Penina, Enge, *Global Positioning System, Volume 1 - Theory and Applications*. American Institute of Aeronautics and Astronautics, 1996. [Online]. Available: http://app.knovel.com/hotlink/toc/id:kpGPSVTA03/global-positioning-system/global-positioning-system

[7] B. Gavish and K. Srikanth, "An optimal solution method for large-scale multiple traveling salesmen problems," *Operations Research*, vol. 34, no. 5, pp. 698–717, 1986. [Online]. Available: http://www.jstor.org/stable/170727

[8] V. Pillac, M. Gendreau, C. Guret, and A. L. Medaglia, "A review of dynamic vehicle routing problems," *European Journal of Operational Research*, vol. 225, no. 1, pp. 1 – 11, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221712006388

[9] J. Xiong and K. Jamieson, "Arraytrack: A fine-grained indoor location system," in *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, ser. nsdi'13. Berkeley, CA, USA: USENIX Association, 2013, pp. 71–84. [Online]. Available: http://dl.acm.org/citation.cfm?id=2482626.2482635

[10] B. C. Levy, *Principles of Signal Detection and Parameter Estimation*, 1st ed. Springer Publishing Company, Incorporated, 2010.

[11] V. Kumar, J.-M. Park, and K. Bian, "Blind transmitter authentication for spectrum security and enforcement," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '14. New York, NY, USA: ACM, 2014, pp. 787–798. [Online]. Available: http://doi.acm.org/10.1145/2660267.2660318

[12] D. M. W. Powers, "Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.

[13] OpenStreetMap, "OpenStreetMap - http://www.openstreetmap.org." [Online]. Available: http://www.openstreetmap.org

[14] N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem," Graduate School of Industrial Administration, Carnegie Mellon University, Technical Report 388, 1976.

[15] R. C. Prim, "Shortest connection networks and some generalizations," *The Bell System Technical Journal*, vol. 36, no. 6, pp. 1389–1401, Nov 1957.

[16] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*, 3rd ed. The MIT Press, 2009, pp. 1111–1115.

[17] P. Fraigniaud, L. Gasieniec, D. R. Kowalski, and A. Pelc, "Collective tree exploration," *Networks*, vol. 48, no. 3, pp. 166–177, 2006. [Online]. Available: http://dx.doi.org/10.1002/net.20127

[18] D. Pajak, "Algorithms for deterministic parallel graph exploration," Ph.D. dissertation, Universite Sciences et Technologies - Bordeaux I, September 2014.

[19] J. Yu and S. M. LaValle, "Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1163–1177, Oct 2016.

[20] K. C. Wang and A. Botea, "MAPP: a scalable multi-agent path planning algorithm with tractability and completeness guarantees," *CoRR*, vol. abs/1401.3905, 2014. [Online]. Available: http://arxiv.org/abs/1401.3905

[21] G. G. Tamas Kalmar-Nagy and B. D. Bak, "The multiagent planning problem," *Complexity*, vol. 2017, p. 12, 2017. [Online]. Available: https://doi.org/10.1155/2017/3813912

[22] K. Sundar and S. Rathinam, "An exact algorithm for a heterogeneous, multiple depot, multiple traveling salesman problem," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2015, pp. 366–371.

[23] G. Sharon, R. Stern, M. Goldenberg, and A. Felner, "The increasing cost tree search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 195, pp. 470 – 495, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0004370212001543

[24] K. Sundar, S. Venkatachalam, and S. Rathinam, "Formulations and algorithms for the multiple depot, fuel-constrained, multiple vehicle routing problem," *CoRR*, vol. abs/1508.05968, 2015. [Online]. Available: http://arxiv.org/abs/1508.05968

[25] S. K. Yadlapalli, S. Rathinam, and S. Darbha, "An approximation algorithm for a 2-depot, heterogeneous vehicle routing problem," in *2009 American Control Conference*, June 2009, pp. 1730–1735.

[26] J. Czyzowicz, L. Gasieniec, A. Kosowski, and E. Kranakis, *Boundary Patrolling by Mobile Agents with Distinct Maximal Speeds*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 701–712.