Online Allocation of Sensing and Computation in Large Graphs

Xinlin Li†, Merve Karakas†, Osama A. Hanna†, Mehrdad Kiamari*, Jared Coleman*,
Christina Fragouli†, Bhaskar Krishnamachari* and Gunjan Verma‡
†University of California, Los Angeles * University of Southern California ‡ DEVCOM Army Research Laboratory
Email:{xinlinli, mervekarakas, ohanna}@ucla.edu, {kiamari,jaredcol}@usc.edu,
christina.fragouli@ucla.edu, bkrishna@usc.edu, gunjan.verma.civ@army.mil

***Abstract*—We consider vehicle tracking over a large territory equipped with sensors and computational units, and propose online resource allocation algorithms that decide which sensor nodes to activate, and on which computational units to perform the corresponding tracking tasks. We show through numerical evaluation that our approach can notably outperform state of art algorithms in terms of delay, communication cost, and the number of required sensor measurements.**

I. Introduction

A myriad of defense and civilian applications center around vehicle tracking, e.g. for search-and-rescue operations; towards this end, work has looked at accurately identifying trajectories [10], [18], use of neural network approaches [4], predicting directions of movement [12], and combining multimodal data for tracking [18]. However, most of these works assume the availability of sensor measurements that cover the vehicle to track, as well as the availability of computational resources to support the needed computations for tracking. In this paper, we ask: how do we decide which sensor nodes to activate, and on which computational units to perform the corresponding tracking tasks?

We consider the following abstract scenario: A large number of sensor nodes (e.g., acoustic sensors) and comparatively sparse computational units (CUs) have been deployed over a given territory to help track target vehicles. The sensors collect data and communicate it to nearby CUs, depicted in Fig. 1, for processing and tracking. Clearly, to track vehicle(s) crossing a territory, all sensors could actively transmit their measurements at all times to all CUs; however, this rapidly depletes the sensor batteries and may lead to communication bottlenecks. Moreover, sensors' measurements are informative only when there is a target vehicle in their vicinity. Similarly, due to other computational tasks, all CUs may not be able to support the processing of all sensor data at all times. Ideally, we would like some nearby CUs to be available to collect and process data from informative sensors to estimate the vehicle's position. Based on these observations, our goal is to maximize the accuracy of tracking target vehicles, using minimal sensing and communication resources.

This work was supported in part by Army Research Laboratory under Cooperative Agreement W911NF-17-2-0196.

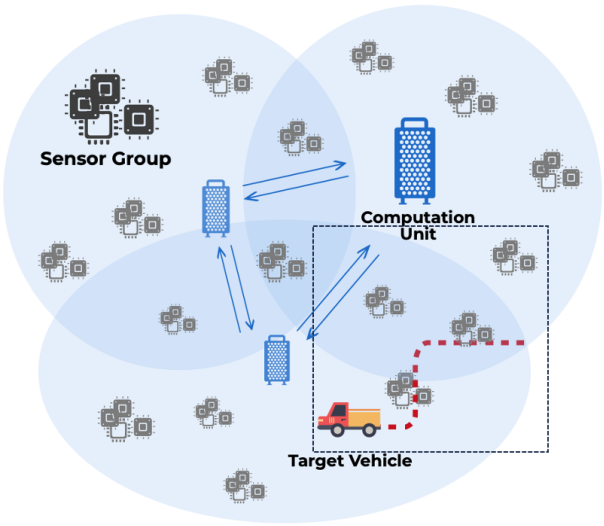

Fig. 1. System model.

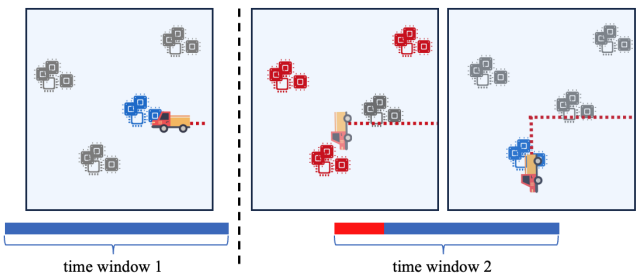

Fig. 2. Sensor selection example. Red represents candidate sensors to activate at the current time window, while blue represents the activated one for tracking.

The example depicted in Fig. 2 illustrates our setup. We assume operation in time windows. At time window 1, a target vehicle is close to the blue sensor. At time window 2, the vehicle may move in a direction unknown to us. Therefore, we want to take measurements from the current as well as the colored in red neighbor sensor (i.e., all sensors the vehicle may be close to) to decide which sensors to activate over time window 2. However, at this stage, heavy processing on these measurements is undesirable (e.g. run the tracking algorithm

on each) both because of delay and computational constraints; we want to use some simple measures (eg., SNR for acoustic sensors) to make our decision. Note that one measurement from each sensor is not sufficient: sensor measurements are noisy, and a direct comparison of individual measurements from each sensor is likely to lead to errors. Thus, we want to take a small number of simple measurements to identify the sensor to activate in the current time window. Furthermore, our performance depends on the allocation of sensor data to available computational nodes, to achieve timely processing of the data. The processing task may consist of, for example, running suitable sensor signal processing algorithms or ML models to detect, locate, and classify the target.

This paper proposes an approach to solve the following online and coupled resource allocation problem: allocate sensing, communication, and computational resources, in an online manner, such that the a priori unknown trajectory of a vehicle is tracked as accurately as possible, in a manner we make precise in Section III. To identify the the best sensor to use with a small number of sensor measurements, (see also Section II), we propose an algorithm that combines maximum a posteriori (MAP) detection with elimination, and we show through extensive simulation results that it outperforms state-of-the-art online learning algorithms such as UCB [1], [9] and best arm identification algorithms such as T3C [14] across metrics that capture communication cost and delay. We also utilize online reservation techniques to ensure that CUs who may be close to the (unknown in advance) vehicle trajectory are available to support our task.

II. Related Work

In the following we discuss the main used techniques and their positioning within related work; a detailed description of the system model and notation are provided in Section III-B2.

Online learning: Online learning algorithms, in particular, multi-armed bandits (MAB), have gained significant attention as they enable modeling real-time decision-making under uncertainty. In an MAB setup, an agent is faced with a set of actions, often referred to as arms, each associated with an unknown reward distribution. The agent sequentially selects an action to play and receives a reward, and it aims to maximize its cumulative reward over time.

Perhaps the most well-known algorithms in this field are the Upper Confidence Bound (UCB) algorithm [1], [9] and Thompson sampling [15]. UCB [1], [9] maintains confidence bounds on the estimated rewards of each arm. Thompson sampling takes a Bayesian approach to utilize prior beliefs and maintains a posterior distribution over the reward parameters of each arm. Both algorithms and their variations have been extensively utilized in different applications as they achieve near-optimal theoretical guarantees.

A subset of multi-armed bandit algorithms, pertinent to our study, named best arm identification (BAI) algorithms, introduced by [3], focuses on determining the arm with the highest expected reward. Unlike multi-armed bandit algorithms, BAI algorithms prioritize identifying the optimal arm with a limited number of samples. While there are multiple settings, Bayesian-based BAI algorithms appear to be most related to our case which takes into account prior information on the arms. [13] proposes Bayesian-based algorithms and proves they have an asymptotically optimal posterior convergence rate; these algorithms introduce randomization to obtain a second candidate for sampling; hence, they provide more exploration compared to the original Thompson sampling. [14] proposes T3C to alleviate the computational burden problem of [13] while preserving the theoretical guarantees.

As our main aim is to correctly identify the best sensor to use with a small number of sensor measurements, the sensor activation problem can be modeled as a best-arm identification problem. The act of sampling the measurements from a sensor corresponds to pulling an arm, the number of arms is determined by the number of currently considered sensors, and the reward is chosen to be a function of the sensor measurement metric. All algorithms mentioned above can be used to solve this problem; however, they do not exploit the problem structure fully, e.g., in the current setup, it is possible to pull more than one arm at each time and a pulled arm gives information about other arms.

Computation scheduling: The problem of allocating a complex application represented by a task graph (a directed acyclic graph, or DAG, where nodes represent tasks and directed edges represent precedence constraints between tasks) to a network of computers has been explored recently in the context of dispersed computing for various tactical applications [6], [11]. Complex scheduling algorithms such as the classic HEFT scheduler [16] and throughput-optimized alternatives [7], [17] are needed to optimize for objectives such as minimal makespan and maximal throughput in the case of a general task graph. Recent work has shown that machine learning techniques such as graph convolutional networks can be used to significantly speed up the scheduling process, which yields benefits in dynamic networks [5], [8]. In this work, as a starting point we consider a relatively simple task graph consisting of the sensor data being sent to a single compute point at each time for processing. In this case, the total latency incurred is a sum of the communication and computation cost. If all compute points are homogeneous, it suffices to optimize for the communication cost when performing task placement (since the computation cost is the same on all nodes), as we consider in section III. To our knowledge, no prior work has explored reserving computation points in advance of task allocation using target movement predictions, which we explore in this paper.

III. Problem Formulation

A. Task Overview

We assume that time is divided into T time windows, where each time window contains multiple timeslots and τ timeslots are used for sensor selection. Given a set of sensor and CUs in a territory (to be described in more detail later in this section), we follow the timeflow depicted in Fig. 3: in (Block A) we

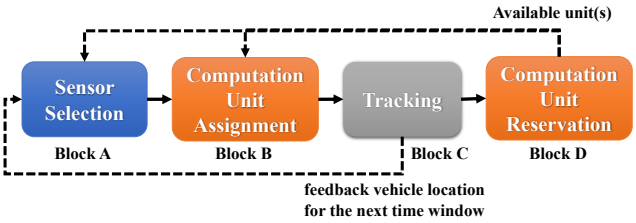

Fig. 3. Workflow in one time window.

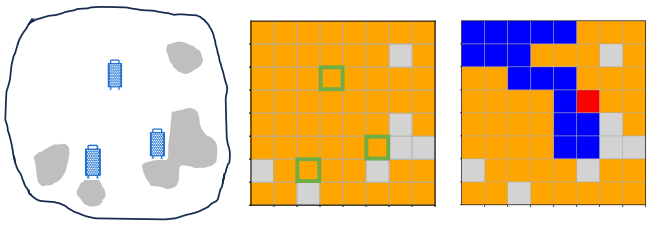

Fig. 4. Example of territory modeling and trajectory. Gray represents blocked areas, green represents locations of compute units, blue marks the historical trajectory, and red represents the current location of the vehicle.

decide which sensor to activate using as input the estimated current location of the target vehicle, the availability of CUs and online samples of measurements; in (Block B) we decide to which CU to allocate the task; in (Block C) we track the current position of the vehicle using sensor measurements; in (Block D) we potentially reserve CUs in anticipation of future movements of the vehicle; and with (Feedback) we provide to the sensor selection the correct current position of the vehicle as well as the CUs availability.

B. System Model

We here provide the system model that we use to the abstract territory, sensing and computational resources, and target vehicle movement. We believe this to be a reasonable model for evaluation, yet, we note that our proposed algorithms are not restricted to it.

1) Territory modeling: As depicted in Fig. 4 we abstract the territory using a grid-based representation, where each square contains a sensor. Specifically, given a territory, we approximate it as an $M \times M$ grid denoted as G, where $G_{i,j}$ refers to the square in i^{th} row and j^{th} column; thus we have in total $N = M^2$ sensors. To capture realistic conditions, we assume that some squares are blocked (inaccessible), for instance, due to the territory morphology.[1]

2) Sensor modeling: We assume that we are provided with a partition (clustering) of the sensors into N clusters of sensors, and will opt to activate one such sensor at a time. For the sake of simplicity, we will use "sensor" to refer to a collection/cluster of sensing assets in the remainder of the paper. We use the sensor measurements for two goals: (i) sensor selection and (ii) vehicle tracking. For (i), we are going to use a fast-to-estimate (and low-cost to transmit) quality metric of the sensor measurements, such as SNR. We expect that such measurements can be significantly corrupted by noise. For (ii), we will assume that communication-expensive measurements are forwarded to a selected computational unit.

3) Computational Units (CUs) modeling: In our system, a subset of cells contains CUs, and we examine the parameter known as the $\frac{K}{N}$ ratio of CUs to sensor cells, where K and N represent the number of CUs and the number of sensors, respectively. Intuitively, the smaller K is, the more computationally constrained the problem will get.

We assume a random process governing the background activity of CUs, where a unit is either available (1) or not (0). The simplest independent and identically distributed (iid) model for availability suggests that at each time step, each unreserved unit is available with probability p. We also assume that during a current time-window we can "reserve" a CU for the next time-window; in this case, any reserved unit will be available with probability 1.

4) Movement model: In each time window t, we consider a vehicle that can move to one of the adjacent squares $\mathcal{L}_t = \{l_i\}_{i=1}^{L}$ (if it is not blocked), and there are L possible movements in total. These movements do not need to happen with the same probability: for instance, the target vehicle may be more inclined to move straight forward rather than changing its direction, while it may be less likely to return to a previously visited square. We assume that conditioned on the history there is a probability associated with each possible movement, denoted as $\mathcal{P}_t = \{p_i\}_{i=1}^{L}$, and $\sum_{p\in\mathcal{P}} = 1$. We allow these probabilities to change over time based on the historical trajectory. Fig. 4 (c) shows an example of a random trajectory.

Prior information: We assume that our algorithms have access to the movement probabilities $\mathcal{P}_t$ [2]. This allows to capture prior information, for instance, from observing past target vehicle trajectories or from knowledge of the territory.

C. Performance metrics

To measure the trade-off between tracking accuracy and resource utilization, we use the following metrics.

- **Accuracy.** Given the trajectory of a target vehicle within a square grid, we measure the accuracy of tracking as the percentage of times we correctly identify the vehicle location.
- **Communication Cost.** Given a trajectory and associated selected sensors, we measure the Euclidean distance between the activated sensors and the used computational point. In particular, the communication cost is $\frac{1}{T}\sum_{t=1}^{T} \|u^{(t)} - v^{(t)}\|_2$, where $\boldsymbol{v} = (v^{(1)}, v^{(2)}, \cdots, v^{(T)})$ is the positions of sensors activated at each time window t, and $\boldsymbol{u} =$

[1] We note that although we use this abstraction in our simulation results, our proposed algorithms are not specific to this model and can be applied over other territory models as well.

[2] When the movement probabilities are unknown, we simply assume a uniform distribution.

$(u^{(1)}, u^{(2)}, \cdots, u^{(T)})$ is the positions of activated CUs for corresponding time windows.

• **Sensing Delay and Cost.** Delay captures the number of timeslots needed to identify the sensors to use, and cost measures the number of arms pulled (samples collected). Note that during each timeslot, we could in parallel collect more than one sample from different sensor locations, under the assumption that sensors do not "interfere" (i.e., the activation of one sensor does not impact what is sensed by any others.)

IV. Proposed Approach

A. Sensor Activation

At each time window t, we consider an individual online learning detection problem as follows: the sensor selection algorithm receives the current tracking information, i.e., the location of the vehicle l_{t-1}, based on the measurements obtained in the previous time window, $t-1$. As the current location of the vehicle is known to the algorithm, the set of possible next positions can be inferred and denoted as $\mathcal{L}_t$. As vehicle moves to the next location, we draw samples from a group of nearby sensors $\mathcal{S}_t$ and measure a simple metric, such as signal-to-noise ratio (SNR), to capture the benefit of selecting a particular sensor; accordingly, we define reward as an increasing function of this metric. Considering that the sensor located in the true vehicle location would return the highest reward, we choose $\mathcal{S}_t$ to be the same as $\mathcal{L}_t$. (The choice the sampling set S_t also depends on the availability of sensors and can differ from the locations $\mathcal{L}_t$, in which our algorithm also applies.)

We make the following observations: (i) When we draw a sample from a sensor in the set $\mathcal{S}_t$, the reward will give us information about all near-by locations (not just the one we sample from). Indeed, for any (fixed but unknown) position of the vehicle, the rewards are correlated as the SNR depends on the distance from the same vehicle location. Thus we could think of our problem as hypothesis testing, where our four hypotheses correspond to the possible locations of the vehicle. (ii) During each timeslot, we can in parallel collect rewards from more than one sensors in $\mathcal{S}_t$, to reduce delay. However, we may not want to collect samples from all sensors during all timeslots, since then we may have an unnecessarily high sensing cost (number of samples we draw).

Proposed Algorithm: Inspired by the previous observations, we propose to use a maximum a posteriori (MAP) detector, combined with an online elimination algorithm, as we describe next.

Maximum a Posteriori Probability (MAP) detector. Based on the knowledge of the trajectory, we construct a prior belief $\Pi_0 = \mathcal{P}_t$ of the possible locations (i.e., hypotheses), which will be updated at each timeslot n to form a posterior distribution Π_n after receiving samples. Specifically, given a sequence of n reward-sensor pairs $\{(r^{(i)}, s_i)\}_{i=1}^n$ collected at the current time window t, where the sequence $\{s_i\}_{i=1}^n$ is selected ahead of time, and $r^{(i)}$ is a reward sampled from the sensor $s_i \in \mathcal{S}_t$, the MAP detector decides on a location of the tracked vehicle by maximizing the posterior of hypothesises as follows

$$
\begin{aligned}
l_{\text{MAP}}(t) &= \arg\max_{l \in L_t} \mathbb{P}[l_t = l | \{s_i\}_{i=1}^n, \{r^{(i)}\}_{i=1}^n] \\
&= \arg\max_{l \in \mathcal{L}_t} \mathbb{P}[\{s_i\}_{i=1}^n, \{r^{(i)}\}_{i=1}^n | l_t = l] \mathbb{P}[l_t = l] \\
&= \arg\max_{l \in \mathcal{L}_t} \mathbb{P}[\{r^{(i)}\}_{i=1}^n | l_t = l, \{s_i\}_{i=1}^n] \mathbb{P}[l_t = l] \quad (1) \\
&= \arg\max_{l \in \mathcal{L}_t} \mathbb{P}[l_t = l] \prod_{i=1}^{n} \mathbb{P}[r^{(i)} | l_t = l, s_i], \quad (2)
\end{aligned}
$$

where l_t is the true location of the vehicle at window t, and $\mathcal{L}_t$ is the set of possible locations at time t. Note that $\mathbb{P}[\{\{s_i\}_{i=1}^n\}_{i=1}^n | l_t = l]$ is omitted in (1) because $\{s_i\}_{i=1}^n$ is pre-determined and is independent of the value of l_t.

MAP with Active Elimination. To reduce the overall number of collected samples, we propose an adaptation of the MAP detector inspired by the active arm elimination algorithm [2]. Specifically, we maintain a set of good locations (i.e., hypotheses), initially defined as all possible locations $\mathcal{L}_t$. At each time slot, we collect one sample from each sensor located in the good locations. Subsequently, we calculate the a posteriori probabilities of each possible location, as described in (2). Locations with probabilities falling below a predefined threshold, denoted as p_{th}, are eliminated from the set of good locations. And we stop drawing samples from the sensor in eliminated locations. The pseudo-code is described in Algorithm 1.

Algorithm 1 MAP with Active Elimination

1: **Inputs:** number of time slots τ, set of possible locations $\mathcal{L}_t$, prior distribution Π_0, threshold probability for arm elimination p_{th}
2: **Initialize** the set of good locations: $\mathcal{G} \leftarrow \mathcal{L}_t$.
3: **Initialize** reward-sensor pairs: $\mathcal{R} \leftarrow \{\}$
4: **for** $n \leftarrow 1, 2, ..., \tau$ **do**
5: Draw samples from all sensors in good locations $\{r, s_l\}_{l \in \mathcal{G}}$
6: Update set of reward-sensor pairs: $\mathcal{R} \leftarrow \mathcal{R} \cup \{r, s_l\}_{l \in \mathcal{G}}$
7: Update distribution: $\Pi_n(l) \leftarrow \mathbb{P}[l_t = l | \mathcal{R}] \ \forall l \in \mathcal{G}$
8: Delete all arms a from $\mathcal{G}$ if $\Pi_n(l) < p_{\text{th}}$
9: **end for**
10: **Output:** $\arg\max_{l \in \mathcal{G}} \Pi_\tau(l)$

B. Computational Unit Allocation

In order to expedite the whole procedure which is reliant on sensor data, it is imperative to minimize both the transmission time from sensors to compute units and the execution time on those units. Intuitively, establishing a strong communication link between sensors and compute units is crucial for optimal performance. Given our assumption that path loss is proportional to distance, connecting a sensor to the nearest compute unit ensures a superior communication link. However, without employing a strategy, the availability of nearby compute units may be compromised due to background processing of lower-priority tasks on those units. To overcome this challenge, we

propose a strategy that reserves compute units for sensors based on predicted vehicle locations in the next time. By proactively allocating compute units, the closest available unit to a sensor can be reliably secured, resulting in efficient data transmission and processing. As a consequence, this approach can significantly reduce the overall time required for data transfer from sensors to compute units and expedites data processing at the compute units.

V. Evaluation

In this section, we evaluate our proposed algorithms through extensive simulations, in terms of all the evaluation metrics mentioned in Section III-C, and compare against the state-of-the-art algorithms as described next.

A. Simulation Setting

We use a $M \times M$ grid to approximate territories, with $M = 20$ and thus $N = 400$ squares in total. We randomly generate territory maps in which each square is randomly blocked with probability 0.1. In addition, we simulate trajectories by assuming the vehicle can move either vertically or horizontally to one of the adjacent squares and the movement probabilities take values in the set $\{c\alpha, c\beta, c\gamma, 0\}$. Here, $\alpha = 4, \beta = 2, \gamma = 0.5$ represent the weights of moving forward, changing direction (either turning left or right), and returning to a previously visited square, respectively, while the value 0 is used to indicate a blocked direction. Note that the weights of directions can change over time, as some cells may be blocked or become previously visited as the trajectory evolves; to make the probabilities sum to 1, we use the normalization constant c that may change across different time windows. For each setting and algorithm, we average the evaluation over 50 trials. In each trial, we simulate a 100-step trajectory (i.e. $T = 100$ time windows) on a randomized map.

Sensor Selection. We assume that the measurements of each sensor, i.e., rewards of each sensor l, follow a Gaussian distribution $\mathcal{N}(\mu_s, \sigma^2)$, with a common known variance $\sigma^2 = 10$ and unknown mean μ_s. The reward mean is calculated by $\mu_s = \frac{2}{(1+d_s)^2}$, where d_s is the Euclidean distance between the sensor s and vehicle. And we assume all sensors and vehicles are located in the center of a square with unit length. The choice of the reward means reflects the SNR of sensors. We also assume that at each time window, we can take at most 100 samples from all candidate sensors in total. Within each timeslot, we can take at most one sample from each sensor.

CUs Allocation. In each map, we randomly assign K compute units. We evaluate how reserving the compute unit closest to the next predicted target location can reduce the communication cost, as a function of K.

B. Evaluation on Sensor Selection

Algorithms. For the sensor selection task, we compare the following algorithms:

- The Upper Confidence Bound (UCB) algorithm [1], [9]. As UCB is not designed for best arm identification, we modify it as follows. We run UCB for 100 iterations, then output the arm with the maximum number of pulls as our estimate of the best arm
- The Successive arm elimination algorithm (SAE) [2]. In this case, we declare an arm with the highest estimated mean among the arms that are not eliminated as our estimate of the best arm.
- The Top-Two Transportation Cost (T3C) [14] assumes a prior distribution over the reward means and updates the beliefs upon receiving each sample. It pulls either the best possible arm or an alternative at each timeslot and outputs the one with the highest empirical mean as the best arm. In our simulation, we construct a Gaussian prior leveraging the prior information about the vehicle.
- MAP: We collect $\lfloor \frac{100}{|A_t|} \rfloor$ samples from each sensor, then use the MAP detector described in the previous section to output the estimate of vehicle location.
- MAP with arm elimination (MAP_e) described in Algorithm 1. We set $p_{\text{th}} = 0.01$ as the elimination threshold.

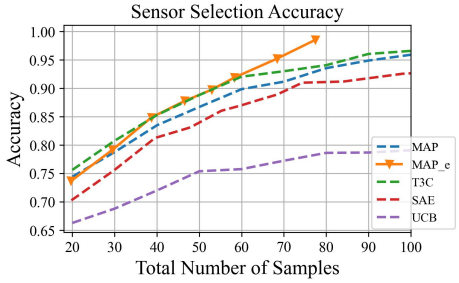

Fig. 5. Sensor Selection Accuracy v.s. Number of Measurements.

Fig. 5 plots the accuracy of correctly activating the sensor located in the square where the target vehicle appears. We compare the accuracy achieved by different algorithms using a fixed number of samples at each time window, among which MAP and T3C show similar results and outperform others.

In addition, Fig. 6 compares the accuracy, sample cost, and delay (see definitions in Section III-C) that the different algorithms achieve under the restriction of at most 100 samples. With comparable accuracy, our proposed elimination scheme significantly reduces the sample cost and delay of vanilla MAP and outperforms other algorithms across all metrics.

C. Computation Reservation Techniques

Fig. 7 illustrates the overall cost associated with transmitting data from sensors to compute units versus 20%, 50%, 80% available compute units as well as reservation technique, for three different numbers of compute units, i.e. $K = 4$, $K = 9$, and $K = 16$. As the availability of compute units decreases, primarily due to the presence of background data traffic, the cost of data transmission experiences a significant increase. However, our proposed reservation technique offers a highly effective solution to mitigate this issue: we can notably decrease the overall cost by reserving the nearest compute unit for each corresponding sensor, considering the predicted location of the vehicle in the next time window. Furthermore, as shown in Fig. 7, as expected the overall cost decreases as the number of all compute units increases.

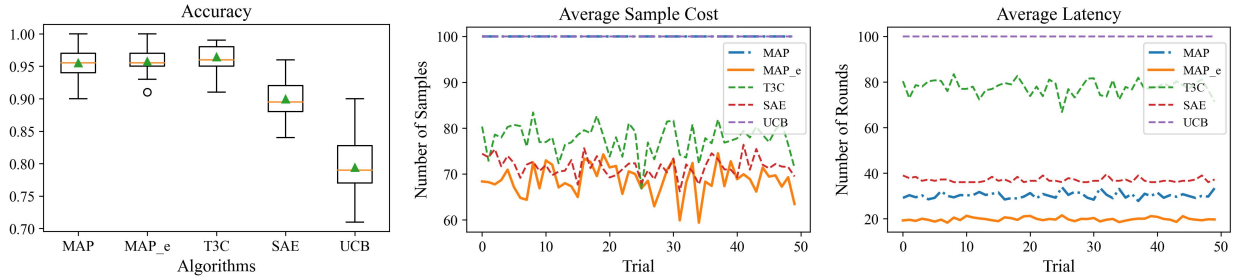

Fig. 6. (a) Accuracy, (b) Sample Cost, (c) Delay

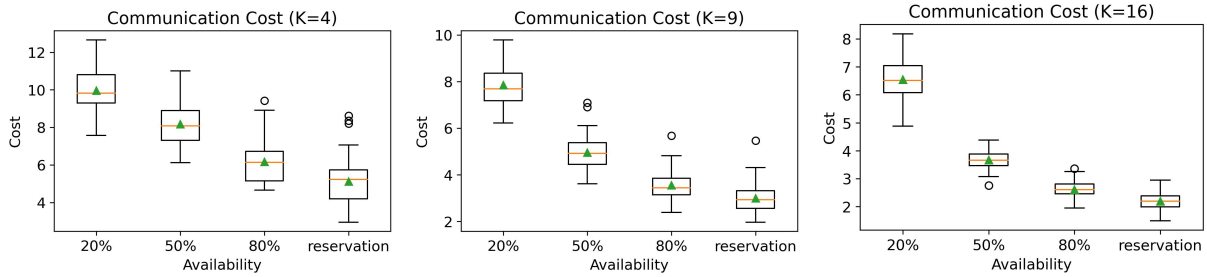

Fig. 7. Average Communication Cost per Trajectory for $K = 4$, $K = 9$, and $K = 16$

VI. Conclusions and Future Work

In this work, we considered the problem of online allocation of sensing and computational resources, under low delay, low measurements and low communication cost constraints, while tracking a vehicle of interest. Our proposed algorithms outperform the state of the art and offer an attractive approach that easily scales over large territories. Open questions that could be explored in future work include taking into account errors in estimating target trajectory and how they may propagate, unknown transition probabilities for target movement, and considering simultaneously tracking multiple vehicles. Currently, we assume that reservation incurs no cost and we are able to preempt ongoing processing on compute units without degrading the system. Exploring reservation schemes that consider these aspects is an important area for future research. It would also be of interest to consider more complex computations expressed in the form of a general task graph. In this case, the scheduling of the processing tasks will require the use of a sophisticated scheduler. In case of dynamic and uncertain network conditions, where rapid scheduling is needed, it may be helpful to explore recently developed ML-based schedulers such as GCNScheduler [5], [8]. In particular, it would be of interest to explore novel enhancements to such schedulers that take into account probabilistic predictions of sensing locations in order to make advanced reservations for compute points, which our present work indicates is a strategy that offers cost reduction. In the current work, we have decoupled sensing and communication for computation to some extent; it would also be of interest to explore how the estimated communication and computation costs for optimal task placement could be taken into account in sensor selection.

References

[1] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 05 2002.

[2] P. Auer and R. Ortner. Ucb revisited: Improved regret bounds for the stochastic multi-armed bandit problem. *Periodica Mathematica Hungarica*, 61(1-2):55–65, 2010.

[3] S. Bubeck, R. Munos, and G. Stoltz. Pure exploration in multi-armed bandits problems. In *Algorithmic Learning Theory: 20th International Conference, ALT 2009, Porto, Portugal, October 3-5, 2009. Proceedings 20*, pages 23–37. Springer, 2009.

[4] G. Ciaparrone, F. L. Sánchez, S. Tabik, L. Troiano, R. Tagliaferri, and F. Herrera. Deep learning in video multi-object tracking: A survey. *Neurocomputing*, 381:61–88, 2020.

[5] J. Coleman, M. Kiamari, L. Clark, D. D'Souza, and B. Krishnamachari. Graph convolutional network-based scheduler for distributing computation in the internet of robotic things. In *MILCOM 2022-2022 IEEE Military Communications Conference (MILCOM)*, pages 1070–1075. IEEE, 2022.

[6] P. Ghosh, Q. Nguyen, P. K. Sakulkar, J. A. Tran, A. Knezevic, J. Wang, Z. Lin, B. Krishnamachari, M. Annavaram, and S. Avestimehr. Jupiter: a networked computing architecture. In *Proceedings of the 14th IEEE/ACM International Conference on Utility and Cloud Computing Companion*, pages 1–8, 2021.

[7] D. Hu and B. Krishnamachari. Throughput optimized scheduler for dispersed computing systems. In *2019 7th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, pages 76–84. IEEE, 2019.

[8] M. Kiamari and B. Krishnamachari. Gcnscheduler: Scheduling distributed computing applications using graph convolutional networks. In *Proceedings of the 1st International Workshop on Graph Neural Networking*, pages 13–17, 2022.

[9] T. L. Lai. Adaptive treatment allocation and the multi-armed bandit problem. *Annals of Statistics*, 15:1091–1114, 1987.

[10] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, and T.-K. Kim. Multiple object tracking: A literature review. *Artificial intelligence*, 293:103448, 2021.

[11] A. Poylisher, A. Cichocki, K. Guo, J. Hunziker, L. Kant, B. Krishnamachari, S. Avestimehr, and M. Annavaram. Tactical jupiter: Dynamic scheduling of dispersed computations in tactical manets. In *MILCOM 2021-2021 IEEE Military Communications Conference (MILCOM)*, pages 102–107. IEEE, 2021.

[12] Z. Rahman, A. M. Ami, and M. A. Ullah. A real-time wrong-way vehicle detection based on yolo and centroid tracking. In *2020 IEEE Region 10 Symposium (TENSYMP)*, pages 916–920. IEEE, 2020.

[13] D. Russo. Simple bayesian algorithms for best arm identification. In *Conference on Learning Theory*, pages 1417–1418. PMLR, 2016.

[14] X. Shang, R. Heide, P. Menard, E. Kaufmann, and M. Valko. Fixed-confidence guarantees for bayesian best-arm identification. In *International Conference on Artificial Intelligence and Statistics*, pages 1823–1832. PMLR, 2020.

[15] W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25:285–294, 1933.

[16] H. Topcuoglu, S. Hariri, and M.-Y. Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE transactions on parallel and distributed systems*, 13(3):260–274, 2002.

[17] C.-S. Yang, R. Pedarsani, and A. S. Avestimehr. Communication-aware scheduling of serial tasks for dispersed computing. *IEEE/ACM Transactions on Networking*, 27(4):1330–1343, 2019.

[18] W. Zhang, H. Zhou, S. Sun, Z. Wang, J. Shi, and C. C. Loy. Robust multi-modality multi-object tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2365–2374, 2019.