Cost Aware Asynchronous Multi-Agent Active Search

Arundhati Banerjee
School of Computer Science
Carnegie Mellon University
Pittsburgh, U.S.A
arundhat@cs.cmu.edu

Ramina Ghods
School of Computer Science
Carnegie Mellon University
Pittsburgh, U.S.A
rghods@andrew.cmu.edu

Jeff Schneider
School of Computer Science
Carnegie Mellon University
Pittsburgh, U.S.A
schneide@cs.cmu.edu

***Abstract*—Multi-agent active search requires autonomous agents to choose sensing actions that efficiently locate targets. In a realistic setting, agents also must consider the costs that their decisions incur. Previously proposed active search algorithms simplify the problem by ignoring uncertainty in the agent's environment, using myopic decision making, and/or overlooking costs. In this paper, we introduce an online active search algorithm to detect targets in an unknown environment by making adaptive cost-aware decisions regarding the agent's actions. Our algorithm combines principles from Thompson Sampling (for search space exploration and decentralized multi-agent decision making), Monte Carlo Tree Search (for long horizon planning) and pareto-optimal confidence bounds (for multi-objective optimization in an unknown environment) to propose an online lookahead planner that removes all the simplifications. We analyze the algorithm's performance in simulation to show its efficacy in cost aware active search.**

***Index Terms*—Planning under Uncertainty; Integrated Planning and Learning; Path Planning for Multiple Mobile Robots or Agents**

I. Introduction

Active search [1], [2] in real world applications like environmental monitoring, or search and rescue, involves autonomous robots (agents) accurately detecting targets by making sequential adaptive data collection decisions while minimizing the usage of resources like energy and time. Previous studies have used various constraints and reductions for resource efficient adaptive search. Such algorithms generally include parameters to trade-off the informativeness of the collected data with the cost of such data collection. One approach to adaptive sensing in robotics is to reduce it to a planning problem assuming full observability of the environment [3], [4]. Imposing a cost budget then reduces to constrained path planning between the known start and goal locations. Unfortunately, this is in contrast with the real world where the agent's environment, the number of targets and their locations may be unknown and the agent may have access only to noisy observations from sensing actions. All these factors increase the difficulty of cost effective active search.

Besides cost efficiency, executing active search with multiple agents creates an additional challenge. Centralized planning in a multi-agent setting is often impractical due to communication constraints [5], [6]. Further, a real world system dependent on a central coordinator that expects synchronicity from all agents is susceptible to communication or agent failure.

In our problem formulation, the agents are not entirely independent actors and therefore still share information with their peers in the team when possible. However, we do not require them to communicate synchronously and instead assume that each agent is able to independently plan and execute its next sensing action using whatever information it already has and happens to receive.

In this paper, we propose a novel cost-aware asynchronous multi-agent active search algorithm called CAST (Cost Aware Active Search of Sparse Targets) to enable agents to detect sparsely distributed targets in an unknown environment using noisy observations from region sensing actions, without any central control or synchronous inter-agent communication. CAST performs cost-aware active search knowing only the costs of its feasible actions without requiring a pre-specified cost budget. It combines Thompson sampling with Monte Carlo Tree Search for lookahead planning and multi-agent decision making, along with Lower Confidence Bound (LCB) style pareto optimization to tradeoff expected future reward with the associated costs. We demonstrate the efficacy of CAST with a set of simulation results across different team sizes and number of targets in the search space.

II. Problem formulation

Consider a team of autonomous agents actively sensing regions of the search space looking for targets. To plan its next sensing action, each cost-aware agent has to trade-off the expected future reward of detecting a target with the overall costs it will incur in travelling to the appropriate location and executing the action. Given previous observations, it adaptively makes such data-collection decisions online while minimizing the associated costs as much as possible. Unfortunately, this problem is NP-hard [7].

Sensing setup: We first describe our setup for active search with region sensing. Consider a gridded search environment with ground truth described by a sparse matrix having k non-zero entries at the locations of the k targets. We define the flattened ground truth vector as $\boldsymbol{\beta} \in \mathbb{R}^n$ where each entry is either 1 (target) or 0 (no target). $\boldsymbol{\beta}$ is the search vector that we want to recover. The sensing model for an agent j at time t is

$$y_t^j = {\mathbf{x}_t^j}^{\mathrm{T}} \boldsymbol{\beta} + \epsilon_t^j, \text{ where } \epsilon_t^j \sim \mathcal{N}(0, \sigma^2). \tag{1}$$

$\mathbf{x}_t^j \in \mathbb{R}^n$ is the (flattened) sensing action at time t. We define our action space $\mathcal{A}$ ($\mathbf{x}_t^j \in \mathcal{A}$) to include only hierarchical spatial pyramid sensing actions [8]. y_t^j is the agent's observation and ϵ_t^j is a random, i.i.d added noise. $(\mathbf{x}_t^j, y_t^j)$ is agent j's measurement at time t. Note that the agents using this region sensing model must trade-off between sensing a wider area with lower accuracy versus a highly accurate sensing action over a smaller region. The support of the vector $\mathbf{x}_t$ is appropriately weighted so that $\|\mathbf{x}_t\|_2 = 1$ to ensure each sensing action has a constant power. This helps us in modeling observation noise as a function of the agent's distance from the region [9]. Since each action has a constant power and every observation has an i.i.d added noise with a constant variance, the signal to noise ratio in the unit squares comprising the rectangular sensing block reduces as the size of the sensing region is increased.

Cost model: We introduce the additional realistic setting that the sensing actions have different associated costs. First, we consider that the agent travelling from location a to location b incurs a travel time cost $c_d(a, b)$.[1] Second, we assume that executing each sensing action at location b incurs a time cost $c_s(b)$. Therefore, T time steps after starting from location x_0, an agent j has executed actions $\{\mathbf{x}_t^j\}_{t=1}^T$ and incurs a total cost defined by $C^j(T) = \sum_{t=1}^T c_d(x_{t-1}^j, x_t^j) + c_s(x_t^j)$.

Communication setup: We assume that communication, although unreliable, will be available sometimes and the agents should utilize it when possible. The agents share their own past measurements asynchronously with teammates, but do not wait for communication from their teammates at any time since this wait could be arbitrarily long and thus cause a loss of valuable sensing time. In the absence of synchronicity, we also do not require that the set of available past measurements remain consistent across agents since communication problems can disrupt it. We will denote the set of measurements available to an agent j at time t by $\mathbf{D}_t^j = \{(\mathbf{x}_{t'}, y_{t'}) | \{t'\} \subseteq \{1, \ldots, t-1\}\}$, $|\mathbf{D}_t^j| \leq t-1$, which includes its own past observations and those received from its teammates till time t.

III. Related work

Autonomous target search has diverse applications in environment monitoring [10], wildlife protection [11] as well as search and rescue operations [12]. In robotics, informative path planning (IPP) problems focus on adaptive decision making to reach a specified goal state or region. In contrast to our setting, common IPP algorithms consider a known environment [13], are myopic or use non-adaptive lookahead [14], and assume weakly coupled sensing and movement actions [15].

Bayesian optimization and active learning methods are another approach to active search [16], [17], [18]. Unfortunately, they mostly address single-agent systems, or if multi-agent they assume central coordination [19], [20] and except for [21] lack any realistic assumptions on sensing actions. Multi-agent asynchronous active search algorithms proposed in [9], [22] tackle several of these challenges but they are myopic in nature. Further, [23] introduced cost effective non-myopic active search but their simplified setting excludes simultaneous evaluation of multiple search points with different costs.

Our active search formulation has close similarities with planning under uncertainty using a Partially Observable Markov Decision Process (POMDP) [24]. Monte Carlo Tree Search (MCTS) [25], [26] has found success as a generic online planning algorithm in large POMDPs [27], but is mostly limited to the single agent setting [28], [29].

Decentralized POMDP (Dec-POMDP) [30], [31] is another framework for decentralized active information gathering using multiple agents which is typically solved using offline, centralized planning followed by online, decentralized execution [32], [33]. Decentralized MCTS (Dec-MCTS) algorithms have also been proposed for multi-robot active perception under a cost budget [34], [35] but they typically rely on each agent maintaining a joint probability distribution over its own belief as well as those of the other agents.

Finally, cost aware active search can be viewed as a multi-objective sequential decision making problem. [36] developed an MCTS algorithm for cost budgeted POMDPs using a scalarized version of the reward-cost trade-off whereas [37] introduced multi-objective MCTS (MO-MCTS) for discovering global pareto-optimal decision sequences in the search tree. Unfortunately, MO-MCTS is computationally expensive and unsuitable for online planning. [38] proposed the Pareto MCTS algorithm for multi-objective IPP but they ignore uncertainty due to partial observability in the search space.

IV. Our proposed algorithm: CAST

A. Background

We first briefly describe the concepts essential to the planning and decision making components of our algorithm.

Monte Carlo Tree Search (MCTS) is an online algorithm that combines tree search with random sampling in a domain-independent manner. In our setup, a cost-aware agent would benefit from the ability to lookahead into the adaptive evolution of its belief about the target's distribution in the environment in response to possible observations from the actions it might execute. We therefore consider MCTS as the basis for developing our online planning method with finite horizon lookahead. Unfortunately, the presence of uncertainty about targets (number and location) in the unknown environment together with the noisy observations introduces additional challenges in our problem formulation.

Pareto optimality: Our formulation of cost aware active search described in Section II can be viewed as a multi-objective sequential decision making problem. A common approach to solving such multi-objective optimization problems is scalarization i.e. considering a weighted combination resulting in a single-objective problem that trades off the different objectives. However, tuning the weight attributed to each objective is challenging since they might be scaling quantities having different units and their relative importance might be context dependent. In contrast, pareto optimization builds on the idea that some solutions to the multi-objective optimization

[1] We assume a constant travelling speed and compute the Euclidean distance between locations a and b.

problem are categorically worse than others and are *dominated* by a set of pareto-optimal solution vectors forming a pareto front for the optimization objective. Considering a set of D-dimensional vectors $\mathbf{g} \in \mathcal{G}$, we define the following:

- $\mathbf{g}$ dominates $\mathbf{g}'$ (i.e. $\mathbf{g} \succ \mathbf{g}'$) iff: (1) $\forall d \in \{1, \ldots, D\}$, $[\mathbf{g}]_d \geq [\mathbf{g}']_d$ (2) $\exists d \in \{1, \ldots, D\}$, $[\mathbf{g}]_d > [\mathbf{g}']_d$
- $\mathbf{g}$ and $\mathbf{g}'$ are incomparable (i.e. $\mathbf{g} || \mathbf{g}'$) iff: $\exists d_1, d_2 \in \{1, \ldots, D\}$, $[\mathbf{g}]_{d_1} > [\mathbf{g}']_{d_1}$ *and* $[\mathbf{g}]_{d_2} < [\mathbf{g}']_{d_2}$
- $\mathcal{G}^* \subseteq \mathcal{G}$ is the pareto-front of $\mathcal{G}$ iff: (1)$\forall \mathbf{g} \in \mathcal{G}$ and $\forall \mathbf{g}' \in \mathcal{G}^*$, $\mathbf{g} \not\succ \mathbf{g}'$ (2) $\forall \mathbf{g}, \mathbf{g}' \in \mathcal{G}^*$, $\mathbf{g} || \mathbf{g}'$.

In our algorithm, each agent estimates a reward-cost vector to evaluate its candidate actions and chooses the next sensing action from a pareto-optimal set of such vectors.

Thompson sampling (TS) [39], studied in a number of bandit and reinforcement learning (RL) settings [40], [41], [42], balances exploration with exploitation by choosing actions that maximize the expected reward assuming that a sample drawn from the posterior distribution is the true state of the world. As a result, exploration in TS is reward oriented, leaning heavily on the drawn posterior sample [43], [44]. Moreover TS only explores on the seemingly optimal policies, which is a disadvantage in environments where knowledge-seeking actions having lower immediate reward are crucial for the agent's efficient long-term performance. This is especially relevant for cost-aware active search wherein actions which are not immediately rewarding in terms of having detected a target may still be informative, for example, by reducing the uncertainty regarding the presence of targets in a certain part of the search space. In this work, we build upon these insights and adapt TS in combination with MCTS to develop a search tree building strategy for online lookahead planning under partial observability.

Additionally, posterior sample based action selection makes TS an excellent candidate for a decentralized multi-agent decision making algorithm [45] and it has been shown to be effective in multi-agent active search with myopic planning [9], [22]. In this work, we will show that our TS based lookahead planning algorithm enables decentralized multi-agent active search with no pre-coordination and minimum communication overhead among agents, in contrast with existing multi-agent algorithms that rely on pre-designed movement coordination or communication and update of joint probability distributions.

B. Our approach

Following the setting described in Section II, consider J agents searching for k sparsely located targets in an unknown environment and $\boldsymbol{\beta}$ is the search vector we want to recover. The agent's belief $b(\boldsymbol{\beta})$ over $\boldsymbol{\beta}$ is a continuous probability distribution over the search space. For any agent j, the prior belief is modeled by $b_0^j = \mathrm{P}(\boldsymbol{\beta}) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ and the likelihood function following the sensing model (1) is given by $\mathrm{P}(y_t^j | \boldsymbol{\beta}, \mathbf{x}_t^j) = \mathcal{N}({\mathbf{x}_t^j}^T \boldsymbol{\beta}, \sigma^2)$. Therefore, at time step t, its posterior belief over $\boldsymbol{\beta}$ is denoted $b_t^j = \mathrm{P}(\boldsymbol{\beta} | \mathbf{D}_t^j \cup \{\mathbf{x}_t^j, y_t^j\}) = \mathcal{N}(\boldsymbol{\mu}_t^j, \boldsymbol{\Sigma}_t^j)$. In the multi-agent setting, each agent j maintains its own posterior belief $b_t^j(\boldsymbol{\beta})$ and estimate $\hat{\boldsymbol{\beta}}(\mathbf{D}_t^j \cup \{\mathbf{x}_t^j, y_t^j\}) = \big(\sigma^2 * {\boldsymbol{\Sigma}_0}^{-1} + \begin{bmatrix} {\mathbf{X}_t^j}^{\mathrm{T}} & \mathbf{x}_t^j \end{bmatrix} \begin{bmatrix} \mathbf{X}_t^j \\ {\mathbf{x}_t^j}^{\mathrm{T}} \end{bmatrix} \big)^{-1} \begin{bmatrix} {\mathbf{X}_t^j}^{\mathrm{T}} & \mathbf{x}_t^j \end{bmatrix} \begin{bmatrix} \mathbf{y}_t^j \\ y_t^j \end{bmatrix}$ where $\{\mathbf{X}_t^j, \mathbf{y}_t^j\}$ consist of the measurements in $\mathbf{D}_t^j$.

Reward formulation: We first observe that our active search problem can be categorized as parameter estimation in active learning, developed in [46] with the name Myopic Posterior Sampling (MPS). Like MPS, our objective is to actively recover the search vector $\boldsymbol{\beta}$. However, MPS being myopic, chooses $\mathbf{x}_t^j$ that maximizes $\mathbb{E}_{y_t^j | \mathbf{x}_t^j, \boldsymbol{\beta}_t^j}\big[\lambda(\boldsymbol{\beta}_t^j, \mathbf{D}_{t+1}^j)\big]$ where $\lambda(\boldsymbol{\beta}_t^j, \mathbf{D}_{t+1}^j) = -\|\boldsymbol{\beta}_t^j - \hat{\boldsymbol{\beta}}(\mathbf{D}_{t+1}^j)\|_2^2$, $\boldsymbol{\beta}_t^j \sim b_t^j$ and $\mathbf{D}_{t+1}^j = \mathbf{D}_t^j \cup \{\mathbf{x}_t^j, y_t^j\}$. Essentially, $\lambda(\boldsymbol{\beta}_t^j, \mathbf{D}_{t+1}^j)$ is designed so that the agents will keep exploring the search space as long as there is uncertainty in the posterior samples $\boldsymbol{\beta}_t^j$. Simultaneously, the posterior belief distribution will contain uncertainty as long as there are unexplored or less explored locations in the search space.

In contrast with MPS, to be cost efficient, our active search agents would benefit from non-myopic reasoning about the trade-off between potential reward obtained by identifying targets versus cost incurred from executing such sensing actions over a finite horizon lookahead. But we note that $\lambda(\boldsymbol{\beta}_t^j, \mathbf{D}_{t+1}^j) \leq 0$, therefore if we simply extend the MPS reward over multiple lookahead steps and try to maximize the value of cumulative discounted reward divided by total incurred cost, it would erroneously favour costlier actions for the same reward. Instead, we propose using $\lambda^-(\boldsymbol{\beta}_t^j, \mathbf{D}_{t+1}^j) = \max\{0, \|\boldsymbol{\beta}_t^j - \hat{\boldsymbol{\beta}}(\mathbf{D}_t^j)\|_2^2 - \|\boldsymbol{\beta}_t^j - \hat{\boldsymbol{\beta}}(\mathbf{D}_{t+1}^j)\|_2^2\}$ as the one-step lookahead reward. We design λ^- to encourage information gathering by favoring actions $\mathbf{x}_t$ that reduce the uncertainty in the posterior sample $\boldsymbol{\beta}_t^j$ over consecutive time steps. Additionally, $\lambda^-(\boldsymbol{\beta}_t^j, \mathbf{D}_{t+1}^j) \geq 0$. Now, we can compute the u-step lookahead reward $R^u(\mathbf{x}_t, \boldsymbol{\beta}_t^j)$ over action sequence $\mathbf{x}_{t:t+u}$ as the γ-discounted expected sum of λ^- over u steps.

$$R^u(\mathbf{x}_t, \boldsymbol{\beta}_t^j) = \mathbb{E}_{y_{t:t+u}}\big[\sum_{\Delta t=1}^{u} \gamma^{\Delta t - 1} \lambda^-(\boldsymbol{\beta}_t^j, \mathbf{D}_{t+\Delta t}^j)\big] \qquad (2)$$

Following the discussion in Section IV-A about TS, we observe that the reward computation in (2) is dependent on the posterior sample $\boldsymbol{\beta}_t^j$. Particularly, $\lambda^-(\boldsymbol{\beta}_t^j, \mathbf{D}_{t+1}^j)$ is higher for sensing actions $\mathbf{x}_t$ that identify the non-zero support elements of the vector $\boldsymbol{\beta}_t^j$. Further, maximizing $R^u(\mathbf{x}_t, \boldsymbol{\beta}_t^j)$ over all sequences $\mathbf{x}_{t:t+u}$ for a sampled $\boldsymbol{\beta}_t^j$ would exacerbate this problem by choosing a series of point sensing actions that identify the non-zero support of the particular sample. Section 8.2 of [41] also highlights this drawback of employing TS based exploration in active learning problems that require a careful assessment of the information gained from actions. In order to overcome these challenges, we propose generalizing the posterior sampling step to a sample size greater than one and combine the information from these samples using confidence bounds over λ^- to evaluate the corresponding sensing actions. To further clarify these design details, we now describe our new algorithm CAST, outlined in Algorithm 1.

CAST: At each time step t, on the basis of its history $\mathbf{D}_t^j$ of past measurements, the agent j decides its next region sensing action $\mathbf{x}_t^j$ using the SEARCH procedure of Algorithm 1. It starts

with an empty tree $\mathcal{T}_t^j$ having just a root node and gradually builds it up over m episodes. We assume a maximum tree depth $d_{\max}$. Our search tree has two types of nodes - belief nodes and action nodes. A belief node h is identified by the history of actions and observations accumulated in reaching that node. An action node (h, a) is identified by the action $\mathbf{a}$ taken at the immediately preceding belief node h in the search tree. The root as well as the leaves are belief nodes. Each

Algorithm 1 Cost Aware Active Search of Sparse Targets

1: **procedure** MAIN ▷ Executed on each agent j
2: **for** t in $\{1, 2, \ldots\}$ **do**
3: $\mathbf{x}_t^j$ = SEARCH($\mathbf{D}_t^j$)
4: Execute $\mathbf{x}_t^j$. Observe y_t^j. $\mathbf{D}_{t+1}^j = \mathbf{D}_t^j \cup \{\mathbf{x}_t^j, y_t^j\}$
5: Share $\{\mathbf{x}_t^j, y_t^j\}$ asynchronously with teammates.
6: Update belief b_{t+1}^j and estimate $\hat{\boldsymbol{\beta}}(\mathbf{D}_{t+1}^j)$.
7: **procedure** SEARCH($\mathbf{D}_t$)
8: **At time t:** History $\mathbf{D}_t = \{(\mathbf{x}_i, y_i)\}_{i=1}^t$, Agent location x_t, Belief $b_t = \mathrm{P}(\boldsymbol{\beta}|\mathbf{D}_t)$, Search tree $\mathcal{T}_t = \phi$
9: **for** each episode $m' \in \{1 \ldots m\}$ **do**
10: Sample $\boldsymbol{\beta} \sim b_t$. Discretize $\boldsymbol{\beta}$ to get $\boldsymbol{\beta}_{m',t}$.
11: SIMULATE($\boldsymbol{\beta}_{m',t}, \mathbf{D}_t, x_t, 0$)
12: $\mathcal{A}_t^*$ =ParetoOptimalActionSet($\mathcal{T}_t$)
13: $\mathbf{a}_t^* = \mathrm{argmax}_{\mathbf{a}}\{\frac{\mathbf{a}.r^{LCB}}{\mathbf{a}.cost}|\mathbf{a} \in \mathcal{A}_t^*\}$
14: **return** $\mathbf{a}_t^*$
15: **procedure** SIMULATE($\boldsymbol{\beta}, \mathbf{D}, x, d$)
16: **Input:** Posterior sample $\boldsymbol{\beta}$, Root node history $\mathbf{D}$, Agent's current location x, Root node depth d
17: $n(h) \leftarrow n(h) + 1$ ▷ Denote root (belief) node as h
18: **if** $d = d_{\max}$ **then return** $0, 0$ ▷ Reached leaf node
19: **if** $\lfloor n(h)^{\alpha_s} \rfloor > \lfloor (n(h)-1)^{\alpha_s} \rfloor$ **then**
20: add new child action node (h, a)
21: **else** select action node (h, a) using (3)
22: $n(h, a) \leftarrow n(h, a) + 1$
23: $o \leftarrow \mathbf{a}^{\mathrm{T}}\boldsymbol{\beta}$, $\mathbf{D}' \coloneqq \mathbf{D} \cup \{\mathbf{a}, o\}$
24: **if** o was not previously observed at (h, a) **then**
25: append new node h' due to o in branch hah'
26: $r_{h'} = \lambda^-(\boldsymbol{\beta}, \mathbf{D}), c_{h'}(x, a) = c_d(x, a) + c_s(a)$
27: Update $r_{h'}^{LCB}$ and $\mathbf{g}_{h'} = \begin{bmatrix} r_{h'}^{LCB} & -c_{h'}(x, a) \end{bmatrix}^{\mathrm{T}}$
28: r', c' = SIMULATE($\boldsymbol{\beta}, \mathbf{D}', a, d+1$)
29: $r'' = r_{h'} + \gamma \times r'$, $c'' = c_{h'} + c'$
30: $\bar{Q}^{UCT}(h, a) = \frac{\bar{Q}^{UCT}(h,a) \times (n(h,a)-1) + \frac{r''}{c''}}{n(h,a)}$
31: LCBParetoFrontUpdate(h')
32: LCBParetoFrontUpdate((h, a))
33: **return** r'', c''

episode $m' \in \{1, \ldots, m\}$ comprises the following:

1) *Sampling*: First, a posterior sample is drawn at the root node from the belief $b_t^j = \mathrm{P}(\boldsymbol{\beta}|\mathbf{D}_t^j)$ and discretized into a binary vector $\boldsymbol{\beta}_{m',t}^j \in \{0, 1\}^n$ (Line 10).
2) *Selection and Expansion*: Starting at the root node, a child action node selection policy (tree policy) is applied at every belief node h in a top-down depth-first traversal till a leaf node is reached. In order to prevent tree width explosion with increasing size of the action space, the progressive widening parameter α_s (Line 19) [47] determines when a new action node is added to the tree. Arriving at any action node (h, a), the corresponding maximum likelihood (ML) observation $o = \mathbf{a}^T\boldsymbol{\beta}_{m',t}^j$ is computed (Line 23) which helps transition to its child belief node h'. The 1-step reward $\lambda_{m'}^-$ for $\boldsymbol{\beta} = \boldsymbol{\beta}_{m',t}^j$ and associated execution cost is computed at each belief node visited in m' (Line 26). Every belief and action node in m' also updates the number of times it has been visited so far (Lines 17 and 22).
3) *Backpropagation*: Once the maximum depth is reached, the lookahead rewards and associated costs are backpropagated up from the leaf to each belief and action node visited in m'. Each action node stores the discounted reward per unit cost averaged over $n(h, a)$ simulations in the subtree rooted at that node (Line 30). Further, each belief and action node builds a reward-cost pareto front (Lines 31 and 32) using the backed up values from their respective subtrees which is utilized in deciding $\mathbf{x}_t$ after m episodes (Line 12).

Remark 1. *The size of action space in active search is larger than what MCTS algorithms commonly deal with, unless they are augmented with a neural policy network [48], [49]. Having a continuous state vector gives rise to additional challenges of exploding width at the belief nodes, making the tree too shallow to be useful and may cause collapse of belief representations resulting in overconfidence in the estimated policy. The added observation noise would exacerbate these challenges. Therefore, discretization of the posterior sample $\boldsymbol{\beta}$ and using the ML observation in updating the belief nodes are important modifications to make MCTS work in our setting.*

Tree policy: UCT (Upper Confidence Bound applied to trees) [25] is the tree policy used in most MCTS implementations to balance exploration-exploitation in building the search tree. UCT exploits action nodes based on their lookahead reward estimates averaged over past episodes but does not account for the inter-episode variance in such rewards. Particularly in our setting, the lookahead reward at any action node in an episode m' depends on the posterior sample $\boldsymbol{\beta}_{m',t}^j$ drawn at the root node and this stochasticity leads to sample variance especially when the particular action node has been visited in only a few episodes. We can account for this variance using the UCB-tuned policy [50] to guide action node selection. Besides, [51] formalized a correction to the UCT formula in an MDP framework replacing its logarithmic exploration term with an appropriate polynomial. We extend it to our tree policy in CAST, called CAST-UCT (3), by combining it with UCB-tuned to balance exploration with exploitation while building the search tree in our partially observable state space. Specifically, CAST-UCT chooses

$$\mathbf{a}^* = \underset{\mathbf{a}}{argmax}\, Q(h, a) + \sqrt{\frac{2\sigma_{h,a}\sqrt{n(h)}}{n(h, a)}} + \frac{16\sqrt{n(h)}}{3n(h, a)}. \quad (3)$$

$\sigma_{h,a}^2$ is the variance of the terms averaged in $Q(h, a)$.

Pareto front construction with confidence bounds: During the selection and expansion phase in any episode m', the one-step lookahead reward $\lambda^-_{m'}$ is computed at each visited belief node h (Line 26). We note that $\lambda^-_{m'}$ depends on the posterior sample $\boldsymbol{\beta}_{m'}$ drawn for that episode. Assuming that a belief node h is visited in $n(h)$ episodes so far while building the search tree $\mathcal{T}_t$, we account for the stochasticity in the computed λ^- by maintaining the Lower Confidence Bound (LCB) of these rewards (denoted r_h^{LCB}) using the Student's t-distribution to estimate a 95% confidence interval (Line 27). Denoting the cost of executing the action that transitions into the belief node h as c_h (Line 26), we define a LCB based immediate (one-step lookahead) reward-cost vector at h, $\mathbf{g}_h = \begin{bmatrix} r_h^{LCB} & -c_h \end{bmatrix}^{\mathrm{T}}$ which is essential to our multi-objective decision making as described next. Fig. 1 highlights, in blue, these variables updated during the selection and expansion phase in one episode.

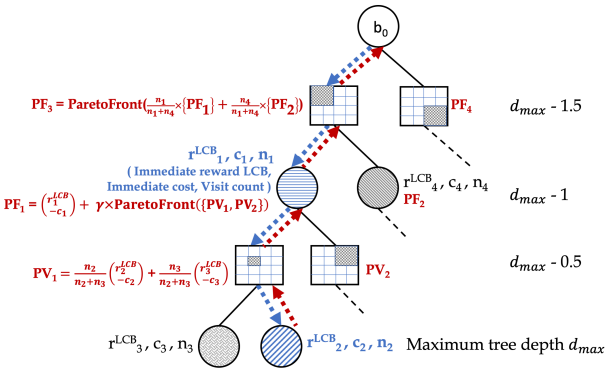

Fig. 1: Illustration of a search tree $\mathcal{T}_t$ with $d_{\max} = 2$. b_0 is the belief at the root node. The action nodes (rectangles) indicate region sensing actions. The belief nodes (circles) are shaded to indicate the evolving posterior belief in the search tree. Blue arrows illustrate top-down traversal for episode m'. r_1^{LCB}, c_1, n_1, r_2^{LCB}, c_2 and n_2 are updated. Red arrows illustrate bottom-up traversal for episode m'. $PV_{\{1,2\}}$ are vectors, $PF_{\{1,2,3,4\}}$ are the pareto fronts at the respective belief and action nodes. γ is the discount factor. ParetoFront() obtains the pareto-optimal vectors from an input set. During backpropagation, only PV_1, PF_1 and PF_3 are updated corresponding to nodes encountered during top-down traversal.

Next, we compute the pareto front over the multi-step lookahead reward-cost vectors at tree nodes visited during the backpropagation phase in episode m'. Fig. 1 illustrates this process. Note that the search tree depth at the leaf nodes is $d_{\max}$ and consecutive action and belief nodes differ in depth by 0.5. The lookahead reward-cost vector at the action node at depth $d_{\max} - 0.5$ is the weighted average of the reward-cost vectors $\mathbf{g}_{h_\ell}$ of all its leaves h_ℓ, weights being in proportion of their visits. Next, the belief node at depth $d_{\max} - 1$ builds a pareto front from the lookahead reward-cost vectors of all its children action nodes. It then takes a discounted sum of its immediate reward-cost vector with this pareto front to build its lookahead reward-cost vector *set* since the pareto front may comprise multiple non-dominated pareto-optimal vectors. Repeating these steps in episode m' all the way up to the root node, we alternate between the following: 1) every action node builds its lookahead reward-cost vector set as the pareto front computed from the weighted average of the lookahead vectors of its children belief nodes 2) every belief node builds its lookahead reward-cost vector set by taking the discounted sum of its immediate reward-cost vector with the pareto front obtained from its children action nodes. Note that all reward-cost vectors use the LCB of the rewards. Therefore, at the end of m episodes, each child action node of the root has an LCB based pareto front of lookahead reward-cost vectors. $\mathcal{A}_t^*$ (Line 12) is the pareto front at the root node comprising the non-dominated lookahead reward-cost vectors among its children action nodes. Finally, the agent selects the action node at the root having the maximum value of reward per unit cost among its vectors in $\mathcal{A}_t^*$ (Line 13). This completes the agent's decision making step at time t.

Remark 2. *In the multi-agent setting as described in Section II, we merely have the agents asynchronously broadcast their action and observation history and each agent independently incorporates in its belief any such data it happens to receive (Line 5). Note that when the agents share their actions and observations, they can consequently hold similar posterior beliefs. However an agent at time t will not necessarily have access to all other agents' histories from the previous $t-1$ time steps (i.e. $|\mathbf{D}_t^j| \leq t-1$) since agents may plan, execute and communicate at different rates. Thus the stochastic nature of our posterior sampling based algorithm enables multi-agent cost-aware active search without a central controller.*

V. Results

We now evaluate CAST by comparing in simulation the total cost incurred during multi-agent active search using cost-aware agents against the cost agnostic active search algorithms SPATS [9] and RSI [21]. SPATS is a TS based algorithm for asynchronous multi-agent active search, whereas RSI chooses sensing actions that maximize its information gain. We also consider sequential point sensing (PS) as a baseline for exhaustive coverage.

In our experiments, we focus on 2-dimensional (2D) search spaces discretized into 16×16 square grid cells of width 10m. An agent can move horizontally or vertically at a constant speed of 5m/s. Each sensing action incurs a fixed cost of c_s seconds (s), in addition to the travel time between sensing locations. We note that the cost-aware active search strategy may differ depending on the relative magnitudes of per action sensing cost and per unit travel cost. Hence, for each setting, we will vary $c_s \in \{0\text{s}, 50\text{s}\}$ to simulate high travel cost and high sensing cost respectively. Our goal is to estimate a k-sparse signal $\boldsymbol{\beta}$ by detecting all the k targets with J agents.

The search vector $\boldsymbol{\beta}$ is generated as a randomly uniform k-sparse vector in the search space. The agents are unaware of k and the generative prior. We set the signal to noise variance to 16. For CAST, we set $\gamma = 0.97$ and $\alpha_s = 0.5$. The hyperparameters in SPATS and RSI follow [9], [21]. We allow

the agents to continue searching the space until all targets have been recovered. Then, across 10 random trials we measure the mean and standard error (s.e.) of the total cost incurred by the team in recovering all k targets. We also plot the mean and s.e. of the full recovery rate achieved as a function of the total cost incurred. The full recovery rate is defined as the fraction of targets in $\boldsymbol{\beta}$ that are correctly identified. All agents start from the same location at one corner of the search space, fixed across trials. However, the exact instantiation of the search space varies across trials in terms of the position of the targets.

Fig. 2 shows full recovery rate versus total cost incurred with J agents looking for $k = 5$ targets in a 16×16 search space. We vary the team size $J \in \{4, 8, 12\}$. Table I indicates the corresponding total cost to correctly detect all targets. CAST simulates $m = 25000$ episodes with a lookahead horizon of 2 actions ($d_{\text{max}} = 2$). Each agent can choose from 341 region sensing actions over successive time steps. We observe that

TABLE I: Total cost (mean and s.e. over 10 trials) to achieve full recovery in a 16×16 grid with J agents, $k = 5$ targets.

Algorithm	J	$c_s = 0$s	$c_s = 50$s
CAST	4	**655.9 (39.4)**	**6852.3 (314.1)**
SPATS		2988.8 (285.6)	12563.8 (1132.7)
RSI		797.4 (37.2)	6862.4 (252.8)
PS		1654.1 (64.4)	42753.3 (1742.6)
CAST	8	**827.0 (48.4)**	**9529.7 (350.6)**
SPATS		2482.3 (255.7)	10242.3 (1033.6)
RSI		1455.5 (59.8)	12815.5 (513.6)
PS		3414.9 (143.7)	88839.9 (3735.3)
CAST	12	**991.6 (39.6)**	**7647.59 (445.4)**
SPATS		2699.2 (240.1)	10764.2 (948.8)
RSI		2118.9 (71.0)	19023.9 (551.1)
PS		4827.0 (167.4)	125582.0 (4352.2)

CAST outperforms SPATS, RSI and PS, incurring a lower cost and a higher full recovery rate across different team sizes and cost scenarios. RSI is information greedy and deterministic in its decision making, so all agents choose the same actions leading to an increasing total cost with larger team sizes. On the other hand, the stochastic nature of TS based active search in SPATS is suited to the asynchronous and decentralized multi-agent setup and becomes competitive especially when sensing actions are more expensive than travelling ($c_s = 50s$) which aligns best with the objective of active information gathering. Exhaustive coverage in PS is comparable only with a smaller team size in case when travelling is expensive ($J = 4, c_s = 0s$) but outperforms SPATS in that setting, showing the need for cost-awareness in active search. Unfortunately in cases that do not match their most favorable scenarios, all of these algorithms exhibit poor cost efficiency. In contrast, the cost-aware agents using CAST's posterior sampling based lookahead pareto-optimal planning and stochastic decision making are able to achieve cost efficiency across different cost scenarios with teams of varying sizes.

We also evaluate the robustness of CAST by comparing the total cost incurred to correctly identify all targets as the number of targets increases in the search space. Table II shows that CAST not only outperforms all others across multi-target and cost scenarios, additionally the total cost incurred is hardly

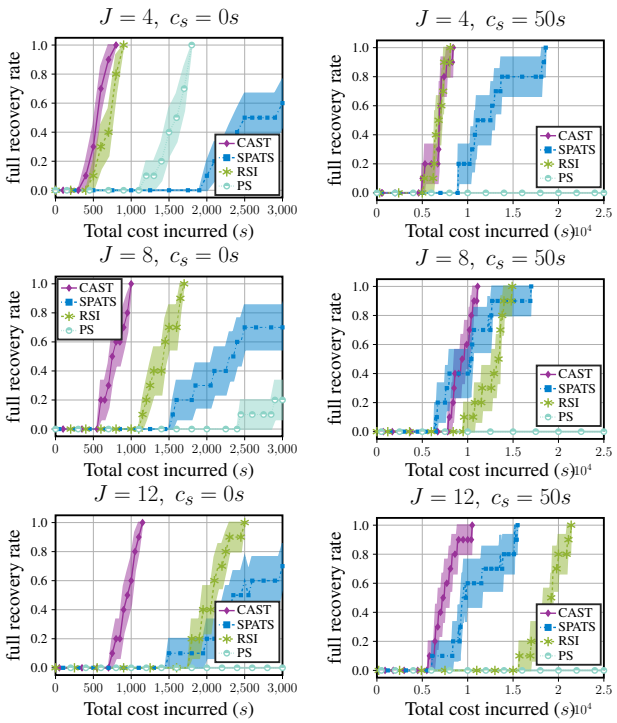

Fig. 2: Full recovery rate versus total cost incurred in seconds in a 16×16 grid with J agents, $k = 5$ targets.

TABLE II: Total cost (mean and s.e. over 5 trials) to achieve full recovery in a 16×16 grid with $J = 8$ agents, k targets.

Algorithm	k	$c_s = 0$s	$c_s = 50$s
CAST	4	**740.7 (26.5)**	**8130.4 (293.1)**
SPATS		3404.4 (432.9)	13574.4 (1792.2)
RSI		1262.8 (73.4)	10242.8 (685.8)
PS		2698.3 (438.6)	70208.3 (11404.6)
CAST	8	**735.3 (57.9)**	**9157.0 (476.7)**
SPATS		3217.2 (461.5)	13267.2 (1737.3)
RSI		1968.4 (72.2)	18398.4 (500.3)
PS		3339.9 (151.7)	86889.9 (3945.2)
CAST	16	**843.9 (29.9)**	**8880.8 (316.2)**
SPATS		3032.7 (54.3)	13212.7 (321.0)
RSI		2734.4 (58.9)	30524.4 (871.3)
PS		3559.1 (83.7)	92589.1 (2176.8)

affected by k since CAST enables cost awareness through decentralized decision making independent of team size J and sparsity rate k. In contrast, SPATS being myopic in nature exhibits more randomness in the actions selected, whereas RSI approximates its mutual information objective assuming $k = 1$, thereby requiring more sensing actions to recover all targets as k increases. For further visualization of the cost-aware multi-agent behavior of CAST compared to RSI and SPATS, we refer to the webpage we created at this link.

VI. Conclusion

We have proposed CAST for detecting sparsely distributed targets without a central planner. Interesting directions of future work include predicting cost-aware trajectories for continuous sensing as well as cost-aware active search and tracking of dynamic targets.

REFERENCES

[1] R. Garnett, Y. Krishnamurthy, D. Wang, J. Schneider, and R. Mann, "Bayesian optimal active search on graphs," in *Ninth Workshop on Mining and Learning with Graphs*, 2011.

[2] R. Garnett, Y. Krishnamurthy, X. Xiong, J. Schneider, and R. Mann, "Bayesian optimal active search and surveying," in *Proceedings of the 29th International Coference on International Conference on Machine Learning*, 2012.

[3] R. Pěnička, J. Faigl, M. Saska, and P. Váňa, "Data collection planning with non-zero sensing distance for a budget and curvature constrained unmanned aerial vehicle," *Autonomous Robots*, vol. 43, no. 8, 2019.

[4] D. Kent and S. Chernova, "Human-centric active perception for autonomous observation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[5] Z. Yan, N. Jouandeau, and A. A. Cherif, "A survey and analysis of multi-robot coordination," *International Journal of Advanced Robotic Systems*, vol. 10, no. 12, 2013.

[6] C. Robin and S. Lacroix, "Multi-robot target detection and tracking: taxonomy and survey," *Autonomous Robots*, vol. 40, no. 4, 2016.

[7] Z. W. Lim, D. Hsu, and W. S. Lee, "Adaptive informative path planning in metric spaces," *The International Journal of Robotics Research*, vol. 35, no. 5, 2016.

[8] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, 2006.

[9] R. Ghods, A. Banerjee, and J. Schneider, "Decentralized multi-agent active search for sparse signals," in *Uncertainty in Artificial Intelligence*, 2021.

[10] M. Popović, T. Vidal-Calleja, G. Hitz, I. Sa, R. Siegwart, and J. Nieto, "Multiresolution mapping and informative path planning for uav-based terrain monitoring," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[11] J. Linchant, J. Lisein, J. Semeki, P. Lejeune, and C. Vermeulen, "Are unmanned aircraft systems (uas s) the future of wildlife monitoring? a review of accomplishments and challenges," *Mammal Review*, vol. 45, no. 4, 2015.

[12] A. Gupta, D. Bessonov, and P. Li, "A decision-theoretic approach to detection-based target search with a uav," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[13] A. A. Meera, M. Popović, A. Millane, and R. Siegwart, "Obstacle-aware adaptive informative path planning for uav-based target search," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019.

[14] A. Singh, A. Krause, and W. J. Kaiser, "Nonmyopic adaptive informative path planning for multiple robots," in *Proceedings of the 21st international jont conference on Artifical intelligence*, 2009.

[15] S. Choudhury, N. Gruver, and M. J. Kochenderfer, "Adaptive informative path planning with multimodal sensing," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 30, 2020.

[16] R. Marchant, F. Ramos, S. Sanner, *et al.*, "Sequential bayesian optimisation for spatial-temporal monitoring." in *UAI*, 2014.

[17] P. Rajan, W. Han, R. Sznitman, P. Frazier, and B. Jedynak, "Bayesian multiple target localization," *Journal of Machine Learning Research*, vol. 37, 2015.

[18] S. Jiang, G. Malkomes, G. Converse, A. Shofner, B. Moseley, and R. Garnett, "Efficient nonmyopic active search," in *International Conference on Machine Learning*, 2017.

[19] J. Azimi, A. Fern, X. Z. Fern, G. Borradaile, and B. Heeringa, "Batch active learning via coordinated matching," in *Proceedings of the 29th International Coference on International Conference on Machine Learning*, 2012.

[20] S. Jiang, G. Malkomes, M. Abbott, B. Moseley, and R. Garnett, "Efficient nonmyopic batch active search," in *Advances in Neural Information Processing Systems*, 2018.

[21] Y. Ma, R. Garnett, and J. Schneider, "Active search for sparse signals with region sensing," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[22] R. Ghods, W. J. Durkin, and J. Schneider, "Multi-agent active search using realistic depth-aware noise model," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[23] S. Jiang, R. Garnett, and B. Moseley, "Cost effective active search," in *Advances in Neural Information Processing Systems*, 2019.

[24] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, vol. 101, no. 1-2, 1998.

[25] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European conference on machine learning*, 2006.

[26] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in games*, vol. 4, no. 1, 2012.

[27] D. Silver and J. Veness, "Monte-carlo planning in large pomdps," in *Advances in neural information processing systems*, 2010.

[28] G. Flaspohler, V. Preston, A. P. Michel, Y. Girdhar, and N. Roy, "Information-guided robotic maximum seek-and-sample in partially observable continuous environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, 2019.

[29] J. Fischer and Ö. S. Tas, "Information particle filter tree: An online algorithm for pomdps with belief-based rewards on continuous domains," in *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[30] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein, "The complexity of decentralized control of markov decision processes," *Mathematics of operations research*, vol. 27, no. 4, 2002.

[31] F. A. Oliehoek, C. Amato, *et al.*, *A concise introduction to decentralized POMDPs*. Springer, 2016, vol. 1.

[32] M. Lauri, J. Pajarinen, and J. Peters, "Multi-agent active information gathering in discrete and continuous-state decentralized pomdps by policy graph improvement," *Autonomous Agents and Multi-Agent Systems*, vol. 34, no. 42, 2020.

[33] M. Lauri and F. Oliehoek, "Multi-agent active perception with prediction rewards," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[34] F. Sukkar, G. Best, C. Yoo, and R. Fitch, "Multi-robot region-of-interest reconstruction with dec-mcts," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019.

[35] G. Best, O. M. Cliff, T. Patten, R. R. Mettu, and R. Fitch, "Decentralised monte carlo tree search for active perception," in *Algorithmic Foundations of Robotics XII*. Springer, 2020.

[36] J. Lee, G.-H. Kim, P. Poupart, and K.-E. Kim, "Monte-carlo tree search for constrained pomdps," in *Advances in Neural Information Processing Systems*, 2018.

[37] W. Wang and M. Sebag, "Multi-objective monte-carlo tree search," in *Asian conference on machine learning*, 2012.

[38] W. Chen and L. Liu, "Pareto monte carlo tree search for multi-objective informative planning." in *Robotics: Science and Systems*, 2019.

[39] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, no. 3/4, 1933.

[40] A. Gopalan, S. Mannor, and Y. Mansour, "Thompson sampling for complex online problems," in *International Conference on Machine Learning*, 2014.

[41] D. Russo, B. Van Roy, A. Kazerouni, I. Osband, and Z. Wen, "A tutorial on thompson sampling," *arXiv:1707.02038*, 2017.

[42] A. Gopalan and S. Mannor, "Thompson sampling for learning parameterized markov decision processes," in *Conference on Learning Theory*, 2015.

[43] A. Bai, F. Wu, Z. Zhang, and X. Chen, "Thompson sampling based monte-carlo planning in pomdps," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 24, no. 1, 2014.

[44] J. Leike, T. Lattimore, L. Orseau, and M. Hutter, "Thompson sampling is asymptotically optimal in general environments," *arXiv:1602.07905*, 2016.

[45] K. Kandasamy, A. Krishnamurthy, J. Schneider, and B. Póczos, "Parallelised bayesian optimisation via thompson sampling," in *International Conference on Artificial Intelligence and Statistics*, 2018.

[46] K. Kandasamy, W. Neiswanger, R. Zhang, A. Krishnamurthy, J. Schneider, and B. Poczos, "Myopic posterior sampling for adaptive goal oriented design of experiments," in *International Conference on Machine Learning*, 2019.

[47] R. Coulom, "Computing "elo ratings" of move patterns in the game of go," *ICGA journal*, vol. 30, no. 4, 2007.

[48] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, 2016.

[49] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, 2017.

[50] J.-Y. Audibert, R. Munos, and C. Szepesvari, "Use of variance estimation in the multi-armed bandit problem," in *NIPS Workshop on On-line Trading of Exploration and Exploitation*, 2006.

[51] D. Shah, Q. Xie, and Z. Xu, "Non-asymptotic analysis of monte carlo tree search," in *Abstracts of the 2020 SIGMETRICS/Performance Joint International Conference on Measurement and Modeling of Computer Systems*, 2020.