2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

A Contextual Bandit Approach for Learning to Plan in Environments with Probabilistic Goal Configurations

Sohan Rudra*, Saksham Goel*, Anirban Santara*, Claudio Gentile*, Laurent Perron, Fei Xia,
Vikas Sindhwani, Carolina Parada, Gaurav Aggarwal
Google

***Abstract*— Object-goal navigation (Object-nav) entails searching, recognizing and navigating to a target object. Object-nav has been extensively studied by the Embodied-AI community, but most solutions are often restricted to considering static objects (e.g., television, fridge, etc.). We propose a modular framework for object-nav that is able to efficiently search indoor environments for not just static objects but also movable objects (e.g. fruits, glasses, phones, etc.) that frequently change their positions due to human intervention. Our contextual-bandit agent efficiently explores the environment by showing optimism in the face of uncertainty and learns a model of the likelihood of spotting different objects from each navigable location. The likelihoods are used as rewards in a weighted minimum latency solver to deduce a trajectory for the robot. We evaluate our algorithms in two simulated environments and a real-world setting, to demonstrate high sample efficiency and reliability.**

I. Introduction

Personal robotics is an important domain of Embodied AI [1] that aims at enhancing human productivity by developing physical assistants for everyday tasks. In this paper, we study the task of Object-Goal Navigation (object-nav) [2], [3], which is a core component of many personal robotics applications like Mobile Manipulation [4], Embodied Question Answering [5], and Vision-and-Language Navigation [6]. Object-nav is defined as the task of searching (and optionally, retrieving) a given object within a designated space, e.g., indoor spaces like homes or offices, which are typically *known* environments. The target object is often specified in natural language, e.g., "a blue soccer ball with stripes". This work is motivated by an everyday *object-nav* task: searching for our keys or cellphone at home. Analogous to how humans search, we propose an algorithm that learns to look for these items in the most likely places first.[1]

Object-nav algorithms work by learning semantic relationships between the target object and the topology of the environment. They can be classified into two categories: map-based and map-free [7]. Map-free algorithms [8], [9], [10], [11], [12], [13], [14], [2] do not require a map of the environment and can decide where to go based directly on the current observations and past memories without having to maintain a global representation of the environment. This requires them to solve the problem of localization and mapping in conjunction with the object-nav problem, making their sample complexity very high. As a result, these algorithms are often trained in simulation, with inevitably simplified human and environment representations, resulting in regressions in real environments due to the sim-to-real gap [15], which is further exacerbated by the fact that many of them work with low-level observations.

Map-based algorithms [16], [17], [18], [19], [20], [21] assume that a map of the environment is available at the time of path-planning. The map could be an occupancy map showing the probability of obstacles being at each location. It could also be a topological map [22] that is comprised of a graph where nodes represent characteristic places and edges contain reachability information (distances, times, etc.) between pairs of nodes. A few of these algorithms construct a map of the current region before planning in that region [23], [24], [25], [2]. Map-based methods are typically modular, hence, sample-efficient [26] and easier to deploy in the real world. However, the validity of the solution devised by these algorithms is a strong function of the accuracy of the map and localization of the robot. Thanks to advancements in Simultaneous Localization and Mapping (SLAM) algorithms [27], constructing an occupancy map of reasonable accuracy has become relatively easy in modern robotic systems.

In this paper, we aim to achieve robust Object-Goal Navigation in indoor human-centric environments. Our algorithms are modular and map-based. They assume access to a binary 2D occupancy map of the environment with navigable and non-navigable parts marked out. In our day to day life, quite frequently, we have to search portable objects that we happened to lose track of. We will refer to such objects as "movable objects". Let us consider the example of *cellphone*. We begin our search by trying to recall a list of places that we are used to finding our *cellphone* at. The algorithm presented in this paper takes a similar approach where we aim to model the likelihood of spotting a given object from different places in the environment. For example, a *cellphone* is more likely to be found on the study-table, work-desk and bedside-table, rather than in the kitchen floor or on top of the refrigerator. These likelihoods are learned online as the agent explores a realistic environment. For stationary objects, we just assign probability values of 0 or 1. These likelihoods are then used for planning a path that minimizes the average distance travelled by the agent to reach the target object.

Fig. 1 provides an overview of our approach. (1) The robot is randomly initialized in the environment with the

(*) Denotes equal contribution. Author contact: santara@google.com
[1]See website https://sites.google.com/view/find-my-glasses/home

task of finding a given target object. (2) A set of reachable vantage points are sampled across the entire environment using the current $2D$ occupancy map via farthest point sub-sampling [28], [29]. (3) A Contextual Bandit Agent [30] estimates the likelihood of spotting the target object from each vantage point. (4) A Weighted Minimum Latency Problem (WMLP) solver [31] is used to generate an ordering of the vantage points taking into account their likelihood scores, the initial position of the robot and the geometry of the room. Finally, (5) The robot visits the vantage points in the planned order while inspecting its surroundings. As soon as it spots the object, it heads directly to it.

A prominent challenge faced by learning-based algorithms in robotics is bridging the sim-to-real gap [15]. Learning agents are usually trained in a simulator (sim) like Matterport [32], AI2Thor [33], Gibson [34] and Habitat [35] before deploying in the real world. However, due to limited fidelity of a simulator, observations of the same event might be different in the simulator and in reality (domain mismatch). The modular design of our approach not only allows us to disentangle systems which are sensitive to sim-to-real change (e.g., vision) from the systems which are not (e.g., navigation) but also to switch out domain-specific systems when transferred from sim to real. Moreover, our formulation of object-nav as a bandit learning problem significantly reduces the sample complexity, making it feasible to train an agent in real from scratch.

II. Model and Algorithm

Model. We formalize our problem as follows. In the 2D occupancy map of the environment, let us denote all the points by a set $\mathcal{X}$. Set $\mathcal{X}$ is partitioned into the set of *feasible* points $\mathcal{F}$ (the points the robot can freely navigate) and the set of *non-feasible* points $\mathcal{N}$ (the points occupied by obstacles like furnitures), so that $\mathcal{X} = \mathcal{F} \cup \mathcal{N}$, and $\mathcal{F} \cap \mathcal{N} = \emptyset$. The three sets $\mathcal{X}$, $\mathcal{F}$, and $\mathcal{N}$ are available to us before planning. Each $x \in \mathcal{F}$ comes with a *visibility set* $V(x) \subseteq \mathcal{X}$, that is, a set of points that the robot can inspect while standing[2] at x. For concreteness, visibility is defined in terms of Euclidean distance as $V(x) = \{x' \in \mathcal{X} : ||x - x'|| \leq r_{vis}\}$, for some visibility range $r_{vis} > 0$, e.g., $r_{vis} = 2.5$ meters (the effective range of the object detectors on the system).

We have n movable objects of interest (glasses, keys, etc.). We use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. For each pair $(i, x) \in [n] \times \mathcal{F}$, denote by $p_i(x)$ the probability that the robot spots object i while standing at position x. These probabilities are in turn defined by n (unknown) probability distributions $\{\mathbb{P}_i,\ i \in [n]\}$, with support $\mathcal{X}$, that determine where objects are located, so that $p_i(x) = \mathbb{P}_{y \sim \mathbb{P}_i}(y \in V(x))$. Notice that an object can, in principle, be anywhere in the scene. The probabilities $p_i(x)$ are unknown to the planner, and have to be learned through interactions with the environment. We model them as $p_i(x) = f(\phi(i, x); \theta)$, where $\phi : [n] \times \mathbb{R}^d \to \mathbb{R}^D$ is a mapping that featurizes the pair (i, x) into a D-dimensional real vector, for some feature dimension $D \geq d$, $f : \mathbb{R}^D \times \mathbb{R}^m \to [0, 1]$ is a known function, and $\theta \in \mathbb{R}^m$ is an unknown vector of parameters, for some parameter dimension m. For instance, $f(\phi; \theta) = \sigma(\theta^\top \phi)$, where σ is the sigmoidal function $\sigma(z) = \frac{e^{zs}}{1+e^{zs}}$ with slope s at the origin, and $m = D$. Our theoretical analysis uses the above generalized linear model, while in our experimental evaluation, we compare both linear and neural models for $f(\cdot; \theta)$.

Planning and Regret. In each navigation episode $t = 1, 2, \ldots$, there will be only one object $i_t \in [n]$ in the scene,[3] and the identity of this object is *known* to the robot. The environment generates the position y_t of object i_t by drawing y_t from $\mathbb{P}_{i_t}$. Let $x_{0,t} \in \mathcal{F}$ be the starting position of the robot in episode t. The algorithm begins by sampling a total of k vantage points $x_{1,t}, \ldots, x_{k,t} \in \mathcal{F}$. The planner has to generate a path $J_t = \langle x_{\pi_t(1),t}, \ldots, x_{\pi_t(k),t} \rangle$ across them, where $\pi_t(\cdot)$ is a permutation of the indices $\{0\} \cup [k]$, with $\pi_t(0) = 0$. The robot traverses the path in the order dictated by J_t, and stops as soon as the object is spotted, as allowed by the visibility structure $V(x_{\ell,t})$, $\ell \in \{0\} \cup [k]$. It is also reasonable to admit that the robot may incur some detection failures during an episode, an event we denote by $\mathcal{E}_t$, to which we shall assign an independent probability $\mathbb{P}(\mathcal{E}_t)$ to occur. We henceforth drop the episode subscript t for notational convenience.

The *path length* loss $L(y, J)$ of J is the actual distance traversed by the robot over the points $x_0, x_1, \ldots, x_k$ *before* spotting object i in position y. On the other hand, if the object is not found, it is reasonable to stipulate that the loss incurred will be a large number $L_M > \sum_{\ell=1}^{k} \sum_{j=1}^{\ell} \mathrm{dist}_\star(x_{\pi(j-1)}, x_{\pi(j)})$, bigger than the total path length of any length-k path $\langle x_{\pi(1)}, \ldots, x_{\pi(k)} \rangle$. Overall

$$L(y, J) = (1 - \mathbb{1}\{\mathcal{E}\}) \times \left(\sum_{\ell=1}^{k} \underbrace{\mathbb{1}\{y \in V(x_{\pi(\ell)}) \setminus (V(x_{\pi(0)}) \cup \ldots \cup V(x_{\pi(\ell-1)}))\}}_{V(x_{\pi(\ell)}) \text{ is the first ball in the traversal order where object is located}} \times \sum_{j=1}^{\ell} \mathrm{dist}_\star(x_{\pi(j-1)}, x_{\pi(j)}) \right) + \mathbb{1}\{\mathcal{E}\} L_M \ , \tag{1}$$

where $\mathbb{1}\{\cdot\}$ is the indicator function of the predicate at argument, and $\mathrm{dist}_\star(x_1, x_2)$ denotes the $A^\star$ distance between the two vantage points x_1 and x_2 on the scene, i.e., the robot path length between x_1 and x_2 (notice that $\mathrm{dist}_\star(x_1, x_2) \geq ||x_1 - x_2||$). Observe that, in the absence of a failure, we are assuming the object will eventually be found during each episode. Hence, a failure will be ascertained only at the end of an episode. We approximate the above by disregarding the overlap among the visibility balls $V(x_{\pi(\ell)})$ (hence somehow assuming these balls do not influence one another), and then take expectation over $y \sim \mathbb{P}_i$ and the independent Bernoulli variables $\mathbb{1}\{\mathcal{E}\}$, with expectation $\mathbb{P}(\mathcal{E})$. This yields

[2]We are assuming here that the robot can sample from x any pose via 360 degree rotation.

[3]This is not a strict requirement, our analysis can be extended to multiple instances of the same object simultaneously present on the scene, since the planner operates on the domain of distributions rather than exact numbers.

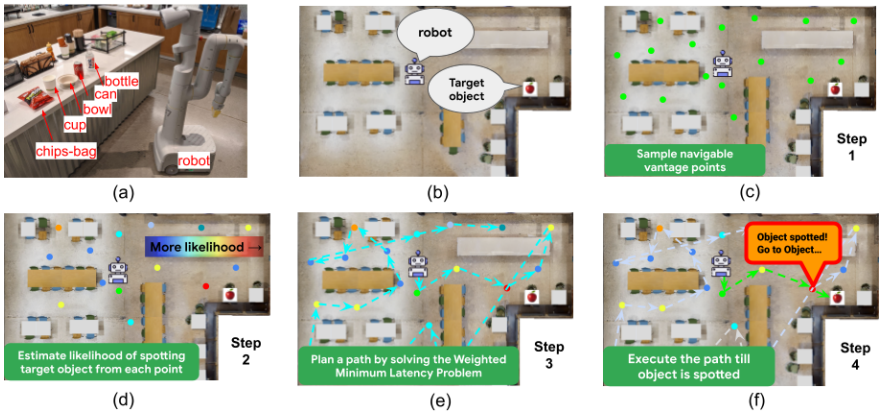


Fig. 1: Overview of our approach. (a) Picture of our robot and the target objects studied in our experiment. (b) The robot is randomly initialized in the environment with the task of finding a given target object. (c) A set of reachable vantage points (green dots) are sampled across the entire environment using the current $2D$ occupancy map. (d) The Contextual-Bandit agent estimates the likelihood of spotting the target object from each vantage point. (e) A Weighted Minimum Latency Problem (WMLP) solver is used to generate an ordering of the vantage points taking into account their likelihood scores, the initial position of the robot and the geometry of the room. (f) The robot visits the vantage points in the planned order while inspecting its surroundings. As soon as it spots the object, it heads directly to it.

the (approximate) average path length

$$\mathbb{E}_i[L(y,J)] = \mathbb{P}(\mathcal{E})\, L_M \tag{2}$$
$$+ (1-\mathbb{P}(\mathcal{E}))\left(\sum_{\ell=1}^{k} p_i(x_{\pi(\ell)}) \sum_{j=1}^{\ell} \mathrm{dist}_\star(x_{\pi(j-1)}, x_{\pi(j)})\right),$$

where $\mathbb{E}_i[\cdot]$ is a short-hand for $\mathbb{E}_{y\sim\mathbb{P}_i,\mathcal{E}}[\cdot]$. We are now in a position to define our benchmark performance measure against which regret will be defined. The *benchmark planner* knows the distributions $\{\mathbb{P}_i,\ i \in [n]\}$ and the probability $\mathbb{P}(\mathcal{E})$, and computes a path $J^\star = \langle x_1^*, \ldots, x_k^\star\rangle$ whose elements are taken from $\mathcal{F}$, such that $J^\star$ minimizes $\mathbb{E}_i[L(y,J)]$ over all length-k paths (and permutations thereof) that can be constructed out of points from $\mathcal{F}$. Given a sequence of episodes $t = 1, \ldots, T$ with corresponding objects $i_1, \ldots, i_T$ (and starting positions $x_{0,1}, \ldots, x_{0,T}$), we define the cumulative *regret* $R_T(J_1, \ldots, J_T)$ of a planner that generates paths $J_1, \ldots, J_T$ as

$$R_T(J_1,\ldots,J_T) = \sum_{t=1}^{T} \mathbb{E}_{i_t}[L(y_t,J_t)] - \mathbb{E}_{i_t}[L(y_t,J_t^\star)]\ .$$

We would like this cumulative regret to be *sublinear* in T with high probability (in the random draw of position y_t at the beginning of each episode t). Notice that the last term in the RHS of (2) is independent of J, hence L_M will play no role in the regret computation.

Algorithm. Our "theoretical" algorithm is described as Algorithm 1. The algorithm operates on an ϵ-cover[4] $\mathcal{F}_\epsilon$ of $\mathcal{F}$ and generalized linear probabilities $p_{i_t}(x) = \sigma(\theta^\top \phi(i_t, x))$. The algorithm replaces the above true probabilities in the average path length (2) with lower confidence estimations $\sigma\Big(\widehat{\Delta}_t(x) - \epsilon_t(x)\Big)$, and then computes J_t by minimizing (2) over the choice of k points within $\mathcal{F}_\epsilon$, as well as their order (permutation π). A sequence of signals $\langle s_{1,t}, \ldots, s_{k'_t,t}\rangle = \langle -1, \ldots, -1, +1\rangle$ observed during episode t is associated with the successful path J_t in that a sequence of $k'_t - 1$ negative signals ("-1" = object not spotted from $x_{\pi_t(\ell),t}$, for $\ell = 1, \ldots, k'_t - 1$) precede a positive signal ("$+1$" = object spotted at $x_{\pi_t(k'_t),t}$). On the other hand, when the object is not found throughout the length-k path, the sequence of signals becomes $\langle s_{1,t}, \ldots, s_{k,t}\rangle = \langle -1, \ldots, -1, -1\rangle$ (this happens with probability $\mathbb{P}(\mathcal{E})$).

When the object is found, Algorithm 1 uses the observed signals to update over time a D-dimensional weight vector $\widehat{\theta}$, and a $(D \times D)$-dimensional matrix M. Vector $\widehat{\theta}_t$ is used to estimate $\theta^\top \phi(i_t, x)$, through $\widehat{\Delta}_t(x)$, while matrix M_t delivers a standard confidence bound via $\epsilon_t^2(x)$. The update rule implements a second-order descent method on logistic loss trying to learn the unknown vector θ out of the signals $s_{j,t}$. In particular, the update $\widehat{\theta}_{c_t+j-1} \rightarrow \widehat{\theta}_{c_t+j}$ is done by computing a standard online Newton step (e.g., [36]). Notice that at each episode t, both matrix M and vector $\widehat{\theta}$ get updated $m_t = k'_t$ times, which corresponds to the number of (valid) signals received in that episode. Counter c_t accumulates the number of such updates across (non-failing) episodes. On the contrary, when the object is not found, we know that there has been a failure, hence we disregard all (negative) signals received and jump to the next episode with no updates ($m_t = 0$).

From a computational standpoint, calculating J_t as described in Algorithm 1 is hard, since the planning problem the algorithm is solving at each episode is essentially equivalent to a *Weighted Minimum Latency Problem* (WMLP) [31], also called *traveling repairman problem*, which, on a generic metric space is NP-hard and also MAX-SNP-hard [37]. Fast algorithms are available only for very special metric graphs, like paths [38], [39] edge-unweighted trees [40], trees of diameter 3 [37], trees of constant number of leaves [41], and the like [42]. Even for weighted trees the problem remains NP-hard [43]. Approximation algorithms are indeed available [44], but they are not practical enough for real-world deployment. In our experiments (Sections III–IV),

[4]Recall that $\mathcal{F}_\epsilon$ is an ϵ-cover of $\mathcal{F}$ if $\mathcal{F}_\epsilon \subseteq \mathcal{F}$ and for all $x \in \mathcal{F}$ there is $x' \in \mathcal{F}_\epsilon$ for which $\mathrm{dist}_\star(x,x') \le \epsilon$. It is easy to see that, given a 2D-scene, the cardinality $|\mathcal{F}_\epsilon|$ of $\mathcal{F}_\epsilon$ is $O(1/\epsilon^2)$.

we implement and compare fast planning approximations to Algorithm 1.

In both the planning and the training of the contextual bandit algorithm, we are using the average path length as a minimization objective because it is easier for the WMLP solver to handle. However, when it comes to evaluating performance in our experiments, we use the Success weighted by Path Length (SPL) metric [45] because it is a more established metric in the object-nav literature.

Regret Analysis. From a statistical standpoint, Algorithm 1 is a simplified version of the slightly more complex algorithm which is the one our regret analysis applies to (details omitted due to space limitations).

Theorem 1: Let $D_M = \max_{x,x' \in \mathcal{F}} \text{dist}_\star(x, x')$ be the diameter of the scene. Also, let the feature mapping $\phi : [n] \times \mathbb{R}^d \to \mathbb{R}^D$ be such that $||\phi(i, x)|| \leq 1$ for all $i \in [n]$ and $x \in \mathcal{F}$, and let constant B be such that $||\theta|| \leq B$. Then a variant of Algorithm 1 exists that operates in episode t with an ϵ-covering $\mathcal{F}_\epsilon$ of $\mathcal{F}$ with $\epsilon = k/\sqrt{t}$, such that with probability at least $1 - \delta$, with $\delta < 1/e$, the cumulative regret of this algorithm satisfies, on any sequence of objects $i_1, \ldots, i_T$,

$$R_T(J_1, \ldots, J_T) = O\Big((1-p)k^2\sqrt{T} + D_M\, k\, \sqrt{(1-p)kT\, \alpha(k, D, T, \delta, B)\, D \log(1+kT)}\Big)\,,$$

where $\alpha(k, D, T, \delta, B) = O\left[e^{2B}\left(k + D\,\log\left(1 + \frac{kDT}{\delta}\right)\right)\right]$, and $p = \mathbb{P}(\mathcal{E}_t)$ is the (constant) failure probability. In the above, the big-oh notation hides additive and multiplicative constants independent of T, D, B, k, p, and δ.

In a nutshell, the above analysis provides a high-probability regret guarantee of the form $k^2 D\sqrt{T}$. The learning rate η and the exploration parameter α in Algorithm 1 are hyper-parameters.

III. Proposed Methodology

In this section, we present a practical algorithm that replicates Algorithm 1. We have three main steps: (a) Sampling k vantage-points that maximally cover the navigable part $\mathcal{F}$ of the given scene; (b) Importance assignment to the sampled points using an online learning contextual bandit algorithm; and (c) Path planning through the vantage points. We use Model Predictive Control (MPC) [46], [47], [48], [49] to execute the selected path. This guarantees collision-free shortest-path trajectories between vantage points while satisfying given safety constraints, optimality criteria, and kinodynamic models. The resulting system is interpretable and safe. We will elaborate on each of these.

A. Sampling vantage points

Our algorithm begins by sampling a sparse set of vantage points that are navigable and spread all over the scene in a way that the agent's vision is able to cover the space to a large extent by visiting these points and looking around. First, we extract all the navigable points (at a resolution of 0.1 meter) from the robot's occupancy map. Next, we select our vantage points using the Farthest Point Subsampling (FPS) algorithm. FPS is a simple, yet effective

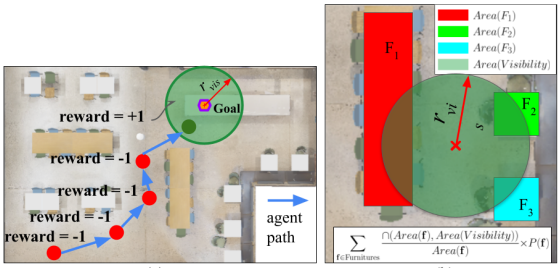


Fig. 2: **(a)** Positive sample augmentation for improved sample efficiency. Radius r_{vis} is the robot's visibility range. **(b)**: Calculation of ground-truth likelihood scores.

method of extracting a small number of points from a given point-cloud to cover the extremities of the point-cloud and capture prominent point features. Given a point cloud of size N and a distance metric d_f, the FPS algorithm works by picking a starting point (randomly or by some heuristic criterion) from the point cloud and iteratively adding new points that are maximally distant – according to d_f – from the current set of selected points, till the required number of points k is reached. This method is popularly used in image processing [28] and computer vision [29]. Since FPS requires $O(N)$ distance calculations in each iteration, we choose the Eucledian distance for d_f in our experiments for computational ease.

B. Node Level Importance Assignment

Once we have the vantage points, we estimate the importance of each point in the context of the target object. An important vantage point is one that has a high likelihood of the robot spotting the target object standing there. For ease of modelling, we assume that our robot is able to spot the target object equally well from any yaw-angle if it is within its visibility range r_{viz}. In order to estimate the likelihood of the robot spotting an object (which can be movable) from a given point, we need to explore the environment. For efficient exploration, we formulate the problem as a contextual bandit with vantage point as an arm and follow the principle of *optimism in the face of uncertainty* (e.g., [50]) as described in Algorithm 1. Each vantage point is described by a vector with positional, geometric and semantic features of the point along with the identity of the target object. We triangulate the position of the target object once the agent spots it from a vantage point. In order to improve learning efficiency, we assign positive training signal ("+1") to all the navigable points within r_{vis} radius from the object. Fig. 2 (a) illustrates this procedure.

In our experiments, we either rely on a generalized linear model (as in Algorithm 1) for estimating the importance of points or on a simple two-layer fully connected neural network model, approximated via Neural Tangent Kernels (NTK), as contained in Algorithms 1-2 in [51] ("Neural") for computing uncertainties $\epsilon_t(x)$ (Step 2 in Algorithm 1) and updating $\widehat{\theta}$ (Step 5 in Algorithm 1).

C. Planning

After estimating the importance-scores of the vantage points, we derive their sequence of visitation by representing

Algorithm 1 Simplified contextual bandit planning algorithm.

Input: Learning rate $\eta > 0$, exploration parameter $\alpha \geq 0$, ϵ-cover $\mathcal{F}_\epsilon$ of $\mathcal{F}$, $\epsilon > 0$, path length k.
Init: $M_0 = kI \in \mathbb{R}^{D\times D}$, $\widehat{\theta}_1 = 0 \in \mathbb{R}^D$, $c_1 = 1$.
For $t = 1, 2, \ldots, T$

1) Get object identity i_t , and initial position of the robot $x_{0,t}$;
2) **For** $x \in \mathcal{F}$, set
$$\widehat{\Delta}_t(x) = \widehat{\theta}_{c_t}^\top \phi(i_t, x) \qquad \text{and} \qquad \epsilon_t^2(x) = \alpha\, \phi(i_t, x)^\top M_{c_t-1}^{-1} \phi(i_t, x)$$
3) Compute $J_t = \langle x_{\pi_t(1),t}, \ldots, x_{\pi_t(k),t} \rangle$ as `//solve WMLP at episode t`
$$J_t = \arg \min_{\substack{x_1 \ldots x_k \in \mathcal{F}_\epsilon \\ \text{permutation } \pi}} \sum_{\ell=1}^{k} \sigma\Big(\widehat{\Delta}_t(x_{\pi(\ell)}) - \epsilon_t(x_{\pi(\ell)})\Big) \sum_{j=1}^{\ell} \text{dist}_\star(x_{\pi(j-1)}, x_{\pi(j)})$$
4) Observe signal $\begin{cases} \langle s_{1,t}, \ldots, s_{k'_t,t} \rangle = \langle -1, \ldots, -1, +1 \rangle & \text{set } m_t = k'_t \\ \text{or} \\ \langle s_{1,t}, \ldots, s_{k,t} \rangle = \langle -1, \ldots, -1, -1 \rangle & \text{set } m_t = 0 \end{cases}$
5) **For** $j = 1, \ldots, m_t$ (in the order of occurrence of items x_j in J_t) update:
$$M_{c_t+j-1} = M_{c_t+j-2} + \phi(i_t, x_j)\phi(i_t, x_j)^\top,$$
$$\widehat{\theta}_{c_t+j} = \widehat{\theta}_{c_t+j-1} + \eta\, \sigma\Big(-s_{j,t}\, \widehat{\theta}_{c_t+j-1}^\top \phi(i_t, x_j)\Big)\, s_{j,t}\, M_{c_t+j-1}^{-1} \phi(i_t, x_j)$$
6) $c_{t+1} \leftarrow c_t + m_t$.

them as a graph – where each node is a vantage point and the edges contain the $A^\star$ distance between vantage points – and solving the Weighted Minimum Latency Problem (WMLP) (e.g., [37], [31]). WMLP tries to minimize the average waiting time of each node in a graph, weighted by its importance score being reached by a travelling agent. In our case, the solution of the WMLP minimizes the average distance traveled by the robot to reach the target object – the average being over the position of the object and the initial position of the robot. We consider two different ways of solving the WMLP in our setting.

CP-SAT. The first approach directly faces the underlying optimization problem. We relied on a satisfiability (SAT)-based constraint programming (CP) solver from Google OR-Tools [52] that uses a lazy clause generation solver on top of a SAT solver to reach its solution conditioned on vantage-points, starting position, and predicted relevance of the points. Although this approach is principled, its running time increases dramatically with the number of points.

One-step greedy. Starting at x, the next point x' is chosen to maximize a weighted combination of the estimated likelihood $\hat{p}_i(x')$ of spotting the target object i, and the inverse of the $A^\star$ distance traveled to reach it:

$$x' = \underset{x'' \in \{\text{unvisited points}\}}{\arg\max} \frac{\alpha_p}{\text{dist}_\star(x, x'')} + (1 - \alpha_p)\, \hat{p}_i(x'') \ , \quad (3)$$

where $\alpha_p \in [0, 1]$ is a hyper-parameter whose value should be chosen to achieve a good trade-off between minimizing the traveled distance in the next step and maximizing the likelihood of spotting the target object. The case of $\alpha_p = 0$ corresponds to greedily choosing the unvisited point with the highest estimated likelihood, while the case of $\alpha_p = 1$ would greedily choose the closest unvisited point. In the above, $\hat{p}_i(x)$ will be computed as suggested by Algorithm 1 via a lower confidence scheme of the form $\hat{p}_i(x) = \sigma\Big(\widehat{\Delta}(x) - \epsilon(x)\Big)$. The one-step greedy approach myopic, as it greedily optimizes for the next step only, but is also much faster to run, hence it should be interpreted here as a fast approximation to the CP-SAT solution.

IV. Experiments

We run experiments in 2 simulated and 1 real office kitchen environments. The environments have areas $80m^2$, $120m^2$ & $120m^2$, respectively. Each experiment involves training the agent for 200 episodes followed by evaluation with frozen parameters in the same environment. For the real kitchen experiment, we train the bandit model on a snapshot of the map in our simulator and evaluate in the real environment. The simulated environments have photo-realistic scenes generated from Matterport scans [53] and Bullet [54] based physics simulation. Our robot is a differential-drive wheeled robot, which has a 3D LiDAR in the front, and depth sensors mounted on its head. It is capable of accurate localization and safe point-to-point navigation. We use Model Predictive Control [55] to execute a path. Each vantage point x is described by a feature-vector $\phi(i, x)$ consisting of a one-hot encoding of the target object i and a flattened 16×16 patch of the wall-distance map centered at the point x. A sinusoidal positional encoding [56] vector is appended to represent the location of the point in the map. Map-resolution and positional encoding dimension are hyper-parameters. The normalizer for the feature-vector is also a hyper-parameter that is chosen from among: a) zero mean, unit standard-deviation, and b) unit l^2-norm. All hyper-parameters (η and α for Algorithm 1, α_p for One-step greedy, the learning rate and batch size in Algorithms

TABLE I: Experimental comparison of performance of our agents on the 3 environments against the metrics mentioned.

	Real Kitchen Env.			Simulated Kitchen 1							Simulated Kitchen 2						
	TSP	Neural		TSP	Gen-Lin		Neural		GT-Scores		TSP	Gen-Lin		Neural		GT-Scores	
		Greedy	CP-SAT		Greedy	CP-SAT	Greedy	CP-SAT	Greedy	CP-SAT		Greedy	CP-SAT	Greedy	CP-SAT	Greedy	CP-SAT
Train SPL	-	-	-	-	0.32	0.31	0.30	0.27	-	-	-	0.26	0.27	0.23	0.13	-	-
Eval. Succ.	0.88	0.80	0.92	0.89	0.88	0.89	0.90	0.90	0.90	0.90	0.56	0.80	0.78	0.74	0.67	0.82	0.80
Eval. SPL	0.37	0.38	0.42	0.38	0.42	0.47	0.40	0.39	0.43	0.51	0.22	0.26	0.29	0.25	0.20	0.33	0.34

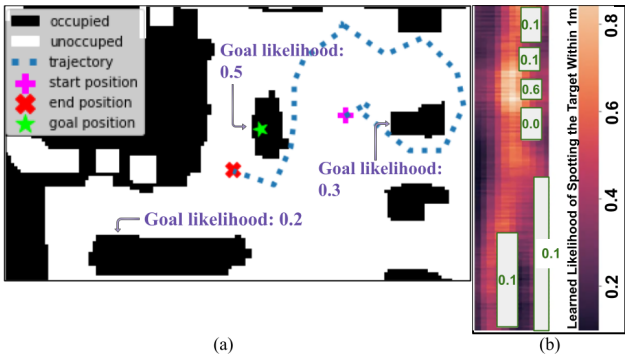
1-2 in [51] for Neural, the positional embededing size and the feature vector normalization for mapping $\phi(i, x)$, and the sigmoidal scale s for the sigmoid in Algorithm 1) are tuned across suitable ranges using a Gaussian-Process Bandit based Blackbox optimizer [57] to maximize **success weighted by path length (SPL)** [58] over the training episodes. Leveraging the modular design of our approach, we train the real-kitchen agent in simulated Kitchen Environment 2, thereby reducing the turnaround time of the overall experiments.

For training in simulation, we consider the agent has successfully reached its goal if and when it visits a vantage point that is within r_{vis} radius from the target object. For learning good quality likelihood maps, we set $r_{vis} = 1$m during training although the default value of $r_{vis} = 2.5$m is used during evaluation in the simulated environments. For success during evaluation in the real environment, the robot must visually detect the target object and drive up to a grasping range of the object. For object detection, we use our implementation of the ViLD detector [59] that has a true positive rate of 84.6% across our test objects.

We have five categories of *movable* target objects: "bottle", "can", "cup", "bowl" and "chips-bag" each of which has the same frequency of occurrence across episodes and in any given episode we have a single instance of the target object in the environment. The same object may appear in different locations in subsequent episodes. We compare the performances of our proposed framework using the generalized linear ("Gen-Lin") and neural ("Neural") models for training the contextual bandit agent and "CP-SAT" and one-step greedy ("Greedy") algorithms for path planning to a purely geometric approach that solves the Travelling Salesman Problem [60] ("TSP"). We assign a time budget of 30 seconds to the CP-SAT solver to have a realistic bound on robot response time. Each algorithm is evaluated over 50 episodes in real and 300 episodes in simulated environments. Fig. 3 (a) shows a sample trajectory during the real kitchen evaluation. Fig. 3 (b) shows a sample learned likelihood map for the Simulated Kitchen-1 environment.

We compare the performance of the agents on the following metrics: a) rate of success during evaluation ("Eval. Succ."), b) (SPL) [58] during training ("Train SPL"), and c) SPL during evaluation ("Eval. SPL"). Table I presents the results of our first study, where we compare different learning and planning approaches for an arbitrary spatial distribution of test objects. The columns labeled "GT-Scores" use ground truth likelihoods of the vantage points computed as shown in 2 (b). For both training and evaluation, we use 25 vantage points for the real environment and the kitchen environment 1 and 50 for Simulated Kitchen 2.

Our first observation is the significant improvement in



(a) (b)

Fig. 3: **(a)** Real-kitchen sample trajectory for CP-SAT planner with the Neural model. **(b)** Heat map of the estimated likelihoods in Kitchen Environment 1 along with ground truth likelihoods (in green) of the goal object occurring on the surface of each furniture in random positions.

performance achieved by our planners using "GT-Scores" over "TSP" in all the environments and this validates the importance of estimating the importance of the vantage points in the context of the target object in addition to optimizing for the room geometry. The performances for "GT-Scores" provide an upper bound for the agents that learn the likelihood function through exploration. Although the performance of "CP-SAT" shines in the real evaluation, under the planning time budget of 30 seconds, the performance of our proposed one-step greedy solver regularly matches up and often beats the CP-SAT solver, especially in larger and more cluttered Simulated Kitchen 2 environment. The performance of the "Neural" model often lags behind the "Gen-Lin" model due to the computational limitations imposed by the NTK approximation.

V. Limitations and Ongoing Work

Our proposed approach can get adversely affected due to: 1) detection failure, 2) slowness of WMLP, and 3) early stopping of CP-SAT solver (not running the CP-SAT solver until the end may give us feasible but poor solutions). Inclusion of orientation along with the robot's base position can help in mitigating missed detections. Using richer feature embeddings can also improve object detection from distance. Related to that, the choice of the network architecture in the Neural model is severely limited by our usage of the NTK approximation to compute confidence bounds. Leveraging more time-efficient approximation schemes may allow for more complex (and potentially more accurate) network architectures. Faster convergence is possible in training by using a likelihood-guided sampling scheme but this may also create opportunities for local minima. These are among the missing aspects we are currently investigating.

REFERENCES

[1] R. Pfeifer and F. Iida, "Embodied artificial intelligence: Trends and challenges," *Embodied artificial intelligence*, pp. 1–26, 2004.

[2] D. S. Chaplot, D. Gandhi, A. Gupta, and R. Salakhutdinov, "Object goal navigation using goal-oriented semantic exploration," in *In Neural Information Processing Systems (NeurIPS)*, 2020.

[3] X. Ye and Y. Yang, "From seeing to moving: A survey on learning for visual indoor navigation (vin)," *arXiv preprint arXiv:2002.11310*, 2020.

[4] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.

[5] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra, "Embodied question answering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1–10.

[6] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. Van Den Hengel, "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3674–3683.

[7] F. Bonin-Font, A. Ortiz, and G. Oliver, "Visual navigation for mobile robots: A survey," *Journal of intelligent and robotic systems*, vol. 53, no. 3, pp. 263–296, 2008.

[8] A. Talukder, S. Goldberg, L. Matthies, and A. Ansar, "Real-time detection of moving objects in a dynamic scene from moving robotic vehicles," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, vol. 2. IEEE, 2003, pp. 1308–1313.

[9] J. Santos-Victor and G. Sandini, "Visual-based obstacle detection: a purposive approach using the normal ow," in *Proc. of the International Conference on Intelligent Autonomous Systems, Karlsruhe, Germany*. Citeseer, 1995.

[10] A. Mousavian, A. Toshev, M. Fišer, J. Košecká, A. Wahid, and J. Davidson, "Visual representations for semantic target driven navigation," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8846–8852.

[11] T. Chen, S. Gupta, and A. Gupta, "Learning exploration policies for navigation," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/pdf?id=SyMWn05F7

[12] S. K. Ramakrishnan, D. S. Chaplot, Z. Al-Halah, J. Malik, and K. Grauman, "Poni: Potential functions for objectgoal navigation with interaction-free learning," in *Computer Vision and Pattern Recognition (CVPR), 2022 IEEE Conference on*. IEEE, 2022.

[13] A. Wahid, A. Stone, K. Chen, B. Ichter, and A. Toshev, "Learning object-conditioned exploration using distributed soft actor critic," *CoRR*, vol. abs/2007.14545, 2020.

[14] G. Georgakis, B. Bucher, K. Schmeckpeper, S. Singh, and K. Daniilidis, "Learning to map for active semantic goal navigation," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=swrMQttr6wN

[15] S. Höfer, K. Bekris, A. Handa, J. C. Gamboa, M. Mozifian, F. Golemo, C. Atkeson, D. Fox, K. Goldberg, J. Leonard, *et al.*, "Sim2real in robotics and automation: Applications and challenges," *IEEE transactions on automation science and engineering*, vol. 18, no. 2, pp. 398–400, 2021.

[16] M. Meng and A. C. Kak, "Neuro-nav: a neural network based architecture for vision-guided mobile robot navigation using non-metrical models of the environment," in *[1993] Proceedings IEEE International Conference on Robotics and Automation*. IEEE, 1993, pp. 750–757.

[17] M. Meng and A. Kak, "Mobile robot navigation using neural networks and nonmetrical environmental models," *IEEE Control Systems Magazine*, vol. 13, no. 5, pp. 30–39, 1993.

[18] J. Pan, D. Pack, A. Kosaka, and A. Kak, "Fuzzy-nav: A vision-based robot navigation architecture using fuzzy inference for uncertainty-reasoning," in *Procs. of the World Congress on Neural Networks*, 1995, pp. 602–607.

[19] D. Kim and R. Nevatia, "Symbolic navigation with a generic map," *Autonomous Robots*, vol. 6, no. 1, pp. 69–88, 1999.

[20] J. Borenstein, Y. Koren, *et al.*, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE transactions on robotics and automation*, vol. 7, no. 3, pp. 278–288, 1991.

[21] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Transactions on systems, Man, and Cybernetics*, vol. 19, no. 5, pp. 1179–1187, 1989.

[22] G. N. DeSouza and A. C. Kak, "Vision for mobile robot navigation: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 2, pp. 237–267, 2002.

[23] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov, "Learning to explore using active neural slam," in *International Conference on Learning Representations (ICLR)*, 2020.

[24] D. S. Chaplot, R. Salakhutdinov, A. Gupta, and S. Gupta, "Neural topological slam for visual navigation," in *CVPR*, 2020.

[25] K. Fang, F.-F. Li, S. Savarese, and A. Toshev, "Scene memory transformer for embodied agents in long time horizon tasks," in *CVPR 2019*, 2019.

[26] S. Shalev-Shwartz and A. Shashua, "On the sample complexity of end-to-end training vs. semantic abstraction training," *arXiv preprint arXiv:1604.06915*, 2016.

[27] G. Dissanayake, S. Huang, Z. Wang, and R. Ranasinghe, "A review of recent developments in simultaneous localization and mapping," in *2011 6th International Conference on Industrial and Information Systems*. IEEE, 2011, pp. 477–482.

[28] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Y. Zeevi, "The farthest point strategy for progressive image sampling," *IEEE Transactions on Image Processing*, vol. 6, no. 9, pp. 1305–1315, 1997.

[29] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

[30] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 661–670.

[31] Z. Wei, "New methods for solving the minimum weighted latency problem," 2018.

[32] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3d: Learning from rgb-d data in indoor environments," *International Conference on 3D Vision (3DV)*, 2017.

[33] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, D. Gordon, Y. Zhu, A. K. Gupta, and A. Farhadi, "Ai2-thor: An interactive 3d environment for visual ai," *ArXiv*, vol. abs/1712.05474, 2017.

[34] F. Xia, A. R. Zamir, Z.-Y. He, A. Sax, J. Malik, and S. Savarese, "Gibson env: real-world perception for embodied agents," in *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE, 2018.

[35] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra, "Habitat: A Platform for Embodied AI Research," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[36] E. Hazan, A. Agarwal, and S. Kale, "Logarithmic regret algorithms for online convex optimization," *Mach. Learn.*, vol. 69, no. 2-3, pp. 169–192, 2007.

[37] A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, and M. Sudan, "The minimum latency problem," in *STOC*, 1994, pp. 163–171.

[38] F. Afrati, S. Cosmadakis, C. Papadimitriou, G. Papageorgiou, and N. Papakostantinou, "The complexity of the traveling repairman problem," *Theoretical Informatics and Applications*, vol. 20, no. 1, pp. 79–86, 1986.

[39] A. Garcia, P. Jodra, and J. Tejel, "A note on the traveling repairman problem," *Networks*, vol. 40, no. 1, pp. 27–31, 2002.

[40] E. Minieka, "The delivery man problem on a tree network," *Ann. Oper. Res.*, vol. 18, pp. 261–266, 1989.

[41] E. Koutsoupias, C. Papadimitriou, and M. Yannakakis, "Searching a fixed graph," in *Proc. 23nd Colloquium on Automata, Languages and Programming*, 1996, pp. 280–289.

[42] B. Wu, "Polynomial time algorithms for some minimum latency problems," *Inf. Process. Lett.*, vol. 75, no. 5, pp. 225–229, 2000.

[43] R. Sitters, "The minimum latency problem is np-hard for weighted trees," in *Proc. 9th International IPCO Conference*, 2002, pp. 230–239.

[44] K. Chaudhuri, B. Godfrey, S. Rao, and K. Talwar, "Paths, trees, and minimum latency tours," in *Proc. 44th Symposium on Foundations of Computer Science (FOCS 2003)*, 2003, pp. 36–45.

[45] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, *et al.*, "On evaluation of embodied navigation agents," *arXiv preprint arXiv:1807.06757*, 2018.

[46] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice - A survey," *Autom.*, vol. 25, no. 3, pp. 335–348, 1989.

[47] Y. Wang and S. P. Boyd, "Fast model predictive control using online optimization," *IEEE Trans. Control. Syst. Technol.*, vol. 18, no. 2, pp. 267–278, 2010.

[48] S. Maniatopoulos, D. Panagou, and K. J. Kyriakopoulos, "Model predictive control for the navigation of a nonholonomic vehicle with field-of-view constraints," in *American Control Conference, ACC 2013, Washington, DC, USA, June 17-19, 2013*. IEEE, 2013, pp. 3967–3972.

[49] T. Fork, H. E. Tseng, and F. Borrelli, "Models and predictive control for nonplanar vehicle navigation," in *24th IEEE International Intelligent Transportation Systems Conference, ITSC 2021, Indianapolis, IN, USA, September 19-22, 2021*. IEEE, 2021, pp. 749–754.

[50] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[51] D. Zhou, L. Li, and Q. Gu, "Neural contextual bandits with ucb-based exploration," in *International Conference on Machine Learning*. PMLR, 2020, pp. 11 492–11 502.

[52] L. Perron and V. Furnon, "Or-tools," Google. [Online]. Available: https://developers.google.com/optimization/

[53] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3d: Learning from rgb-d data in indoor environments," *arXiv preprint arXiv:1709.06158*, 2017.

[54] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2019.

[55] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer science & business media, 2013.

[56] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[57] D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. Karro, and D. Sculley, "Google vizier: A service for black-box optimization," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 1487–1495.

[58] D. Batra, A. Gokaslan, A. Kembhavi, O. Maksymets, R. Mottaghi, M. Savva, A. Toshev, and E. Wijmans, "Objectnav revisited: On evaluation of embodied agents navigating to objects," *CoRR*, vol. abs/2006.13171, 2020. [Online]. Available: https://arxiv.org/abs/2006.13171

[59] X. Gu, T.-Y. Lin, W. Kuo, and Y. Cui, "Open-vocabulary object detection via vision and language knowledge distillation," *arXiv preprint arXiv:2104.13921*, 2021.

[60] J. K. Lenstra and A. R. Kan, "Some simple applications of the travelling salesman problem," *Journal of the Operational Research Society*, vol. 26, no. 4, pp. 717–733, 1975.

[61] A. Santara, G. Aggarwal, S. Li, and C. Gentile, "Learning to plan variable length sequences of actions with a cascading bandit click model of user feedback," in *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, vol. 151, 2022, pp. 767–797.

[62] C. Gentile and F. Orabona, "On multilabel classification and ranking with partial feedback," in *Advances in Neural Information Processing Systems*, vol. 25. Curran Associates, Inc., 2012, pp. 1151–1159.

[63] N. Cesa-Bianchi and C. Conconi, A. Gentile, "A second-order perceptron algorithm," *SIAM J.Comput.*, vol. 34(3), pp. 640–668, 2005.

[64] Y. Abbasi-Yadkori, D. Pál, and C. Szepesvári, "Improved algorithms for linear stochastic bandits," in *NIPS*, 2011.