Online State Exploration: Competitive Worst Case and Learning-Augmented Algorithms

Sungjin Im[1(✉)], Benjamin Moseley[2(✉)], Chenyang Xu[3(✉)], and Ruilong Zhang[4(✉)]

[1] Electrical Engineering and Computer Science, University of California at Merced, Merced, CA, USA
sim3@ucmerced.edu

[2] Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA, USA
moseleyb@andrew.cmu.edu

[3] Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, China
cyxu@sei.ecnu.edu.cn

[4] Department of Computer Science and Engineering, University at Buffalo, Buffalo, NY, USA
ruilongz@buffalo.edu

Abstract. This paper introduces the online state exploration problem. In the problem, there is a hidden d-dimensional target state. We are given a distance function between different states in the space and a penalty function depending on the current state for each incorrect guess. The goal is to move to a vector that dominates the target state starting from the origin in the d-dimensional space while minimizing the total distance and penalty cost. This problem generalizes several natural online discrete optimization problems such as multi-dimensional knapsack cover, cow path, online bidding, and online search. For online state exploration, the paper gives results in the worst-case competitive analysis model and in the online algorithms augmented with the prediction model. The results extend and generalize many known results in the online setting.

Keywords: Online Search · Online Algorithms · Competitive Ratio · Learning-augmented Algorithms · Worst-case Analysis

1 Introduction

A recent trend in algorithmic design under uncertainty is making use of machine learning to augment online algorithms [22]. In this emerging setting, we are given some predictions of the future. These predictions are learned from historical data, and thus, their actual accuracy is unknown. The goal is to develop

All authors (ordered alphabetically) have equal contributions.

D. Koutra et al. (Eds.): ECML PKDD 2023, LNAI 14172, pp. 333–348, 2023.
https://doi.org/10.1007/978-3-031-43421-1_20

an algorithm with these predictions so that the algorithm can outperform traditional algorithms (without predictions) if the predictions are accurate, while still retaining a theoretical worst-case guarantee even when the predictions are arbitrarily wrong. The performance measure of an algorithm is *competitive analysis*. Take a minimization problem as an example. An online algorithm is said to be c-competitive or have a competitive ratio c if the algorithm's objective value is at most a factor c larger than the optimal objective value *on any instance*. We also follow the standard terminology stated in [23]: an algorithm with predictions is said to be α-*consistent* and β-*robust* if its competitive ratio is β with any predictions and it improves to α with perfect predictions.

Many classical online problems have been considered in this novel model. For instance, see the work by [15,20,24] on caching; [5,12] on the classic secretary problem; [14,18] on scheduling; [3,23] on ski rental; and [8] on set cover. Among them, the recent two works [2,4] inspire our paper. In [2], the authors introduced the online search framework, and provided a 4-competitive algorithm and a learning-augmented algorithm that is $(1+\epsilon)$-consistent and $5(1+1/\epsilon)$-robust for any $\epsilon > 0$. [4] considered the cow path problem with predictions and gave an algorithm with $(1+\epsilon)$-consistency and $(\epsilon(1+2/\epsilon)^2+1)$-robustness. For both the two problems, we find that whether in their pure online algorithms or learning-augmented algorithms, the well-known *guess-and-double* technique is applied. This key observation motivates our paper.

Guess-and-Double. This technique is one of the most widely used techniques in the field of machine learning and algorithmic design under uncertainty. Many problems build on this technique to design algorithms, such as incremental clustering [9], online k-center [9] and online load balancing [6]. The basic algorithmic idea is first developing competitive algorithms parameterized by the optimal value (OPT) under the assumption that OPT is known, and then leveraging guess-and-double to remove the assumption. More specifically, we keep running the parameterized algorithm with a guessed value OPT, and once the guessed value is found to be wrong, we geometrically increase the guess. The analysis can guarantee that such an operation only loses a constant factor on the competitive ratio.

Due to the rich applications of the guess-and-double technique, a natural question then arises in the context of learning-augmented algorithms:

> *Is there any unified framework to integrate machine-learned predictions with the guess-and-double technique?*

The main contribution of this paper is proposing such a general framework. We first introduce the online state exploration problem, which unifies applications where guess-and-double is used, and then design learning-augmented algorithms for the problem.

1.1 The Online State Exploration Problem

The online state exploration problem (OSEP) is a generalization of many online discrete optimization problems. This problem is defined in d-dimensional (*state*) space $\mathbb{R}^d_{\geq 0}$, where each point $\mathbf{v} \in \mathbb{R}^d_{\geq 0}$ is referred to as a *state*. For any two states

$\mathbf{u} = (u_1, \ldots, u_d)$ and $\mathbf{v} = (v_1, \ldots, v_d)$, we define that $\mathbf{u}$ *dominates* $\mathbf{v}$ ($\mathbf{u} \succeq \mathbf{v}$) if $u_i \geq v_i \ \forall i \in [d]$. There is a hidden *target* state $\mathbf{t}$ known only to an oracle, and the algorithm is required to move from $\mathbf{0}$ to a state that dominates $\mathbf{t}$. When the algorithm moves from $\mathbf{u}$ to $\mathbf{v}$, it needs to pay the moving cost $c(\mathbf{u}, \mathbf{v})$ and then is told whether $\mathbf{v}$ dominates $\mathbf{t}$. The goal is to minimize the total moving cost.

To see how this captures the guess-and-double framework, intuitively think of $\mathbf{t}$ as the optimal value in the guess-and-double framework, which is generalized to multiple dimensions. Then, the moving cost captures how much an algorithm pays when using a particular guess of optimal.

Define that the moving cost $c(\mathbf{u}, \mathbf{v}) = \mathsf{D}(\mathbf{u}, \mathbf{v}) + \mathsf{P}(\mathbf{u})$ consists of two parts: the distance cost $\mathsf{D}(\mathbf{u}, \mathbf{v})$ and the penalty cost $\mathsf{P}(\mathbf{u})$. The distance function is assumed to satisfy the following three properties.

- (Identity) $\mathsf{D}(\mathbf{u}, \mathbf{u}) = 0$ for all $\mathbf{u}$. The distance of a point to itself is 0.
- (Domination Monotonicity) For any two states $\mathbf{u} \succeq \mathbf{v}$, $\mathsf{D}(\mathbf{0}, \mathbf{u}) \geq \mathsf{D}(\mathbf{0}, \mathbf{v})$. If $\mathbf{u}$ dominates $\mathbf{v}$ then $\mathbf{u}$ has no smaller distance from the origin than $\mathbf{v}$.
- (Domination Submodularity) For any two states $\mathbf{u} \succeq \mathbf{v}$ and any $\mathbf{w} \in \mathbb{R}^d_{\geq 0}$ with at most one non-zero entry, we have $\mathsf{D}(\mathbf{u}, \mathbf{u}+\mathbf{w}) \leq \mathsf{D}(\mathbf{v}, \mathbf{v}+\mathbf{w})$. In other words, making an affine move by $\mathbf{w}$ from $\mathbf{u}$ is no costlier than making the same move from $\mathbf{v}$. That is, distances are submodular.

For convenience, we call the last two assumptions *monotonicity* and *submodularity*, respectively. For a state $\mathbf{u}$ in the space, define its *distance vector* $\tilde{\mathbf{u}} = (\tilde{u}_1, \ldots, \tilde{u}_d)$, where $\tilde{u}_i = \mathsf{D}(\mathbf{0}, u_i \mathbf{e}_i)$ and $\mathbf{e}_i$ is the standard basis vectors where the 1 appears in the i-th position. We define penalty $\mathsf{P}(\mathbf{u}) \leq \gamma \cdot ||\tilde{\mathbf{u}}||_\infty$ where $\gamma \geq 0$ is an input *penalty factor*. Intuitively, the penalty is incurred when visiting an incorrect state $\mathbf{u}$. The penalty generalizes the cost function and is useful for capturing some applications that a submodular distance function is not general enough to capture. Given a target $\mathbf{t}$, we have a standard assumption that the optimal solution is moving from $\mathbf{0}$ to $\mathbf{t}$ directly, i.e., $\mathrm{OPT}(\mathbf{t}) = c(\mathbf{0}, \mathbf{t})$, and $\mathrm{OPT}(\mathbf{t})$ is scaled to always be at least 1.

In the learning-augmented algorithms model, the algorithm is given access to a prediction. We consider predicting the target state $\mathbf{t}'$. This prediction is constructed from data corresponding to prior instances of the problem considered. The prediction helps the algorithm cope with uncertainty in the online setting. As machine learning is often imperfect, the algorithm must cope with having an erroneous prediction.

1.2 Applications of Online State Exploration

Online state exploration captures many natural and known problems. We introduce three examples here. The detailed discussions of the reductions from these problems to online state exploration are omitted in this version.

Online Bidding. In this problem, an algorithm can make a bid u for cost u. The algorithm must make bids until the algorithm bids larger than some unknown

Table 1. We only show the results of deterministic algorithms in the table. In the table, d is the number of dimensions, $\gamma \geq 0$ is the penalty factor, and e is the base of the natural logarithmic. The pair (α, β) represents an α-consistent and β-robust algorithm. For the full spectrum of (α, β) pairs that can be obtained, see the theorems and corollaries below.

Problems	Worst Case Algorithms	Learning-Augmented Algorithms
Online State Exploration	$(\gamma+1)ed+e$ (Theorem 1)	$(2(1+\gamma)d-\gamma, 2(1+\gamma)e^2d-\gamma e^2)$ (Theorem 3)
Online Bidding	4 [10]	$(2, 4)$ (Theorem 5)
MD Cow Path	$2ed+1$ [7]	$(2d+1, 2ed+1)$ (Corollary 3)
MD Knapsack Cover	$ed+e$ (Corollary 1)	$(2d, 2e^2d)$ (Corollary 2)

target T. In the online setting, the target is revealed only when the algorithm bids an amount larger than T. The goal is to minimize the summation of all bids. See [10,13]. The problem admits a 4-competitive deterministic algorithm and an e-competitive randomized algorithm, which are shown to be optimal.

Multi-Directional Cow Path. A cow is located at the meeting point of several rays. There is only one gate that is located on some ray. The cow must locate the gate by moving along the rays where each unit of distance traveled costs one unit. The gate is only discovered if the cow reaches that location. The goal is to discover the gate at a minimum cost. See [7,11,16,17]. The problem admits a deterministic algorithm that can achieve $2ed+1$ and a randomized algorithm that can achieve $3.088d+o(d)$. Both ratios are optimal.

Online Multi-dimensional Knapsack Cover. In this problem, there is a collection of items that can be used to cover d dimensional space. Each item i costs c_i units and covers dimension $j \in [d]$ by a non-negative amount $w_{i,j}$. Each dimension j must be covered to an amount at least $h_j \geq 0$. We refer to vector $\mathbf{h} = (h_1, \ldots, h_d)$ as the *demand vector*. If a set of items S are selected then dimension j is covered by $\sum_{i \in S} w_{i,j}$ and the cost is $\sum_{i \in S} c_i$. The goal is to choose items of minimum cost to cover all dimensions. In the online setting, each dimension i covering the amount h_i is unknown, and it is only known when the algorithm purchases enough items to cover all dimensions. And we assume that every item can be used an infinite number of times, and the items cannot be revoked once they are included in the solution. This problem is a natural generalization of the Multi-Optional Ski-Rental problem. See [1,19,21,25] for the multi-optional ski-rental problem. To our best knowledge, we introduce this problem for the first time.

1.3 Our Contributions

Our main contributions are in the following two senses (Table 1):

- We introduce the online state exploration problem, which unifies many classical applications where guess-and-double is used, such as online bidding, multidimensional cow path, and multidimensional knapsack cover;
- We design learning-augmented algorithms for the problem which have good performance on each problem as long as the specific properties of each problem are utilized during the analysis.

We also consider the problem without a prediction in the traditional worst-case competitive analysis model. This paper develops a deterministic algorithm that has a competitive ratio linear in d when the penalty factor γ is given (Theorem 1). We show that this is essentially tight deterministically (Theorem 2). Then we show competitive ratio can be slightly improved via randomization.

We then consider the learning-augmented algorithm model. In this case, the algorithm is given a prediction $\mathbf{t}'$ of the target state that may be erroneous. We give a deterministic algorithm that has a trade-off on consistency and robustness with respect to the prediction error $||\tilde{\mathbf{t}} - \tilde{\mathbf{t}}'||_\infty$ (Theorem 3). We remark that the trade-off can also be improved by the randomized algorithm, which is omitted in this version. Finally, we show that any algorithm with a bounded robustness ratio has no smaller consistent ratio than ours (Theorem 2).

We show the consistency and robustness trade-offs can be further improved for some special cases by using the specific properties of specific problems.

For online bidding, we show that there is a deterministic algorithm that achieves the consistent ratio $(1+\epsilon)$ and the robust ratio $\frac{2(1+\epsilon)^2}{\epsilon(1+\epsilon/2)}$ when $\epsilon < 1$ (Theorem 5), which slightly improves the previous result $(1+\epsilon)$-consistency and $5(1+\frac{1}{\epsilon})$-robustness [2]. Moreover, if we do not pursue a consistency ratio smaller than 2, the robustness ratio can always be guaranteed no worse than the worst-case bound 4 [10]. We also remark that the trade-offs can be further improved if allowing for randomization.

For multi-directional cow path (MD cow path for short), we show that our algorithm achieves the consistency ratio $(1+\epsilon)$ and the robustness ratio $(\epsilon(1+\frac{2}{\epsilon})^d + 1)$ for any $\epsilon > 0$. Notice that if setting $\epsilon = 2d$, the robustness ratio is linearly dependent on d. When $d = 2$, our algorithm is $(1+\epsilon)$-consistent and $(\epsilon(1+\frac{2}{\epsilon})^2 + 1)$-robust, this matches the previous work [4] for 2-directional cow path which is shown to be Pareto optimal.

For online multi-dimensional knapsack cover (MD knapsack cover for short), we show that the problem is a special case of OSEP. Thus, we can directly get a worst-case algorithm and learning-augmented algorithm by setting the penalty factor $\gamma = 0$ (Corollary 1 and Corollary 2).

In Sect. 4, we verify the theory empirically. The experiments show that our algorithms can achieve desirable empirical trade-offs between consistency and robustness. We also discuss the learnability of our prediction. Say that $\mathbf{t}$ is drawn from an unknown distribution $\mathcal{D}$. Then we prove that only a small (polynomial in d) number of samples need to be drawn from $\mathcal{D}$ to efficiently learn $\mathbf{t}'$ that minimizes the expected prediction error. Due to space, this part and some proofs are omitted and can be found in the full version.

2 Algorithms and Analysis

In this section, we start by introducing an algorithmic framework that is used in our learning-augmented algorithms (Sect. 2.1). Then in Sect. 2.2, we discuss worst-case algorithms for OSEP as a warm-up. Finally, we show the main algorithmic results—the learning-augmented algorithms in Sect. 2.3.

Algorithm 1. Algorithmic Framework for Online State Exploration (Budget Solver)

Input: A target point $\mathbf{t}$; a budget sequence $\mathcal{B} = (B_1, B_2, \ldots)$; the cost function $c(\cdot, \cdot)$.
Output: A feasible solution $\mathcal{S}$.

```
S ← {0}; k ← 0; u ← 0.
while the target point t has not been dominated do
    φ(k) ← 1 + k (mod d). // the current dimension
    Find the maximum x such that c(0, x e_{φ(k)}) ≤ B_k.
    Move to v, where v_{φ(k)} = x and v_i = u_i for all i ≠ φ(k).
    S ← S ∪ {v}; u ← v; k ← k + 1.
end while
return Solution S.
```

2.1 Algorithmic Intuition and Framework

The main difficulty of online state exploration is that the algorithm has very little information available. Even if the algorithm arrives at a state with only one dimension's entry smaller than the target, the algorithm only knows that the current state is yet to dominate the target.

Thus, there is a trade-off between distance and penalty costs. If the algorithm increases all entries equally when deciding the next state, the total distance cost may be far from optimal because perhaps only one entry needs to increase. We call such a move a "big move" because all entries increase. Contrarily, if the algorithm explores one dimension at a time, this will control the distance cost, but the penalty is large. We refer to this strategy as a "small move" as only one entry increases.

Algorithmic Intuition. We will say that two states are *adjacent* if they differ in only one coordinate. Formally, $\mathbf{u}$ and $\mathbf{v}$ are adjacent if there exists a unique $r \in [d]$ such that $u_r \neq v_r$ and $u_i = v_i$ for all $i \neq r$. We first observe the following property for two adjacent states.

Proposition 1. *For any two adjacent states* $\mathbf{u}$ *and* $\mathbf{v}$, $c(\mathbf{u}, \mathbf{v}) \leq \gamma \cdot ||\tilde{\mathbf{u}}||_\infty + ||\tilde{\mathbf{v}}||_\infty$.

We can prove Proposition 1 easily by the monotonicity and submodularity. The proof is omitted in this version. We will use this property to carefully choose the next state each time so that the cost can be bounded. To see some intuition, say $||\tilde{\mathbf{u}}||_\infty$ doubles each time the algorithm moves. Then it is the case that the algorithm's total moving cost is bounded by $(2+2\gamma)$ times $||\tilde{\mathbf{u}}||_\infty$ where $\tilde{\mathbf{u}}$ is the final state the algorithm visits. If this cost is bounded by optimal, then we can bound the overall competitive ratio. Thus, we have the following framework.

Algorithmic Framework. Now, we present an algorithmic framework (Algorithm 1) that will be of use in several algorithms developed in this paper. The algorithm runs in rounds. Inspired by the intuition above, our algorithm will make a small move in each round. Based on the state selected by the previous round, the

algorithm will increase the coordinate value of this state in a particular dimension. The key to the algorithm is selecting an upper bound of the moving cost (budget) each time the algorithm moves states. Once the budget is determined, Algorithm 1 will increase the coordinate value of the chosen dimension as much as possible under the budget constraint.

The framework specifies a family of algorithms depending on different budget sequences. All our algorithms will employ this algorithmic framework to obtain a feasible solution, and as such, the budget sequence is the main remaining technical challenge.

Our algorithms compute the budget sequence online, and together with Algorithm 1 this yields a fully online algorithm. We use BUDSOL($\mathcal{B}$) to denote the solution returned by Algorithm 1 when the budget sequence is $\mathcal{B}$.

2.2 Warm-Up: Worst Case Algorithms

For the completeness of the story, before stating the learning-augmented algorithm, we give a quick discussion of worst-case algorithms for the new defined problem. The deterministic worst-case algorithm is technically simple: we set the budget sequence $\mathcal{B}$ to be an infinite geometric progression, i.e., $\mathcal{B} = \{1, a, a^2, \ldots\}$, where a is a parameter we can choose. The pseudo-code is omitted in this version. Use $\mathcal{A}$ to denote this algorithm.

Theorem 1. *Given an arbitrary target point* $\mathbf{t}$*, use* $\mathrm{ALG}(\mathbf{t})$ *to denote the objective value of* $\mathcal{A}$*. We have* $\mathrm{ALG}(\mathbf{t}) \leq ((\gamma+1)ed + e) \cdot \mathrm{OPT}(\mathbf{t})$*, where* d *is the number of dimensions,* $\gamma \geq 0$ *is the penalty factor and* e *is the base of natural logarithm.*

We only present the statements of two key lemmas in this version.

Lemma 1. *Given an arbitrary target point* $\mathbf{t}$*, algorithm* $\mathcal{A}$ *terminates in at most* $d + \lceil \log_a(\mathrm{OPT}(\mathbf{t})) \rceil$ *iterations, where* $\mathrm{OPT}(\mathbf{t})$ *is the optimal cost.*

Lemma 2. *The competitive ratio of algorithm* $\mathcal{A}$ *is at most* $\left(\frac{a+\gamma}{a-1} \cdot a^d\right)$.

The first lemma shows that $\mathcal{A}$ terminates in a polynomial number of iterations. Then, by setting $a = 1 + 1/d$ in the second lemma, Theorem 1 can be proved. Notice that $1 + 1/d$ is not the best choice of a. By taking the derivation, one can find a better multiplier a.

2.3 Algorithms Leveraging Predictions

This section gives the learning-augmented algorithm where the algorithm is given an erroneous prediction of the target state. Given an arbitrary target point $\mathbf{t}$, let $\mathbf{t}' = (t'_1, \ldots, t'_d)$ be its predicted target point. We use a natural error measurement as the prediction error, namely the infinity norm. The formal definition can be found in Definition 1.

Definition 1. *(Prediction Error) Given a target point* $\mathbf{t}$ *and its prediction* $\mathbf{t}'$*, define the prediction error* $\eta(\mathbf{t}, \mathbf{t}') := ||\tilde{\mathbf{t}} - \tilde{\mathbf{t}}'||_\infty$.

When the parameter is clear in the context, write $\eta(\mathbf{t}, \mathbf{t}')$ as η for short. We first prove a lower bound of the consistency ratio and then show that our algorithms can approach the bound arbitrarily close.

Lower Bound of Consistency

We give an $\Omega(\gamma d)$ lower bound for (randomized) learning-augmented algorithms.

Theorem 2. *In online state exploration, given the predicted target, for any algorithm with a bounded robustness ratio, the consistency ratio is at least* $(1+\gamma)d-\gamma$.

The main idea of this proof is to construct a specific instance such that the moving cost between two non-adjacent states is infinitely large. Such an instance will force any algorithm to move to the adjacent state in each step. Otherwise, it cannot obtain a bounded robustness ratio. Thus, any algorithm has to go through the dimensions one by one and visit many incorrect states, which will incur an $\Omega(\gamma d)$ penalty cost.

Remark. Note that the lower bound $\Omega(\gamma d)$ also applies to traditional worst-case algorithms since a worst-case algorithm is essentially a special learning-augmented algorithm whose consistency and robustness are the same. From the proof sketch of Theorem 2, we see that a big move may make the competitive ratio unbounded. Thus, in our algorithms, only small moves are considered.

Deterministic Learning-Augmented Algorithm

In this section, we give a deterministic learning-augmented algorithm for OSEP. For notational convenience, use $\psi(\gamma, d) := (1+\gamma)d-\gamma$ to denote the lower bound stated in Theorem 2.

Theorem 3. *Given any* $\epsilon > 0$*, there is a deterministic algorithm with a consistency ratio of* $\psi(\gamma, d) \cdot (1 + \epsilon)$ *and a robustness ratio of* $\psi(\gamma, d) \cdot (1 + \epsilon) \cdot \left(1 + \frac{1+2/\epsilon}{d-\gamma/(\gamma+1)}\right)^{2d}$*. The ratio degrades at a rate of* $O(\gamma\epsilon^{-(2d+1)}d^{-(2d-1)})$ *as the error* η *increases. Moreover, by setting appropriate parameters, the algorithm can be* $(2(1+\gamma)d-\gamma)$*-consistent and* $(2(1+\gamma)e^2d-\gamma e^2)$*-robust.*

Notice that for the online bidding problem, we have $\gamma = 1$ and $d = 1$. In this case, the algorithm is $(1 + \epsilon)$-consistent and $O(1/\epsilon^2)$-robust for any $\epsilon > 0$. Later in Sect. 3.1, we will show that by a more careful analysis specific to online bidding (online search), the algorithm obtains a robustness ratio of $O(1/\epsilon)$ when it is $(1+\epsilon)$-consistent, which has been proved to be the best possible trade-off [2].

To prove Theorem 3, we first state a parameterized algorithm, and then show that choosing appropriate parameters gives the claimed ratio. The algorithm is described in Algorithm 2.

The algorithm is parameterized by a and θ, where a is the common ratio of the geometric progression. The parameter θ can be viewed as the degree of trusting the prediction with smaller values reflecting high confidence in the prediction. Define $\mathcal{B}^1 := \bigcup_{i\in[d-1]} \mathcal{B}_{(i)}$ and $\mathcal{B}^2 := \mathcal{B}_{(d)}$. The budgets in $\mathcal{B}^1$ are computed based on the prediction, while the budgets in $\mathcal{B}^2$ are computed following the manner of the deterministic worst-case algorithm.

Intuitively, $\mathcal{B}^1$ and $\mathcal{B}^2$ are two different types of budget sequences, and thus, they split Algorithm 2 into two different phases. In the first phase, Algorithm 2 is indicated by the predicted target state. Informally, the algorithm will utilize each coordinate of the predicted state to compute a initial budget (line 1 of Algorithm 2). Thus, there are d initial budgets, one for each dimension. Starting from the smallest initial budget, Algorithm 2 grows the budget in a geometric manner until Algorithm 2 reaches a state that dominates the current coordinate of the predicted target point. At the end of each round, Algorithm 2 carefully choose the next initial budget. If the actual target state is still not dominated by Algorithm 2 at the end of $\mathcal{B}_{(d-1)}$, the algorithm will be switched to the traditional online deterministic algorithm and grow the budget in a pure geometric manner.

Algorithm 2. Deterministic Learning-Augmented Algorithm for Online State Exploration

Input: Parameter $a > 1$ and $\theta \in (0, 1]$; the predicted target point $\mathbf{t}'$; the cost function $c(\cdot, \cdot)$.
Output: A feasible solution $\mathcal{S}$.

1: $\forall j \in [d]$, compute the unique $r_j \in [-1, 0)$ such that $(r_j + z_j)d = T'_j + \theta d$ for some integer z_j, where $T'_j = \log_a(\tilde{t}'_j)$.
2: Reindex dimensions in the non-decreasing order of r_j.
3: $k \leftarrow 0$; $p \leftarrow 1$; $q \leftarrow 1$;
4: $B_0 \leftarrow a^{r_1 d}$; $\mathcal{B}_{(1)}, \ldots, \mathcal{B}_{(d)} \leftarrow \emptyset$.
5: **while** $q < d$ **do**
6: $\quad \phi(k) \leftarrow 1 + k \pmod d$. // the current dimension
7: $\quad$ **if** $B_k \le a^{T'_q + \theta d}$ **then**
8: $\quad\quad \mathcal{B}_{(q)} \leftarrow \mathcal{B}_{(q)} \cup \{B_k\}$; $B_{k+1} \leftarrow B_k \cdot a$.
9: $\quad$ **else**
10: $\quad\quad q \leftarrow q + 1$.
11: $\quad\quad$ **if** $B_k/a < a^{T'_q + \theta d}$ **then**
12: $\quad\quad\quad B_k \leftarrow B_k \cdot a^{(r_q - r_p)d}$; $p \leftarrow q$.
13: $\quad\quad$ **end if**
14: $\quad\quad \mathcal{B}_{(q)} \leftarrow \mathcal{B}_{(q)} \cup \{B_k/a\}$; $B_{k+1} \leftarrow B_k$.
15: $\quad$ **end if**
16: $\quad k \leftarrow k + 1$.
17: **end while**
18: Let $\mathcal{B}_{(d)}$ be the infinite geometric progression $\{B_k, B_k \cdot a, B_k \cdot a^2, \ldots\}$.
19: $\mathcal{B} \leftarrow \mathcal{B}_{(1)} \cup \ldots \cup \mathcal{B}_{(d)}$.
20: $\mathcal{S} \leftarrow \text{BUDSOL}(\mathcal{B})$.
21: **return** Solution $\mathcal{S}$.

The algorithm is a bit subtle. So we first give two observations to help to understand why we design $\mathcal{B}$ in this way.

Observation 1. *For any $q \in [d]$, $\mathcal{B}_{(q)}$ contains at least one value and is a geometric sequence with a common ratio a.*

Observation 2. *For any $q \in [d-1]$, let B_k and B_{k+1} be the last term in $\mathcal{B}_{(q)}$ and the first term in $\mathcal{B}_{(q+1)}$ respectively. We have $B_k \le B_{k+1}$.*

The first observation is because if q increases by 1 in an iteration, we always add a budget into $\mathcal{B}_{(q)}$, while if q does not increase, the budget added in that iteration must be a times the budget added in the previous iteration. The second observation is due to $r_q \le r_{q+1}$ for any $q \in [d-1]$. When q increases, the added budget either remains the same or increases by a factor $a^{(r_q - r_p)d} \ge 1$. According to the two observations, if we remove the last term of each $\mathcal{B}_{(q)}$ ($q \in [d-1]$) from $\mathcal{B}$, we can round up each remaining term in $\{\mathcal{B}_{(q)}\}_{q\in[d-1]}$ such that the sequence becomes a geometric progression with common ratio a. Denote such a sequence by $\mathcal{B}'$. Note that the increase rate of these rounded terms is at most a^d because $|r_j - r_i| \le 1$ for any i, j.

Now we are ready to analyze the algorithm. We first prove that the algorithm always terminates in a polynomial number of iterations.

Lemma 3. *Algorithm 2 terminates in at most $3d - 1 + \lceil \log_a(\mathrm{OPT}) \rceil$ iterations.*

Proof. Since the last term $\mathcal{B}_{(d)}$ is an infinite geometric progression, we can always find the term $B_h \in \mathcal{B}$ which is the first $B_h \ge \mathrm{OPT}$. Following the proof of Lemma 1, the algorithm terminates within $h + d$ iterations. Now we show that $h \le 2d - 1 + \lceil \log_a(\mathrm{OPT}) \rceil$.

Due to Observation 1 and Observation 2, $\mathcal{B}$ is a non-decreasing sequence. If removing the last term of $\mathcal{B}_{(q)}$ for each $q \in [d-1]$, the ratio between any two neighboring budgets in the new sequence is at least a. Observing that the initial budget is at least a^{-d}, the index of B_h in the new sequence is at most $d + \lceil \log_a(\mathrm{OPT}) \rceil$. Since the number of removed budgets is at most $d-1$, we have $h \le 2d - 1 + \lceil \log_a(\mathrm{OPT}) \rceil$, completing the proof.

We bound the consistency and the robustness of the algorithm by the following two technical lemmas respectively.

Lemma 4. *Given any target point $\mathbf{t}$ and its predicted point $\mathbf{t}'$, let $\mathrm{ALG}(\mathbf{t})$ be the solution returned by Algorithm 2. Then, we have*

$$\mathrm{ALG}(\mathbf{t}) \le f(a) \cdot a^{\theta d} \cdot \left(\mathrm{OPT}(\mathbf{t}) + \frac{a^{2d} - a^{\theta d}}{a^{\theta d} - 1} \cdot \eta \right)$$

where $a > 1, \theta \in (0, 1]$ and $f(a) = \left((1+\gamma)(\frac{1}{a-1} + d) - \gamma \right)$.

Lemma 5. *Given an arbitrary target point $\mathbf{t}$ and its prediction $\mathbf{t}'$, let $\mathrm{ALG}(\mathbf{t})$ be the solution returned by Algorithm 2. Then, we have: $\mathrm{ALG}(\mathbf{t}) \le f(a) \cdot a^{2d} \cdot \mathrm{OPT}(\mathbf{t})$. where $a > 1$ is the parameter and $f(a) = \left((1+\gamma)(\frac{1}{a-1} + d) - \gamma \right)$.*

Lemma 4 and Lemma 5 are sufficient to prove Theorem 3. By setting $a = 1 + \frac{2/\epsilon+1}{d-\gamma/(\gamma+1)}$ and $\theta = \frac{1}{d}\log_a(1+\epsilon/2)$, the first claimed ratio in Theorem 3 can be proved. The second mentioned ratio is obtained by setting $a = 1 + 1/d$ and $\theta \to 0$.

Remark for The Randomized Learning-Augmented Algorithm. Note that we can also use the randomized technique to improve the competitive ratio in the learning-augmented setting. For Algorithm 2 with any $\epsilon > 0$, our randomized algorithm can improve both of the consistency ratio and robustness ratio by at least a factor of $(1 + 2/\epsilon)\ln(1 + \epsilon/2)$.

3 Applications of OSEP: Problem-Specific Analyses and Better Results

3.1 Online Bidding

We first give a formal reduction from online bidding to online state exploration.

Theorem 4. *A c-competitive algorithm for the OSEP implies a c-competitive ratio for online bidding.*

Proof. Given an arbitrary instance of online bidding, we construct an instance I of OSEP as follows. There is only one dimension in the instance I. In the online bidding problem, every bid u costs u. Thus, for every bid u in the online bidding problem, we create a point $\mathbf{u} = u$. Given two points $\mathbf{u}$ and $\mathbf{v}$, we define the distance cost from the point $\mathbf{u}$ to $\mathbf{v}$ as follows: $\mathsf{D}(\mathbf{u}, \mathbf{v}) := |v - u|$. Note that, for an arbitrary point $\mathbf{u}$, the penalty cost $\mathsf{P}(\mathbf{u}) = u$ since there is only one dimension. Let the penalty factor $\gamma = 1$. Thus, the moving cost $c(\mathbf{u}, \mathbf{v}) = |v - u| + u$. Clearly, the distance function defined above satisfies the identity, monotonicity and submodularity property. Thus, the constructed instance is a special case of the online state exploration problem.

Given an arbitrary feasible solution $\mathcal{S}_b = \{v_1, v_2, \ldots, v_k\}$ of the online bidding problem, we can assume that $v_i \leq v_j$ if $i \leq j$ since any reasonable solution will not include a smaller bid in the next iteration. Now, we construct a solution $\mathcal{S}_e = \{\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \ldots, \mathbf{v}^{(k)}\}$ to the OSEP. Clearly, $\mathcal{S}_e$ is a feasible solution to the OSEP. Let $\mathrm{F}(\mathcal{S}_b)$ be the total cost of the solution $\mathcal{S}_b$. Then, we have $\mathrm{F}(\mathcal{S}_b) = \sum_{i\in[k]} v_i = \mathrm{F}(\mathcal{S}_e)$. Conversely, any reasonable solution $\mathcal{S}_e = \{\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \ldots, \mathbf{v}^{(k)}\}$ to the constructed online state exploration instance also satisfies $\mathbf{v}^{(i)} \preceq \mathbf{v}^{(j)}$ if $i \leq j$. Thus, $\mathcal{S}_e$ can also be easily converted into a feasible solution to the online bidding problem with the same cost.

Moreover, an arbitrary online bidding instance has the same optimal solution as the constructed online state exploration problem. Thus, a c-competitive algorithm for the general online state exploration problem will imply a c-competitive algorithm for the online bidding problem.

Due to the reduction, applying Algorithm 2 to online bidding directly gives a competitive ratio claimed in Theorem 3. We further show that a better analysis can be obtained by taking advantage of specific properties of online bidding.

Theorem 5. *Given an arbitrary instance of online bidding, if we want a near optimal consistency ratio* $(1+\epsilon)$ *for a small* $\epsilon > 0$*, Algorithm 2 with appropriate parameters can obtain*

$$\mathrm{ALG} \leq \min\left\{\frac{2(1+\epsilon)^2}{\epsilon(1+\epsilon/2)}\mathrm{OPT}, (1+\epsilon)\left(\mathrm{OPT} + \left(\frac{4}{\epsilon^2} + \frac{2}{\epsilon} - 1\right)\eta\right)\right\}.$$

On the other hand, if we want an optimal robustness ratio 4[1]*, Algorithm 2 with appropriate parameters can obtain*

$$\mathrm{ALG} \leq \min\left\{4\mathrm{OPT}, 2(1+\epsilon)\mathrm{OPT} + (\frac{1}{\epsilon} - \epsilon)\eta\right\}.$$

Since $\frac{2(1+\epsilon)^2}{\epsilon(1+\epsilon/2)} < 5(1+\frac{1}{\epsilon})$, the trade-off is slightly better than [2]. Moreover, we can also use randomization to further improve the consistent and robust ratio by a factor of at least $(1+\frac{2}{\epsilon})\ln(1+\frac{\epsilon}{2})$. In Sect. 4, we show that both of these two algorithms beat the algorithm in [2] in practice. Note that the consistency and robustness trade-off is not unique.

Extended to Online Search. Our results for the online bidding problem can be directly extended to the online search problem considered by [2]. Online search is a generalization of online bidding, but we can show it is still captured by our framework. The reduction from online search to OSEP is similar to the reduction for online bidding.

3.2 Multi-dimensional Knapsack Cover

Multi-dimensional knapsack cover is a special case of the OSEP when $\gamma = 0$. Intuitively, if the demand vector $\mathbf{h}$ is known in advance, we can solve the multi-dimensional knapsack cover problem by standard dynamic programming (DP). The state in the dynamic programming is defined to be the optimal value of the sub-problem with a demand vector $\mathbf{x} \preceq \mathbf{h}$, which is denoted by $S(\mathbf{x})$. The key idea of the reduction is to map the DP's states to points in the OSEP. For a point $\mathbf{x}$ in online state exploration, we let the distance between $\mathbf{x}$ and the origin $\mathbf{0}$ be $S(\mathbf{x})$. For two different points $\mathbf{x}, \mathbf{y}$, we define their distance as $S(\mathbf{y}-\mathbf{x})$. Then, a c-competitive algorithm for the OSEP can imply a c-competitive algorithm for multi-dimensional knapsack cover.

Corollary 1. *Given an arbitrary instance of multi-dimensional knapsack cover and a target value vector* $\mathbf{t}$*, there exist a deterministic algorithm such that* $\mathrm{ALG}(\mathbf{t}) \leq (ed+e)\cdot\mathrm{OPT}(\mathbf{t})$*, where* d *is the number of dimensions and* e *is the base of natural logarithmic.*

[1] The lower bound 4 of worst case algorithms is the best possible robustness ratio.

Corollary 2. *Given any $\epsilon > 0$, there is a deterministic learning-augmented algorithm for multi-dimensional knapsack cover with a consistency ratio of $d(1+\epsilon)$ and a robustness ratio of $d(1+\epsilon)(1+\frac{1+2/\epsilon}{d})^{2d}$. The ratio degrades at a rate of $O(\epsilon^{-(2d+1)}d^{-(2d-1)})$ as the error η increases. Moreover, by setting appropriate parameters, the algorithm can be $2d$-consistent and $2e^2d$-robust.*

3.3 Multi-directional Cow Path

This section shows that online state exploration captures the multi-directional cow path problem. Although both of them travel in a d-dimensional space, the reduction is not that obvious, because, in MD cow path, the algorithm can only visit a point with at most one non-zero entry while there is no such restriction for online state exploration.

To build the intuition of the reduction, we define states on MD cow path. Consider an algorithm $\mathcal{A}$ for MD cow path and an arbitrary time t during $\mathcal{A}$'s travel. Define the state that $\mathcal{A}$ reaches at time t to be a d-dimensional vector $\mathbf{v}$, where v_i is the furthest point that $\mathcal{A}$ has visited in the i-th ray. Such a state definition implies that $\mathcal{A}$ will terminate if and only if it gets to a state $\mathbf{v}$ that dominates the target point $\mathbf{t}$. We can also obtain a better competitive ratio on MD cow path by more careful analysis.

Corollary 3. *Given an arbitrary instance of the multi-directions cow path problem and its prediction, for any $\epsilon > 0$, there exists a deterministic algorithm with $(1+\epsilon)$-consistent and $(\epsilon(1+\frac{2}{\epsilon})^d + 1)$-robust. When $\epsilon = 2d$, the algorithm is $(2d+1)$-consistent and $(2ed+1)$-robust.*

4 Experiments

This section shows the empirical performance of our algorithms. We investigate our algorithms' trade-offs between consistency and robustness on online bidding and multi-directional cow path. We only present experimental results for online bidding in this version. The experiments[2] are conducted on a machine running Ubuntu 18.04 with an i7-7800X CPU and 48 GB memory.

Setup. We compare the deterministic and randomized algorithm to the algorithm Predict-And-Double (PAD) [2] on the online bidding problem. To show the trade-off between consistency and robustness, we construct a set of (T, T') pairs, where T is the target bidding and T' is the prediction, and test the worst empirical performance of each algorithm when they share the same consistency ratio $1+\epsilon$. In the experiment, we let T, T' be integers in $[1, 1000]$, thus, there are total 1000^2 pairs. We investigate 10 different values of $\epsilon = 0.1, 0.2, \ldots, 1.0$.

[2] The code is available at https://github.com/Chenyang-1995/Online-State-Exploration.

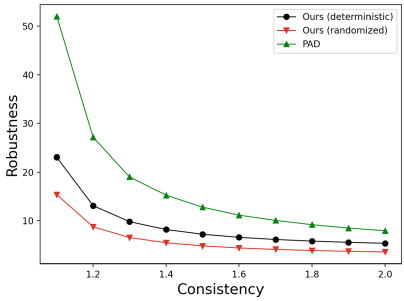

Fig. 1. The consistency and robustness trade-offs on online bidding.

Results. The results are given in Fig. 1. We see that the experiments corroborate the theory. For each learning-augmented algorithm, the curve in the figure matches our theoretical analysis, i.e., $(1+\epsilon)$-consistency and $O(\frac{1}{\epsilon})$-robustness. Both of our algorithms obtain better consistency and robustness trade-offs than the baseline. Moreover, given the same consistency ratio, our randomized algorithm always obtains the best robustness ratio.

5 Conclusion

This paper introduces the online state exploration problem (OSEP), which generalizes many new and hitherto studied problems and gives online algorithms that can benefit from ML predictions. The problem formulation is distinguished from the previous work for its multidimensional aspect and thus can be used to capture a rich body of applications. Further, our results match or improve upon the best-known results for problems like cow path and online bidding. One interesting open problem is whether it is possible to further improve the trade-off between consistency and robustness for OSEP. Another direction would be to study the OSEP with different types of feedback. For example, if we get notified when the current state dominates the target state on each dimension, can we obtain stronger results? Also, it would be interesting to consider different types of predictions.

Acknowledgements. Chenyang Xu was supported in part by Science and Technology Innovation 2030 -"The Next Generation of Artificial Intelligence" Major Project No.2018AAA0100900, and the Dean's Fund of Shanghai Key Laboratory of Trustworthy Computing, East China Normal University. Sungjin Im was supported in part by NSF grants CCF-1844939 and CCF-2121745. Benjamin Moseley was supported in part by a Google Research Award, an Infor Research Award, a Carnegie Bosch Junior Faculty Chair, and NSF grants CCF-2121744 and CCF-1845146. Ruilong Zhang was supported by NSF grant CCF-1844890.

Ethical Issues

The current paper is a theoretical work that explores various ideas and concepts related to the topic which aims to strengthen the traditional worst-case algorithm via machine learning advice. As such, there are no ethical issues associated with the research presented here. The paper includes some experiments which aims to verify the efficiency of the proposed algorithms. But this paper does not involve any experiments or studies that involve human and no personal information or data is used in the analysis. Instead, the focus is on developing theoretical models and frameworks that can help to advance our understanding of the subject matter.

References

1. Ai, L., Wu, X., Huang, L., Huang, L., Tang, P., Li, J.: The multi-shop ski rental problem. In: SIGMETRICS. pp. 463–475. ACM (2014)
2. Anand, K., Ge, R., Kumar, A., Panigrahi, D.: A regression approach to learning-augmented online algorithms. In: NeurIPS, vol. 34 (2021)
3. Anand, K., Ge, R., Panigrahi, D.: Customizing ML predictions for online algorithms. In: ICML. Proceedings of Machine Learning Research, vol. 119, pp. 303–313. PMLR (2020)
4. Angelopoulos, S.: Online search with a hint. In: ITCS. LIPIcs, vol. 185, pp. 51:1–51:16 (2021)
5. Antoniadis, A., Gouleakis, T., Kleer, P., Kolev, P.: Secretary and online matching problems with machine learned advice. In: NeurIPS (2020)
6. Azar, Y.: On-line load balancing. In: Fiat, A., Woeginger, G.J. (eds.) Online Algorithms. LNCS, vol. 1442, pp. 178–195. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0029569
7. Baeza-Yates, R.A., Culberson, J.C., Rawlins, G.J.E.: Searching in the plane. Inf. Comput. **106**(2), 234–252 (1993)
8. Bamas, É., Maggiori, A., Svensson, O.: The primal-dual method for learning augmented algorithms. In: NeurIPS (2020)
9. Charikar, M., Chekuri, C., Feder, T., Motwani, R.: Incremental clustering and dynamic information retrieval. SIAM J. Comput. **33**(6), 1417–1440 (2004)
10. Chrobak, M., Kenyon, C., Noga, J., Young, N.E.: Incremental medians via online bidding. Algorithmica **50**(4), 455–478 (2008)
11. Demaine, E.D., Fekete, S.P., Gal, S.: Online searching with turn cost. Theor. Comput. Sci. **361**(2–3), 342–355 (2006)
12. Dütting, P., Lattanzi, S., Leme, R.P., Vassilvitskii, S.: Secretaries with advice. In: EC, pp. 409–429. ACM (2021)
13. Epstein, L., Levin, A.: Randomized algorithms for online bounded bidding. Inf. Process. Lett. **110**(12–13), 503–506 (2010)
14. Im, S., Kumar, R., Qaem, M.M., Purohit, M.: Non-clairvoyant scheduling with predictions. In: SPAA, pp. 285–294. ACM (2021)
15. Jiang, Z., Panigrahi, D., Sun, K.: Online algorithms for weighted paging with predictions. In: ICALP, pp. 69:1–69:18 (2020)
16. Kao, M., Ma, Y., Sipser, M., Yin, Y.L.: Optimal constructions of hybrid algorithms. J. Algorithms **29**(1), 142–164 (1998)

17. Kao, M., Reif, J.H., Tate, S.R.: Searching in an unknown environment: an optimal randomized algorithm for the cow-path problem. Inf. Comput. **131**(1), 63–79 (1996)
18. Lattanzi, S., Lavastida, T., Moseley, B., Vassilvitskii, S.: Online scheduling via learned weights. In: SODA, pp. 1859–1877 (2020)
19. Lotker, Z., Patt-Shamir, B., Rawitz, D.: Rent, lease, or buy: randomized algorithms for multislope ski rental. SIAM J. Discret. Math. **26**(2), 718–736 (2012)
20. Lykouris, T., Vassilvitskii, S.: Competitive caching with machine learned advice. In: ICML, Proceedings of Machine Learning Research, vol. 80, pp. 3302–3311. PMLR (2018)
21. Meyerson, A.: The parking permit problem. In: FOCS, pp. 274–284. IEEE Computer Society (2005)
22. Mitzenmacher, M., Vassilvitskii, S.: Algorithms with predictions. In: Beyond the Worst-Case Analysis of Algorithms, pp. 646–662. Cambridge University Press (2020)
23. Purohit, M., Svitkina, Z., Kumar, R.: Improving online algorithms via ML predictions. In: NeurIPS, pp. 9684–9693 (2018)
24. Rohatgi, D.: Near-optimal bounds for online caching with machine learned advice. In: SODA, pp. 1834–1845 (2020)
25. Wang, S., Li, J., Wang, S.: Online algorithms for multi-shop ski rental with machine learned advice. In: NeurIPS (2020)