Smoothed Online Combinatorial Optimization Using Imperfect Predictions

Kai Wang[1*], **Zhao Song**[2], **Georgios Theocharous**[2], **Sridhar Mahadevan**[2]
[1]Harvard University, Cambridge, MA
[2]Adobe Research, San Jose, CA
kaiwang@g.harvard.edu, {zsong, theochar, smahadev}@adobe.com

Abstract

Smoothed online combinatorial optimization considers a learner who repeatedly chooses a combinatorial decision to minimize an unknown changing cost function with a penalty on switching decisions in consecutive rounds. We study smoothed online combinatorial optimization problems when an imperfect predictive model is available, where the model can forecast the future cost functions with uncertainty. We show that using predictions to plan for a finite time horizon leads to regret dependent on the total predictive uncertainty and an additional switching cost. This observation suggests choosing a suitable planning window to balance between uncertainty and switching cost, which leads to an online algorithm with guarantees on the upper and lower bounds of the cumulative regret. Empirically, our algorithm shows a significant improvement in cumulative regret compared to other baselines in synthetic online distributed streaming problems.

Introduction

We consider the *smoothed online combinatorial optimization* problem, which is an extension of online convex optimization (Hazan 2019; Shalev-Shwartz et al. 2011; Zinkevich 2003; Hazan, Agarwal, and Kale 2007) and smoothed online convex optimization (Lin et al. 2012a,b). In the smoothed online combinatorial optimization problem, an online learner is repeatedly optimizing a cost function with unknown changing parameter. In every time step, the learner chooses a feasible decision from a combinatorial feasible region before observing the parameter of the cost function. After the learner chooses the decision, the learner receives (i) the cost function parameter and the associated cost (ii) an additional known switching cost function dependent on the chosen decision and the previous decision. The goal of the learner is to minimize the cumulative cost in T time steps, including cost produced by the cost function and the switching cost.

Smoothed online combinatorial optimization is commonly seen in applications with online combinatorial decisions and switching penalty, including ride sharing with combinatorial driver-customer assignment (Jia, Xu, and Liu 2017), distributed streaming system with bipartite data-to-server assignment (Garg 2013; Thein 2014), and A-B testing in advertisement (Bhat et al. 2020). All these examples incur a potential switching cost when the decisions are changed, e.g., reassigning drivers or data to different locations or servers is costly, and changing advertisement campaign requires additional human resources. The challenge of online combinatorial decision-making and the presence of hidden switching cost motivate the study of smoothed online combinatorial optimization.

In this paper, we study the smoothed online combinatorial optimization where an imperfect predictive model is available. We assume that the predictive model can forecast the future cost parameters with uncertainties, and the uncertainties can evolve over time. We measure the performance of online algorithms by *dynamic regret*, which assumes a dynamic offline benchmark, i.e., the optimal performance when the cost function parameters are given a priori and the sequential decisions are allowed to change. The same use of predictions and dynamic regret are also studied in receding horizon control (Mattingley, Wang, and Boyd 2011; Camacho and Alba 2013) in smoothed online convex optimization under different assumptions on the predictions (Chen et al. 2015; Badiei, Li, and Wierman 2015; Chen et al. 2016; Li and Li 2020; Li, Qu, and Li 2020). In our case, the challenges of bounding dynamic regret inherit from smoothed online convex optimization, while the additional combinatorial structure further complicates the analysis.

Main Contribution

Our main contribution is an online algorithm that plans ahead using the imperfect predictions within a dynamic planning window determined based on the predictive uncertainty of the predictive model. We summarize our contributions as follows:

- Given imperfect predictions with uncertainties, we show that planning based on predictions within a finite time horizon leads to a regret bound that is a function of the total predictive uncertainty with an additional potential switching cost. This bound quantifies one source of regret corresponding to the imperfectness of the predictions, while the other source comes from the additional switching cost (Theorem 0.5).

[*]Work done during an internship at Adobe Research.

- Our regret bound in finite time horizon suggests using a dynamic planning window to optimally balance two sources of regret coming from predictive uncertainty and the switching cost, respectively. Iteratively selecting a dynamic planning window to plan ahead leads to a regret bound in infinite time horizon (Theorem 0.6).
- Specifically, when the uncertainties converge to 0 when more data is collected, we show that the cumulative regret is always sublinear (Theorem 0.7), which guarantees the no-regretness of Algorithm 1. We also quantify the dependency of the cumulative regret on the convergence rate of the uncertainty in some special cases (Corollary 0.8).
- Lastly, we show a lower bound on the total regret for any randomized online algorithm when predictive uncertainty is present. The order of the lower bound matches to the order of the upper bound in some special cases, which guarantees the tightness of our online algorithm and the corresponding regret bounds (Corollary 0.9).

Lastly, given predictions and dynamic planning windows, the smoothed online combinatorial optimization problem reduces to an offline combinatorial problem. We use an iterative algorithm to find an approximate solution to the offline problem efficiently, which largely reduces the computation cost compared to solving the large combinatorial problem using mixed-integer linear program.

Empirically, we evaluate our algorithm on the online distributed streaming problem motivated from Apache Kafka with synthetic traffic. We compare our algorithm using predictions and dynamic planning windows with various baselines. Our algorithm using predictions outperforms baselines without using predictions. Our experiments show an improvement of choosing the right dynamic planning windows against algorithms using fixed planning window, which demonstrates the importance of balancing uncertainty and the switching cost. The use of iterative algorithm also largely reduces the computation cost while keeping a comparable performance, leading to an effective scalable online algorithm that can be applied to real-world problems.

Related Work

Online convex optimization Online convex optimization (Gemp and Mahadevan 2016; Hazan 2019; Shalev-Shwartz et al. 2011; Zinkevich 2003) assumes the objective function is convex and no switching cost. In online convex optimization, *static regret* is most commonly used, which assumes a static benchmark with full information but the decisions over the entire time steps have to be static. Various variants of online gradient descents (Zinkevich 2003; Hazan, Agarwal, and Kale 2007; Hazan, Rakhlin, and Bartlett 2007; Srebro, Sridharan, and Tewari 2011; Flaxman, Kalai, and McMahan 2004) were proposed with bounds on the static regret. However, the gradient-based approaches and the regret bounds do not directly generalize to the combinatorial setting due to the discreteness of the feasible region.

Smoothed online convex optimization with predictions Smoothed online convex optimization generalizes online convex optimization by assuming a switching cost that defines the cost of moving from the previous decision to the current one. (Andrew et al. 2013) showed that smoothed online convex optimization can achieve the same static regret bound using the algorithms in online convex optimization without switching cost. In terms of dynamic regret, receding horizon control (Mattingley, Wang, and Boyd 2011) was proposed to leverage the predictions of future time step to make decision. Perfect (Lin et al. 2012b,a) and imperfect (Chen et al. 2015, 2016; Li and Li 2020; Li, Qu, and Li 2020) predictions are used to bound the performance of receding horizon control with fixed planning window size. Separately, chasing convex bodies (Sellke 2020; Bubeck et al. 2019, 2020; Friedman and Linial 1993) shares the same challenge of smoothed online convex optimization but focuses on the competitive ratio.

Nonetheless, the analyses in the convex objectives and feasible regions do not apply to the combinatorial setting. The planning window in receding horizon control is also restricted to be fixed across different time steps.

Online combinatorial optimization and metrical task system Online combinatorial optimization assumes a discrete feasible region that the learner can choose from *before* seeing the cost function. Existing results (Audibert, Bubeck, and Lugosi 2014; Koolen et al. 2010) focus on bounding dynamic regret in the case of linear objectives without switching cost. On the other hand, metrical task system assumes n discrete states that the learner can choose *after* seeing the cost function, and there is a metrical switching cost associated to every switch. Existing results focus on bounding *competitive ratio*, where the competitive ratio is lower bounded by $\Omega(\frac{\log n}{\log \log n})$ (Bartal, Bollobás, and Mendel 2006; Bartal et al. 2003) and upper bounded by $O(\log^2 n)$ (Bubeck et al. 2021). In contrast, *dynamic regret* is a stronger additive guarantee and is more challenging to analyze.

Our work shows that analyzing dynamic regret in an arbitrary smoothed online combinatorial optimization problem becomes tractable when an imperfect predictive model is given.

Problem Statement

An instance of smoothed online combinatorial optimization is composed of a cost function $f : \mathcal{X} \times \mathbf{\Theta} \rightarrow \mathbb{R}_{\geq 0}$ where $\boldsymbol{x} \in \mathcal{X}$ denotes all the feasible decisions that can be taken and $\boldsymbol{\theta} \in \mathbf{\Theta}$ denotes all the possible unknown parameters of the cost function, and a metric $d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}_{\geq 0}$ that is used to measure the distance of different decisions. At each time step t, the learner receives a feature $\xi_t \in \Xi$ that is correlated to the unknown parameters in the future. Based on the given feature ξ_t, the learner can predict the future parameters and choose a feasible decision $\boldsymbol{x}_t \in \mathcal{X}$ without seeing the future parameter $\boldsymbol{\theta}_t$. The parameter $\boldsymbol{\theta}_t$ is revealed after the decision is executed and the learner receives an objective cost $f(\boldsymbol{x}_t, \boldsymbol{\theta}_t)$ with a switching cost $d(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t)$ which measures the movement of the decisions made by time step t and $t-1$. The total cost of an online algorithm ALG up to time T is the summation of both the objective cost and the

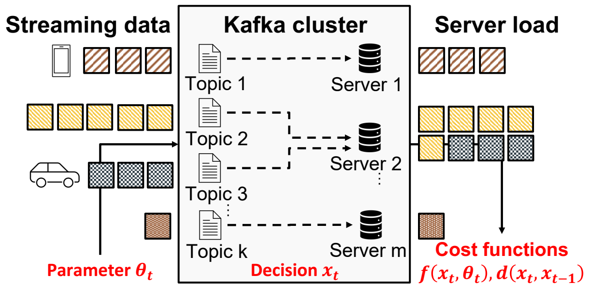


Figure 1: Apache Kafka maintains a bipartite assignment $\boldsymbol{x}_t$ between k topics and m servers to prepare for processing the streaming data. The streaming traffic $\boldsymbol{\theta}_t$ comes later and gets routed to the corresponding servers. A server imbalance cost $f(\boldsymbol{x}_t, \boldsymbol{\theta}_t)$ and a switching cost $d(\boldsymbol{x}_t, \boldsymbol{x}_{t-1})$ due to assignment change are received.

switching cost across all time steps:

$$\text{cost}_T(\text{ALG}) = \sum\nolimits_{t=1}^{T} f(\boldsymbol{x}_t, \boldsymbol{\theta}_t) + d(\boldsymbol{x}_t, \boldsymbol{x}_{t-1}).$$

We want to compare to the offline benchmark OPT in time T that knows all the parameters in advance, which minimizes the total cost defined below:

$$\text{cost}_T(\text{OPT}) = \min_{\boldsymbol{x}_t \in \mathcal{X} \ \forall t} \sum\nolimits_{t=1}^{T} f(\boldsymbol{x}_t, \boldsymbol{\theta}_t) + d(\boldsymbol{x}_t, \boldsymbol{x}_{t-1})$$

Definition 0.1. An online algorithm ALG has a dynamic regret $\rho(T)$ if we have:

$$\text{Reg}_T := \text{cost}_T(\text{ALG}) - \text{cost}_T(\text{OPT}) \leq \rho(T) \quad \forall T.$$

The goal of the learner is to design an online algorithm with a small dynamic regret bound $\rho(T)$.

Example: Online Distributed Streaming Systems

One application of smoothed online combinatorial optimization problems is the online load balancing problem in the distributed streaming system known as Apache Kafka (Garg 2013; Thein 2014). The system is composed of k topics of streaming data and m servers as shown in Figure 1. At each time step t, the system maintains a bipartite assignment $\boldsymbol{x}_t$ between k topics and m servers so that the servers can process the streaming data in real time. Specifically, each topic must be assigned to exactly one server. We use $\boldsymbol{x}_t \in \mathcal{X}_t \subseteq \{0,1\}^{k \times m}$ with $\boldsymbol{x}_{t,i,j} = 1$ to denote assigning the topic i to server j at time t. The learner can use the parameters in the prior H time steps as the feature ξ_t that is correlated to the unknown future parameters. After the assignment is chosen, a new traffic vector $\boldsymbol{\theta}_t \in \mathbb{R}^k$ arrives with each entry representing the number of incoming messages associated to the topic. Figure 1 illustrates how the data-to-server assignment works. A commonly used server imbalance cost is defined as makespan $f(\boldsymbol{x}_t, \boldsymbol{\theta}_t) = \left\|\boldsymbol{x}_t^\top \boldsymbol{\theta}_t\right\|_\infty$, the largest load across all servers.

Paper structure We first discuss how planning based on predictions works and how to bound the associated dynamic regret using predictive uncertainty. Second, we discuss two different sources of regret, predictive uncertainty and the number of planning windows used. We propose to use a dynamic planning window to balance the tradeoff with a regret bound derived. Third, we propose an iterative algorithm to solve an offline problem by decoupling the temporal dependency caused by switching cost. Lastly, an application in distributed streaming system and Apache Kafka is discussed and used in our experiments.

Planning Using Predictions

Motivated by the use of predictions in smoothed online convex optimization (Chen et al. 2016; Li and Li 2020; Antoniadis et al. 2020), this section studies the connection of predictions and predictive uncertainties to the dynamic regret. To conduct the regret analysis below, we require the following assumptions to hold:

Assumption 0.2. The cost function $f(\boldsymbol{x}, \boldsymbol{\theta})$ is Lipschitz in $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ with Lipschitz constant L, i.e., $\|\frac{\partial f(\boldsymbol{x}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\| \leq L$ for all $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{\theta} \in \boldsymbol{\Theta}$.

Assumption 0.3. The switching cost is upper bounded in the feasible region $\mathcal{X}$ by $B = \sup_{x,y \in \mathcal{X}} d(\boldsymbol{x}, \boldsymbol{y})$.

Assumption 0.2 quantifies the change of the cost function with respect to the parameter. Assumption 0.3 quantifies the upper bound of switching cost.

Predictions with Uncertainty

Assumption 0.4. We assume there is a predictive model that is trained based on the revealed parameters prior to time t. At time t, the predictive model takes the feature ξ_t and produces a sequence of predicted future parameters $\{\boldsymbol{\theta}_s^{(t)}\}_{s \in \mathbb{N}, s \geq t}$ with uncertainty $\{\epsilon_s^{(t)}\}_{s \in \mathbb{N}, s \geq t}$, where the distance between the prediction $\boldsymbol{\theta}_s^{(t)}$ and the true parameter $\boldsymbol{\theta}_s$ at time s is bounded by $\|\boldsymbol{\theta}_s - \boldsymbol{\theta}_s^{(t)}\| \leq \epsilon_s^{(t)}$.

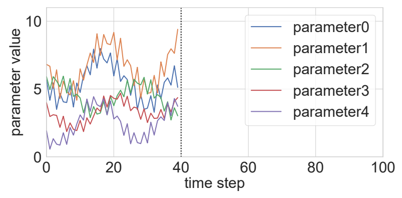
We also assume that the predictive uncertainty $\epsilon_s^{(t)}$ increases in s due to the difficulty of predicting further future parameters, while the predictive uncertainty decreases in t due to more training data available to train the predictive model.

Planning in Fixed Time Horizon

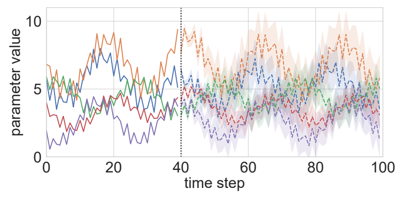
We first analyze the regret in fixed time horizon when we use the predictions to plan accordingly. More precisely, at time t, given the previous decision $\boldsymbol{x}_{t-1}$ at time $t-1$ and the prediction $\{\boldsymbol{\theta}_s^{(t)}\}_{s \in \mathbb{N}, s \geq t}$ of the future time steps, the learner selects a planning window $S \in \mathbb{N}$ to plan for the next S time steps by solving a minimization problem:

$$\begin{aligned} &\{\boldsymbol{x}_s\}_{s \in \{t, t+1, \cdots, t+S-1\}} \\ &= \operatorname*{arg\,min}_{\boldsymbol{x}_s \in \mathcal{X} \ \forall s} \sum\nolimits_{s=t}^{t+S-1} f(\boldsymbol{x}_s, \boldsymbol{\theta}_s^{(t)}) + d(\boldsymbol{x}_s, \boldsymbol{x}_{s-1}). \end{aligned} \quad (1)$$

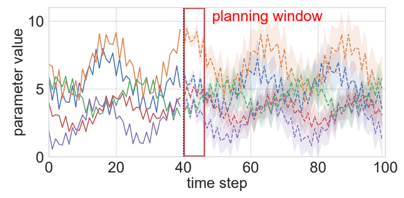
Solving the above finite time horizon optimization problem suggests a solution $\{\boldsymbol{x}_s\}_{s \in \{t, t+1, \cdots, t+S-1\}}$ in the next S time steps to execute starting from time t. This process is summarized in Fig. 2.



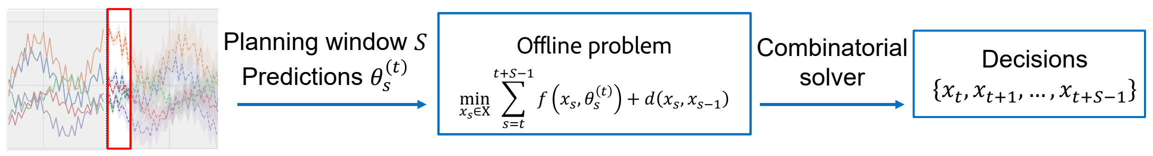
a The learner has access to the historical parameters $\{\boldsymbol{\theta}_s \in \mathbb{R}^k\}_{s<t}$. We plot each entry of the parameter prior to time t as a time series to visualize the trend.



b The learner predicts the future parameters with uncertainty. Each entry of the parameter corresponds to a time series prediction problem.



c Given the predictions, we choose a dynamic planning window such that the total uncertainty within the window is of the same order of the switching cost.



Planning window S
Predictions $\theta_s^{(t)}$
Offline problem
$\min_{x_s \in X} \sum_{s=t}^{t+S-1} f\left(x_s, \theta_s^{(t)}\right) + d(x_s, x_{s-1})$
Combinatorial solver
Decisions
$\{x_t, x_{t+1}, \dots, x_{t+S-1}\}$

d Given the predictions and the planning window, the planning problem reduces to an offline combinatorial problem. We can use any combinatorial solver to find a solution to the offline problem. The solution is executed in the planning window.

Figure 2: This flowchart summarizes how predictions are used to derive planning decisions. Fig. 2a shows the historical data prior to time t as multiple time series. Fig. 2b visualizes the predictions and uncertainty intervals learned from the historical parameters. Fig. 2c demonstrates how to determine the dynamic planning window. Fig. 2d solves an offline problem and executes accordingly.

However, since the predictions are not perfect, the suggested solution might not be the true optimal solution when the true cost function parameters are present. To compare with the true offline optimal solution using the true cost function parameters, we express the offline solution by:

$$\begin{aligned}&\{\boldsymbol{x}'_s\}_{s\in\{t,t+1,\cdots,t+S-1\}}\\&= \mathop{\arg\min}_{\boldsymbol{x}_s\in\mathcal{X}_s\ \forall s} \sum\nolimits_{s=t}^{t+S-1} f(\boldsymbol{x}_s,\boldsymbol{\theta}_s) + d(\boldsymbol{x}_s,\boldsymbol{x}_{s-1})\end{aligned} \quad (2)$$

The only difference between Eq. (1) and Eq. (2) is that Eq. (2) has full access to the future cost parameters, while Eq. (1) uses the predictions instead. We can define the difference by the following regret:

$$\begin{aligned}\mathrm{Reg}_t^{t+S-1}(\boldsymbol{x}_{t-1}) &= \left(\sum\nolimits_{s=t}^{t+S-1} f(\boldsymbol{x}_s,\boldsymbol{\theta}_s) + d(\boldsymbol{x}_s,\boldsymbol{x}_{s-1})\right)\\&- \left(\sum\nolimits_{s=t}^{t+S-1} f(\boldsymbol{x}'_s,\boldsymbol{\theta}_s) + d(\boldsymbol{x}'_s,\boldsymbol{x}'_{s-1})\right).\end{aligned} \quad (3)$$

We have the following bound on the regret:

Theorem 0.5. *Under Assumption 0.2, the regret from time step t to $t+S-1$ in Eq. 3 is upper bounded by:* $\mathrm{Reg}_t^{t+S-1}(\boldsymbol{x}_{t-1}) \leq 2L\sum_{s=t}^{t+S-1}\epsilon_s^{(t)}$. *where L is the Lipschitz constant in Assumption 0.2.*

Theorem 0.5 links the dynamic regret with the total predictive uncertainty in finite time horizon. Notice that the switching cost terms in Eq. (3) are misaligned. Therefore, the proof requires not only the Lipschitzness of the objective function f but also the optimality conditions of both the offline and online planning problems to bound the total cumulative regret.

Algorithm 1: Dynamic Future Planning

Input: Total time steps T. Maximal switching cost B. A predictive model that can produce predictions $\{\boldsymbol{\theta}_{t+s}^{(t)}\}_{s\in\mathbb{N}}$ at time t.

```
1: **Initialization** $t = 1$, # of planning windows $I = 0$
2: **while** $t \leq T$ **do**
3:     Get predictions $\{\boldsymbol{\theta}_s^{(t)}\}_{s\in\mathbb{N},s\geq t}$ and predictive uncertainty $\{\epsilon_s^{(t)}\}_{s\in\mathbb{N},s\geq t}$ from the model.
4:     Find the largest $S$ s.t. $2L\sum_{s=t}^{t+S-1}\epsilon_s^{(t)} \leq B$.
5:     Solve the optimization problem in Eq. (1) with starting time $t$ and planning window $S$ to get $\{\boldsymbol{x}_s\}_{s\in\{t,t+1,\cdots,t+S-1\}}$.
6:     Execute $\boldsymbol{x}_s$ and receive $\boldsymbol{\theta}_s$ with cost $f(\boldsymbol{x}_s,\boldsymbol{\theta}_s) + d(\boldsymbol{x}_s,\boldsymbol{x}_{s-1})$ at time $s \in \{t,\cdots,t+S-1\}$.
7:     Set $t = t + S$, $I = I + 1$.
8: **end while**
```

Infinite Time Horizon and Dynamic Planning Window

In the inifinite time horizon problem, the main idea is to reduce the problem to multiple finite time horizon problems with different planning window sizes.

Recall that the predictive uncertainty often increases when we try to predict the parameters in the far future, i.e., $\epsilon_s^{(t)}$ is increasing in s. Since the regret in Theorem 0.5 directly relates to the predictive uncertainty in the planning window, it suggests keeping the planning window small to reduce the regret.

On the other hand, Theorem 0.5 assumes an identical initial decision $\boldsymbol{x}_{t-1}$ in the online problem (Eq. (1)) and offline problem (Eq. (2)). In the infinite time horizon case, two algorithms may start from different initial decisions, which may create an additional regret upper bounded by the maximum switching cost B due to the misalignment of the initial decision. This observation suggests using larger planning windows to avoid changing between different planning windows.

Therefore, we propose to balance two sources of regret by choosing the largest planning window S such that:

$$2L \sum\nolimits_{s=t}^{t+S-1} \epsilon_s^{(t)} \leq B \tag{4}$$

The choice of the dynamic planning window can ensure that the total excessive predictive uncertainty is upper bounded by cost B, while we also plan as far as possible to reduce the number of planning windows incurred during switching between different finite time horizons. The algorithm is described in Algorithm 1.

Theorem 0.6. *Given Lipschitzness L in Assumption 0.2 and the maximal switching cost B in Assumption 0.3, in T time steps, Algorithm 1 achieves cumulative regret upper bounded by $2BI$, where I is the total number of planning windows used in Algorithm 1.*

Proof sketch. The regret of our algorithm comes from two parts: (i) regret from the discrepancy of the initial decision $\boldsymbol{x}_{t-1}$ and the initial decision of the offline optimal $\boldsymbol{x}^*_{t-1}$ at time t, the start of every planning window, and (ii) the incorrect predictions used in the optimization, which is bounded by Theorem 0.5.

The regret in part (i) is bounded by $d(\boldsymbol{x}_{t-1}, \boldsymbol{x}^*_{t-1}) \leq B$ for every planning window because it would take at most the maximal switching cost B to align different initial decisions before we can compare. Thus the total regret in part (i) is bounded by BI, where I is the number of planning windows executed in Algorithm 1.

The regret in part (ii) is bounded by Theorem 0.5 and the choice of the dynamic planning window in Eq. (4). We have $\text{Reg}_t^{t+S-1}(\boldsymbol{x}^*_{t-1}) \leq 2L \sum_{s=t}^{t+S_i-1} \epsilon_s^{(t)} \leq B$ for the i-th window. We can take summation over all planning windows to bound the total regret in part (ii) by: $\sum_{i=1}^{I} B = BI$. where combining two bounds concludes the proof. ☐

Theorem 0.6 links the excessive dynamic regret to I, the number of planning windows that Algorithm 1 uses. The next step is to bound the number of planning windows I by the total time steps T. In Theorem 0.7, we first show that the cumulative regret is always sublinear in T when the predictive uncertainty converges to 0 when more data is collected.

Theorem 0.7. *Under Assumption 0.2 and 0.3, if $\epsilon_{t+s-1}^{(t)} = o(1)$ in t for all $s \in \mathbb{N}$, i.e., $\epsilon_{t+s-1}^{(t)} \to 0$ when $t \to \infty$, then the cumulative regret of Algorithm 1 is sublinear in T.*

Proof. When the predictive uncertainty $\epsilon_s^{(t)} \to 0$ when $t \to \infty$, the window size S_t that satisfies $2L \sum_{s=t}^{t+S_i-1} \epsilon_s^{(t)} \leq B$ at time t converges to ∞ when $t \to \infty$. This suggests that the number of windows I required in total number of time steps T is strictly smaller than $\Theta(T)$, i.e., $I = o(T)$. By Theorem 0.6, the cumulative regret is upper bounded by $2BI = o(T)$, which is sublinear in T. ☐

Theorem 0.7 guarantees that the cumulative regret of Algorithm 1 in Theorem 0.6 is sublinear when the uncertainty converges to 0. This establishes the no-regretness of Algorithm 1 in dynamic regret, which is only known to be possible in the smoothed online *convex* optimization but not known in the smoothed online *combinatorial* optimization.

In some special cases of the predictive uncertainty, we can further provide a more precise bound on the cumulative regret in the following corollary.

Corollary 0.8. *If the uncertainty satisfies $\epsilon_{t+s-1}^{(t)} = O(\frac{s^a}{t^b})$, $\forall s, t \in \mathbb{N}$ with $a, b \in \mathbb{R}_{\geq 0}$, we have:*

$$\text{Reg}_T \leq \begin{cases} O(T^{1-\frac{b}{a+1}}) & \textit{if } b < a+1 \\ O(\log T) & \textit{if } b = a+1 \\ O(\log\log T) & \textit{if } b > a+1 \end{cases}.$$

Corollary 0.8 is proved by providing a more concrete bound on the number of planning windows I in Theorem 0.6. Corollary 0.8 also quantifies the dependency of the cumulative regret on the convergence rate of predictive uncertainty. When $b > 0$, the cumulative regret is always sublinear, which matches our result in Theorem 0.7.

Lower Bound on The Cumulative Regret

In this section, we provide a lower bound on the expected cumulative regret, showing that no randomized algorithm can achieve an expected cumulative regret lower than a term similar to the upper bound.

Corollary 0.9. *Given $\epsilon_{t+s-1}^{(t)} = \Omega(\frac{s^a}{t^b})$ for all $t, s \in \mathbb{N}$ with $0 \leq b$, there exist instances such that for any randomized algorithm, the expected regret is at least:*

$$\mathbb{E}[\text{Reg}_T] \geq \begin{cases} \Omega(T^{1-b}) & \textit{if } b < 1 \\ \Omega(\log T) & \textit{if } b = 1 \\ \Omega(1) & \textit{if } b > 1 \end{cases}.$$

The lower bound suggests that there is no online learning algorithm that can achieve a cumulative regret that is smaller than the regret in Corollary 0.9. Specifically, we can see that the lower bound matches to the upper bound up to a logarithm factor when $a = 0$, which guarantees the tightness of our upper bound in Corollary 0.8 and Theorem 0.6 in the case of $a = 0$.

Extension to Probabilistic Bounds

In this paper, we primarily focus on the deterministic uncertainty bounds of the predictive model. The same analyses in Section also generalize to probabilistic bounds of the predictive model that hold with high probability, e.g., with probability $1 - \delta_i$ for each prediction in the i-th planning window with size S_i. This kind of probabilistic bounds is commonly seen in the literature of probably approximately correct (PAC) learning, where the predictive error bound can be bounded by the number of training samples used in fitting the underlying hypothesis class. In this case, the regret analysis in Theorem 0.5 needs to additionally consider the event when the uncertainty bounds do not hold, which leads to an additional regret term with order $O(S_i \delta_i)$ in Theorem 0.5, leading to a linear term $\sum_{i=1}^{I} S_i \delta_i$ in Theorem 0.6.

Fortunately, we can also select a decreasing failure probability δ_i in the later planning windows when more samples are collected. As long as we can guarantee that the choice of uncertainty bound $\epsilon_s^{(t)}$ and the failure probability δ_i at time t converge to 0 when more samples are collected, we can obtain a similar result as Theorem 0.7 showing the cumulative regret bound is sublinear in T. This generalizes our results of deterministic bounds to probabilistic bounds.

Experiment Setup

In our experiment, we use the distributed streaming system problems with synthetic data to compare our algorithm with other baselines.

Cost function and switching cost In the distributed streaming system, the learner maintains a bipartite assignment $\boldsymbol{x}_t \in \mathcal{X}_t \subseteq \{0,1\}^{k \times m}$ between k topics and m servers at time step t to process the streaming data, where $\boldsymbol{x}_{t,i,j} = 1$ denotes that topic i is assigned to server j at time t to process the incoming traffic. Once the decision $\boldsymbol{x}_t$ is chosen at time t, a traffic vector $\boldsymbol{\theta}_t \in \boldsymbol{\Theta} \subseteq \mathbb{R}^k$ is revealed.

Given traffic $\boldsymbol{\theta}_t$ and the chosen assignment $\boldsymbol{x}_t$, we define the cost function by $f(\boldsymbol{x}_t, \boldsymbol{\theta}_t) = \|\boldsymbol{x}_t^\top \boldsymbol{\theta}_t\|_\infty$ as the resulting server imbalance cost, which is also known as makespan, i.e., the maximal number of messages a server needs to process across all servers. Minimizing makespan is a well-studied strongly NP-complete problem (Garey and Johnson 1979) with various approximation algorithms (Hochbaum and Shmoys 1987; Leung 1989). Additionally, we define the switching cost by $d(\boldsymbol{x}, \boldsymbol{y}) := \mathbf{1}_k^\top |\boldsymbol{x} - \boldsymbol{y}| \boldsymbol{u}$, where $|\boldsymbol{x} - \boldsymbol{y}| \in \mathbb{R}_{\geq 0}^{k \times m}$ represents the number of switches of each pair of topic and server, and each entry of $\boldsymbol{u} \in \mathbb{R}^m$ denotes the unit switching cost associated to the corresponding server, which is randomly drawn from a uniform distribution $U[0,2]$.

Data generation We assume that there are $k = 10$ topics to be assigned to $m = 3$ servers. We generate k time series, where each represents the trend of incoming traffic $\{\boldsymbol{\theta}_{t,i}\}_{t \in [T]}$ of topic $i \in [k]$ as the cost function parameter. Each time series is generated by a composition of sine waves, an autoregressive process, and a Gaussian process to model the seasonality, trend, and the random process. We use sine waves with periods of 24 and 2 with amplitudes drawn from $U[1,2]$ and $U[0.5,1]$ to model the daily and hourly changes. We use an autoregressive process AR(1) that takes the weighted sum of 0.9 of the previous signal and a 0.1 of a white noise to generate the next signal. Lastly, we use a rational quadratic kernel as the Gaussian process kernel.

Predictive model At time step t, to predict the incoming traffic $\boldsymbol{\theta}_s \in \boldsymbol{\Theta} \subseteq \mathbb{R}^k$ for all $s \geq t$, we collect all the historical data $\{\boldsymbol{\theta}_{s'}\}_{s' < t}$ prior to time t and apply Gaussian process regression using the same rational quadratic kernel on the historical data to generate predictions $\{\boldsymbol{\theta}_s^{(t)}\}_{s \geq t}$ of the future time steps. We use the standard deviation learned from Gaussian process regression as the uncertainty $\{\epsilon_s^{(t)}\}_{s \geq t}$.

Experimental setup For each instance of the load balancing problem, we assume 50 historical data have been collected a priori to stabilize Gaussian process regression. We run different online algorithms for another 100 time steps with hidden incoming traffic to measure the performance of online algorithms. For each setup, we run 10 independent trials with different random seeds to estimate the average performance. All the results are plotted with average value and the corresponding standard deviation.

Experimental Results

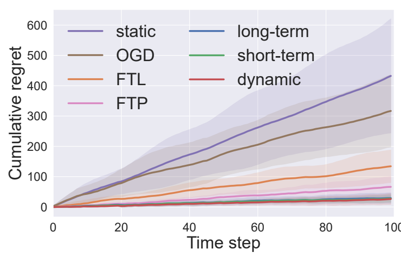
We compare with our algorithm with baselines in the literature of online convex optimization:

- The **static** approach uses the initial assignment and never adjusts dynamically.
- The Online Gradient Descent (**OGD**) updates the assignment by running gradient descent on the cost function received previously and project back to the discrete feasible region.
- The Follow-The-Leader (**FTL**) aggregates all the cost functions received in the past and finds the optimal decision that optimizes the historical cost functions with switching cost.
- The Follow-The-Previous (**FTP**) optimizes the cost function in the last time step.
- The **short-term** algorithm and the **long-term** algorithm both use predictions but with deterministic planning window sizes 1 and 10, respectively.
- The **dynamic** algorithm refers to our algorithm using a dynamic planning window determined by the predictive uncertainty.

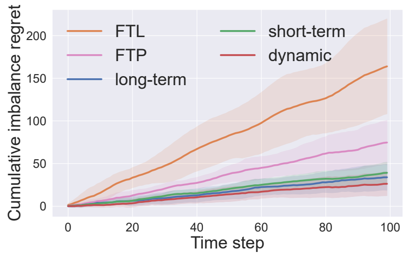
All the algorithms compare with an offline benchmark with full information. Since the offline problem is NP-hard to solve, we split the offline problem into chunks of size 5 and solve each of them optimally using mixed integer program to get the offline performance.

Effect of predictions In Fig. 3, we compare the performance of baselines (static, OGD, FTL, FTP) with approaches using predictions with different planning window sizes (short-term, long-term, dynamic). We first notice that OGD and FTL perform worse than FTP, which simply follows the previous cost function to update solution. Due to the smoothness of the cost function parameters, optimizing over the previous cost function can be a strong baseline.

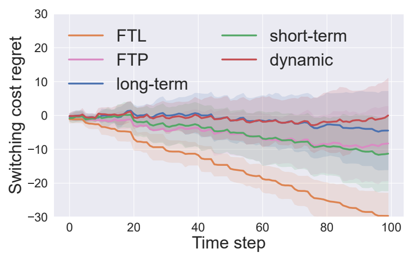
Secondly, the methods using predictions further improve the solution quality. Using predictions can help leverage the seasonality and trend information, and leave the uncertainty to the planning part. On the other hand, the OGD and the FTL algorithms are designed to deal with the case without predictable pattern and switching cost. The different purposes of algorithm design make our algorithm more applicable to our problem.



a Average cumulative regret that includes the imbalance cost and switching cost.

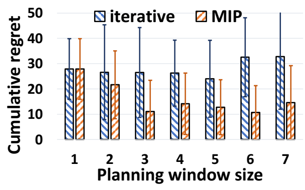


b Average cumulative imbalance regret compared to the offline benchmark.

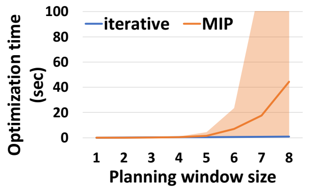


c Average cumulative switching cost regret compared to the offline benchmark.

Figure 3: We compare the performance of our approaches with various baselines without using predictions. The first takeaway is that methods using predictions largely outperform the methods without using predictions in Fig. 3a. Secondly, choosing the right planning window can achieve a better imbalance cost in Fig. 3b with a small increase in the amount of switching cost in Fig. 3c. All the algorithms are compared with an offline benchmark with full information. The shaded area refers to the region within first standard deviation.



a Cumulative regret of methods using different planning window sizes and different optimization approaches.



b Average running time per optimization of different optimization methods with different planning window sizes.

Figure 4: Comparison of different methods of solving Eq. (1) and different planning window sizes.

Lastly, in Fig. 3a, we can see that the dynamic algorithm achieves the smallest cumulative regret compared to the short-term algorithm and the long-term algorithm using planning window with size 1 and 10, respectively. Fig. 3b and Fig. 3c further compare different performance metrics. We can see that our approach of choosing proper planning window can achieve much smaller server imbalance performance while requiring slightly more switching cost only. Methods considering less future effect (FTL, FTP, short-term) can be reluctant to switch and underestimate the benefit of switching, which results in a smaller switching cost but larger imbalance cost. In contrast, the long-term algorithm using larger planning window instead can be harmed by the increasing predictive uncertainty, which leads to incorrect planning decision due to the uncertainty. This result justifies the benefit of predictions and the right planning window to balance between uncertainty and the switching cost.

Effect of planning window size In Fig. 4a, we compare the performance of different choices of planning window size and different ways of solving the offline problem in Eq. (1). First, if we use mixed integer program (**MIP**), we can see a clear improvement by using a larger planning window and a slightly degraded performance after window size exceeds 3. This empirical result matches to our analysis of shorter and longer planning windows, where the dynamic planning window suggests a planning window with size around 3. We also compare with an **iterative** algorithm (Algorithm 2 in Appendix) that is used to approximately solve the NP-hard offline problem in Eq. (1). The effect of planning window size is less significant due to the suboptimality of the iterative algorithm. But we can still see a similar benefit while using an appropriate planning window size.

Fig. 4b compares the runtime of solving Eq. (1) using different approaches and planning window sizes. Runtime of solving the optimization problem is important because decisions have to be made in real time. We can see that MIP requires an exponentially increasing runtime because the combinatorial structure and the linearly increasing number of binary variables when the window size grows. On the other hand, the iterative algorithm solves the problem approximately and more efficiently. In short, the MIP algorithm achieves the best performance but with an expensive computation, while the iterative algorithm scales better but with a loss in the solution quality.

Conclusion

This paper studies the smoothed online combinatorial optimization problem with switching cost. We show that when predictions with uncertainty are available, we can bound the dynamic regret by the convergence of the predictive uncertainty, which links the bound on dynamic regret to the predictability of the incoming cost function parameters. Our analysis suggests using a dynamic planning window dependent on the sequence of predictive uncertainties. Our dynamic planning window can optimize the regret, where we empirically show in our experiments that using a predictive model and an appropriate planning window can further improve the performance.

References

Andrew, L.; Barman, S.; Ligett, K.; Lin, M.; Meyerson, A.; Roytman, A.; and Wierman, A. 2013. A tale of two metrics: Simultaneous bounds on competitiveness and regret. In *COLT*, 741–763.

Antoniadis, A.; Coester, C.; Elias, M.; Polak, A.; and Simon, B. 2020. Online metric algorithms with untrusted predictions. In *ICML*, 345–355.

Audibert, J.-Y.; Bubeck, S.; and Lugosi, G. 2014. Regret in online combinatorial optimization. *Mathematics of Operations Research*, 39(1): 31–45.

Badiei, M.; Li, N.; and Wierman, A. 2015. Online convex optimization with ramp constraints. In *2015 54th IEEE Conference on Decision and Control (CDC)*, 6730–6736.

Bartal, Y.; Bollobás, B.; and Mendel, M. 2006. Ramsey-type theorems for metric spaces with applications to online problems. *Journal of Computer and System Sciences*, 72(5): 890–921.

Bartal, Y.; Linial, N.; Mendel, M.; and Naor, A. 2003. On metric Ramsey-type phenomena. In *STOC*, 463–472.

Bhat, N.; Farias, V. F.; Moallemi, C. C.; and Sinha, D. 2020. Near-optimal AB testing. *Management Science*, 66(10): 4477–4495.

Bubeck, S.; Cohen, M. B.; Lee, J. R.; and Lee, Y. T. 2021. Metrical task systems on trees via mirror descent and unfair gluing. *SIAM Journal on Computing*, 50(3): 909–923.

Bubeck, S.; Klartag, B.; Lee, Y. T.; Li, Y.; and Sellke, M. 2020. Chasing nested convex bodies nearly optimally. In *SODA*, 1496–1508. SIAM.

Bubeck, S.; Lee, Y. T.; Li, Y.; and Sellke, M. 2019. Competitively chasing convex bodies. In *STOC*, 861–868.

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

Chen, N.; Agarwal, A.; Wierman, A.; Barman, S.; and Andrew, L. L. 2015. Online convex optimization using predictions. In *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, 191–204.

Chen, N.; Comden, J.; Liu, Z.; Gandhi, A.; and Wierman, A. 2016. Using predictions in online optimization: Looking forward with an eye on the past. *ACM SIGMETRICS Performance Evaluation Review*, 44(1): 193–206.

Farrell, J.; and Klemperer, P. 2007. Coordination and lock-in: Competition with switching costs and network effects. *Handbook of industrial organization*, 3: 1967–2072.

Flaxman, A. D.; Kalai, A. T.; and McMahan, H. B. 2004. Online convex optimization in the bandit setting: gradient descent without a gradient. *arXiv preprint cs/0408007*.

Friedman, J.; and Linial, N. 1993. On convex body chasing. *Discrete & Computational Geometry*, 9(3): 293–321.

Garey, M. R.; and Johnson, D. S. 1979. *Computers and intractability*, volume 174. freeman San Francisco.

Garg, N. 2013. *Apache kafka*. Packt Publishing Birmingham.

Gemp, I.; and Mahadevan, S. 2016. Online Monotone Optimization. *arXiv preprint arXiv:1608.07888*.

Hazan, E. 2019. Introduction to online convex optimization. *arXiv preprint arXiv:1909.05207*.

Hazan, E.; Agarwal, A.; and Kale, S. 2007. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3): 169–192.

Hazan, E.; Rakhlin, A.; and Bartlett, P. 2007. Adaptive online gradient descent. *Advances in Neural Information Processing Systems*, 20.

Hochbaum, D. S.; and Shmoys, D. B. 1987. Using dual approximation algorithms for scheduling problems theoretical and practical results. *Journal of the ACM (JACM)*, 34(1): 144–162.

Jia, Y.; Xu, W.; and Liu, X. 2017. An optimization framework for online ride-sharing markets. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, 826–835. IEEE.

Koolen, W. M.; Warmuth, M. K.; Kivinen, J.; et al. 2010. Hedging Structured Concepts. In *COLT*, 93–105.

Krishnasamy, S.; Akhil, P.; Arapostathis, A.; Sundaresan, R.; and Shakkottai, S. 2018. Augmenting max-weight with explicit learning for wireless scheduling with switching costs. *IEEE/ACM Transactions on Networking*, 26(6): 2501–2514.

Leung, J. Y. 1989. Bin packing with restricted piece sizes. *Information Processing Letters*, 31(3): 145–149.

Li, Y.; and Li, N. 2020. Leveraging predictions in smoothed online convex optimization via gradient-based algorithms. *arXiv preprint arXiv:2011.12539*.

Li, Y.; Qu, G.; and Li, N. 2020. Online optimization with predictions and switching costs: Fast algorithms and the fundamental limit. *IEEE Transactions on Automatic Control*.

Lin, M.; Liu, Z.; Wierman, A.; and Andrew, L. L. 2012a. Online algorithms for geographical load balancing. In *2012 international green computing conference (IGCC)*, 1–10. IEEE.

Lin, M.; Wierman, A.; Andrew, L. L.; and Thereska, E. 2012b. Dynamic right-sizing for power-proportional data centers. *IEEE/ACM Transactions on Networking*, 21(5): 1378–1391.

Mattingley, J.; Wang, Y.; and Boyd, S. 2011. Receding horizon control. *IEEE Control Systems Magazine*, 31(3): 52–65.

Sellke, M. 2020. Chasing convex bodies optimally. In *SODA*, 1509–1518. SIAM.

Shalev-Shwartz, S.; et al. 2011. Online learning and online convex optimization. *Foundations and trends in Machine Learning*, 4(2): 107–194.

Srebro, N.; Sridharan, K.; and Tewari, A. 2011. On the universality of online mirror descent. In *NIPS*, 2645–2653.

Thein, K. M. M. 2014. Apache kafka: Next generation distributed messaging system. *International Journal of Scientific Engineering and Technology Research*, 3(47): 9478–9483.

Zinkevich, M. 2003. Online convex programming and generalized infinitesimal gradient ascent. In *ICML*, 928–936.