Online Covering with Multiple Experts

Enikő Kevi ENIKO.KEVI@UNIV-GRENOBLE-ALPES.FR
Nguyễn Kim Thắng KIM-THANG.NGUYEN@UNIV-GRENOBLE-ALPES.FR
150 Pl. du Torrent, 38400 Saint-Martin-d'Heres, France

Editors: Matus Telgarsky and Jonathan Ullman

Abstract

Designing online algorithms with machine learning predictions is a recent approach beyond the worst-case paradigm for various practically relevant online problems like scheduling, caching, and clustering. While most previous learning-augmented algorithms focus on integrating the predictions of a single oracle, we study the design of online algorithms with *multiple* prediction sources (experts). To go beyond the performance guarantee of the popular static best expert in hindsight benchmark, we introduce a new benchmark that can be seen as the linear combination of predictions that change over time. We present competitive algorithms in the new dynamic benchmark for 0-1 online covering problems with a performance guarantee of $O(\log K)$ if the objective is linear and $O(\ln(K)) \cdot \frac{\lambda}{(1-\mu \ln(K))}$ if the objective is non-linear, where K is the number of experts and (λ, μ) are parameters of the objective function. Our approach gives a new perspective on combining multiple algorithms in an online manner (a central subject in the online algorithm research community) using machine learning techniques.

Keywords: learning-augmented algorithm, multiple experts, multiple oracles, primal-dual algorithm, covering problem, predictions, online algorithm with advice

1. Introduction

Designing online competitive algorithms with *multiple* prediction sources (experts) is an important research agenda in the domain of Algorithms with Predictions. Given a set of algorithms/experts, can we combine the algorithms/experts in some generic way to obtain a better competitive online strategy? Combining online algorithms into a new algorithm to achieve better results than the individual input algorithms has been a long-standing online algorithm design question (Azar et al., 1993; Blum and Burch, 2000). Its intrinsic difficulty is that when the performance of the given input algorithms (or heuristics, randomized methods) is unclear (especially in the online setting), it is challenging to create a combination that can surpass the performance of the given algorithms. Following the current development of online algorithm design techniques with multiple predictions, this subject has been renewed with different approaches in several settings Gollapudi and Panigrahi (2019b); Almanza et al. (2021); Dinitz et al. (2022); Anand et al. (2022); Antoniadis et al. (2023).

An algorithm's consistency with the experts' suggestions is typically measured by comparing the algorithm's result with the solution of the *best* expert. A representative example

is the popular notion of regret in online learning, which fueled the development of many powerful algorithms and techniques. A natural research question is whether designing competitive algorithms with mathematical performance guarantees with a stronger benchmark than the best expert is possible. This has been a long-standing question in the community of online algorithms (Azar et al., 1993; Blum and Burch, 2000). Comparing an algorithm with a stronger benchmark could provide deeper insights into the learning process and give better ways of exploiting the experts' predictions. In this paper, we propose a new benchmark (the *best linear combination* of the experts) and provide algorithms for online linear/non-linear covering problems in this benchmark.

1.1. Model and Problem

Covering problem with experts. In the *linear* problem setting, we have n resources, and each resource i has a cost per unit c_i that we know in advance ($1 \leq i \leq n$). Let x_i be a non-negative variable representing the amount chosen from resource i. The total cost of a solution is $\sum_{i=1}^{n} c_i x_i$. The problem includes K experts, and the problem's covering-type constraints are revealed online one by one. At each time $t \geq 1$, we receive a covering constraint $\sum_{i=1}^{n} a_i^t x_i \geq 1$ (where $a_i^t \geq 0$) and each expert k provides a solution $(s_{i,k}^t)_{i=1}^n$. Our algorithm can observe the experts' solutions, and afterwards, it must update its own solution (denoted as $(x_i^t)_{i=1}^n$) to satisfy the new constraint while maintaining the satisfaction of the previous ones. This algorithm must update its solution in the sense of online algorithms, formally, $x_i^t \geq x_i^{t-1}\ \forall\ i, t$. Our goal is to design an algorithm that minimizes $\sum_{i=1}^{n} c_i x_i^T$ subject to all online covering constraints t, where $1 \leq t \leq T$. The value T is the last time a constraint is released, and it is not known by the algorithm.

The *non-linear* problem setting is analogous to the linear one; however, the total cost of the solution $(x_i^t)_{i=1}^n$, becomes $f(x^t)$, where f is a non-linear function. This setting captures different classes of problems: online mixed packing and covering, submodular optimization, etc.

Experts. In our model, the experts' predictions are also *online solutions* fulfilling these properties:

1. for every expert k and for every time t the solution $(s_{i,k}^t)_{i=1}^n$ is feasible, therefore, every constraint t' where $1 \leq t' \leq t$ is satisfied;
2. for every expert k and for every time t and for every resource i, the previous expert solutions are irrevocable, therefore $s_{i,k}^t \geq s_{i,k}^{t'}$ for all $t' \leq t$.

These properties can be verified online. If some experts do not satisfy them, we simply ignore those experts both in the decision-making and in the benchmark. We provide motivations and justification for the assumptions in Appendix A.2.

Benchmark. We consider a dynamic benchmark that captures the *best linear combination* of all experts' solutions *over time*. Informally, at any online time step, the benchmark can take a linear combination of the experts' solutions. The linear combination can be *dynamically* changed over time, and it can be different from previous combinations. However,

the benchmark's decisions are also online, so it cannot decrease the value of the decision variables (x_i). We refer to our benchmark with the name LIN−COMB from now on.

The LIN−COMB benchmark's formal description is given in Figure 1. Let $w_k^t \geq 0$ be the weight assigned by the LIN−COMB benchmark to expert k (where $1 \leq k \leq K$) at time $1 \leq t \leq T$. Since we consider a linear combination, the constraint $\sum_{k=1}^{K} w_k^t = 1$ must hold. The solution x_i^t of LIN−COMB at time t is ideally a combination $\sum_{k=1}^{K} w_k^t s_{i,k}^t$. However, by the nature of online decisions, x_i^t must be larger than x_i^{t-1}. Hence, the solution x_i^t is imposed to be larger than both $\sum_{k=1}^{K} w_k^t s_{i,k}^t$ and x_i^{t-1}.

$$\text{Linear:} \quad \min \sum_{i=1}^{n} c_i x_i^T = \sum_{i=1}^{n} c_i \sum_{t=1}^{T} \left(x_i^t - x_i^{t-1}\right)$$

$$\text{Non-linear:} \quad \min f(x^T)$$

$$\text{s.t:} \quad \sum_{k=1}^{K} w_k^t = 1 \; \forall t, \qquad x_i^t \geq \sum_{k=1}^{K} w_k^t s_{i,k}^t \; \forall i, t$$

$$x_i^t \geq x_i^{t-1} \; \forall i, t \qquad w_k^t \geq 0 \; \forall t, k$$

Figure 1: LIN−COMB benchmark

Since every expert's solution is feasible by our assumptions, at each time t and for all resource i, the constructed solution $x_i^t \geq \sum_{k=1}^{K} w_k^t s_{i,k}^t$ constitutes a feasible solution to the covering constraints of the original covering problem. Formally, for every constraint t' with $t' \leq t$,

$$\sum_{i=1}^{n} a_i^{t'} x_i^t \geq \sum_{i=1}^{n} a_i^{t'} \left(\sum_{k=1}^{K} w_k^t s_{i,k}^t \right) = \sum_{k=1}^{K} w_k^t \left(\sum_{i=1}^{n} a_i^{t'} s_{i,k}^t \right) \geq \sum_{k=1}^{K} w_k^t \geq 1$$

where the second inequality holds due to the feasibility of the experts' solutions.

We highlight that the best-expert benchmark is included in LIN−COMB. We get the solution of this benchmark by setting $w_{k^*}^t = 1$ for all t, where $1 \leq t \leq T$, and $w_k^t = 0$ for all $k \neq k^*$, where k^* is the best expert (so $x_i^t = s_{i,k^*}^t$ for all i and t). Indeed, compared to the best expert in hindsight, LIN−COMB is equivalent in the linear setting and is stronger in a more general non-linear setting. We briefly provide an example with non-linear objectives. (For a more detailed discussion, see Appendix A.1.) Consider the makespan minimization problem in which one assigns n unit jobs to n identical machines to minimize the maximum machine load. There are n experts and each expert i assigns all jobs to machine i. Hence, the best expert has the maximum machine load of n (the same for every expert). However, the optimal solution in LIN−COMB can choose $w_i = 1/n$, which results in the makespan of 1. The solution corresponds to the assignment of one job to one machine.

1.2. Our approach and contribution

Approach. We rely on the primal-dual approach in our algorithm design. To lower bound the algorithm's objective value, we relax the linear program formulation of LIN−COMB. Then, we take the dual of the relaxation, which is a lower bound on the relaxation. Following the chain of lower bounds, the dual problem is a lower bound on the original LIN−COMB benchmark.

Both of our proposed algorithms set the decision variables at every time step based on the solution of an internal program. Our approach is inspired by the convex regularization

method of Buchbinder et al. (2014). When the objective cost is a linear function, it is well-known that the regularization function is a shifted entropy function. These functions have been widely used, in particular in the recent breakthrough related to k-server (Bubeck et al., 2018; Buchbinder et al., 2019) and metrical task system problems (Bubeck et al., 2021), in which the entropy functions are shifted by constant parameters.

A novel point in our approach is that the entropy function is shifted by the average of the experts' solutions. Moreover, regarding the constraints of the internal program, instead of using the experts' solutions directly, we define auxiliary solutions that guarantee tight constraint satisfaction. Intuitively, this step is useful to avoid misleading experts' suggestions.

It is more challenging to find a regularization function for non-linear objective functions. In our approach, we propose a new regularizer by coupling the entropy-like function with the gradient-Lipschitz property of the original covering problem's objective function. This new regularization function allows us to analyze the non-linear objective setting similarly to the linear case.

Results. Let ρ be the maximum ratio between the experts' solutions on the resources. Informally, ρ represents the discrepancy across the experts' predictions. Formally, $\rho = \max_i \max_{t',t''} \left\{ \frac{\sum_{k=1}^{K} s_{i,k}^{t'}}{\sum_{k=1}^{K} s_{i,k}^{t''}} : \sum_{k=1}^{K} s_{i,k}^{t''} > 0 \right\}$. Our main results are the following.

First, we present an algorithm for the online linear covering problem that achieves an objective cost at most $O(\ln(K\rho))$ times the cost of the LIN−COMB benchmark. Second, for the online non-linear covering problem[1], we provide an algorithm that achieves an objective cost at most $O(\ln(K\rho)) \cdot \frac{\lambda}{(1-\mu\ln(K\rho))}$ times the cost of LIN−COMB, where (λ,μ) are parameters of the non-linear objective function (precisely, (λ,μ) are smoothness parameters of the objective function that we define later in the corresponding section).

In particular, for 0-1 optimization problems, where the experts provide integer (deterministic or randomized) solutions, our first algorithm is $O(\ln(K))$-competitive with LIN−COMB, and the second one is $O(\ln(K)) \cdot \frac{\lambda}{(1-\mu\ln(K))}$-competitive. Notably, our algorithms are resilient against the prediction quality fluctuations (discussion in Section 1.3 and experiments in Section 4).

1.3. Related work and discussions

Much of the research focusing on surpassing worst-case performance guarantees is motivated by the spectacular advances of machine learning (ML). Lykouris and Vassilvtiskii (2018) introduced a general framework to integrate ML predictions into classical algorithm designs to surpass the worst-case performance limit. As a result, many practically relevant online problems were revisited to enhance existing classical algorithms with ML predictions (see the aforementioned (Lattanzi et al., 2020; Mitzenmacher, 2020; Lykouris and Vassilvtiskii, 2018; Rohatgi, 2020; Antoniadis et al., 2020; Gollapudi and Panigrahi,

1. The precise properties of the objective functions are stated in the corresponding section.

2019a; Kumar et al., 2018; Angelopoulos et al., 2020; Hsu et al., 2019; Kraska et al., 2018; Mitzenmacher, 2018; Antoniadis et al., 2023)).

Integrating multiple predictions into online algorithms has been actively studied. Gollapudi and Panigrahi (2019b) studied the ski rental problem with multiple predictions in which they compare the performance of their algorithm to the optimal solution, given that at least one prediction (among the k predictions) is optimal. Almanza et al. (2021) also considered multiple predictions in the online facility location problem. They compared the performance of their algorithm to the best possible solution obtained on the union of the suggestions. Recently, Dinitz et al. (2022) studied the use of multiple predictors for several problems such as matching, load balancing, and non-clairvoyant scheduling. They provided algorithms that were competitive with the best predictor for such problems. Anand et al. (2022) studies the design of algorithms with multiple experts for linear covering problems. They consider a DYNAMIC benchmark that is intuitively the minimum cost solution that is supported by at least one expert solution at each time step, and claim an $O(\ln(K))$-competitive algorithm in the DYNAMIC benchmark. However, there is a counter-example showing that their algorithm is not competitive in the DYNAMIC benchmark. We provide the counter-example in Appendix B, and this example indicates that one would weaken the DYNAMIC benchmark to be the best-expert benchmark so that the algorithm becomes $O(\ln(K))$-competitive.

Furthermore, Antoniadis et al. (2023) proposed an algorithm with multiple experts for the metrical task system problem. Their benchmark allows switching from one expert to another at each time step, but it does not allow combinations of experts or any solution not suggested by one of the experts. In our LIN−COMB benchmark, the linear combination that evolves over time could result in a solution that is not suggested by any of the experts and potentially, it can be much more efficient. Therefore, the results of Antoniadis et al. (2023) are not applicable to our setting.

Combining online algorithms into a new algorithm to achieve better results than the individual input algorithms has been a long-standing online algorithm design question (Azar et al., 1993; Blum and Burch, 2000). Its intrinsic difficulty is similar to the issue we mentioned earlier: when the performance of the given input algorithms (or heuristics, randomized methods) is unclear (especially in the online setting), it is challenging to create a combination that can surpass the performance of the included algorithms. Following the current development of online algorithm design techniques with multiple predictions, this subject has been renewed with different machine learning approaches. Our paper contributes to this line of research.

2. Online linear covering with experts

Benchmark relaxation for the algorithm. Since our LIN−COMB benchmark is a linear combination of the experts' solutions, the equality $\sum_{k=1}^{K} w_k^t = 1$ must hold, where $w_k^t \geq 0$ is the weight assigned to expert k at time t. In the following, we formulate a relaxed version of LIN−COMB, where $\sum_{k=1}^{K} w_k^t \geq 1$. The relaxed formulation also enables us to avoid the

(online) hard constraint requiring $w_k^t s_{i,k}^t \geq w_k^{t-1} s_{i,k}^{t-1}$ to hold. Instead, we introduce a new variable, y_i^t, to represent the increase of x_i^t compared to x_i^{t-1}. When $w_k^t s_{i,k}^t < w_k^{t-1} s_{i,k}^{t-1}$ during the execution, we set the contribution of i at time t to be 0, and therefore, $y_i^t = 0$. The relaxed formulation of LIN−COMB and its dual are given in Figure 2.

$$\min \sum_{t=1}^{T} \sum_{i=1}^{n} c_i y_i^t$$

$$\sum_{k=1}^{K} w_k^t \geq 1 \qquad \forall\, t$$

$$\sum_{k=1}^{K} \left(w_k^t s_{i,k}^t - w_k^{t-1} s_{i,k}^{t-1} \right) \leq y_i^t \qquad \forall\, i, t$$

$$w_k^t,\ y_i^t \geq 0 \quad \forall\, i, t, k$$

$$\max \sum_{t=1}^{T} \alpha^t$$

$$\alpha^t + \sum_{i=1}^{n} s_{i,k}^t (\beta_i^{t+1} - \beta_i^t) \leq 0 \quad \forall\, k, t$$

$$\beta_i^t \leq c_i \quad \forall\, i, t$$

$$\alpha_i^t,\ \beta_i^t \geq 0 \quad \forall\, i, t$$

Figure 2: The relaxation of the LIN−COMB benchmark and its dual

According to the theorem of weak duality, any feasible solution of the dual program lower bounds any feasible solution of the primal program, and therefore, any feasible dual solution also lower bounds our LIN−COMB benchmark. Following the chain of lower bounds, our approach to design a competitive algorithm is as follows. At every time step t, we build solutions for all x_i^t together with the solutions for the dual problem (α^t, β_i^t). Then, we bound the cost of the algorithm to that of the dual. It is important to emphasize that the designed solution for every x_i^t must be feasible to the covering constraints, but it may *not necessarily* be a linear combination of the experts' solutions.

2.1. Algorithm description

We recall that by our assumptions, the experts' solutions are always feasible and non-decreasing. At the arrival of the t^{th} constraint, expert k provides a feasible solution $s_k^t = (s_{i,k}^t)_{i=1}^n$, such that $s_{i,k}^t \geq s_{i,k}^{t'}$ for all $t' \leq t$ and all i. These assumptions do not exclude the possibility for the experts to provide malicious solutions that instruct the algorithm to use an unnecessarily large amount of resources. Note that one difficulty of designing competitive algorithms comes from the lack of obvious ways to distinguish good expert solutions from (probably many) non-efficient/misleading ones.

To circumvent the issue of malicious suggestions, we preprocess the experts' solutions at each iteration. During the preprocessing, every solution s_k^t is scaled down to make it as tight as possible on the t^{th} constraint while always maintaining $s_{i,k}^t \geq s_{i,k}^{t-1}$ for all i. Additionally, after the down-scaling, we create an auxiliary solution $\hat{s}_k^t$ that is tight for the t^{th} constraint. An important highlight: this auxiliary solution is useful for our algorithm and its analysis, but it does *not* directly participate in forming the actual solution. (Our algorithm only sets the weights of the experts and does not change the experts' solutions.)

Moreover, this is necessary (as shown in Appendix A.3); one cannot simply assume the expert solutions are tight. The auxiliary solution $\hat{s}_k^t$ is created with the following procedure.

Scaling preprocessing. Do the following for each expert k:

1. If $(s_{i,k}^t)_{i=1}^n$ is tight on the t^{th} constraint, then for every i set $\hat{s}_{i,k}^t$ to $s_{i,k}^t$.
2. Otherwise, let $\hat{s}_{i,k}^{t-1}$ be the auxiliary solution of expert k at the previous time $t-1$, meaning that, $\sum_{i=1}^n a_i^{t-1}\hat{s}_{i,k}^{t-1} = 1$. Let $I := \{i : s_{i,k}^t > \hat{s}_{i,k}^{t-1} \cdot \frac{a_i^{t-1}}{a_i^t}\}$. If $i \notin I$ we set $\hat{s}_{i,k}^t \leftarrow s_{i,k}^t$. If $i \in I$, we set $\hat{s}_{i,k}^t$ to be some value in $[\hat{s}_{i,k}^{t-1} \cdot \frac{a_i^{t-1}}{a_i^t}, s_{i,k}^t]$ such that the solution $\hat{s}_{i,k}^t$ becomes tight on the t^{th} constraint.

Lemma 1 *One can always obtain the solutions $\hat{s}_{i,k}^t$ such that $\hat{s}_{i,k}^t \le s_{i,k}^t$ and $\sum_{i=1}^n a_i^t \hat{s}_{i,k}^t = 1$.*

Algorithm 1. At the arrival of the t^{th} constraint

Step (1): solve the following inner convex program and set w^t to be the obtained optimal solution

$$\min_w \left\{ \sum_{i=1}^n c_i \left[\left(\sum_{k=1}^K s_{i,k}^t w_{i,k} + \delta_i^t \right) \ln \left(\frac{\sum_{k=1}^K s_{i,k}^t w_{i,k} + \delta_i^t}{\sum_{k=1}^K s_{i,k}^{t-1} w_{i,k}^{t-1} + \delta_i^{t-1}} \right) - \sum_{k=1}^K s_{i,k}^t w_{i,k} \right] \right\}$$

$$(\gamma^t) \qquad \sum_{i=1}^n a_i^t \left(\sum_{k=1}^K \hat{s}_{i,k}^t w_{i,k} \right) \ge 1 \qquad \forall\, t$$

$$(\lambda_i) \qquad \sum_{k=1}^K w_{i,k} \ge 1 \qquad \forall\, i$$

$$(\mu_i^t) \qquad \sum_{k=1}^K s_{i,k}^t w_{i,k} \ge 0 \qquad \forall\, i,t$$

where $\delta_i^t = \frac{1}{K}\sum_k s_{i,k}^t$. Note that we use the auxiliary solution $\hat{s}_{i,k}^t$ in the first constraint. For every i where $s_{i,k}^t = 0$ for all k, the term related to i is not included in the objective function ($w_{i,k} = 0$ for all k beforehand).

Step (2)**:** For all i, if $\sum_k w_{i,k}^t s_{i,k}^t > x_i^{t-1}$ then set $x_i^t \leftarrow \sum_k w_{i,k}^t s_{i,k}^t$; otherwise set $x_i^t \leftarrow x_i^{t-1}$.

Remark. To set $(x_i^t)_{i=1}^n$ at each time t, our proposed algorithm solves an internal convex program, whose objective uses a shifted entropy function as a convex regularizer. To avoid a possible division by 0, we use a dummy expert. This expert sets initially each variable to some small value, then follows the greedy heuristic at each arriving constraint. The presence of a dummy expert only changes the competitive ratio from $O(\log(K))$ to $O(\log(K+1))$, so to simplify the notation, we display all other occurrences of the competitive ratio as $O(\log(K))$.

2.2. Performance Analysis

As w^t is the optimal solution of the convex program and $(\gamma^t, \lambda_i, \mu_i^t)$ is the optimal solution of its dual, the following Karush-Kuhn-Tucker (KKT) and complementary slackness conditions hold:

$$\left[\sum_{i=1}^{n} a_i^t \left(\sum_{k=1}^{K} \hat{s}_{i,k}^t w_{i,k}^t\right) - 1\right]\gamma^t = 0 \;\forall t, \qquad \left[\sum_{k=1}^{K} w_{i,k}^t - 1\right]\lambda_i = 0 \;\forall i, t,$$

$$\left[\sum_{k=1}^{K} s_{i,k}^t w_{i,k}^t\right]\mu_i^t = 0 \;\forall i, t, \qquad \gamma^t, \lambda_i, \mu_i^t \qquad \geq 0 \;\forall i, t$$

and

$$c_i s_{i,k}^t \ln\left(\frac{\sum_{k=1}^{K} s_{i,k}^t w_{i,k}^t + \delta_i^t}{\sum_{k=1}^{K} s_{i,k}^{t-1} w_{i,k}^{t-1} + \delta_i^{t-1}}\right) - a_i^t \hat{s}_{i,k}^t \gamma^t - \lambda_i - s_{i,k}^t \mu_i^t = 0 \qquad \forall\, i, k, t$$

Moreover, if $\sum_k w_{i,k}^t s_{i,k}^t > 0$, meaning that $\mu_i^t = 0$, then

$$c_i s_{i,k}^t \ln\left(\frac{\sum_{k=1}^{K} s_{i,k}^t w_{i,k}^t + \delta_i^t}{\sum_{k=1}^{K} s_{i,k}^{t-1} w_{i,k}^{t-1} + \delta_i^{t-1}}\right) - a_i^t \hat{s}_{i,k}^t \gamma^t - \lambda_i = 0 \tag{1}$$

Dual variables and feasibility. We set the dual variables of the linear program relaxation of our `LIN-COMB` benchmark based on the dual variables of the convex program used inside the algorithm. The dual formulation is visible on Figure 2.

$$\alpha^t = \frac{1}{\ln(K\rho)}\left(\gamma^t + \sum_{i=1}^{n} \lambda_i\right), \qquad \beta_i^t = \frac{1}{\ln(K\rho)} c_i \ln\left(\frac{(1 + 1/K) \cdot \max_{t'} \sum_{k=1}^{K} s_{i,k}^{t'}}{\sum_{k=1}^{K} s_{i,k}^{t-1} w_{i,k}^{t-1} + \delta_i^{t-1}}\right)$$

where recall that $\rho = \max_{i,t',t''} \left\{ \frac{\sum_{k=1}^{K} s_{i,k}^{t'}}{\sum_{k=1}^{K} s_{i,k}^{t''}} : \sum_{k=1}^{K} s_{i,k}^{t''} > 0 \right\}$.

Lemma 2 *The x_i^t solutions set by Algorithm 1 for the original covering problem and the dual variables (α^t, β_i^t) of the `LIN-COMB` benchmark's linear program relaxation are feasible.*

The proofs of Lemmas 1 and 2 are represented in Appendix C.

Theorem 3 *Algorithm 1 is $O(\ln(K\rho))$-competitive with the `LIN-COMB` benchmark.*

Proof We show that the algorithm's solution increases the primal objective value of the original covering problem by at most $O(\ln(K\rho))$ times the value of the relaxation's dual solution, which serves as the lower bound on the LIN−COMB benchmark.

$$\sum_{i=1}^{n} c_i(x_i^t - x_i^{t-1}) = \sum_{i:x_i^t > x_i^{t-1}} c_i(x_i^t - x_i^{t-1}) \leq \sum_{i:x_i^t > x_i^{t-1}} c_i(x_i^t + \delta_i^t) \ln\left(\frac{x_i^t + \delta_i^t}{x_i^{t-1} + \delta_i^t}\right) \tag{2}$$

$$\leq \sum_{i:x_i^t > x_i^{t-1}} c_i(x_i^t + \delta_i^t) \ln\left(\frac{x_i^t + \delta_i^t}{x_i^{t-1} + \delta_i^{t-1}}\right) \tag{3}$$

$$= \sum_{i:x_i^t > x_i^{t-1}} c_i \left[\left(\sum_{k=1}^{K} s_{i,k}^t w_{i,k}^t + \frac{1}{K}\sum_{k=1}^{K} s_{i,k}^t\right) \ln\left(\frac{\sum_{k=1}^{K} s_{i,k}^t w_{i,k}^t + \delta_i^t}{x_i^{t-1} + \delta_i^{t-1}}\right)\right] \tag{4}$$

$$\leq \sum_{i:x_i^t > x_i^{t-1}} c_i \left[\left(\sum_{k=1}^{K} s_{i,k}^t w_{i,k}^t + \frac{1}{K}\sum_{k=1}^{K} s_{i,k}^t\right) \ln\left(\frac{\sum_{k=1}^{K} s_{i,k}^t w_{i,k}^t + \delta_i^t}{\sum_{k=1}^{K} s_{i,k}^{t-1} w_{i,k}^{t-1} + \delta_i^{t-1}}\right)\right] \tag{5}$$

$$= \sum_{i:x_i^t > x_i^{t-1}} \sum_{k=1}^{K} (w_{i,k}^t + 1/K) c_i s_{i,k}^t \ln\left(\frac{\sum_{k=1}^{K} s_{i,k}^t w_{i,k}^t + \delta_i^t}{\sum_{k=1}^{K} s_{i,k}^{t-1} w_{i,k}^{t-1} + \delta_i^{t-1}}\right)$$

$$= \sum_{i:x_i^t > x_i^{t-1}} \sum_{k=1}^{K} (w_{i,k}^t + 1/K)\left(a_i^t \hat{s}_{i,k}^t \gamma^t + \lambda_i\right) \tag{6}$$

$$\leq \sum_{i=1}^{n}\sum_{k=1}^{K} (w_{i,k}^t + 1/K)\left(a_i^t \hat{s}_{i,k}^t \gamma^t + \lambda_i\right)$$

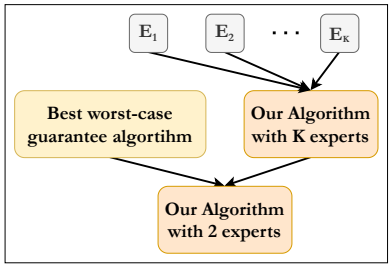
$$= \sum_{i=1}^{n} a_i^t \left(\sum_{k=1}^{K} w_{i,k}^t \hat{s}_{i,k}^t\right)\gamma^t + \sum_{i=1}^{n}\left(\sum_{k=1}^{K} w_{i,k}^t\right)\lambda_i + \frac{1}{K}\sum_{k=1}^{K}\left(\sum_{i=1}^{n} a_i^t \hat{s}_{i,k}^t\right)\gamma^t + \frac{1}{K}\sum_{k=1}^{K}\sum_{i=1}^{n}\lambda_i$$

$$= 2\gamma^t + 2\sum_{i=1}^{n}\lambda_i = 2\ln(K\rho)\alpha^t \tag{7}$$

The above corresponding transformations hold since:
(2) follows from the inequality $a - b \leq a\ln(a/b)$ for all $0 < b \leq a$;
(3) holds since $\delta_i^t \geq \delta_i^{t-1}$ (because $s_{i,k}^t \geq s_{i,k}^{t-1}$ for all i, k, t);
(4) is valid because $x_i^t > x_i^{t-1}$, so $x_i^t = \sum_k s_{i,k}^t w_{i,k}^t$;
(5) is by the design of the algorithm: $x_i^{t-1} \geq \sum_k s_{i,k}^{t-1} w_{i,k}^{t-1}$;
(6) holds since given that $x_i^t > x_i^{t-1} \geq 0$ (so $\sum_k s_{i,k}^t w_{i,k}^t = x_i^t > 0$), the KKT condition (1) applies;
(7) is true due to the complementary slackness conditions and that $\sum_i a_i^t \hat{s}_{i,k}^t = 1$.
By weak duality, the theorem follows. ■

Corollary 4 *For* 0-1 *optimization problems in which the experts provide integer (deterministic or randomized) solutions, our algorithm is* $O(\ln(K))$*-competitive with the* LIN−

COMB benchmark. Further, there exists an algorithm whose performance is $O(\ln(K))$-competitive with the LIN−COMB benchmark and is up to a constant factor compared to the best guarantee in the worst-case benchmark.



Proof Recall that for the 0-1 optimization problems, deterministic or randomized experts provide solution $s^t_{i,k}$ such that $s^t_{i,k} \in \{0, 1\}$ for every i, k, t. (In particular, a randomized solution is a distribution over $0-1$ supports.) Therefore,

$$\rho = \max_i \max_{t',t''} \left\{ \frac{\sum_{k=1}^K s^{t'}_{i,k}}{\sum_{k=1}^K s^{t''}_{i,k}} : \sum_{k=1}^K s^{t''}_{i,k} > 0 \right\} \leq \frac{K}{1}$$

Therefore, the competitive ratio of our algorithm with the LIN−COMB benchmark is $O(\log(K\rho)) = O(\log(K^2))$. To obtain an algorithm that is competitive with both the LIN−COMB and the worst-case benchmarks, we proceed as follows. (The figure serves as an illustration in which $E_1, \ldots, E_K$ correspond to the K experts.) We first apply Algorithm 1 on the K experts' predictions to obtain an online algorithm, named A. Algorithm A is $O(\ln(K))$-competitive with the LIN−COMB benchmark. Let B be the algorithm with the best-known worst-case guarantee. We apply Algorithm 1 one more time to the two algorithms, A and B. The final online algorithm is $O(\ln(2))$-competitive with both A and B. In other words, its performance is $O(\ln(K))$-competitive with the LIN−COMB benchmark and is up to a constant factor to the best-known guarantee in the worst-case benchmark. ■

3. Online non-linear covering with experts

While linear programs are suitable to represent many optimization problems, they are not expressive enough to capture some practically relevant problems (for example, makespan minimization and congestion management). This section takes a step further towards a general framework and proposes an algorithm with multiple experts for 0-1 optimization problems with non-linear objectives and linear constraints.

The performance guarantee proof of our previous algorithm relies on the primal-dual method, which bounds the algorithm's total cost by bounding at each time step the primal with the dual increase. The LIN−COMB benchmark's non-linear objective function (visible on Figure 1) does not allow us to express the objective's cost by additive terms like in the linear case. To keep the primal-dual proving technique, we reformulate LIN−COMB as finding the minimum cost linear combination among all possible linear combinations. This formulation has an exponential number of variables and constraints, but it transforms the non-linear objective function into a linear one.

Benchmark reformulation. Denote $\mathcal{E} = \{1, 2, \ldots, n\}$ the set of indices of variable x in the original covering problem. Then, let $S \subseteq \mathcal{E}$ be a solution if $\mathbf{1}_S$ corresponds to a feasible

$$\min \sum_{t=1}^{T} \sum_{S} c_S y_S^t$$

$$(\alpha^t) \quad \sum_{k=1}^{K} w_k^t \geq 1 \qquad \forall\, t$$

$$(\beta_i^t) \quad \sum_{S: i \in S} z_S^t \geq \sum_{k=1}^{K} s_{i,k}^t w_k^t \qquad \forall\, i, t$$

$$(\gamma^t) \quad \sum_{S} z_S^t = 1 \qquad \forall\, t$$

$$(\xi_S^t) \quad y_S^t = z_S^t - z_S^{t-1} \qquad \forall\, S, t$$

$$w_k^t, y_S^t, z_S^t \geq 0 \qquad \forall\, k, S, t$$

$$\max \sum_{t=1}^{T} (\alpha^t + \gamma^t)$$

$$\alpha^t - \sum_{i=1}^{n} s_{i,k}^t \beta_i^t \leq 0 \qquad \forall\, k, t$$

$$\xi_S^t \leq c_S \qquad \forall\, S, t$$

$$\sum_{i \in S} \beta_i^t + \gamma^t - \xi_S^t + \xi_S^{t+1} \leq 0 \qquad \forall\, S, t$$

$$\alpha^t, \beta_i^t \geq 0 \qquad \forall\, S, i, t$$

Figure 3: Reformulation and relaxation of the LIN−COMB benchmark and its dual

solution. Recall that we consider 0-1 optimization problems, so each coordinate x_i of a feasible solution has a value at most 1. Let z_S be the indicator variable for solution S: $z_S = 1$ iff every variable $x_i = 1$ if $i \in S$, and $x_i = 0$ if $i \notin S$; and $z_S = 0$ otherwise. In other words, z_S indicates if the final solution is S. The formulation and its dual are on Figure 3. On the left of the figure, the first constraint guarantees that we have a linear combination of the experts' solutions. The second constraint guarantees that if the linear combination of the experts' solutions sets a variable x_i to a non-zero value, then some solutions that includes x_i must be used as well. The third constraint ensures that at each time step, one solution should be selected. In the last constraint, y_S^t intuitively represents the fractional increase of the usage of solution S at time t.

3.1. Non-linear objective function properties

For vectors $x, y \in \mathcal{R}^n$, we say that $x \geq y$ iff $x_i \geq y_i$ for all $1 \leq i \leq n$. We consider objective function $f : [0,1]^n \to \mathbb{R}^+$ that satisfies the following properties:

- f is non-decreasing, i.e., $f(x) \geq f(y)$ for all $x \geq y$,
- the gradient of f is non-decreasing on all coordinates, i.e., $\nabla_i f(x) = \partial f(x)/\partial x_i \geq \nabla_i f(y) := \partial f(y)/\partial y_i$ for all $x \geq y$. For example, if f is a polynomial with non-negative coefficients then it satifies this property.

Moreover, the objective function also satisfies the gradient-Lipschitz and (λ, μ)-locally-smooth properties below.

Definition 5 (L-gradient-Lipschitz) *A function f is L-gradient-Lipschitz, if its derivatives are Lipschitz-continuous with the constant L. Formally, for all x and y:*

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$$

Definition 6 ((λ, μ)-local-smoothness) *A differentiable function $f : [0,1]^n \to \mathbb{R}^+$ is* (λ, μ)-locally-smooth *if for for every set $S \subseteq \{1, 2, \ldots, n\}$, the following inequality holds:*

$$\sum_{i \in S} \nabla_i f(x) \leq \lambda f\left(\mathbf{1}_S\right) + \mu f\left(x\right)$$

where $\nabla_i F(x)$ denotes $\partial F(x)/\partial x_i$ and $\mathbf{1}_S$ is the vector where the entry of the i^{th} coordinate equals 1 if $i \in S$ and equals 0 otherwise.

Note that the notation of gradient-Lipschitz is also called smoothness in literature. The notation of (λ, μ)-locally-smoothness, introduced in Thang (2020), is rooted in the work of Roughgarden (2015) in the domain of algorithmic game theory in which it is termed the smoothness argument. To avoid confusion, we use the terms gradient-Lipschitzness and (λ, μ)-local-smoothness with the abovementioned definitions.

3.2. Algorithm description

Similar to Algorithm 1, we apply the scaling preprocessing procedure (see Section 2.1) to produce auxiliary solution $(s^t_{i,k})_{i=1}^n$ that are tight on the t^{th} constraint. We also include the dummy expert to avoid division by zero.

Algorithm 2. The algorithm follows the same steps of Algorithm 1 but with the following objective function of the inner convex program:

$$\begin{aligned}\min_w \Bigg\{ &\sum_{i=1}^n \nabla_i f\left(x^{t-1}\right) \left[\left(\sum_{k=1}^K s^t_{i,k} w_{i,k} + \delta^t_i \right) \ln \left(\frac{\sum_{k=1}^K s^t_{i,k} w_{i,k} + \delta^t_i}{\sum_{k=1}^K s^{t-1}_{i,k} w^{t-1}_{i,k} + \delta^{t-1}_i} \right) - \sum_{k=1}^K s^t_{i,k} w_{i,k} \right] \\ &+ \sum_{i=1}^n \frac{L}{2} \left(\sum_{k=1}^K s^t_{i,k} w_{i,k} + \delta^t_i - 2 \sum_{k=1}^K s^{t-1}_{i,k} w^{t-1}_{i,k} - 2\delta^{t-1}_i \right) \left(\sum_{k=1}^K s^t_{i,k} w_{i,k} + \delta^t_i \right) \cdot \\ &\qquad \cdot \ln \left(\frac{\sum_{k=1}^K s^t_{i,k} w_{i,k} + \delta^t_i}{\sum_{k=1}^K s^{t-1}_{i,k} w^{t-1}_{i,k} + \delta^{t-1}_i} \right) \\ &- \sum_{i=1}^n \frac{L}{4} \left(\sum_{k=1}^K s^t_{i,k} w_{i,k} + \delta^t_i - 2 \sum_{k=1}^K s^{t-1}_{i,k} w^{t-1}_{i,k} - 2\delta^{t-1}_i \right)^2 \Bigg\}\end{aligned}$$

where $\delta^t_i = \frac{1}{K} \sum_k s^t_{i,k}$ and L is the gradient-Lipschitz parameter of function f.

The objective function of the inner convex program is based on the entropy regularizer with the gradient of the objective function f. However, due to the difficulty raised by the

non-linearity of f, we leverage the gradient-Lipschitz property of f and Introduce additional terms into the inner convex program in order to bound the algorithm's cost. Note that our algorithm does not require knowledge of the parameters (λ, μ)-local-smoothness of the function. The latter is only used in the analysis of the performance guarantee.

Theorem 7 *Algorithm 2 is $\frac{O(\ln(K\rho))\cdot\lambda}{1-2\mu\ln(K\rho)}$-competitive within the* LIN−COMB *benchmark. Precisely, the algorithm's objective at most $\frac{O(\ln(K\rho))\cdot\lambda}{1-2\mu\ln(K\rho)}$ times that of the optimal* LIN−COMB *solution plus a fixed term $\frac{8Ln\sqrt{n}\ln(K\rho)}{1-2\mu\ln(K\rho)}$.*

Subsequently, by the same combination as in Theorem 4, one can guarantee the performance both in the LIN−COMB benchmark and in comparison to the best-known algorithm in the worst-case model.

Corollary 8 *For 0-1 optimization problems in which the experts provide integer (deterministic or randomized) solutions, there exists an algorithm whose performance is $\frac{O(\ln K)\lambda}{1-2\mu\ln(K)}$-competitive with the* LIN−COMB *benchmark and is up to a constant factor compared to the best-known guarantee in the worst-case benchmark.*

An immediate application of this corollary is to ℓ_p-norm objective functions. The class of ℓ_p-norms encompasses several important problems in combinatorial (0–1) optimization, such as makespan minimization (introduced earlier), online vector scheduling, and energy minimization, among others. For degree-p polynomials with non-negative coefficients (including p-powers of ℓ_p-norms), it was shown in Thang (2020), that they are $\left(O(p\ln K)^p, \frac{p-1}{2p\ln K}\right)$-local-smooth. By applying our corollary on ℓ_p^p functions and taking the p^{th}-root, the framework yields (for instance, for the running example of makespan minimization discussed in the introduction) a $O(p\cdot\ln K)$-competitive algorithm in the LIN−COMB benchmark, and within a constant factor of the best-known guarantees in the worst-case benchmark.

4. Experiments

In this section we briefly mention our empirical findings with Algorithm 1 for online linear covering problems. The analysis in-depth is given in Appendix E.

Comparison. The best standard online algorithm for general covering problems without experts is the online multiplicative weight update (MWU) algorithm. When a new constraint arrives in the online problem, the MWU algorithm increases each variable x_i in the constraint with a rate of $\frac{a_i^t}{c_i}(x_i+1/n)$, where n is the total number of variables. We compare our algorithm with MWU algorithm.

Input. We evaluated the performance of Algorithm 1 on the MWU's pathological input. This instance includes n variables and n constraints with uniform costs and coefficients. Each arriving constraint includes one less variable. While the optimal solution is 1, the worst-case guarantee of MWU is $O(\log n)$. For our algorithm, we provided n experts, where $(n-1)$ experts suggest an adversarial trivial solution to set all variables to 1, while 1 expert suggests the optimal offline solution.

Highlight. Our algorithm managed to identify the good expert among the majority of adversaries, obtaining a better objective value, than MWU. To investigate the impact of the number of variables and the number of experts on our algorithm, we executed this example with several input sizes. The result of this experiment is visible on Figure 4.

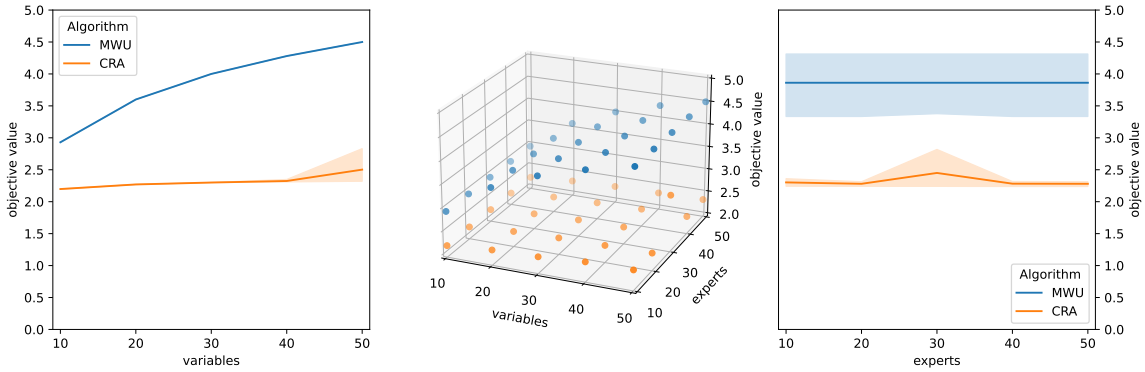


Figure 4: Experiment with varying numbers of variables and experts on the MWU worst-case instance. The left and right plots show the result visible on the middle 3D plot with two dimensions. The shaded areas correspond to the 95% confidence intervals.

5. Conclusion

We introduce a dynamic LIN−COMB benchmark in the setting of multiple expert predictions that encapsulates and extends the traditional static best expert in hindsight benchmark. We give competitive algorithms for the online linear and non-linear covering problems in this benchmark. Many practical problems can be formalized in our setting, such as network design, energy minimization, TCP acknowledgment, facility location, etc. Our work naturally gives rise to open directions of designing competitive algorithms for online packing problems (with both linear and non-linear) objectives.

Acknowledgments

This work was funded by the project PREDICTIONS, grant ANR-23-CE48-0010 from the French National Research Agency (ANR).

References

Matteo Almanza, Flavio Chierichetti, Silvio Lattanzi, Alessandro Panconesi, and Giuseppe Re. Online facility location with multiple advice. In *Advances in Neural Information Processing Systems*, volume 34, pages 4661–4673, 2021.

Keerti Anand, Rong Ge, Amit Kumar, and Debmalya Panigrahi. Online algorithms with multiple predictions. In *Proc. 39th International Conference on Machine Learning*, 2022.

Spyros Angelopoulos, Christoph Dürr, Shendan Jin, Shahin Kamali, and Marc Renault. Online Computation with Untrusted Advice. In *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*, volume 151, pages 52:1–52:15, 2020.

Antonios Antoniadis, Christian Coester, Marek Elias, Adam Polak, and Bertrand Simon. Online metric algorithms with untrusted predictions. In *International Conference on Machine Learning*, pages 345–355, 2020.

Antonios Antoniadis, Christian Coester, Marek Eliáš, Adam Polak, and Bertrand Simon. Mixing predictions for online metric algorithms, 2023.

Yossi Azar, Andrei Z Broder, and Mark S Manasse. On-line choice of on-line algorithms. In *Symposium of Discrete Algorithms (SODA)*, pages 432–440, 1993.

Avrim Blum and Carl Burch. On-line learning and the metrical task system problem. *Machine Learning*, 39(1):35–58, 2000.

Sébastien Bubeck, Michael B Cohen, Yin Tat Lee, James R Lee, and Aleksander Madry. K-server via multiscale entropic regularization. In *Proc. 50th Symposium on Theory of Computing*, pages 3–16, 2018.

Sébastien Bubeck, Michael B Cohen, James R Lee, and Yin Tat Lee. Metrical task systems on trees via mirror descent and unfair gluing. *SIAM Journal on Computing*, 50(3):909–923, 2021.

Niv Buchbinder, Shahar Chen, and Joseph (Seffi) Naor. Competitive analysis via regularization. In *Proc. 25th Symposium on Discrete Algorithms*, pages 436–444, 2014.

Niv Buchbinder, Anupam Gupta, Marco Molinaro, and Joseph Naor. k-servers with a smile: Online algorithms via projections. In *Proc. 30th Symposium on Discrete Algorithms*, pages 98–116, 2019.

Michael Dinitz, Sungjin Im, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Algorithms with prediction portfolios. In *Advances in Neural Information Processing Systems*, 2022.

Sreenivas Gollapudi and Debmalya Panigrahi. Online algorithms for rent-or-buy with expert advice. In *International Conference on Machine Learning*, pages 2319–2327, 2019a.

Sreenivas Gollapudi and Debmalya Panigrahi. Online algorithms for rent-or-buy with expert advice. In *Proceedings of the 36th International Conference on Machine Learning*, 2019b.

Chen-Yu Hsu, Piotr Indyk, Dina Katabi, and Ali Vakilian. Learning-based frequency estimation algorithms. In *Proc. Conference on Learning Representations*, 2019.

Tim Kraska, Alex Beutel, Ed H Chi, Jeffrey Dean, and Neoklis Polyzotis. The case for learned index structures. In *Proc. Conference on Management of Data*, pages 489–504, 2018.

Ravi Kumar, Manish Purohit, and Zoya Svitkina. Improving online algorithms via ML predictions. In *Proc. 32nd Conference on Neural Information Processing Systems*, pages 9684–9693, 2018.

Silvio Lattanzi, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Online scheduling via learned weights. In *Proc. Symposium on Discrete Algorithms*, pages 1859–1877, 2020.

Thodoris Lykouris and Sergei Vassilvtiskii. Competitive caching with machine learned advice. In *International Conference on Machine Learning*, pages 3296–3305, 2018.

Michael Mitzenmacher. A model for learned bloom filters, and optimizing by sandwiching. In *Proc. Conference on Neural Information Processing Systems*, pages 464–473, 2018.

Michael Mitzenmacher. Scheduling with predictions and the price of misprediction. In *Proc. 11th Innovations in Theoretical Computer Science Conference*, 2020.

Dhruv Rohatgi. Near-optimal bounds for online caching with machine learned advice. In *Proc. Symposium on Discrete Algorithms*, pages 1834–1845, 2020.

Tim Roughgarden. Intrinsic robustness of the price of anarchy. *Journal of the ACM*, 62(5): 32, 2015.

Nguyen Kim Thang. Online primal-dual algorithms with configuration linear programs. In *Proc. 31st International Symposium on Algorithms and Computation*, 2020.

Appendix A. Models

A.1. Comparaison between LIN−COMB and the best-expert in hindsight models

In this section, we argue that (1) in the linear covering setting, LIN−COMB and the best-expert in hindsight benchmark are equivalent; and (2) in the non-linear covering one, LIN−COMB is stronger than the best-expert in hindsight benchmark.

First, recall the best expert in hindsight benchmark. There are K experts, and the problem's covering-type constraints are revealed online one by one. At each time $t \geq 1$, we receive a covering constraint $\sum_{i=1}^{n} a_i^t x_i \geq 1$ (where $a_i^t \geq 0$) and each expert k (where $1 \leq k \leq K$) provides a solution $(s_{i,k}^t)_{i=1}^n$. The objective is to minimize a (cost) function $f(x_1, \ldots, x_n)$. The optimal value in the best expert benchmark, up to time T, is

$$\min_k f(s_{1,k}^T, \ldots, s_{n,k}^T)$$

Note that the best experts can be changed over time (i.e., the best expert at time T might be different to the one at time $T-1$). In the following, let T be the last time a constraint is released.

Linear Covering Setting. In this setting, the objective function $f(x_1, \ldots, x_n)$ can be expressed as $\sum_{i=1}^{n} c_i x_i$ where $c_i \geq 0$. Let k^* be the best expert, i.e., $k^* = \arg\min_k \sum_{i=1}^{n} c_i s_{i,k}^T$. The LIN−COMB can achieve this value by setting $w_{k^*}^t = 1$ and $w_k^t = 0$ for all $k \neq k^*$ over all $1 \leq t \leq T$. Inversely, assume that LIN−COMB benchmark provides $w_1^T, \ldots, w_k^T$ at time T. Then, the optimal value of LIN−COMB (at time T) is:

$$\begin{aligned}\sum_{i=1}^{n} c_i x_i &\geq \sum_{i=1}^{n} c_i \left(\sum_{k=1}^{K} w_k^T s_{i,k}^T \right) = \sum_{k=1}^{K} w_k^T \left(\sum_{i=1}^{n} c_i s_{i,k}^T \right) \\ &\geq \left(\sum_{k=1}^{K} w_k^T \right) \cdot \left(\sum_{i=1}^{n} c_i s_{i,k^*}^T \right) = \sum_{i=1}^{n} c_i s_{i,k^*}^T.\end{aligned}$$

Consequently, the two benchmarks are equivalent.

Non-Linear Covering Setting. By the previous argument, LIN−COMB always captures the best expert benchmark by setting $w_{k^*}^t = 1$ and $w_k^t = 0$ for all $k \neq k^*$ over all $1 \leq t \leq T$. We provide an example showing that LIN−COMB is stronger than the best-expert. Consider the makespan minimization problem in which one assigns n unit jobs to n identical machines to minimize the maximum machine load (makespan). This problem can be formulated as a non-linear covering problem:

$$\min \left\| \left(\sum_j x_{1j}, \sum_j x_{2j}, \ldots, \sum_j x_{nj} \right) \right\|_\infty \qquad \text{s.t} \qquad \sum_{i=1}^{n} x_{ij} \geq 1 \ \forall j, \quad x_{ij} \in \{0,1\} \ \forall i,j$$

where x_{ij} is a $0-1$ variable indicating if job j is assigned to machine i for $1 \leq i \leq n, 1 \leq j \leq n$. Consider n experts, and each expert i assigns *all* jobs to machine i for $1 \leq i \leq n$.

The best expert has the maximum machine load of n (the same for every expert). However, the optimal solution in LIN−COMB can choose $w_i = 1/n$, which results in the makespan of 1. The solution corresponds to the assignment of one job to one machine, i.e., job i to machine i for all $1 \leq i \leq n$.

A.2. Assumptions on the feasibility and the monotonicity of experts

We consider two natural assumptions on the suggestions/solutions of experts.

The first one is *feasibility*, i.e., experts' solutions must satisfy all the constraints so far. One can easily check and detect if a solution is infeasible and ignore the corresponding expert if that is the case.

The second one is the *monotonicity* of the expert solution, i.e., $s^t_{i,k} \geq s^{t-1}_{i,k}$ for all i, k, t. We argue that any well-defined benchmark with the best expert in hindsight must have this property.

Consider a linear covering problem $Nx_1 + x_2$ where N is large and constraint at time 1 is $x_1 + x_2 \geq 1$ and constraint at time 2 is $2x_1 + x_2 \geq 1$. There are two experts. At time 1, experts 1 and 2 provide solution $(1, 0)$ and (∞, ∞) respectively. The best expert up to time 1 is expert 1 with an objective cost of N. At time 2, expert 1 always provides $(1, 0)$ while expert 2 gives a solution $(0, 1)$ by *decreasing* its previous solution. Now, the best expert up to time 2 is expert 2 with an objective cost of 1. One can generalize this toy example to create an unstable objective cost (very large, then very small, and so on) of the best expert benchmark.

In the above example, no algorithm based on expert solutions can be competitive since an adversary can create a dummy expert (providing a trivial large solution) all the time but the last step and in the last step, the dummy expert, by decreasing its solution, gives an optimal solution. An online algorithm cannot decrease its solution (as decisions made are irrevocable) and becomes uncompetitive. Hence, all stable and well-defined benchmark that includes the best expert in hindsight as a candidate should have the monotone property.

Besides, one motivation for our work is to combine online algorithms into a new one with a better guarantee. Such online algorithms, represented by experts, maintain monotone solutions. That motivates the monotone property.

A.3. No assumption on the tightness of online expert solutions

In some settings, one can assume that the provided expert solutions/predictions are tight (i.e., constraints hold with equality) without loss of generality. However, the following example shows that we cannot expect online expert solutions (in the sense of online algorithms, i.e., without modifying the past decisions) to be tight on the arriving constraints and one needs to non-trivially handle that.

In the example, we display the experts' solutions after each constraint.

$$
\begin{array}{lcc}
\min & x_1 + & x_2 \\
 & x_1 + \frac{1}{2} x_2 \geq 1 & \\
\text{Expert}_1: & 1 & 0 \\
\text{Expert}_2: & 0 & 2 \\
 & & x_2 \geq 1 \\
\text{Expert}_1: & 1 & 1 \\
\text{Expert}_2: & 0 & 2
\end{array}
$$

To have tight a suggestion from Expert_2 on the second constraint, Expert_2 not only has to decrease its value of x_2 (which is not allowed), but even increase the value of x_1 for the first constraint. In other words, Expert_2 has to completely modify its past decisions.

Appendix B. Counter example for the performance of the algorithm of Anand et al. (2022)

Anand, Ge, Kumar, and Panigrahi Anand et al. (2022) recently proposed online algorithms for online linear covering problems with multiple expert solutions. We show here a counter-example that contradicts Theorem 2.1 presented in Section 3 of their paper. In the proof of Theorem 2.1, it is stated that *the total cost of the algorithm is at most* 3 *times the potential* ϕ *at the beginning, i.e., at most* $O(\log K)$ *times the* `DYNAMIC` *benchmark*. However, in our counter-example, the total cost of their algorithm is $O(L\log(K))$ times the `DYNAMIC` benchmark, where L is an arbitrarily large number.

Besides, we also attempt to fix the definition of `DYNAMIC` benchmark. If one considers modifications on the order of the quantifiers in the `DYNAMIC`'s definition, to overcome the counter-example, one has to modify `DYNAMIC` benchmark to be the best-expert one.

B.1. Setting

First, we recall the `DYNAMIC` benchmark and the algorithm in Anand et al. (2022). Note that in this section, we consider only linear objectives.

`DYNAMIC` benchmark. Given K experts, denote $x_i(j,s)$ the solution from expert s for variable i on constraint j. It is also assumed that the expert solutions are tight, formally:

$$\sum_{i=1}^{n} a_{ij}\, x_i(j,s) = 1 \quad \forall s \in [K]$$

The `DYNAMIC` benchmark is defined as the minimum cost solution supported by at least one expert at each step. Formally:

$$\texttt{DYNAMIC} = \min_{\hat{\mathbf{x}}\in\hat{X}} \sum_{i=1}^{n} c_i\hat{x}_i, \text{ where}$$

$$\hat{X} = \{\hat{\mathbf{x}} : \forall\, i \in [n],\ \forall\, j \in [m],\ \exists\, s \in [K] \text{ where the solution } x_i(j,s) \le \hat{x}_i\}$$

Let's consider modified definitions based on the `DYNAMIC` benchmark by reordering the quantifiers in $\hat{X}$.

$$\texttt{m-DYNAMIC}(1) = \min_{\hat{\mathbf{x}}\in\hat{X}(1)} \sum_{i=1}^{n} c_i\hat{x}_i, \text{ where}$$

$$\hat{X}(1) = \{\hat{\mathbf{x}} : \forall\, i \in [n],\ \exists\, s \in [K],\ \forall\, j \in [m] \text{ where the solution } x_i(j,s) \le \hat{x}_i\}$$

$$\texttt{m-DYNAMIC}(2) = \min_{\hat{\mathbf{x}}\in\hat{X}(2)} \sum_{i=1}^{n} c_i\hat{x}_i, \text{ where}$$

$$\hat{X}(2) = \{\hat{\mathbf{x}} :\ \exists\, s \in [K], \forall\, i \in [n],\ \forall\, j \in [m] \text{ where the solution } x_i(j,s) \le \hat{x}_i\}$$

We notice that the m−DYNAMIC(2) benchmark is equivalent to the best-expert benchmark. Moreover, given the same problem (constraints, objective) and the same experts' predictions, the values DYNAMIC $\leq$ m−DYNAMIC(1) $\leq$ m−DYNAMIC(2). In other words, DYNAMIC is stronger than m−DYNAMIC(1) which is stronger than m−DYNAMIC(2).

Algorithm in Anand et al. (2022). Algorithm 1 (from Anand et al. (2022)) receives solutions from K experts. While a constraint is not satisfied, their algorithm updates each variable with an increasing rate of

$$\frac{dx_i}{dt} = \frac{a_{ij}}{c_i}(x_i + \delta_{ij})$$

where $\delta_{ij} = \frac{1}{K}\sum_{s=1}^{K} x_i(j,s)$ is the average of the experts' solutions for x_i at the arrival of constraint j. Algorithm 1 of Anand et al. (2022) scales down the problem with 0.5, so it does not increase any variable above 0.5 and satisfies each constraint with value 0.5. The exact solution is obtained by doubling the variables at the end of the execution. (This descaling is an important aspect in the authors' proof.)

B.2. Counter-example

In the following example, we reveal in an online manner a linear program parametrized by L with K experts and observe the behavior of Algorithm 1 (from Anand et al. (2022)). This example is an extension of the pathological input for the multiplicative weight update algorithm.

Objective. The example has $(L \cdot K + 1)$ variables with uniform cost:

$$\min\ x_1 + x_2 + \cdots + x_K + \cdots + x_{2K} + \cdots + x_{LK} + x_{LK+1}$$

Constraints. There are L batches of $(K-1)$ constraints. The first constraint of each batch has $(K+1)$ variables. The last variable (x_{LK+1}) is present in every constraint in every batch, but none of the experts suggests to use this variable. Within a batch, each consecutive constraint has one less variable. The experts set each variable that appears in

later batches to 0. The first batch:

$$
\begin{array}{rcccccc}
 & x_1 + x_2 + \cdots + x_{(K-1)} + x_K + x_{LK+1} \geq 1 & & & & & \\
\text{Expert}_1: & 1 & 0 & \ldots & 0 & 0 & 0 \\
\text{Expert}_2: & 0 & 1 & \ldots & 0 & 0 & 0 \\
 & \vdots & & & & & \\
\text{Expert}_{K-1}: & 0 & 0 & \ldots & 1 & 0 & 0 \\
\text{Expert}_K: & 0 & 0 & \ldots & 0 & 1 & 0 \\
 & x_2 + \cdots + x_{(K-1)} + x_K + x_{LK+1} \geq 1 & & & & & \\
\text{Expert}_1: & 1 & 1 & \ldots & 0 & 0 & 0 \\
\text{Expert}_2: & 0 & 1 & \ldots & 0 & 0 & 0 \\
 & \vdots & & & & & \\
\text{Expert}_{K-1}: & 0 & 0 & \ldots & 1 & 0 & 0 \\
\text{Expert}_K: & 0 & 0 & \ldots & 0 & 1 & 0 \\
 & \vdots & & & & & \\
 & x_{(K-1)} + x_K + x_{LK+1} \geq 1 & & & & & \\
\text{Expert}_1: & 1 & 1 & \ldots & 1 & 0 & 0 \\
\text{Expert}_2: & 0 & 1 & \ldots & 1 & 0 & 0 \\
 & \vdots & & & & & \\
\text{Expert}_{K-1}: & 0 & 0 & \ldots & 1 & 0 & 0 \\
\text{Expert}_K: & 0 & 0 & \ldots & 0 & 1 & 0
\end{array}
$$

During the first constraint of every batch, the experts' solutions form an identity matrix. With each disappearing variable in the consecutive constraints, experts who suggested to use variables which are no longer available, choose to set the variable with the smallest index. Consequently, $(K-1)$ experts suggest to use variable $x_{(K-1)}$ and one expert suggests to use x_K during the last constraint in the first batch. The pattern of the experts' solutions are identical for each batch. The constraints of the l^{th} batch ($1 \leq l \leq L$) are:

$$
\begin{aligned}
x_{(l-1)K+1} + x_{(l-1)K+2} + \cdots + x_{(l-1)K+(K-1)} + x_{lK} + x_{LK+1} &\geq 1 \\
x_{(l-1)K+2} + \cdots + x_{(l-1)K+(K-1)} + x_{lK} + x_{LK+1} &\geq 1 \\
&\vdots \\
x_{(l-1)K+(K-1)} + x_{lK} + x_{LK+1} &\geq 1
\end{aligned}
$$

Lemma 9 *In the example above, the objectives of the* `DYNAMIC` *and the* `m-DYNAMIC(1)` *benchmarks are both at most 1.*

Proof Consider solution $\mathbf{x}^*$ in which $x^*_{LK+1} = 1$ and $x^*_i = 0$ for $i \neq LK+1$. We verify that $\mathbf{x}^* \in \hat{X}$. For each $i \neq lK$ where $1 \leq l \leq L$, and for each constraint j, there exists an expert, indeed expert K, such that $x^*_i \geq 0 = x_i(j,K)$. For $i = lK$, and for each constraint j, there exists an expert, indeed any expert in $1, 2, \ldots, K-1$, such that $x^*_i \geq 0 = x_i(j,1) = x_i(j,2) = \ldots = x_i(j,K-1)$. Moreover, $\mathbf{x}^*$ satisfies all constraints (since variable x_{LK+1} appears in all constraints). Hence, $\mathbf{x}^* \in \hat{X}$. Subsequently, the objective value of the `DYNAMIC` benchmark is at most 1.

Similarly, we verify that $\mathbf{x}^* \in \hat{X}(1)$. For each $i \neq lK$ where $1 \leq l \leq L$, there exists an expert, indeed expert K, and for each constraint j, $x^*_i \geq 0 = x_i(j,K)$. Besides, for $i = lK$, there exists an expert, indeed any expert in $1, 2, \ldots, K-1$, and for each constraint j, $x^*_i \geq 0 = x_i(j,1) = x_i(j,2) = \ldots = x_i(j,K-1)$.Hence, $\mathbf{x}^* \in \hat{X}(1)$. Therefore, the objective value of the `m-DYNAMIC(1)` benchmark is at most 1. ■

Lemma 10 *The objective value of Algorithm* 1 *(from Anand et al. (2022)) on the example above is* $O(L\log(K))$.

Proof By the design of Algorithm 1, the increasing rate of x_{LK+1} is zero throughout the execution, and the variables which are not part of the current constraint are not increased. During the first constraint of each batch, the increasing rate of the first K variables in the batch is $1/K$, since the increasing rate of variable x_i is $(x_i + \frac{1}{K}\sum_{s=1}^{K} x_i(1,s))$ and initially every variable is set to zero. At the second constraint, the increasing rate of the second variable in the batch is higher than the other variables' increasing rate, because the first expert also uses this variable in its solution. Therefore, the increasing rate of the second variable is $(x_{(l-1)K+2} + 2/K)$, while the other remaining expert variables in the constraint have an increasing rate of $(x_i + 1/K)$. Following the same reasoning (apart from the first constraint in the batch), the variable with the smallest index in the constraint has a higher increasing rate, than the other variables. During the last constraint of each batch, the increasing rate of the last two remaining expert variables are $(x_{(l-1)K+(K-1)} + (K-1)/K)$ and $(x_{lK} + 1/K)$. Keeping the increasing rates and the constraint satisfaction in mind, we

can lower bound the value of each variable:

$$\begin{aligned}
\frac{1}{K} &\leq x_{(l-1)K+1} \\
\frac{1}{K-1} &\leq x_{(l-1)K+2} \\
\frac{1}{K-2} &\leq x_{(l-1)K+3} \\
&\vdots \\
\frac{1}{3} &\leq x_{(l-1)K+(K-2)} \\
\frac{1}{2} &\leq x_{(l-1)K+(K-1)} \\
\frac{1}{K} &\leq x_{lK}
\end{aligned}$$

Summing the terms together, we get that the objective value increases at least with $O(\log K)$ during each batch. There are L batches, so the total cost of Algorithm 1 is at least $O(L \log(K))$. ■

The above claims lead to the following one.

Proposition 11 *The competitive ratio of Algorithm* 1 *(from Anand et al. (2022)) on the example above in both* `DYNAMIC` *and* `m-DYNAMIC(1)` *benchmarks is at least* $O(L \log(K))$ *for arbitrarily large* L*.*

B.3. Comparison

In this specific counter-example, the `LIN-COMB` and `m-DYNAMIC(2)` benchmarks are both equivalent to the static best-expert benchmark, i.e., the solution of Expert$_K$. The objective value of the best expert benchmark is L (since the optimal solution sets x_{lK} variables for $1 \leq l \leq L$ to 1 and other variables to 0). In this counter-example, the objective value of our Algorithm 1 is $O(L \log K)$. Consequently, our proposed algorithm is $O(\log K)$ competitive in the `LIN-COMB` (so the best-expert) benchmark.

Appendix C. Online linear covering with experts

Lemma 1 *One can always obtain the solutions $\hat{s}_{i,k}^t$ such that $\hat{s}_{i,k}^t \leq s_{i,k}^t$ and $\sum_{i=1}^n a_i^t \hat{s}_{i,k}^t = 1$.*

Proof Let us fix an expert k. We prove the lemma by induction. At time step $t = 1$, we can always scale down the solution $s_{i,k}^1 \geq 0$ such that the first constraint becomes tight. Assume that the lemma holds until $t-1$, so $\sum_{i=1}^n a_i^{t-1} \hat{s}_{i,k}^{t-1} = 1$ and $\hat{s}_{i,k}^{t-1} \leq s_{i,k}^{t-1}$. Consider time t. If $(s_{i,k}^t)_{i=1}^n$ is tight on the t^{th} constraint then we are done. Otherwise, we have

$$1 < \sum_{i=1}^n a_i^t s_{i,k}^t = \sum_{i \in I} a_i^t s_{i,k}^t + \sum_{i \notin I} a_i^t s_{i,k}^t$$
$$1 = \sum_{i=1}^n a_i^{t-1} \hat{s}_{i,k}^{t-1} = \sum_{i=1}^n a_i^t \left(\hat{s}_{i,k}^{t-1} \cdot \frac{a_i^{t-1}}{a_i^t} \right) \geq \sum_{i \in I} a_i^t \left(\hat{s}_{i,k}^{t-1} \cdot \frac{a_i^{t-1}}{a_i^t} \right) + \sum_{i \notin I} a_i^t s_{i,k}^t$$

Hence, there exists $\hat{s}_{i,k}^t \in \left[\hat{s}_{i,k}^{t-1} \cdot \frac{a_i^{t-1}}{a_i^t}, s_{i,k}^t \right]$ for every i, where $1 \leq i \leq n$, such that $\sum_{i=1}^n a_i^t \hat{s}_{i,k}^t = 1$. ■

Lemma 2 *The x_i^t solutions set by Algorithm 1 for the original covering problem and the dual variables (α^t, β_i^t) of the* `LIN-COMB` *benchmark's linear program relaxation are feasible.*

Proof We first prove that the x_i^t variables satisfy the covering constraints by induction. At time 0, no constraint has been released yet, and every variable is set to 0. This all-zero solution is feasible. Let us assume that the algorithm provides feasible solutions up to time $t-1$. At time t, the algorithm maintains the inequality $x_i^t \geq x_i^{t-1}$, so all constraints t' where $t' < t$ are satisfied. Besides, x_i^t is always at least $\sum_k w_{i,k}^t s_{i,k}^t$, which is larger than $\sum_k w_{i,k}^t \hat{s}_{i,k}^t$ since $s_{i,k}^t \geq \hat{s}_{i,k}^t$ for all i, k due to the preprocessing step. Hence, the constraint t is also satisfied. Formally,

$$\sum_{i=1}^n a_i^t x_i^t \geq \sum_{i=1}^n a_i^t \left(\sum_{k=1}^K w_{i,k}^t \hat{s}_{i,k}^t \right) \geq 1.$$

In the remaining part of the proof, we show the feasibility of α^t and every β_i^t. Since $\gamma^t \geq 0$ and $\lambda_i \geq 0$ for all i and t, we get that $\alpha^t \geq 0$. When we set β_i^t, the nominator of the logarithm term is always larger than the denominator, and it is smaller than $(K\rho)$-times the denominator. Consequently, $0 \leq \beta_i^t \leq c_i$. Furthermore,

$$\beta_i^{t+1} - \beta_i^t = -\frac{1}{\ln(K\rho)} c_i \ln \left(\frac{\sum_{k=1}^K s_{i,k}^t w_{i,k}^t + \delta_i^t}{\sum_{k=1}^K s_{i,k}^{t-1} w_{i,k}^{t-1} + \delta_i^{t-1}} \right).$$

Since $\sum_i a_i^t \hat{s}_{i,k}^t = 1$, using the KKT conditions, we get:

$$\begin{aligned}
&\alpha^t + \sum_{i=1}^{n} s_{ik}^t \left(\beta_i^{t+1} - \beta_i^t\right) \\
&\quad = \frac{1}{\ln(K\rho)} \left(\gamma^t + \sum_{i=1}^{n} \lambda_i\right) - \frac{1}{\ln(K\rho)} \sum_{i=1}^{n} s_{i,k}^t c_i \ln\left(\frac{\sum_{k=1}^{K} s_{i,k}^t w_{i,k}^t + \delta_i^t}{\sum_{k=1}^{K} s_{i,k}^{t-1} w_{i,k}^{t-1} + \delta_i^{t-1}}\right) \\
&\quad = \frac{1}{\ln(K\rho)} \left[\gamma^t + \sum_{i=1}^{n} \lambda_i - \sum_{i=1}^{n} \left(a_i^t \hat{s}_{i,k}^t \gamma^t + \lambda_i + s_{i,k}^t \mu_i^t\right)\right] \leq 0
\end{aligned}$$

■

Appendix D. Online non-linear covering with experts

Complete analysis

As w^t is the locally optimal solution of the non-linear program and $(\theta^t,\ \chi_i,\ \nu_i^t)$ is the locally optimal solution of its dual, the following Karush-Kuhn-Tucker (KKT) and complementary slackness conditions hold:

$$\left[\sum_{i=1}^{n} a_i^t \left(\sum_{k=1}^{K} \hat{s}_{i,k}^t w_{i,k}^t\right) - 1\right]\theta^t = 0 \qquad \forall\, t$$

$$\left[\sum_{k=1}^{K} w_{i,k}^t - 1\right]\chi_i = 0 \qquad \forall\, i,t$$

$$\left[\sum_{k=1}^{K} s_{i,k}^t w_{i,k}^t\right]\nu_i^t = 0 \qquad \forall\, i,t$$

$$s_{i,k}^t \left[\nabla_i f(x^{t-1}) + L\left(\sum_{k=1}^{K} s_{i,k}^t w_{i,k}^t + \delta_i^t - \sum_{k=1}^{K} s_{i,k}^{t-1} w_{i,k}^{t-1} - \delta_i^{t-1}\right)\right] \cdot$$

$$\cdot \ln\left(\frac{\sum_{k=1}^{K} s_{i,k}^t w_{i,k}^t + \delta_i^t}{\sum_{k=1}^{K} s_{i,k}^{t-1} w_{i,k}^{t-1} + \delta_i^{t-1}}\right) - a_i^t \hat{s}_{i,k}^t \theta^t - \chi_i - s_{i,k}^t \nu_i^t = 0 \qquad \forall\, i,k,t$$

$$\theta^t, \chi_i, \nu_i^t \geq 0 \qquad \forall\, i,t$$

Moreover, if $\sum_k w_{i,k}^t s_{i,k}^t > 0$, meaning that $\nu_i^t = 0$, then

$$s_{i,k}^t \left[\nabla_i f(x^{t-1}) + L\left(\sum_{k=1}^{K} s_{i,k}^t w_{i,k}^t + \delta_i^t - \sum_{k=1}^{K} s_{i,k}^{t-1} w_{i,k}^{t-1} - \delta_i^{t-1}\right)\right] \ln\left(\frac{\sum_{k=1}^{K} s_{i,k}^t w_{i,k}^t + \delta_i^t}{\sum_{k=1}^{K} s_{i,k}^{t-1} w_{i,k}^{t-1} + \delta_i^{t-1}}\right)$$
$$- a_i^t \hat{s}_{i,k}^t \theta^t - \chi_i^t = 0 \tag{8}$$

Dual variables and feasibility. We set the dual variables of the reformulation of our `LIN-COMB` benchmark based on the dual variables of the non-linear program used inside the algorithm. The dual formulation is visible on Figure 3. The constants λ and μ correspond to the (λ, μ)-local-smoothness, and L to the L-gradient-Lipschitz of f (see the definitions at Section 3.1).

$$\alpha^t = \frac{1}{\lambda}\left(\theta^t + \sum_{i=1}^{n} \chi_i\right),$$

$$\beta_i^t = \frac{1}{\lambda}\left[\nabla_i f(x^{t-1}) + L\left(\sum_{k=1}^{K} s_{i,k}^t w_{i,k}^t + \delta_i^t - \sum_{k=1}^{K} s_{i,k}^{t-1} w_{i,k}^{t-1} - \delta_i^{t-1}\right)\right] \ln\left(\frac{\sum_{k=1}^{K} s_{i,k}^t w_{i,k}^t + \delta_i^t}{\sum_{k=1}^{K} s_{i,k}^{t-1} w_{i,k}^{t-1} + \delta_i^{t-1}}\right)$$

$$\xi_S^t = \frac{1}{\lambda}\sum_{i\in S} \nabla_i f(x^{t-1}) \ln\left(\frac{(1+1/K)\cdot \max_{t'} \sum_{k=1}^{K} s_{i,k}^{t'}}{\sum_{k=1}^{K} s_{i,k}^{t-1} w_{i,k}^{t-1} + \delta_i^{t-1}}\right) - \frac{\mu}{\lambda}\ln(K\rho) f(x^{t-1})$$

$$\begin{aligned}\gamma^t = &-\frac{1}{\lambda}\sum_{i=1}^{n}\left[\nabla_i f(x^{t-1}) + L\left(\sum_{k=1}^{K} s_{i,k}^t w_{i,k}^t + \delta_i^t - \sum_{k=1}^{K} s_{i,k}^{t-1} w_{i,k}^{t-1} - \delta_i^{t-1}\right)\right] \ln\left(\frac{\sum_{k=1}^{K} s_{i,k}^t w_{i,k}^t + \delta_i^t}{\sum_{k=1}^{K} s_{i,k}^{t-1} w_{i,k}^{t-1} + \delta_i^{t-1}}\right) \\ &- \frac{1}{\lambda}\sum_{i=1}^{n}\left[\nabla_i f(x^t) \ln\left(\frac{(1+1/K)\cdot \max_{t'} \sum_{k=1}^{K} s_{i,k}^{t'}}{\sum_{k=1}^{K} s_{i,k}^{t} w_{i,k}^{t} + \delta_i^{t}}\right) - \nabla_i f(x^{t-1}) \ln\left(\frac{(1+1/K)\cdot \max_{t'} \sum_{k=1}^{K} s_{i,k}^{t'}}{\sum_{k=1}^{K} s_{i,k}^{t-1} w_{i,k}^{t-1} + \delta_i^{t-1}}\right)\right. \\ &+ \frac{\mu}{\lambda}\ln(K\rho) f(x^t) - \frac{\mu}{\lambda}\ln(K\rho) f(x^{t-1})\end{aligned}$$

Lemma 12 *The original covering problem's x_i^t solution set by the algorithm and the dual variables $(\alpha^t, \beta_i^t, \xi_S^t, \gamma^t)$ of the reformulated* `LIN-COMB` *benchmark are feasible up to a factor of* $\ln(K\rho)$.

Proof We first prove that the x_i^t variables satisfy the covering constraints by induction. At time 0, no constraint has been released yet, and every variable is set to 0. This all-zero solution is feasible. Let us assume that the algorithm provides feasible solutions up to time $t-1$. At time t, the algorithm maintains the inequality $x_i^t \geq x_i^{t-1}$, so all constraints t' where $t' < t$ are satisfied. Besides, x_i^t is always at least $\sum_k w_{i,k}^t s_{i,k}^t$, which is larger than $\sum_k w_{i,k}^t \hat{s}_{i,k}^t$ since $s_{i,k}^t \geq \hat{s}_{i,k}^t$ for all i, k due to the preprocessing step. Hence, constraint t is also satisfied, formally,

$$\sum_{i=1}^{n} a_i^t x_i^t \geq \sum_{i=1}^{n} a_i^t \left(\sum_{k=1}^{K} \hat{s}_{i,k}^t w_{i,k}^t\right) \geq 1.$$

In the remaining part of the proof, we show the feasibility of the dual solution. The dual constraints are visible on Figure 3. The first constraint follows from the KKT conditions and from the fact that $\sum_i a_i^t \hat{s}_{i,k}^t = 1$. The second constraint is satisfied up to a factor of $\ln(K\rho)$ because

$$\begin{aligned}\xi_S^t &= \frac{1}{\lambda}\sum_{i\in S} \nabla_i f(x^{t-1}) \ln\left(\frac{(1+1/K)\cdot \max_{t'} \sum_{k=1}^{K} s_{i,k}^{t'}}{\sum_{k=1}^{K} s_{i,k}^{t-1} w_{i,k}^{t-1} + \delta_i^{t-1}}\right) - \frac{\mu}{\lambda}\ln(K\rho) f(x^{t-1}) \\ &\leq \frac{1}{\lambda}\sum_{i\in S} \nabla_i f(x^{t-1}) \ln(K\rho) - \frac{\mu}{\lambda}\ln(K\rho) f(x^{t-1}) \leq \ln(K\rho) c_S\end{aligned}$$

where the last inequality holds due to the (λ, μ)-local-smoothness. Important: after down-scaling the variables by a factor of $\ln(K\rho)$, the second dual constraint holds.

The last constraint is satisfied due to the definition of the dual variables. We need

$$\sum_{i \in S} \beta_i^t + \gamma^t - \xi_S^t + \xi_S^{t+1} \leq 0$$
$$\sum_{i \in S} \beta_i^t - \xi_S^t + \xi_S^{t+1} \leq -\gamma^t$$

to hold. We can rewrite γ^t as follows:

$$\begin{aligned}
\gamma^t = &-\frac{1}{\lambda} \sum_{i=1}^{n} \left[L\left(\sum_{k=1}^{K} s_{i,k}^t w_{i,k}^t + \delta_i^t - \sum_{k=1}^{K} s_{i,k}^{t-1} w_{i,k}^{t-1} - \delta_i^{t-1} \right) \right] \ln \left(\frac{\sum_{k=1}^{K} s_{i,k}^t w_{i,k}^t + \delta_i^t}{\sum_{k=1}^{K} s_{ik}^{t-1} w_{i,k}^{t-1} + \delta_i^{t-1}} \right) \\
&- \frac{1}{\lambda} \sum_{i=1}^{n} \left[\nabla_i f(x^t) - \nabla_i f(x^{t-1}) \right] \ln \left(\frac{(1 + 1/K) \cdot \max_{t'} \sum_{k=1}^{K} s_{i,k}^{t'}}{\sum_{k=1}^{K} s_{ik}^t w_{i,k}^t + \delta_i^t} \right) \\
&+ \frac{\mu}{\lambda} \ln(K\rho) f(x^t) - \frac{\mu}{\lambda} \ln(K\rho) f(x^{t-1})
\end{aligned}$$

Now let us fix an arbitrary set S. Then, following the simplified form of γ^t, we have

$$\begin{aligned}
-\gamma^t \geq &\frac{1}{\lambda} \sum_{i \in S} \left[L\left(\sum_{k=1}^{K} s_{i,k}^t w_{i,k}^t + \delta_i^t - \sum_{k=1}^{K} s_{i,k}^{t-1} w_{i,k}^{t-1} - \delta_i^{t-1} \right) \right] \ln \left(\frac{\sum_{k=1}^{K} s_{i,k}^t w_{i,k}^t + \delta_i^t}{\sum_{k=1}^{K} s_{ik}^{t-1} w_{i,k}^{t-1} + \delta_i^{t-1}} \right) \\
&+ \frac{1}{\lambda} \sum_{i \in S} \left[\nabla_i f(x^t) - \nabla_i f(x^{t-1}) \right] \ln \left(\frac{(1 + 1/K) \cdot \max_{t'} \sum_{k=1}^{K} s_{i,k}^{t'}}{\sum_{k=1}^{K} s_{ik}^t w_{i,k}^t + \delta_i^t} \right) \\
&- \frac{\mu}{\lambda} \ln(K\rho) f(x^t) + \frac{\mu}{\lambda} \ln(K\rho) f(x^{t-1})
\end{aligned}$$

since the terms in the first two lines are positive and we restrict the sum to a subset. In particilar, $\nabla_i f(x^t) \geq \nabla_i f(x^{t-1})$ by assumption of function f, and either $\sum_{k=1}^{K} s_{i,k}^t w_{i,k}^t + \delta_i^t \geq \sum_{k=1}^{K} s_{i,k}^t w_{i,k}^{t-1} + \delta_i^{t-1}$ and the logarithm term is non-negative, or $\sum_{k=1}^{K} s_{i,k}^t w_{i,k}^t + \delta_i^t < \sum_{k=1}^{K} s_{i,k}^t w_{i,k}^{t-1} + \delta_i^{t-1}$ and the logarithm term is also negative, making their product non-

negative. We have to show that the following is true to satisfy the third constraint:

$$
\begin{aligned}
&\sum_{i\in S} \beta_i^t - \xi_S^t + \xi_S^{t+1} \\
&= \sum_{i\in S} \frac{1}{\lambda}\left[\nabla_i f(x^{t-1}) + L\left(\sum_{k=1}^{K} s_{i,k}^t w_{i,k}^t + \delta_i^t - \sum_{k=1}^{K} s_{i,k}^{t-1} w_{i,k}^{t-1} - \delta_i^{t-1}\right)\right] \ln\left(\frac{\sum_{k=1}^{K} s_{i,k}^t w_{i,k}^t + \delta_i^t}{\sum_{k=1}^{K} s_{i,k}^{t-1} w_{i,k}^{t-1} + \delta_i^{t-1}}\right) \\
&\quad - \frac{1}{\lambda}\sum_{i\in S} \nabla_i f(x^{t-1}) \ln\left(\frac{(1+1/K)\cdot \max_{t'} \sum_{k=1}^{K} s_{i,k}^{t'}}{\sum_{k=1}^{K} s_{i,k}^{t-1} w_{i,k}^{t-1} + \delta_i^{t-1}}\right) + \frac{\mu}{\lambda}\ln(K\rho) f(x^{t-1}) \\
&\quad + \frac{1}{\lambda}\sum_{i\in S} \nabla_i f(x^{t}) \ln\left(\frac{(1+1/K)\cdot \max_{t'} \sum_{k=1}^{K} s_{i,k}^{t'}}{\sum_{k=1}^{K} s_{i,k}^{t} w_{i,k}^{t} + \delta_i^{t}}\right) - \frac{\mu}{\lambda}\ln(K\rho) f(x^{t}) \\
&= \frac{1}{\lambda}\sum_{i\in S}\left[L\left(\sum_{k=1}^{K} s_{i,k}^t w_{i,k}^t + \delta_i^t - \sum_{k=1}^{K} s_{i,k}^{t-1} w_{i,k}^{t-1} - \delta_i^{t-1}\right)\right] \ln\left(\frac{\sum_{k=1}^{K} s_{i,k}^t w_{i,k}^t + \delta_i^t}{\sum_{k=1}^{K} s_{ik}^{t-1} w_{i,k}^{t-1} + \delta_i^{t-1}}\right) \\
&\quad + \frac{1}{\lambda}\sum_{i\in S}\left[\nabla_i f(x^t) - \nabla_i f(x^{t-1})\right] \ln\left(\frac{(1+1/K)\cdot \max_{t'} \sum_{k=1}^{K} s_{i,k}^{t'}}{\sum_{k=1}^{K} s_{ik}^{t} w_{i,k}^{t} + \delta_i^{t}}\right) \\
&\quad - \frac{\mu}{\lambda}\ln(K\rho) f(x^t) + \frac{\mu}{\lambda}\ln(K\rho) f(x^{t-1}) \\
&\leq -\gamma^t
\end{aligned}
$$

where the second equality holds due to $\log(a/b) - \log(a/c) = \log(c/b)$.

It remains to show that α^t and β_i^t are positive. Due to the KKT conditions both variables are always positive, so the lemma holds. ■

Theorem 7 *Algorithm 2 is $\frac{O(\ln(K\rho))\cdot\lambda}{1-2\mu\ln(K\rho)}$-competitive within the* `LIN-COMB` *benchmark. Precisely, the algorithm's objective at most $\frac{O(\ln(K\rho))\cdot\lambda}{1-2\mu\ln(K\rho)}$ times that of the optimal* `LIN-COMB` *solution plus a fixed term $\frac{8Ln\sqrt{n}\ln(K\rho)}{1-2\mu\ln(K\rho)}$.*

Proof Lemma 12 proved that our algorithm creates feasible solutions for the original covering problem and for the dual problem of the reformulated `LIN-COMB` benchmark. We show that the algorithm's solution increases the primal objective value of the original covering problem by at most $O(\ln(K\rho))$ times the value of the dual solution, which serves as the lower bound on the `LIN-COMB` benchmark - the best linear combination of the experts' solutions.

$$f(x^t) - f(x^{t-1}) \le \nabla f(x^{t-1})(x^t - x^{t-1}) + \frac{2L}{2}\|x^t - x^{t-1}\|^2 \tag{9}$$

$$\le \sum_{i:x_i^t > x_i^{t-1}} \left[\nabla_i f(x^{t-1}) + L(x_i^t - x_i^{t-1})\right](x_i^t - x_i^{t-1})$$

$$\le \sum_{i:x_i^t > x_i^{t-1}} \left[\nabla_i f(x^{t-1}) + L(x_i^t - x_i^{t-1})\right](x_i^t + \delta_i^t) \ln\left(\frac{x_i^t + \delta_i^t}{x_i^{t-1} + \delta_i^t}\right) \tag{10}$$

$$\le \sum_{i:x_i^t > x_i^{t-1}} \left[\nabla_i f(x^{t-1}) + L(x_i^t - x_i^{t-1})\right](x_i^t + \delta_i^t) \ln\left(\frac{x_i^t + \delta_i^t}{x_i^{t-1} + \delta_i^{t-1}}\right) \tag{11}$$

$$= \sum_{i:x_i^t > x_i^{t-1}} \left[\nabla_i f(x^{t-1}) + L(x_i^t - x_i^{t-1})\right] \left[\left(\sum_{k=1}^K s_{i,k}^t w_{i,k}^t + \frac{1}{K}\sum_{k=1}^K s_{i,k}^t\right) \ln\left(\frac{\sum_{k=1}^K s_{i,k}^t w_{i,k}^t + \delta_i^t}{x_i^{t-1} + \delta_i^{t-1}}\right)\right] \tag{12}$$

$$\le \sum_{i:x_i^t > x_i^{t-1}} \left[\nabla_i f(x^{t-1}) + L(x_i^t - x_i^{t-1})\right] \left[\left(\sum_{k=1}^K s_{i,k}^t w_{i,k}^t + \frac{1}{K}\sum_{k=1}^K s_{i,k}^t\right) \ln\left(\frac{\sum_{k=1}^K s_{i,k}^t w_{i,k}^t + \delta_i^t}{\sum_{k=1}^K s_{i,k}^{t-1} w_{i,k}^{t-1} + \delta_i^{t-1}}\right)\right] \tag{13}$$

$$= \sum_{i:x_i^t > x_i^{t-1}} \sum_{k=1}^K (w_{i,k}^t + 1/K) s_{i,k}^t \left[\nabla_i f(x^{t-1}) + L(x_i^t - x_i^{t-1})\right] \ln\left(\frac{\sum_{k=1}^K s_{i,k}^t w_{i,k}^t + \delta_i^t}{\sum_{k=1}^K s_{i,k}^{t-1} w_{i,k}^{t-1} + \delta_i^{t-1}}\right)$$

$$\le \sum_{i:x_i^t > x_i^{t-1}} \sum_{k=1}^K (w_{i,k}^t + 1/K) s_{i,k}^t \left[\nabla_i f(x^{t-1}) + L\left(\sum_{k=1}^K s_{i,k}^t w_{i,k}^t + \delta_i^t - \sum_{k=1}^K s_{i,k}^{t-1} w_{i,k}^{t-1} - \delta_i^{t-1}\right)\right] \cdot$$

$$\ln\left(\frac{\sum_{k=1}^K s_{i,k}^t w_{i,k}^t + \delta_i^t}{\sum_{k=1}^K s_{i,k}^{t-1} w_{i,k}^{t-1} + \delta_i^{t-1}}\right) \tag{14}$$

$$= \sum_{i:x_i^t > x_i^{t-1}} \sum_{k=1}^K (w_{i,k}^t + 1/K)\left(a_i^t \hat{s}_{i,k}^t \theta^t + \chi_i^t\right) \tag{15}$$

$$\le \sum_{i=1}^n \sum_{k=1}^K (w_{i,k}^t + 1/K)\left(a_i^t \hat{s}_{i,k}^t \theta^t + \chi_i^t\right)$$

$$= \sum_{i=1}^n a_i^t \left(\sum_{k=1}^K w_{i,k}^t \hat{s}_{i,k}^t\right)\theta^t + \sum_{i=1}^n \left(\sum_{k=1}^K w_{i,k}^t\right)\chi_i^t + \frac{1}{K}\sum_{k=1}^K \left(\sum_{i=1}^n a_i^t \hat{s}_{i,k}^t\right)\theta^t + \frac{1}{K}\sum_{k=1}^K \sum_{i=1}^n \chi_i^t$$

$$= 2\theta^t + 2\sum_{i=1}^n \chi_i^t \tag{16}$$

$$= 2\lambda \cdot \alpha^t$$

The above corresponding transformations hold since:

(9) holds as f is $2L$-smooth;

(10) follows from the inequality $a - b \leq a \ln(a/b)$ for all $0 < b \leq a$;

(11) holds since $\delta_i^t \geq \delta_i^{t-1}$ (because $s_{i,k}^t \geq s_{i,k}^{t-1}$ for all i, k, t);

(12) is valid because $x_i^t > x_i^{t-1}$, so $x_i^t = \sum_{k=1}^K s_{i,k}^t w_{i,k}^t$;

(13) is by the design of the algorithm: $x_i^{t-1} \geq \sum_{k=1}^K s_{i,k}^{t-1} w_{i,k}^{t-1}$;

(14) since again, $x_i^t > x_i^{t-1} \geq 0$, we have $x_i^t = \sum_{k=1}^K s_{i,k}^t w_{i,k}^t$ and it always holds that $x_i^{t-1} \geq \sum_{k=1}^K s_{i,k}^{t-1} w_{i,k}^{t-1}$;

(15) since given that $x_i^t > x_i^{t-1} \geq 0$ (so $\sum_{k=1}^K s_{i,k}^t w_{i,k}^t = x_i^t > 0$), the KKT condition (8) applies;

(16) is true due to the complementary slackness conditions and that $\sum_{i=1}^n a_i^t \hat{s}_{i,k}^t = 1$.

The objective of the dual includes $\sum_{t=1}^T \gamma^t$ as well, so it remains to bound this sum. Recall the simplified version of γ^t:

$$\begin{aligned}\gamma^t = &-\frac{1}{\lambda}\sum_{i=1}^n \left[L\left(\sum_{k=1}^K s_{i,k}^t w_{i,k}^t + \delta_i^t - \sum_{k=1}^K s_{i,k}^{t-1} w_{i,k}^{t-1} - \delta_i^{t-1}\right)\right] \ln\left(\frac{\sum_{k=1}^K s_{i,k}^t w_{i,k}^t + \delta_i^t}{\sum_{k=1}^K s_{ik}^{t-1} w_{i,k}^{t-1} + \delta_i^{t-1}}\right) \\ &- \frac{1}{\lambda}\sum_{i=1}^n \left[\nabla_i f(x^t) - \nabla_i f(x^{t-1})\right] \ln\left(\frac{(1+1/K)\cdot \max_{t'} \sum_{k=1}^K s_{i,k}^{t'}}{\sum_{k=1}^K s_{ik}^t w_{i,k}^t + \delta_i^t}\right) \\ &+ \frac{\mu}{\lambda}\ln(K\rho) f(x^t) - \frac{\mu}{\lambda}\ln(K\rho) f(x^{t-1})\end{aligned}$$

Since f is $2L$-smooth with respect to the ℓ_2-norm, we can observe that

$$\sum_{i=1}^n (\nabla_i f(x^t) - \nabla_i f(x^{t-1}) \ln\left(\frac{(1+1/K)\cdot \max_{t'} \sum_{k=1}^K s_{i,k}^{t'}}{\sum_{k=1}^K s_{i,k}^t w_{i,k}^t + \delta_i^t}\right) \leq \ln(K\rho) 2L\sqrt{n} \sum_{i=1}^n (x_i^t - x_i^{t-1})$$

and after bounding the following term

$$2\left(\sum_{k=1}^K s_{i,k}^t w_{i,k}^t - x_i^{t-1}\right) \ln\left(\frac{\sum_{k=1}^K s_{i,k}^t w_{i,k}^t + \delta_i^t}{\sum_{k=1}^K s_{i,k}^{t-1} w_{i,k}^{t-1} + \delta_i^{t-1}}\right) \leq \ln(K\rho)(x_i^t - x_i^{t-1})$$

we get that

$$\begin{aligned}\sum_{t=1}^T \gamma^t &\geq -\frac{\mu \ln(K\rho)}{\lambda} f(x^T) - \ln(K\rho)\frac{4L}{\lambda}\sqrt{n}\sum_{t=1}^T \sum_{i=1}^n (x_i^t - x_i^{t-1}) \\ &\geq -\frac{\mu \ln(K\rho)}{\lambda} f(x^T) - \ln(K\rho)\frac{4L}{\lambda}\sqrt{n}\sum_{i=1}^n x_i^T.\end{aligned}$$

Considering both parts of the proof so far, we bound the dual objective as follows:

$$\begin{aligned}\sum_{t=1}^{T}(\alpha^t + \gamma^t) &\geq \frac{1/2 - \mu\ln(K\rho)}{\lambda} f(x^T) - \ln(K\rho)\frac{4L}{\lambda}\sqrt{n}\sum_{i=1}^{n} x_i^T \\ &\geq \frac{1/2 - \mu\ln(K\rho)}{\lambda} f(x^T) - \ln(K\rho)\frac{4L}{\lambda} n\sqrt{n}\end{aligned}$$

where $x_i^T \leq 1$ since it is a linear combination of expert 0-1 solutions. Rearranging the terms, the theorem follows. ■

Appendix E. Experiments

In this section we include the experiments of the first algorithm for problems with a linear objective function.

Implementation. The first step of our first proposed algorithm is to solve an internal convex program. In the experiments, we approximate the optimal solution of this program using a vanilla Frank-Wolfe implementation. The linear minimization step within Frank-Wolfe is solved with the Gurobi optimizer.

Comparison. The best standard online algorithm for general covering problems without experts is the online multiplicative weight update (MWU) algorithm. In the experiments we compare our algorithm with the MWU algorithm. When a new constraint arrives in the online problem, the MWU algorithm increases each variable x_i in the constraint with a rate of $\frac{a_i^t}{c_i}(x_i + 1/n)$, where n is the total number of variables. We also compare our results with the optimum offline solution (that knows the whole instance in advance) and the average solution of the experts.

Input. First, we evaluated the result of our algorithm on the pathological input of the MWU algorithm. This instance includes n variables and n constraints with uniform costs and coefficients. Each arriving constraint in this pathological example includes one less variable. While the optimal solution is 1, the worst-case guarantee of MWU is $O(\log n)$. For our algorithm we provided n experts, where $(n-1)$ experts suggest an adversarial trivial solution to set all variables to 1, while 1 expert suggests the optimal offline solution. The result of this experiment is visible on Figure 5. An important highlight: our algorithm managed to identify the good expert among the majority of adversaries, obtaining a better objective value, than MWU. Then, we generated some instances to observe the performance of our algorithm on non-specific inputs. The specification for the instance generation includes several parameters, which we detail on Figure 6. The non-specific instances try to represent instances with different 'shape', meaning that the ratio of the number of constraints and variables varies.

To further investigate the impact of the number of variables and the number of experts on our algorithm, we executed the worst case example for the MWU with several input sizes. The result is visible on Figure 7.

Result. Multiplicative weight update is a simple and well-performing algorithm in practice. On its pathological worst-case example, our algorithm performs better, however on most instances the expert suggestions were significantly worse than MWU, which impacted the performance of our algorithm. To some extent, our algorithm can detect the good experts and is robust against even many adversaries. We think that given a real-life problem, it is possible to construct well-performing experts (for example using past data) and with a fine-tuned convex program solver our algorithm can be of interest for various use-cases.

Algo name	Worst-case for MWU	Inst. 1	Inst. 2	Inst. 3	Inst. 4
OPT Offline	1.0	1.3	1.5	10.6	31.3
MWU Online	2.9	2.0	1.7	28.1	63.7
Our Algo	**2.2**	**2.5**	**2.7**	**17.9**	**59.4**
Avg of experts	9.1	15.2	14.3	1884.6	43.4

Figure 5: Objective value of the experiment instances

Input Generation Parameters	Instance 1	Instance 2	Instance 3	Instance 4
Number of variables	10	10	44	30
Number of constraints	10	25	2	15
Min objective coefficient	1	10	1	1
Max objective coefficient	10	25	100	100
Min constraint coefficient	1	10	1	1
Max constraint coefficient	10	25	1	1
Min number of zero coefficient	0	1	11	5
Max number of zero coefficient	5	5	22	20
Number of perfect experts	1	0	0	2
Number of online experts	2	1	1	2
Number of random experts	1	1	11	0
Number of adversaries	1	1	0	0

Figure 6: Parameters of the generated experiment instances

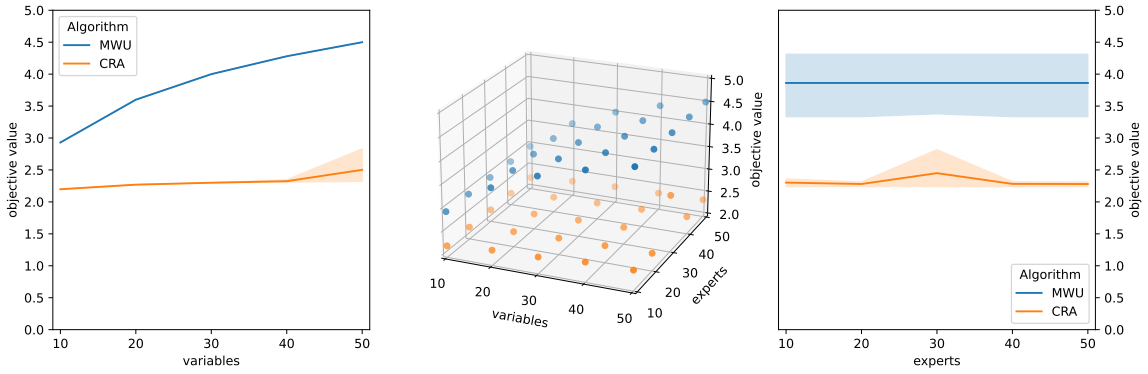


Figure 7: Experiment with varying number of variables and experts on the MWU worst-case instance. The left and right plots show the result visible on the middle 3D plot with two dimensions. The shaded areas correspond to the 95% confidence intervals.