The K-Server Problem via a Modern Optimization Lens

Dimitris Bertsimas, Patrick Jaillet, Nikita Korolko

Massachusetts Institute of Technology, Operations Research Center, Cambridge, MA, 02139

dbertsim@mit.edu*, jaillet@mit.edu, korolko@mit.edu

Abstract

We consider the well-known K-server problem from the perspective of mixed integer, robust and adaptive optimization. We propose a new tractable mixed integer linear formulation of the K-server problem that incorporates both information from the past and uncertainty about the future. By combining ideas from classical online algorithms developed in the computer science literature and robust and adaptive optimization developed in the operations research literature we propose a new method that (a) is computationally tractable, (b) almost always outperforms all other methods in numerical experiments, and (c) is stable with respect to potential errors in the assumptions about the future.

Keywords: Scheduling, adaptive robust optimization, Work Function Algorithm

1. Introduction

The K-server problem, introduced by Manasse et al. (1990), is one of the most fundamental problems considered from the perspective of online algorithms and competitive analysis (Borodin & El-Yaniv, 2005). Given a metric space S, K mobile servers are initially located at some predefined points in that space. Over time, service requests appear successively at different locations in the space. Upon knowing the location of the request, one has to immediately dispatch one of K servers to this location. The cost associated with the assignment is the distance between the location of the server and the location of the request. The objective is to minimize the total cost of serving requests, that is the sum of distances traveled by all servers over a given time horizon.

The typical setting for the K-server problem requires that assignments be made in online fashion, that is, by considering only the current and the past requests. The offline version of the K-server is to find an optimal strategy of serving a finite sequence of known requests.

In the online context, one does not have information about precise locations of future requests, this is why online algorithms may incur significantly worse costs than an optimal offline method. One of the common ways to assess the quality of an online algorithm is via its *competitive ratio*, that is the worst-case ratio between the performance of the online algorithm and the optimal offline clairvoyant over all instances of the problem (Sleator & Tarjan, 1985; Borodin & El-Yaniv, 2005).

*Corresponding author

The K-server problem together with its variants have many practical applications. First of all, the K-server problem is a generalization of important caching/paging problems that arise in disciplines such as computer science (Sleator & Tarjan, 1985), statistics (Raghavan, 1992), mathematics (Du & Hwang, 1993) and many others. The so-called Cable News Network (CNN) problem where servers move along the lines (Koutsoupias & Taylor, 2000), as well as its simpler versions, such as the bridge and the cow path problem, are also examples that possess the structure of the K-server problem. Additionally, this problem has applications in graph theory, certain scheduling problems (Burley, 1996) and in the theory of metrical task systems (Borodin et al., 1992).

Many online algorithms for solving the K-server problem have been suggested in the literature (Rudec et al., 2013; Floratos & Boppana, 1997). They differ in their competitive ratios, tractability and amount of historical data they use in order to make the online assignment decisions. In general, the more information is used, the better the performance of the online method is. The Work Function Algorithm (WFA) (Borodin & El-Yaniv, 2005) is one of the most important online algorithms for the K-server problem from both theoretical perspective of competitiveness and practical performance (Rudec et al., 2013). It considers the full amount of historical data to make its online decisions, as opposed to other simpler online methods.

In practice, it is often the case that information about past observations can lead to some reasonable, data-driven assumptions about the future. In this paper, the online version of the K-server problem is considered from the perspective of mixed integer optimization (MIO), robust optimization (RO) and adaptive optimization. We combine several optimization techniques and present a new holisitic adaptive optimization method that simultaneously incorporates both information from the past and assumptions about the future, giving a significant edge in the algorithm performance.

More specifically, the major contributions of this paper can be summarized as follows:

1. We propose a new MIO formulation of the offline version of the K-server problem. This formulation is computationally tractable and can model many practical generalizations of the classical problem.
2. We introduce robust and affine adaptive counterparts of the online K-server problem by modeling the uncertainty of the positions of future requests as an uncertainty set in the context of the new MIO approach. To the best of our knowledge, this is the first application of robust optimization to online algorithms.
3. We design a new algorithm called the Holistic Adaptive Robust Optimization (HARO) method by combining ideas of the WFA and affine adaptive robust optimization (AARO) techniques. To the best of our knowledge, this is the first algorithm that combines ideas from the computer science and operations research tradition in the context of online algorithms.
4. We empirically demonstrate that the HARO method is tractable and almost always outperforms all other methods considered including WFA and AARO. HARO is the best method for various settings: finite and continuous metric space S, uniform and non-uniform distribution of the requests and different number of servers and locations. Moreover, HARO is stable with respect to potential errors in the uncertainty set describing the future time stages.

The rest of the paper is organized as follows. In Section 2, we briefly discuss existing online algorithms for solving the K-server problem. We represent the offline variant of the K-server problem as a mixed binary optimization problem. We also give a MIO formulation of the WFA procedure. In Section 3, we extend the offline formulation from Section 2 to the online setting of the K-server problem and design its RO and AARO counterparts. In Section 4, we introduce the new HARO method combining the formulations of WFA and AARO algorithms. In Section

5, we consider generalizations of the HARO approach that can be adapted to different settings. We consider various types of the underlying metric space S (multidimensional and/or finite) with different distance functions, as well as a weighted version of the HARO algorithm. In Section 6, we give examples of possible uncertainty sets and describe a way to measure varying amount of information about future requests. In Section 7, we present our numerical results and empirically prove the computational tractability, efficiency and stability of the proposed HARO method.

2. Problem formulation and existing methods

In this section, we first describe in detail the specific variants of the K-server problem that we consider. We also present a brief overview of the most frequently used online algorithms for solving this class of problems. Finally, we introduce a new MIO formulation of the offline K-server problem and a MIO formulation generating the decisions made by the WFA. For the sake of clarity, the formulations in this section are designed for problems with many simplifying assumptions, most of which will be relaxed in Section 5.

2.1. Basic one-dimensional version of the K-server problem

We consider a one-dimensional continuous compact metric space $S = [0, 1]$ with standard Euclidean metric $d(x, y) = |x - y|$. We assume that there are K mobile servers within S that should respond to the requests that appear sequentially in S. The starting locations of the servers, $\hat{\mathbf{x}}^0 = \{\hat{x}_k^0 \in S \,|\, k = 1, \ldots, K\}$, are assumed to be known and form the initial configuration of the system. In what follows, we put a "hat" symbol on variables with known values to emphasize the difference between input data and uncertain parameters of the problem.

The sequence of request locations $\{\sigma_t \in S \,|\, t \geq 1, t \in \mathbb{Z}\}$ appears in an online fashion. At any time-step τ, the locations of only the first τ requests $\{\hat{\sigma}_1, \ldots, \hat{\sigma}_\tau\}$ are known to the online decision maker. At time-step τ the new revealed request at location $\hat{\sigma}_\tau$ should be immediately served by a server $k \in \{1, \ldots, K\}$, where k is chosen according to a proposed algorithm $\mathbb{A}$, and before the next request is revealed at time-step $\tau + 1$. In this case, the configuration of the system, described by the vector of the locations of all K servers, $\mathbf{x}^\tau$, changes according to the rule

$$\mathbf{x}^{\tau-1} = \{x_1^{\tau-1}, \ldots, x_{k-1}^{\tau-1}, x_k^{\tau-1}, x_{k+1}^{\tau-1}, \ldots, x_K^{\tau-1}\} \mapsto \mathbf{x}^\tau = \{x_1^{\tau-1}, \ldots, x_{k-1}^{\tau-1},\, \hat{\sigma}_\tau,\, x_{k+1}^{\tau-1}, \ldots, x_K^{\tau-1}\}.$$

When the server k is assigned to the new request at location $\hat{\sigma}_\tau$ it incurs the cost $d(x_k^{\tau-1}, \hat{\sigma}_\tau) \geq 0$, where $x_k^{\tau-1} \in S$ denotes the position of server k at the end of time-step $\tau - 1$. The objective of an online algorithm $\mathbb{A}$ is to minimize the overall total distance traveled by all K servers. The basic version of the K-server problem assumes that a request cannot be canceled or postponed, does not incur service time while being served, and that the time it takes for a server to move between locations is negligible. As a result, when a new request $\hat{\sigma}_\tau$ appears, all K servers are available to be assigned.

2.2. Offline and online algorithms

In this subsection, we briefly discuss various well-known algorithms (both offline and online) for solving the K-server problem (Rudec et al., 2013; Floratos & Boppana, 1997).

First of all, if the sequence of requests $\boldsymbol{\sigma}$ is finite of known length N, and all components of the vector $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_N)$ are known ahead of time, then the corresponding offline linear optimization problem can be modeled as a standard network flow optimization problem (Chrobak et al., 1991;

Bazaraa et al., 2011) that can be solved efficiently. The objective value of such an offline optimal solution represents a benchmark for evaluating the performance of online algorithms.

One way of measuring the quality of an online algorithm $\mathbb{A}$ is its competitive ratio as introduced in Sleator & Tarjan (1985). Let us assume that $C_{OPT}(\hat{\mathbf{x}}^0, \boldsymbol{\sigma})$ and $C_{\mathbb{A}}(\hat{\mathbf{x}}^0, \boldsymbol{\sigma})$ are the total costs incurred by an offline algorithm OPT and an online algorithm $\mathbb{A}$, respectively, for the K-server problem with initial locations of the servers $\hat{\mathbf{x}}^0$ and a sequence of requests $\boldsymbol{\sigma}$. If there are constants $c_1 > 0$ and c_2 such that for any input data of the problem $\hat{\mathbf{x}}^0$ and $\boldsymbol{\sigma}$ the following inequality holds

$$C_{\mathbb{A}}(\hat{\mathbf{x}}^0, \boldsymbol{\sigma}) \leq c_1 \cdot C_{OPT}(\hat{\mathbf{x}}^0, \boldsymbol{\sigma}) + c_2, \tag{1}$$

then the algorithm $\mathbb{A}$ is said to be c_1-competitive. In that case, the competitive ratio of $\mathbb{A}$ is defined as the infimum of all c_1 such that $\mathbb{A}$ is c_1-competitive. When no such finite constants exist, then the online algorithm $\mathbb{A}$ is sometimes called non-competitive.

Probably the most straightforward deterministic online algorithm for solving the K-server problem is GREEDY. This method assigns the closest server to the request. Despite the fact that it is not competitive for any fixed value of c_1, the empirical experiments presented in (Rudec et al., 2013) demonstrate that in many different settings GREEDY performs reasonably well. BALANCE is another deterministic non-competitive algorithm that tries to keep the total traveled distance by all servers as equal as possible. It assigns a server whose cumulative distance traveled so far plus the distance to the new request location is smallest (Chrobak et al., 1991).

In contrast to deterministic algorithms (like GREEDY and BALANCE), *randomized* algorithms contain random steps as part of their design. In this case, the notion of a competitive ratio can be generalized from the perspective of expected costs produced by the algorithms (Borodin & El-Yaniv, 2005). The competitiveness of randomized online algorithms depends on the adversary model under consideration (oblivious, adaptive online or adaptive offline). We will assume here an oblivious adversary that knows the probability distributions underlying the proposed randomized algorithm $\mathbb{A}$, but not their realizations. In this case, one may modify the definition (1) as

$$\mathbb{E}\, C_{\mathbb{A}}(\hat{\mathbf{x}}^0, \boldsymbol{\sigma}) \leq c_1 \cdot C_{OPT}(\hat{\mathbf{x}}^0, \boldsymbol{\sigma}) + c_2,$$

where $\mathbb{E}[\cdot]$ is the mathematical expectation operator applied to the random choices made by $\mathbb{A}$.

A natural randomized online algorithm, called RAND, chooses a server for each request randomly with equal probability. However, it ignores historical data and distances from the servers to the current request location. The HARMONIC algorithm, introduced by Raghavan & Snir (1989), randomizes between servers with probabilities inversely proportional to distances between their locations and the position of the request. This method is also memoryless and $O(2^K log(K))$-competitive against an adaptive online adversary (Bartal & Grove, 2000), and no better result is known for HARMONIC, even against the weaker oblivious adversary.

One of the most important online algorithms for the K-server problem is WFA (Rudec et al., 2013; Koutsoupias & Papadimitriou, 1995). At time-step τ this deterministic method selects a server k^* as the solution of the following optimization problem:

$$k^* = \arg \min_{k=1,\ldots,K} \Big\{ C_{OPT}\big(\hat{\mathbf{x}}^0, \hat{\sigma}_1, \ldots, \hat{\sigma}_\tau, \hat{\mathbf{x}}^\tau(k)\big) + d(\hat{x}_k^{\tau-1}, \hat{\sigma}_\tau) \Big\}, \tag{2}$$

where the vector $\hat{\mathbf{x}}^\tau(k) = \big(\hat{x}_1^\tau(k), \ldots, \hat{x}_K^\tau(k)\big)$ is defined as follows:

$$\hat{x}_j^\tau(k) = \begin{cases} \hat{x}_j^{\tau-1}, & \text{if } j \neq k \\ \hat{\sigma}_\tau, & \text{if } j = k. \end{cases}$$

The first term in (2), $C_{OPT}\big(\hat{\mathbf{x}}^0, \hat{\sigma}_1, \ldots, \hat{\sigma}_\tau, \hat{\mathbf{x}}^\tau(k)\big)$, is called the *work function*. It represents the optimal value of the offline problem with the known sequence of requests $\hat{\sigma}_1, \ldots, \hat{\sigma}_\tau$ and fixed initial and final configurations $\hat{\mathbf{x}}^0$ and $\hat{\mathbf{x}}^\tau(k)$, respectively.

Intuitively, WFA tries to find a good balance between a strategic solution (represented by the work function term) and a greedy solution (second term of (2)). Learning from past request locations while determining the next move is useful since we can find the sequence of optimal assignments and system configuration that OPT algorithm would generate after servicing the sequence $(\hat{\sigma}_1, \ldots, \hat{\sigma}_\tau)$. Simultaneous minimization of the work function term and the greedy term in (2) means that we would like to be close to the optimal counterfactual configuration and the server assignment sequence determined by the work function, while not moving very far from the current configuration $\hat{\mathbf{x}}^\tau(k)$ (Koutsoupias, 2009). Taking into account past locations of requests often gives a significant performance advantage to the WFA over other online heuristics that ignore the request history (such as GREEDY) particularly in case of non-uniform distribution of requests and initial locations of servers in the metric space. The reason of this advantage is that WFA gradually learns where requests tend to arrive and what the optimal assignments for these requests are (Baumgartner et al., 2012).

A *truncated* method w-WFA is an important generalization of WFA (Rudec et al., 2013), when only the most recent w observations are included into the optimization of the work function

$$C_{OPT}\big(\hat{\mathbf{x}}^{\tau-w}, \hat{\sigma}_{\tau-w}, \ldots, \hat{\sigma}_\tau, \hat{\mathbf{x}}^\tau(k)\big). \tag{3}$$

For many online problems, WFA gives the best possible competitive ratio or it is conjectured to be best possible (Borodin & El-Yaniv, 2005). For example, for the classical K-server problem it has the best known competitive ratio of $2K-1$ for any metric space, and a competitive ratio of K for some special cases of the problem (Chrobak et al., 1991; Koutsoupias & Papadimitriou, 1995; Koutsoupias, 1999). Moreover, in many practical settings, WFA demonstrates superior empirical perfomance in terms of total cost incurred compared to other online algorithms (Rudec et al., 2013; Bartal & Grove, 2000). The empirical ratio between the cost incurred by WFA and OPT in practice is significantly smaller than the theoretical upper bound of $2K-1$.

The truncated method w-WFA (that plays a fundamental role for construction of HARO algorithm as we will discuss below in Section 4) is known to be non-competitive (Baumgartner et al., 2012). For arbitrarily large value of w and constants c_1 and c_2 it is possible to generate an example of input violating Ineq. (1). However, the non-competitive example designed by Baumgartner et al. (2012) is quite ad-hoc, while for practical applications w-WFA has similar good empirical performance as initial WFA even for relatively small values of w.

Taking into account the theoretical and practical value of WFA, we use this algorithm as a starting point for constructing a more efficient method for solving the K-server problem from a robust and adaptive optimization point of view.

2.3. MIO formulation of the offline K-server problem

Let us formulate the offline version of the K-server described in Section 2.1 as a multiperiod mixed binary linear problem. This model will be easily extended to more realistic settings later by virtue of the strong modeling power of MIO.

Following the basic notation introduced in Section 2.1, we first define the decision variables used in our formulation: $x_k^t \in [0,1]$ represents the position of server $k \in \{1, \ldots, K\}$ at the end of time-step $t \in \{1, \ldots, N\}$; $y_k^t \in \{0,1\}$ is an assignment indicator equal to 1 if server k is assigned to request $\hat{\sigma}_t$. Then, the offline K-server problem can be formulated as the following multiperiod optimization problem:

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{y}} \quad & \sum_{k=1}^{K}\sum_{t=1}^{N} |x_k^t - x_k^{t-1}| && \text{(4a)}\\
\text{s.t.} \quad & x_k^t = x_k^{t-1} + y_k^t(\hat{\sigma}_t - x_k^{t-1}), \quad t=1,\dots,N,\ k=1,\dots,K && \text{(4b)}\\
& x_k^0 = \hat{x}_k^0, \quad k=1,\dots,K && \text{(4c)}\\
& \sum_{k=1}^{K} y_k^t = 1, \quad t=1,\dots,N && \text{(4d)}\\
& 0 \le x_k^t \le 1,\ y_k^t \in \{0,1\}, \quad t=1,\dots,N,\ k=1,\dots,K. && \text{(4e)}
\end{aligned}
$$

The objective function (4a) minimizes the cumulative distance traveled by all servers. Eq. (4b) states that the location of server k at time-step t is either the same as it was at the previous time-step $t-1$, if $y_k^t = 0$ (i.e., if server k is not dispatched to request $\hat{\sigma}_t$), or is equal to $\hat{\sigma}_t$, if $y_k^t = 1$.

Formulation (4) is nonlinear due to products of the form $y_k^t x_k^{t-1}$ and absolute values in the objective (4a). These products can be linearized by introduction of substituting variables z_k^t, while absolute values can be modeled by new variables v_k^t. As a result, the offline K-server problem can be formulated as a mixed binary linear optimization problem:

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{v}} \quad & \sum_{k=1}^{K}\sum_{t=1}^{N} v_k^t \\
\text{s.t.} \quad & \text{For } k=1,\dots,K;\ t=1,\dots,N: \\
& \quad v_k^t \ge x_k^t - x_k^{t-1} \\
& \quad v_k^t \ge -(x_k^t - x_k^{t-1}) \\
& \quad x_k^t = x_k^{t-1} + y_k^t\hat{\sigma}_t - z_k^t \\
& \quad z_k^t \le y_k^t \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (5) \\
& \quad z_k^t \le x_k^{t-1} \\
& \quad z_k^t \ge y_k^t + x_k^{t-1} - 1 \\
& \quad 0 \le x_k^t,\ z_k^t \le 1,\ y_k^t \in \{0,1\} \\
& x_k^0 = \hat{x}_k^0, \quad k=1,\dots,K \\
& \sum_{k=1}^{K} y_k^t = 1, \quad t=1,\dots,N.
\end{aligned}
$$

2.4. MIO formulation related to the WFA

WFA method defined by (2) makes successive assignment decisions for the K-server problem. At each time-step τ (for $1 \le \tau \le N$), one may obtain this assignment decision as part of an optimal solution of a MIO formulation that we present in this section.

According to the definition of WFA for a fixed time-step τ, in order to make a locally optimal assignment decision $y_{k^*}^{\tau}$, we need to know the initial locations of the servers $\{\hat{x}_k^0 \,|\, k=1,\dots,K\}$, the sequence of requests $\{\hat{\sigma}_t \,|\, t = 1, \dots, \tau\}$ and the locations of the servers at the end of time-step $\tau - 1$, that is,

$$\hat{\mathbf{x}}^{\tau-1} = \left(\hat{x}_1^{\tau-1}, \dots, \hat{x}_K^{\tau-1}\right).$$

This vector $\hat{\mathbf{x}}^{\tau-1}$ determines a system configuration in which a counterfactual sequence of re-optimized past server assignments should end up. Similarly to (5), instead of solving K optimization problems of the type (2), we find an optimal assignment k^* by solving the following consolidated optimization problem:

$$\begin{aligned}
\min_{\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{v}} \quad & \sum_{k=1}^{K}\sum_{t=1}^{\tau} v_k^t + \sum_{k=1}^{K} y_k^\tau \cdot d(\hat{x}_k^{\tau-1}, \hat{\sigma}_\tau) \\
\text{s.t.} \quad & \text{For } k = 1, \dots, K: \\
& \quad v_k^t \geq x_k^t - x_k^{t-1}, \quad t = 1, \dots, \tau \\
& \quad v_k^t \geq -(x_k^t - x_k^{t-1}), \quad t = 1, \dots, \tau \\
& \quad x_k^t = x_k^{t-1} + y_k^t \hat{\sigma}_t - z_k^t, \quad t = 1, \dots, \tau - 1 \\
& \quad x_k^\tau = \hat{x}_k^{\tau-1} + y_k^\tau(\hat{\sigma}_\tau - \hat{x}_k^{\tau-1}) \\
& \quad z_k^t \leq y_k^t, \quad t = 1, \dots, \tau - 1 \\
& \quad z_k^t \leq x_k^{t-1}, \quad t = 1, \dots, \tau - 1 \\
& \quad z_k^t \geq y_k^t + x_k^{t-1} - 1, \quad t = 1, \dots, \tau - 1 \\
& \quad x_k^0 = \hat{x}_k^0 \\
& \quad 0 \leq x_k^t,\, z_k^t \leq 1,\; y_k^t \in \{0, 1\}, \quad t = 1, \dots, \tau \\
& \sum_{k=1}^{K} y_k^t = 1, \quad t = 1, \dots, \tau.
\end{aligned} \tag{6}$$

The only difference with the offline formulation (5) is that the locations of the servers at the last time-step τ are constrained by the given configuration $\hat{\mathbf{x}}^{\tau-1}$ and the objective function is augmented with the second "greedy" summand as in definition (2). The equivalence of server assignments produced by solving optimization formulations (2) and (6) is stated in the following proposition.

Proposition 1. For any fixed input parameters $\hat{\mathbf{x}}^0, \hat{\sigma}_1, \dots, \hat{\sigma}_\tau, \hat{\mathbf{x}}^{\tau-1}$, if k^* is the optimal server index defined by Eq. (2) and k^{**} is the server index such that $\mathbf{y}^*_{k^{**}} = 1$ in the optimal solution $\mathbf{y}^*$ of problem (4), then k^* is identical to k^{**}.

Proof. Two indices k^* and k^{**} are equivalent since outer minimization with respect to $k = 1, \dots, K$ in (2) is modeled by binary variables y_k^τ in (6). For any fixed $k = 1, \dots, K$, if variable $y_k^\tau = 1$, then optimization formulation (6) is reduced to a much simpler problem that has identical form with the objective function in (2). Since in (6) we optimize over variables y_k^τ for $k = 1, \dots, K$ and we need to pick exactly one index k such that $y_k^\tau = 1$, then by solving (6) we select the server assignment with the smallest objective value as defined in (2). $\square$

The aggregation idea of K separate optimization problems into one (6) is similar to a network-based formulation of WFA algorithm for K-server problem developed by Rudec et al. (2013). The MIO formulations of more complex problem settings with multidimensional and finite metric spaces are discussed in Section 5.

3. The K-server problem from a RO perspective

In the previous section, both formulations (5) and (6) assume that all request locations are certain. More specifically, vector $(\hat{\sigma}_1, \dots, \hat{\sigma}_N)$ is a part of the input data for the offline formulation (5), and vector $(\hat{\sigma}_1, \dots, \hat{\sigma}_\tau)$ is certain in the WFA formulation (6) at time-step τ.

In our RO-based online algorithm framework described in Section 3.1, at each time-step τ (for $1 \le \tau \le N$) the allocation decisions $\{y_k^\tau\}_{k=1}^K$ are made by solving one instance of a robust MIO formulation under uncertainty (11). More precisely, we build upon the deterministic formulation (5), assuming that at time-step τ requests $\hat{\boldsymbol{\sigma}} = (\hat{\sigma}_1, \dots, \hat{\sigma}_\tau)$ are exactly known, while the future requests $\boldsymbol{\sigma} = (\sigma_{\tau+1}, \dots, \sigma_N)$ are only known up to some uncertainty. We model this uncertainty using a robust optimization approach (Ben-Tal & Nemirovski, 2008; Bertsimas et al., 2011). As opposed to stochastic optimization, it does not require the knowledge of the probability distribution for the parameters $\boldsymbol{\sigma}$, but rather assumes that the uncertain data reside in a so-called uncertainty set U. Probably the simplest (yet powerful) choice of set U is represented by a bounded polyhedral set

$$U = \{\mathbf{A}\boldsymbol{\sigma} \le \mathbf{b}, \boldsymbol{\sigma} \ge \mathbf{0}\}, \tag{7}$$

where $\mathbf{A}$ is a given $m \times (N - \tau)$ matrix and $\mathbf{b}$ is a given $m \times 1$ vector. This set of inequalities (7) describes *hard* constraints, that is, constraint violation is impossible for any realization of parameters $\boldsymbol{\sigma}$ in the uncertainty set. In this paper we concentrate on polyhedral uncertainty sets because of their computational tractability and prominent modeling power. The robust optimization approach developed in Section 3.1 leads to computationally tractable online algorithm for the K-server problem. Moreover, it will allow us to design (in Section 3.2) more advanced *adaptive* robust optimization methods that outperform the basic robust counterpart.

3.1. Multiperiod RO formulation of the K-server problem

In order to extend the deterministic formulation (5) to the online setting with partially unknown values of request locations (namely, $\sigma_{\tau+1}, \dots, \sigma_N$ are uncertain at time-step τ), we need to modify two types of equalities.

First, given that any server assignment $y_k^t = 1$ incurs a non-negative traveling cost, we substitute the requirement that exactly one of K servers should be assigned to each of the requests with the following equivalent inequality

$$\sum_{k=1}^K y_k^t \ge 1, \quad t = \tau, \dots, N.$$

Second, we eliminate equations (4b) (used to keep track of the locations of the servers) as well as variables $\mathbf{x}$ from formulation (5). In order to do it, we notice that the location x_k^t of server k at the end of time-step t (for $\tau \le t \le N$) can be expressed as a function of the initial location $\hat{x}_k^{\tau-1}$ (part of the input data at time-step τ), the sequence of requests $\boldsymbol{\sigma}$ and the assignment decisions $\mathbf{y} = \{y_k^t \,|\, k = 1, \dots, K,\, t = \tau, \dots, N\}$. Namely,

$$x_k^t = u[k, t, \tau - 1]\, \hat{x}_k^{\tau-1} + u[k, t, \tau]\hat{\sigma}_\tau + \sum_{j=\tau+1}^{t} u[k, t, j]\sigma_j, \tag{8}$$

where $\mathbf{u} = \big\{u[k, t, j]\big\}$ are new binary variables uniquely determining locations of the servers defined as

$$u[k, t, j] = y_k^j \cdot \prod_{i=j+1}^{t} (1 - y_k^i), \quad k = 1, \dots, K,\, t = \tau - 1, \dots, N,\, j = \tau - 1, \dots, t. \tag{9}$$

For any time-step t, the server k can be in one of the following $t-\tau+2$ locations: $\hat{x}_k^{\tau-1}$, $\hat{\sigma}_\tau$ or σ_j (for $j=\tau+1,\dots t$). Hence, the binary indicators $\mathbf{u}$ must satisfy the following constraint:

$$\sum_{j=\tau-1}^{t} u[k,t,j] = 1.$$

To see this let us consider all possible options. If the server k was not assigned to any of the requests $\sigma_\tau,\dots,\sigma_t$, then it is at its initial location $\hat{x}_k^{\tau-1}$, corresponding in (8) to the case $u[k,t,\tau-1]=1$. If the last time the server k was assigned to a request was j (for some $j\in\{\tau,\dots,t\}$), and was not assigned to any of the subsequent requests $\sigma_{j+1},\dots,\sigma_t$, then at time-step t it is at the location of request σ_j, corresponding to the case $u[k,t,j]=1$. We add auxiliary variables $y_k^{\tau-1}:=1$ (for $k=1,\dots,K$) to make definition (9) correct when $j=\tau-1$.

We also substitute the change of location of the server k at time-step $t\in\{\tau,\dots,N\}$ (previously expressed as $x_k^t-x_k^{t-1}$ in (5) and (6)) by a new expression d_k^t defined as follows:

$$\begin{aligned} d_k^t(\mathbf{u},\boldsymbol{\sigma}) = \quad & \big(u[k,t,\tau-1]-u[k,t-1,\tau-1]\big)\,\hat{x}_k^{\tau-1} + \big(u[k,t,\tau]-u[k,t-1,\tau]\big)\,\hat{\sigma}_\tau + \\ & + \sum_{j=\tau+1}^{t}\big(u[k,t,j]-u[k,t-1,j]\big)\sigma_j. \end{aligned} \tag{10}$$

Expression (10) depends on allocation decisions $\mathbf{u}$ and uncertain parameters $\boldsymbol{\sigma}$, and is derived from the change of variables (8). Hence, in the RO-based algorithm framework, we solve the following formulation (11) at each time-step τ (for $1\le\tau\le N$):

$$\begin{aligned}
\min_{\mathbf{y},\mathbf{v},\mathbf{u}} \quad & \sum_{k=1}^{K}\sum_{t=\tau}^{N} v_k^t && \text{(11a)}\\
\text{s.t.} \quad & \text{For } k=1,\dots,K;\ t=\tau,\dots,N: \\
& \quad v_k^t \ge d_k^t(\mathbf{u},\boldsymbol{\sigma}), \quad \forall\boldsymbol{\sigma}\in U && \text{(11b)}\\
& \quad v_k^t \ge -d_k^t(\mathbf{u},\boldsymbol{\sigma}), \quad \forall\boldsymbol{\sigma}\in U && \text{(11c)}\\
& \quad u[k,t,j] \le y_k^j, \qquad j=\tau-1,\dots,t && \text{(11d)}\\
& \quad u[k,t,j] \le 1-y_k^i, \qquad j=\tau-1,\dots,t,\ i=j+1,\dots,t && \text{(11e)}\\
& \quad u[k,t,j] \ge y_k^j + \sum_{i=j+1}^{t}(1-y_k^i) - (1+(t-j)-1), \\
& \qquad\qquad\qquad j=\tau-1,\dots,t && \text{(11f)}\\
& \quad \sum_{j=\tau-1}^{t} u[k,t,j] = 1 && \text{(11g)}\\
& \quad 0 \le u[k,t,j] \le 1, \qquad j=\tau-1,\dots,t && \text{(11h)}\\
& \quad u[k,t,j] = 0, \qquad j=t+1,\dots,N && \text{(11i)}\\
& \quad y_k^t \in \{0,1\} \\
& y_k^{\tau-1}=1;\quad u[k,\tau-1,\tau-1]=1, \qquad k=1,\dots,K && \text{(11j)}
\end{aligned}$$

$$\sum_{k=1}^{K} y_k^t \geq 1, \quad t = \tau, \ldots, N. \tag{11k}$$

The objective function (11a) is to minimize the total distance traveled as defined by constraints (11b)-(11c) and expressions (10). Given that each variable $u[k,t,j]$ is a product of binary variables (9), inequalities (11d)-(11f) characterize the linear representation of variables $\mathbf{u}$. Identity (11g) states that server k at time-step t must be in exactly one of possible locations indexed by $j = \tau - 1, \ldots, t$. Constraints (11i), (11j) define initial values of variables, while inequalities (11k) guarantee that we assign at least one server for each request.

The formulation (11) includes the uncertain parameters $\boldsymbol{\sigma}$ only in the first two groups of inequalities describing the feasible set. As such, it is a regular robust mixed integer linear optimization problem and has been shown to be computationally tractable by virtue of standard duality techniques (Bertsimas et al., 2011). We solve the RO problem (11), observe an optimal solution $\mathbf{y}^*$ and assign the server k at time-step τ to the request $\hat{\sigma}_\tau$, if $(y_k^\tau)^* = 1$.

Similarly to the idea of the truncated w-WFA method defined in (3), we introduce a new parameter T for the RO formulation, which controls (and fixes) the number of subsequent time periods taken into consideration. That is, in order to make a decision at time-step τ, we consider the uncertainty set U associated to the next T requests $(\sigma_{\tau+1}, \ldots, \sigma_{\tau+T})$ only, as opposed to all remaining $N - \tau$ requests $(\sigma_{\tau+1}, \ldots, \sigma_N)$. The truncated version of the RO formulation with parameter T (that we denote as RO(T)) has an identical form (11) with the only difference in the limits for time index t. For RO(T) algorithm, we substitute $t = \tau, \ldots, N$ with $t = \tau, \ldots, \min(\tau + T, N)$, where the new upper bound on time index guarantees that t is always less or equal than the problem time horizon N.

3.2. Affinely adaptive robust formulation

In this subsection, we develop an online algorithm for the K-server problem based on an affinely adaptive robust optimization (AARO) approach, which is a natural extension to the robust formulation (11). AARO is a simple, but efficient methodology for solving multiperiod optimization problems that was first suggested by Ben-Tal et al. (2004) and proven to yield less conservative solutions compared to the robust optimization approach in a computationally tractable way.

The motivation for designing an AARO extension to our case is twofold. First, the K-server problem is by definition a multiperiod optimization problem, justifying a natural split of variables into two groups: non-adaptive *here-and-now* decisions that should be implemented before uncertain parameters $\boldsymbol{\sigma}$ are realized, and adaptive *wait-and-see* variables that can be determined after realization of the uncertain locations of future requests. Second, when the uncertainty set U is not very restrictive and does not provide a lot of information about positions of future requests, consideration of the worst-case scenario (which is intrinsic to robust optimization perspective) may lead to risk-averse and therefore potentially suboptimal solutions. The AARO approach addresses both of these issues.

Following the idea behind AARO (Ben-Tal et al., 2004), we increase the adaptability of the model by assuming that variables y_k^t are affine functions of the uncertainty. More precisely, at time-step $\tau \in \{1, \ldots, N\}$ we set

$$y_k^t = \beta[k,t,\tau] + \sum_{p=\tau+1}^{t} \beta[k,t,p]\sigma_p, \quad k = 1, \ldots, K, \, t = \tau, \ldots N, \tag{12}$$

where $\boldsymbol{\beta} = \big\{\beta[k,t,p]\big\}$ become new decision variables replacing variables $\mathbf{y}$. We do keep here-and-now decisions (previously modeled by y_k^τ, $k = 1,\ldots,K$) as binary variables by imposing

$$\beta[k,\tau,\tau] \in \{0,1\}, \quad k = 1,\ldots,K,$$

while wait-and-see decisions (modeled by y_k^t, for $k = 1,\ldots,K, t = \tau+1,\ldots,N$) are *relaxed* to be continuous variables $0 \le y_k^t \le 1$ for $t = \tau+1,\ldots,N$, and interpreted as *probabilities* that at time-step t the server k will be assigned to request σ_t. The auxiliary variables $\mathbf{u}, \mathbf{v}$ from formulation (11) remain unchanged. (It is worth mentioning that variables v_k^t can be also substituted by affine expressions similar to Eq. (12). We tested performance of the extended methods AARO and HARO empirically and observed its marginal improvement. Hence, we do not adopt this extension in the definition of the AARO algorithm.)

In the online AARO algorithm framework, we solve one instance of the problem (13) at each time-step τ (for $1 \le \tau \le N$). It is a standard mixed integer linear robust optimization problem obtained from formulation (11) by substituting variables y_k^t according to formula (12) as follows:

$$\begin{aligned}
\min_{\boldsymbol{\beta},\mathbf{u},\mathbf{v}} \quad & \sum_{k=1}^{K}\sum_{t=\tau}^{N} v_k^t \\
\text{s.t.} \quad & \text{For } k = 1,\ldots,K;\ t = \tau,\ldots,N: \\
& \quad v_k^t \ge d_k^t(\mathbf{u},\boldsymbol{\sigma}), \quad \forall \boldsymbol{\sigma} \in U \\
& \quad v_k^t \ge -d_k^t(\mathbf{u},\boldsymbol{\sigma}), \quad \forall \boldsymbol{\sigma} \in U \\
& \quad u[k,t,j] \le \beta[k,j,\tau] + \sum_{p=\tau+1}^{j} \beta[k,j,p]\sigma_p, \quad j = \tau-1,\ldots,t,\ \forall \boldsymbol{\sigma} \in U \\
& \quad u[k,t,j] \le 1 - \Big(\beta[k,i,\tau] + \sum_{p=\tau+1}^{i} \beta[k,i,p]\sigma_p\Big), \\
& \qquad\qquad j = \tau-1,\ldots,t,\ i = j+1,\ldots,t,\ \forall \boldsymbol{\sigma} \in U \\
& \quad u[k,t,j] \ge \Big(\beta[k,j,\tau] + \sum_{p=\tau+1}^{j} \beta[k,j,p]\sigma_p\Big) + \\
& \qquad + \sum_{i=j+1}^{t} \Big(1 - (\beta[k,i,\tau] + \sum_{p=\tau+1}^{i} \beta[k,i,p]\sigma_p)\Big) - (1 + (t-j) - 1), \\
& \qquad\qquad j = \tau-1,\ldots,t,\ \forall \boldsymbol{\sigma} \in U \\
& \quad 0 \le u[k,t,j] \le 1, \qquad j = \tau-1,\ldots,t \\
& \quad \sum_{j=\tau-1}^{t} u[k,t,j] = 1 \\
& \sum_{k=1}^{K} \Big(\beta[k,t,\tau] + \sum_{p=\tau+1}^{t} \beta[k,t,p]\sigma_p\Big) \ge 1, \quad t = \tau,\ldots,N,\ \forall \boldsymbol{\sigma} \in U \\
& 0 \le \beta[k,t,\tau] + \sum_{p=\tau+1}^{t} \beta[k,t,p]\sigma_p \le 1, \quad t = \tau+1,\ldots,N, k = 1,\ldots,K,\ \forall \boldsymbol{\sigma} \in U
\end{aligned} \tag{13}$$

$$\begin{aligned}&\beta[k,\tau,\tau]\in\{0,1\}, \qquad k=1,\dots,K\\ &u[k,\tau-1,\tau-1]=1; \quad \beta[k,\tau-1,\tau]=1, \qquad k=1,\dots,K.\end{aligned}$$

The resulting optimization problem (13) is a relaxation of the robust counterpart (11), since products of binary variables $\mathbf{y}$ defining $u[k,t,j]$ in (9) are now approximated by the corresponding affine expressions in terms of continuous variables $\boldsymbol{\beta}$, rather than binary variables $\mathbf{y}$ (12). However, since we still have constraints (11g) and (11h) in fromulation (13), we now can treat variables $u[k,t,j]$ as probabilities. The last group of constraints in (13) replaces the convention $y_k^{\tau-1}=1$ used in formulation (11).

The optimization problem (13) yields an optimal solution $\boldsymbol{\beta}^*$ that uniquely identifies the optimal server assignment at the current time-step τ. Namely, if $\beta^*[k,\tau,\tau]=1$ for some $k\in\{1,\dots,K\}$, then the server k is assigned to the request σ_τ. It is worth mentioning that potentially fractional future assignment decisions y_k^t for $t=\tau+1,\dots,N$ (modeled by affine expressions (12)) are not intended to be implemented in practice, since at each time-step τ we update the input data and resolve the optimization formulation (13). Fractional wait-and-see decisions are needed only to model a total future cumulative distance traveled by the servers.

The truncated problem formulation AARO(T) is defined similarly to RO(T) as described in Section 3.1, that is, at time-step τ we restrict the number of time periods in the optimization problem (13) from $t=\tau,\dots,N$ to $t=\tau,\dots,\min(\tau+T,N)$.

Let us denote the feasible set in formulation (13) with up to T future time periods ($t=\tau,\dots,\min(\tau+T,N)$) by $\mathcal{F}(\tau,T,\hat{\mathbf{x}}^{\tau-1},K,N,U)$, where paramaters $\tau,T,\hat{\mathbf{x}}^{\tau-1},K,N,U$ uniquely determine the optimization problem (13). In this case, a short representation of the AARO(T) problem (13) that we will use in subsequent sections is as follows:

$$\begin{aligned}\min_{\boldsymbol{\beta},\mathbf{u},\mathbf{v}} \quad & \sum_{k=1}^{K}\sum_{t=\tau}^{N} v_k^t\\ \text{s.t.} \quad & \boldsymbol{\beta},\mathbf{u},\mathbf{v}\in\mathcal{F}(\tau,T,\hat{\mathbf{x}}^{\tau-1},K,N,U).\end{aligned} \tag{14}$$

4. Holistic adaptive robust optimization algorithm

Both the WFA and AARO approaches for the K-server problem have their own merits. The main idea behind the design of the WFA is to reconsider past decisions in order to help find a high-quality assignment for the current one. On the other hand, the AARO method exploits information about future requests using uncertainty set in order to improve the current decision. The main purpose of this section is to introduce a new *holistic* method for solving multiperiod optimization problems which simultaneously incorporates information from the past and assumptions about the future. This holistic adaptive robust optimization (HARO) approach is a combination of the WFA method (involving past observations) and the AARO method (modeling future uncertainty in adaptive manner by means of robust optimization).

The HARO approach can be seen as a natural extension of WFA in the following way. The basic definition of WFA contains a "greedy" term:

$$d(\hat{x}_k^{\tau-1},\hat{\sigma}_\tau) \text{ in definition (2), or equivalently } \sum_{k=1}^{K} y_k^\tau\cdot d(\hat{x}_k^{\tau-1},\hat{\sigma}_\tau) \text{ in formulation (6),}$$

which contains only information about the location of the current request $\hat{\sigma}_\tau$ and provides the distance that server will travel at the current time-step τ. Within the HARO framework, the uncertainty set U now contains additional information about future requests $\boldsymbol{\sigma}$ (not available before), and it is now possible to augment this "greedy" term. Instead of describing only one immediate time period, the "greedy" term is replaced by an expression giving an uncertain cumulative distance traveled by all servers during the next $N-\tau$ stages indexed by $t=\tau+1,\dots,N$.

At each time-step τ (for $1 \le \tau \le N$), an optimal assignment k^* generated by the HARO algorithm is now defined as follows:

$$k^* = \operatorname*{arg\,min}_{k=1,\dots,K} \Big\{ C_{OPT}(\hat{\mathbf{x}}^0, \hat{\sigma}_1, \dots, \hat{\sigma}_\tau, \hat{\mathbf{x}}^\tau(k)) + d(\hat{x}_k^{\tau-1}, \hat{\sigma}_\tau) + \\ + C_{AARO}\big((\sigma_{\tau+1}, \dots, \sigma_N) \in U; \hat{\mathbf{x}}^\tau(k)\big)\Big\}, \quad (15)$$

where the first two terms are identical to the definition in WFA (2). The last term

$$C_{AARO}\big((\sigma_{\tau+1}, ..., \sigma_N) \in U; \hat{\mathbf{x}}^\tau(k)\big) = \sum_{k'=1}^{K} \sum_{t=\tau+1}^{N} (v_{k'}^t)^*$$

represents the total distance that servers will travel over the next $N-\tau$ time-steps ($t=\tau+1,\dots,N$) according to the optimal solution $\mathbf{v}^*$ of the AARO method from Section 3.2 if at time-step τ we assign server k (what described by configuration $\hat{\mathbf{x}}^\tau(k)$). The last time-step of WFA (indexed by τ in (2)) and the first time-step of AARO (indexed by τ in (13) as well) now both simultaneously represent the current time-step of the multiperiod optimization problem (15). This is the only time-step where time periods of WFA and AARO formulations overlap.

Let us define the HARO(w,T) method as a combination of the w-WFA and AARO(T) methods. More specifically, we formulate HARO(w,T) as a mixed binary optimization model with $w+T+1$ time-steps, where the WFA block has w periods, the AARO block has T periods, and the current time-step has index τ. In order to solve one stage of the HARO(w,T) problem at time-step τ we need the following input data:

1. The previous request locations: $\hat{\sigma}_{\max(\tau-w,1)}, \dots, \hat{\sigma}_\tau$, where $\hat{\sigma}_\tau$ is the position of the current request to be served. The definition of the first time index $t=\max(\tau-w,1)$ guarantees that it is always greater or equal than 1.
2. The current configuration of the system, that is, vector $\hat{\mathbf{x}}^{\tau-1} = (\hat{x}_1^{\tau-1}, \dots, \hat{x}_K^{\tau-1})$ of server locations at the end of previous time-step $t=\tau-1$.
3. The configuration of the system at the end of time-step $t=\max(\tau-w-1,0)$, that is, vector $\hat{\mathbf{x}}^{\max(\tau-w-1,0)}$ representing the starting configuration of the WFA block.
4. The description of the uncertainty set U of the form (7) for up to T future request locations $(\sigma_{\tau+1}, \dots, \sigma_{\min(\tau+T,N)})$.

In the HARO(w,T) online framework, at each time-step τ (for $1 \le \tau \le N$) we solve the formulation (16), which is by construction a combination of the w-WFA and AARO(T) formulations:

$$\begin{aligned} \min_{\substack{\mathbf{x},\mathbf{y},\mathbf{z}\\ \boldsymbol{\beta},\mathbf{u},\mathbf{v}}} \quad & \sum_{k=1}^{K} \sum_{t=\max(\tau-w,1)}^{\tau} v_k^t + \sum_{k=1}^{K} y_k^\tau \cdot d(\hat{x}_k^{\tau-1}, \hat{\sigma}_\tau) + \sum_{k=1}^{K} \sum_{t=\tau+1}^{\min(\tau+T,N)} v_k^t \\ \text{s.t.} \quad & \text{For } k=1,\dots,K: \\ & \quad v_k^t \ge x_k^t - x_k^{t-1}, \quad t=\max(\tau-w,1),\dots,\tau \\ & \quad v_k^t \ge -(x_k^t - x_k^{t-1}), \quad t=\max(\tau-w,1),\dots,\tau \end{aligned}$$

$$
\begin{aligned}
& x_k^t = x_k^{t-1} + y_k^t\,\hat{\sigma}_t - z_k^t, \quad t = \max(\tau - w, 1), \ldots, \tau - 1 \\
& x_k^{\max(\tau-w,1)} = \hat{x}_k^{\max(\tau-w-1,0)} + y_k^{\max(\tau-w,1)}\,(\hat{\sigma}_{\max(\tau-w,1)} - \hat{x}_k^{\max(\tau-w-1,0)}) && \text{(16a)} \\
& x_k^\tau = \hat{x}_k^{\tau-1} + y_k^\tau\,(\hat{\sigma}_\tau - \hat{x}_k^{\tau-1}) && \text{(16b)} \\
& z_k^t \le y_k^t, \quad t = \max(\tau - w, 1), \ldots, \tau - 1 \\
& z_k^t \le x_k^{t-1}, \quad t = \max(\tau - w, 1), \ldots, \tau - 1 \\
& z_k^t \ge y_k^t + x_k^{t-1} - 1, \quad t = \max(\tau - w, 1), \ldots, \tau - 1 \\
& y_k^\tau = \beta[k, \tau, \tau] && \text{(16c)} \\
& 0 \le x_k^t,\ z_k^t \le 1,\ y_k^t \in \{0, 1\}, \quad t = \max(\tau - w, 1), \ldots, \tau \\
\sum_{k=1}^{K} & y_k^t = 1, \quad t = \max(\tau - w, 1), \ldots, \tau - 1 \\
\boldsymbol{\beta}, & \mathbf{u}, \mathbf{v} \in \mathcal{F}(\tau, T, \hat{\mathbf{x}}^{\tau-1}, K, N, U). && \text{(16d)}
\end{aligned}
$$

The formulation (16) has a similar structure as the formulation (6) which represented the WFA method. Equations (16a) and (16b) fix configurations of the system at time-steps $t = \max(\tau - w - 1, 0)$ and $t = \tau - 1$, respectively, as required by the w-WFA definition. The most important binary assignment variables for the current time-step $t = \tau$ are modeled by y_k^τ and $\beta[k, \tau, \tau]$, $k = 1, \ldots, K$ in the WFA and AARO blocks, respectively. These decision variables are consistent with each other due to equations (16c). The last group of constraints (16d) duplicates the AARO(T) block (14) complementing the MIO formulation of the HARO(w, T) algorithm.

The HARO approach is a computationally tractable method that is stable with respect to errors in the uncertainty set. As we will see in Section 7 where we present numerical experiments, the HARO framework demonstrates superior performance in terms of objective value as opposed to the existing online algorithms and the adaptive algorithms under uncertainty (RO, AARO).

In this paper we concentrate on *empirical* competitive ratio (defined in Section 7) of RO-based methods rather than a theoretical one (1) for several reasons that distinguish a classical setting for competitive ratio analysis of online algorithms from a setting that we designed HARO algorithm for. First, all presented RO-based algorithms by design have access to approximate locations of future requests, while the key idea of competitive ratio analysis is to measure the price of not knowing the future. Second, in the RO setting by selecting an uncertainty set (and particularly its size) we can control the level of robustness of the server assignments or, equivalently, control the power of model adversary that selects the future realizations of requests. Finally, holistic HARO(w,T) algorithm is partially build upon non-competitive w-WFA method that despite the absence of theoretical guarantees is still known to generate efficient assignment recommendations in practical settings (Baumgartner et al., 2012).

5. Generalizations of the K-server problem and HARO algorithm

In the previous sections, we have considered the simplest version of the K-server problem on a one-dimensional continuous metric space in order to introduce our proposed mixed integer, robust and adaptive optimization approaches. In this section, we discuss extensions of the basic formulation of the K-server problem to more general settings.

Multidimensional continuous metric space

If the metric space is the unit cube of dimension $D \geq 2$, $S = [0,1]^D$, then all requests $\boldsymbol{\sigma}$ and server locations $\mathbf{x}$ become vectors of dimension D and we can adjust the dynamics equation (4b) as follows:

$$x_{k,d}^t = x_{k,d}^{t-1} + y_k^t(\sigma_{t,d} - x_{k,d}^{t-1}), \quad d = 1, \dots, D.$$

If the distance between points in the metric space S is induced by the first norm $\|\cdot\|_1$ or infinity norm $\|\cdot\|_\infty$, then all previous formulations (OPT, WFA, RO, AARO, HARO) remain linear and have almost identical form as before. The only difference is that now we introduce variables $v_{k,d}^t = |x_{k,d}^t - x_{k,d}^{t-1}|$ for $d = 1, \dots, D$, replacing the one-dimensional counterparts v_k^t used when $D = 1$. Moreover, for the case of the infinity norm $\|\cdot\|_\infty$, the distance traveled by all servers at time-step $t = 1, \dots, N$, v_t, can be expressed using inequalities

$$v_t \geq v_{k,d}^t, \quad k = 1, \dots, K, \; d = 1, \dots, D,$$

instead of a sum $v_t = \sum_{k=1}^{K} \sum_{d=1}^{D} v_{k,d}^t$, as is the case for the first norm $\|\cdot\|_1$.

Finite metric space

We consider the setting when the metric space S consists of a finite number of fixed points $\{\alpha_l \,|\, l = 1, \dots, L\} \subset [0,1]^D$, where D is the dimension of the embedding Euclidean space. In this setting, we replace the previously considered uncertain parameters $\boldsymbol{\sigma} = (\sigma_{\tau+1}, \dots, \sigma_N)$ (representing the positions of requests) with new uncertain parameters $\boldsymbol{\nu} = \{\, \nu_l^t \,|\, t = \tau+1, \dots, N, \; l = 1, \dots, L\}$ defined as follows:

$$\nu_l^t = \begin{cases} 1, & \text{if request } \sigma_t \text{ is at location } \alpha_l, \\ 0, & \text{otherwise.} \end{cases} \tag{17}$$

The connection between the old and new uncertain parameters is given by a linear equation

$$\sigma_{t,d} = \sum_{l=1}^{L} \alpha_{l,d} \cdot \nu_l^t, \quad t = \tau+1, \dots, N, \; d = 1, \dots, D. \tag{18}$$

Hence, in case of a finite metric space equipped with distance $d(x,y) = \|x-y\|_1$ or $d(x,y) = \|x-y\|_\infty$ the HARO algorithm for the K-server problem can be modeled similarly to (16) using the linear change of variables (18). The only difference is that now a polyhedral uncertainty set should be expressed in terms of the parameters $\boldsymbol{\nu}$, rather than $\boldsymbol{\sigma}$, in the following way:

$$U_\nu = \{\mathbf{A}_\nu \boldsymbol{\nu} \leq \mathbf{b}_\nu, \; \boldsymbol{\nu} \geq \mathbf{0}\}, \tag{19}$$

where the new parameters $\boldsymbol{\nu}$ are supposed to be binary variables. We relax this integrality assumption and consider $0 \leq \nu_l^t \leq 1$ in order to use the same duality methodology as in (11), (13) and (16). In this setting, we treat the uncertain parameters ν_l^t as probabilities $\Pr(\sigma_t = \alpha_l)$ that request σ_t is located at location α_l.

Weighted HARO algorithm

Burley (1996) introduced generalized WFA that can be parametrized by a weighting scalar of the work function. The author demonstrated that carefully tuned weight may lead to a better performance and a lower competitive ratio of WFA in some settings. Developing this idea, it is possible to consider a weighted counterpart of the HARO method:

$$k^* = \arg\min_{k=1,\ldots,K} \Big\{ C_{OPT}(\hat{\mathbf{x}}^0, \hat{\sigma}_1, \ldots, \hat{\sigma}_\tau, \hat{\mathbf{x}}^\tau(k)) + d(\hat{x}_k^{\tau-1}, \hat{\sigma}_\tau) + \\ + \lambda \cdot C_{AARO}\big((\sigma_{\tau+1}, \ldots, \sigma_N) \in U; \hat{\mathbf{x}}^\tau(k)\big) \Big\}, \quad (20)$$

where a higher weight $\lambda > 1$ in front of C_{AARO} term can be chosen if the quality of information about the future requests is high. Indeed, the more precise information about the future requests we have, the more emphasis we should put on minimization of the C_{AARO} term in (20) while, respectively, the past WFA block becomes less significant. In a limit, when parameter λ is very high, and we know locations of future requests exactly, we can completely ignore the WFA term and fully concentrate on minimization of the C_{AARO} block (this setting by construction will be identical to offline formulation (5)).

6. Construction of uncertainty sets

In this section, we describe simple polyhedral uncertainty sets as examples of sets that can be effectively used in practice with HARO method. These examples present one possible way to measure different levels of available information about future requests. We will use this "varying information level" scale to empirically demonstrate in Section 7 that the performance of all RO-based methods (RO, AARO and HARO) improves as more information becomes available.

In order to quantify the amount of available information about a future request $\bar{\sigma}$, we simply use a metric-based notion of *neighborhood* $\mathcal{N}(\bar{\sigma}, r)$, defined as the ball centered around the true location of the request and of radius r, that is,

$$\mathcal{N}(\bar{\sigma}, r) = \{x \in S \mid d(x, \bar{\sigma}) \le r\},$$

where $d(\cdot, \cdot)$ is a chosen distance of the metric space. For the case of continuous metric space $S = [0,1]^D$ with the uniform norm $\|\cdot\|_\infty$ and radius $r \ge 0$, we have

$$\mathcal{N}(\bar{\sigma}, r) = \{x \in S \mid \|x - \bar{\sigma}\|_\infty \le r\}.$$

In the context of a finite metric space, the neighborhood $\mathcal{N}(\bar{\sigma}, r)$ with integral radius $r \in \{1, \ldots, |S|\}$ consists of r closest neighbors to the point $\bar{\sigma}$ including itself.

Having specified the notion of the neighborhood, one may now consider the following family of uncertainty sets at any given time-step τ:

$$U_\tau(T, r) = \{(\sigma_{\tau+1}, \ldots, \sigma_{\tau+T}) \mid \sigma_t \in \mathcal{N}(\hat{\sigma}_t, r),\ t = \tau+1, \ldots, \tau+T\}, \quad (21)$$

where T denotes time horizon and $\{\hat{\sigma}_t \mid t = \tau+1, \ldots, \tau+T\}$ are true locations of future requests. Generally speaking, we do not know the exact locations of T future requests $\{\hat{\sigma}_t \mid t = \tau+1, \ldots, \tau+T\}$, but only their surrounding neighborhoods of radius r. In this case, the parameter r measures how precise the information is. Moreover, for both cases of continuous and finite metric spaces, the uncertainty set $U_\tau(T, r)$ can be represented as a bounded polyhedron of the form (7) or (19), expressed in terms of $\boldsymbol{\sigma}$ or $\boldsymbol{\nu}$, respectively.

In the next section, we use uncertainty sets of type (21) in order to show the increasing advantage of the HARO method as the amount of available information grows. At the same time, other types of polyhedral uncertainty sets describing future requests are also possible. One possible example is a Central Limit Theorem-motivated set defined as

$$U = \left\{ (\sigma_{\tau+1}, \ldots, \sigma_{\tau+T}) \,\Big|\, \Big| \frac{\sum_{t=\tau+1}^{\tau+T} \sigma_t - \bar{\mu}}{\bar{s}\sqrt{T}} \Big| \le \Gamma \right\},$$

where $\bar{\mu}$ and $\bar{s}$ denote empirical mean and standard deviation of requests $\boldsymbol{\sigma}$, respectively, and Γ is a predefined level of robustness (Bertsimas et al., 2011). In case of a finite metric space, parameters $\boldsymbol{\nu}$ defined in (17) allow a practitioner to incorporate information about the distribution of future requests. For instance, inequalities

$$\underline{\rho} \le \sum_{l:\, \alpha_l \in S_0} \nu_l^t \le \bar{\rho}$$

state that at time-step t probability that request σ_t will emerge somewhere in a subset $S_0 \subset S$ is bounded by pre-specified parameters $\underline{\rho}$ and $\bar{\rho}$. For more complex (and potentially non-polyhedral) settings with correlated or multimodal distributions of uncertain parameters we refer an interested reader to extensive practical guides for data-driven uncertainty sets suggested by Bandi & Bertsimas (2012); Bertsimas et al. (2018); Hanasusanto et al. (2015).

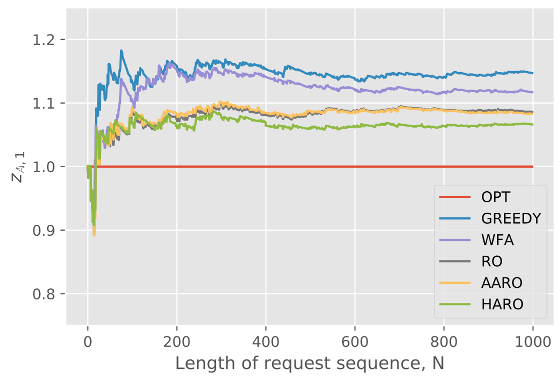
7. Numerical experiments

In this section, we provide empirical evidence that the HARO method (16) designed in Section 4 for the K-server problem is computationally tractable, and almost uniformly outperforms existing online methods, as well as adaptive optimization methods. We show that the HARO algorithm gradually improves as more information about the future requests becomes available, and that it is stable with respect to possible mistakes in the assumptions about uncertain request locations.

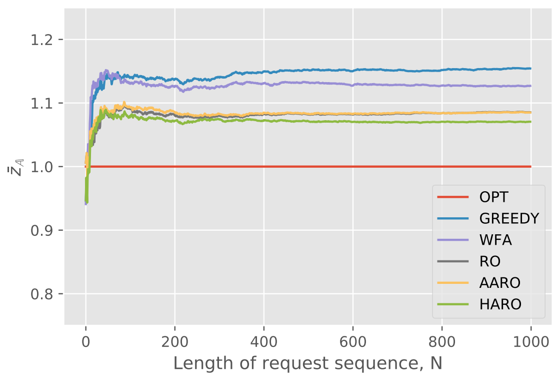
The K-server problem as well as the proposed HARO method are characterized by many different parameters and performance metrics. The most significant ones are:

- The underlying metric space S (continuous or finite) with distance metric d;
- The number of servers K and their initial locations $\mathbf{x}_0$ in S;
- The distribution of requests (uniform or non-uniform) in space S;
- The number of locations L in case of finite metric space S;
- The length of the WFA optimization window w and the AARO optimization window T in the definition of the HARO(w, T) method;
- The uncertainty set U describing future requests;
- The computational time required to find high-quality solutions.

The benchmark method in all our experiments is the optimal offline algorithm OPT. We implemented OPT and WFA as network flow problems according to Rudec et al. (2013). Of all heuristic online algorithms presented in Section 2.1, only results for GREEDY and WFA are reported, since the rest of the methods (RAND, HARMONIC, BALANCE) resulted in all of our experiments in worse objective values. We coded all algorithms in Julia/JuMP (Lubin & Dunning, 2015) and solved them with Gurobi 6.5 on a computer with 2.5Hz processor and 16GB of memory.



(a) Convergence of $z_{\mathbb{A},1}$



(b) Convergence of $\bar{z}_{\mathbb{A}}$

Figure 1: Typical convergence rate of the empirical competitive ratios $z_{\mathbb{A},1}$ (for one instance $i = 1$) and $\bar{z}_{\mathbb{A}}$ (average across $I = 10$ instances) with respect to N. Experiment description: $S = [0,1]^2$, $K = 5$, $T = 5$, $w = 10$, U is defined by (23) with $n = 5$, distribution of requests is uniform.

In all experiments, we run $I = 10$ instances (indexed by i), each of which is determined by a vector of initial server locations $\mathbf{x}^{0,i}$ and by a sequence of requests $\boldsymbol{\sigma}^i$ with a fixed length of $N = 1000$ time-steps. Parameters in the HARO(w,T) method vary from 10 to 30 for w, and from 5 to 15 for T. Each method $\mathbb{A}$ that we test (OPT, GREEDY, WFA, RO(T), AARO(T), HARO(w,T)) generates a sequence of server assignments $\mathbf{y}^*_{\mathbb{A},i} = (y^*_1, \dots, y^*_N)$ that depends on instance $i = 1, \dots, I$. Having obtained a sequence $\mathbf{y}^*_{\mathbb{A},i}$, one can calculate the total distance $C_{\mathbb{A}}(\mathbf{x}^{0,i}, \boldsymbol{\sigma}^i)$ traveled by all servers. In all experiments, we report the average empirical competitive ratio $\bar{z}_{\mathbb{A}}$ as the performance metric of algorithm $\mathbb{A}$:

$$\bar{z}_{\mathbb{A}} = \frac{1}{I}\sum_{i=1}^{I} z_{\mathbb{A},i}, \qquad \text{where } z_{\mathbb{A},i} = \frac{C_{\mathbb{A}}(\mathbf{x}^{0,i}, \boldsymbol{\sigma}^i)}{C_{OPT}(\mathbf{x}^{0,i}, \boldsymbol{\sigma}^i)}. \tag{22}$$

The optimal algorithm OPT (modeled by a single optimization formulation (5) for all $N = 1000$ requests) knows *precisely* the sequence of all future requests $\hat{\boldsymbol{\sigma}} = (\hat{\sigma}_1, \dots, \hat{\sigma}_N)$. Online heuristic algorithms GREEDY and WFA do not use information about future requests by construction. RO-based algorithms (RO(T), AARO(T), HARO(w,T)) have access to *approximate* locations of the next T requests. The approximation comes in a form of the Cartesian product of confidence intervals centered around true locations (denoted by $\hat{\sigma}_{\tau+1}, \dots, \hat{\sigma}_{\tau+T}$) and is modeled by uncertainty sets of type (21).

The length of the request sequence $N = 1000$ and the number of test instances $I = 10$ were empirically proved to be sufficient to statistically differentiate the behavior of the six methods considered in our experiments. The typical convergence rate of the empirical competitive ratios $z_{\mathbb{A},1}$ and $\bar{z}_{\mathbb{A}}$ (22) with respect to the length N is presented in Figure 1.

The metric space S is either $[0,1]^2$ for the continuous case, or a set of L randomly generated points in $[0,1]^2$ for the discrete case. The distance between locations is calculated using the metric $d(x,y) = \|x - y\|_1$. The vector of initial locations for the servers $\mathbf{x}_0$ is generated uniformly at random. By default, the uncertainty sets $U_\tau(T,r)$ (21) do not include possible errors and the points $\hat{\sigma}_t$, $t = \tau + 1, \ldots, \tau + T$ represent the true locations of the future requests.

In case of the continuous metric space S, we test the performance of each RO-based algorithm with the same sequence of requests $\hat{\boldsymbol{\sigma}} = (\hat{\sigma}_1, \ldots, \hat{\sigma}_{1000})$, but different levels of information that is available about the next T requests. We model this varying level of information by the uncertainty sets $U_\tau(T, \frac{1}{n})$, for $n = 1, \ldots, 10$. For any fixed value of n, the decision maker knows that the next T uncertain requests σ_t belong to the corresponding intervals of type

$$\hat{\sigma}_t - \frac{1}{n} \leq \sigma_t \leq \hat{\sigma}_t + \frac{1}{n}. \tag{23}$$

In the context of finite metric space S, for each instance $i = 1 \ldots I$ we run all online algorithms 12 times with identical sequence of requests $\hat{\boldsymbol{\sigma}}^i = (\hat{\sigma}^i_1, \ldots, \hat{\sigma}^i_{1000})$, but with gradually increasing amount of information represented by uncertainty sets

$$U_\tau\Big(T,\, \lceil L\big(1 - \frac{n-1}{12}\big)\rceil\Big).$$

Generally speaking, for any given $t = \tau + 1, \ldots, \tau + T$, parameter $n = 1, \ldots, 12$ controls the number of neighboring locations that contain a true (but uncertain) location $\hat{\sigma}_t$. The units of parameter n are different in two cases. In the setting of a continuous metric space, n is a measure of distance in this space induced by the norm $\|\cdot\|_1$, while for discrete setting, n is a fraction of a total number of locations. However, the range of parameter n in both settings is similar ($1 \leq n \leq 10$ or $1 \leq n \leq 12$) by design and the larger values of n correspond to more precise information about the future requests. In both cases, the case of $n = 1$ corresponds to the scenario when there is no information available about the future requests and these requests can appear anywhere in the metric space.

In our experiments, we consider three types of request distributions over space S. The *uniform* distribution models scenario when requests appear in different locations of S with equal probability. The *non-uniform* distribution is defined as follows. When $S = [0,1]^2$, we consider a right top quadrant of the square $\bar{S} = \{(x,y) \,|\, 0.5 \leq x, y \leq 1\} \subset S$ and generate 50% of all requests in $\bar{S}$ (permuted randomly), and the remaining 50% of requests are sampled outside this subset $\bar{S}$. In case of finite metric space S, we first randomly select 25% of locations L and then again assign 50% of all requests to the specified subset of points $\bar{S}$. Finally, we define distribution *with remote locations* for the case of continuous metric space $S = [0,1]^2$ as follows. With probability 90%, a request is generated uniformly at random in a central subset $\bar{S} = [0.35, 0.65]^2 \subset S$, and with equal probability 5%, a request is generated either at the remote corner $(0,0)$ or $(1,1)$. This setting is inspired by examples of special types of metric spaces (Rudec et al., 2013) demonstrating non-competitiveness of GREEDY algorithm and advantages of WFA method when distribution of requests is highly non-uniform.

In the next Sections 7.1 - 7.4, we conduct numerical experiments to empirically answer the following questions. Does the HARO method generate more efficient solutions than other server assignment algorithms? (Section 7.1) How stable the HARO method is with respect to potential noise or errors in the description of the future modeled by the uncertainty set? (Section 7.2) How does the performance of the HARO(w,T) method change with respect to parameters w and T? (Section 7.3) Does the HARO method preserve its strong performance across various scenarios defined by the number of servers and locations and the distribution of requests? (Section 7.4)

7.1. Performance of the methods

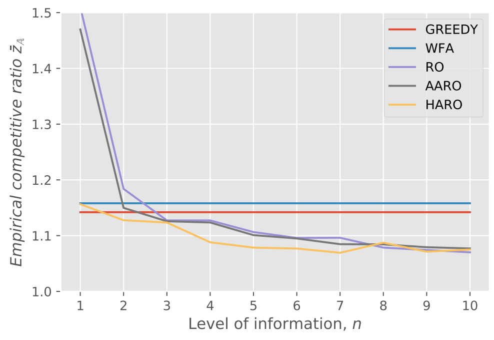
In this experiment, we demonstrate that the HARO(w,T) method outperforms the best existing online algorithms (GREEDY, WFA) and adaptive methods (RO(T), AARO(T)) in terms of the empirical competitive ratio $\bar{z}_{\mathbb{A}}$ for almost all tested levels of information about the future. We consider several different scenarios: when the metric space S is continuous or finite, when distribution of requests is uniform or not, and when there are remote locations in the space or not, as defined in the beginning of this section.

Other parameters of the experiment are: $T = 5$, $w = 20$, $K = 7$, and $L = 25$ (when space S is finite). Given that $N = 1000$ and $I = 10$ are large enough, we report the performance results based on I realizations of the sequences $\hat{\boldsymbol{\sigma}}^i = (\hat{\sigma}_1^i, \ldots, \hat{\sigma}_{1000}^i)$ that yield an accurate estimation of the performance metric $\bar{z}_{\mathbb{A}}$. The simulation results (most of which are visualized in Figure 2 and 3) infer the following conclusions:

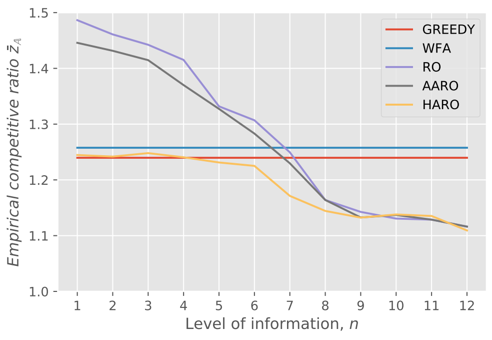
1. All three proposed methods (RO(T), AARO(T), HARO(w, T)) improve as the amount of available information measured by parameter n grows. The performance of online heuristic GREEDY and WFA by construction does not depend on n what can be seen in Figure 2.
2. In all five scenarios, the asymptotic empirical competitive ratio of the RO-based methods is approximately the same, and it is between 1.08 and 1.16 depending on the particular experiment. This level is strictly greater than 1, because even for large values of information precision n, our model incorporates information only about the next $T = 5$ requests.
3. For small values of n, RO(T) method yields solutions that are too conservative with worse objective values. It illustrates the situation when there is almost no information about the next T requests, and RO(T) tries to find server assignments against the worst-case scenario. The AARO(T) method has a slightly better performance than RO(T) for small values of n, but after some point RO(T) and AARO(T) results are almost indistinguishable.
4. The HARO(w, T) algorithm, as a combination of stable w-WFA and adaptive AARO(T) methods, does not have high empirical competitive ratios for small values of n and at the same time improves with increasing of the available information. In most of the cases, even for large values of n, HARO(w, T) method outperforms AARO(T) algorithm. In case when there is no information about next T requests ($n = 1$), the HARO(w, T) method with selected type of uncertainty set is as efficient as online algorithms GREEDY and WFA.
5. In case of the finite metric space S, AARO(T) requires more information (compared to the case of continuous S) in order to approach the quality of existing methods GREEDY and WFA. This is why the benefit of the HARO(w, T) algorithm over AARO(T) becomes more pronounced for finite spaces.
6. The average run time of one step of HARO(w, T) algorithm is $2.35\,s$ when S is continuous and $4.11\,s$ when S is finite.

In addition, we tested a setting with a highly non-uniform request distribution in the continuous metric space S with remote locations. By construction, the locations of five out of $K = 7$ servers were generated uniformly at random inside the central subset $\bar{S}$, while the remaining two servers were initially positioned in opposite corners $(0, 0)$ and $(1, 1)$. In this setting, GREEDY algorithm never assigns a corner server to a request in the central part $\bar{S}$, despite the fact that vast majority of requests emerge in $\bar{S}$. The WFA method, on contrary, quickly learns from historical observations that it is beneficial for minimization of a total distance traveled by all servers to move a corner server to $\bar{S}$. By design, the HARO method also takes into account past request observations and inherits this positive property of the WFA method (Fig. 3).

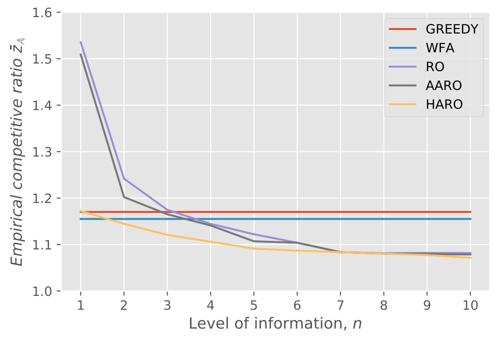
Overall, the empirical performance of the HARO(w, T) algorithm is the best out of all considered methods (GREEDY, WFA, RO(T), AARO(T)) for most of the information levels n, characteristics of the metric space and the distribution of requests.



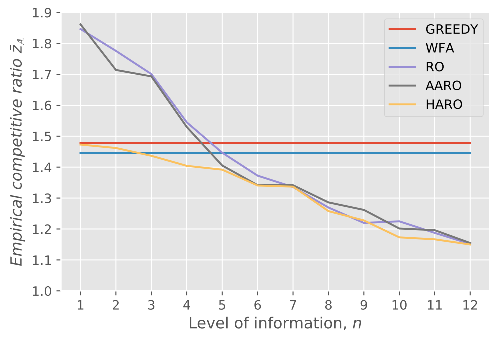
(a) S is continuous; uniform distribution of $\boldsymbol{\sigma}$



(b) S is finite; uniform distribution of $\boldsymbol{\sigma}$



(c) S is continuous; non-uniform distribution of $\boldsymbol{\sigma}$



(d) S is finite; non-uniform distribution of $\boldsymbol{\sigma}$

Figure 2: Empirical performance of online algorithms

7.2. Sensitivity of the HARO approach with respect to noise

The second experiment demonstrates the stability of the HARO(w,T) algorithm with respect to noise in the description of the uncertainty sets $U_\tau(T,r)$. More precisely, we introduce a new parameter μ which is equal to the probability that the true location of future request $\hat{\sigma}_t$ is mistakenly replaced with some random point $\tilde{\sigma}_t(\mu)$ in S, for $t=\tau+1,\ldots,\tau+T$. That is, we define the perturbed version of the uncertainty set as follows:

$$\tilde{U}^\mu_\tau(T,r) = \left\{(\sigma_{\tau+1},\ldots,\sigma_{\tau+T}) \,|\, \sigma_t \in \mathcal{N}\big(\tilde{\sigma}_t(\mu), r\big),\, t=\tau+1,\ldots,\tau+T\right\},$$

where for each $t=\tau+1,\ldots,\tau+T$ the perturbed request $\tilde{\sigma}_t(\mu)$ is defined by

$$\tilde{\sigma}_t(\mu) = \begin{cases} \hat{\sigma}_t, & \text{with probability } 1-\mu, \\ \alpha_l, & \text{with probability } \mu, \text{ for some randomly chosen } l \in \{1,\ldots,L\}. \end{cases}$$

For this experiment we select the finite space S with $L=30$ locations and a uniform distribution of requests $\hat{\boldsymbol{\sigma}}$. We fix the level of available information $n=4$, and set parameters $I=10$, $K=10$, $N=7$, $w=15$. From Figure 4 we infer that when the distortion probability $\mu=0$ and, therefore,

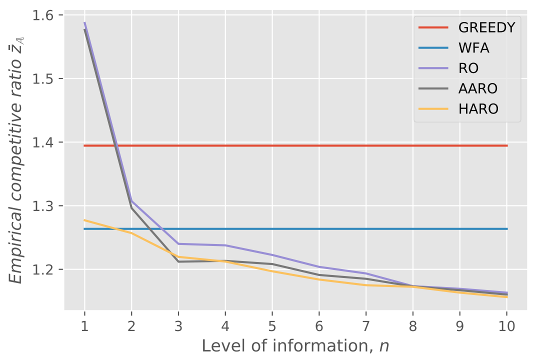


Figure 3: Performance of the algorithms in a setting with remote locations.

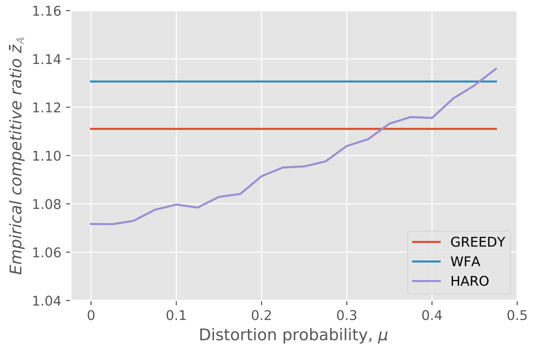


Figure 4: Performance of the HARO method with perturbed sets $\tilde{U}_{\mu}$.

there are no errors in provided information, then the HARO(w,T) method performs significantly better than GREEDY and WFA. Obviously, the mistakes in uncertainty set $\tilde{U}^{\mu}_{\tau}(T,r)$ aggravate the efficiency of the HARO(w,T) algorithm, but deterioration of the method performance is slow. The HARO(w,T) method starts to perform at the level of the best existing online algorithm when the probability of distortion μ is greater than 35% for each of the time-steps $t=\tau+1,\dots,\tau+T$.

We also tested HARO(w,T) algorithm with assymetric uncertainty sets $U_{\tau}(T,r)$, when the true location of request $\hat{\sigma}_t$ was not necessarily in the middle of the uncertainty interval as in Eq. (23), but rather just a random point of an interval of length $\frac{2}{n}$:

$$\hat{\sigma}_t-(1-\theta_t)\frac{2}{n}\leq\sigma_t\leq\hat{\sigma}_t+\theta_t\frac{2}{n},$$

where parameters θ_t were generated uniformly at random between 0 and 1. This pivot in the definition of the uncertainty sets did not observably change the performance of the HARO(w,T) method presented in Figure 2 - 4.

Table 1: Average time needed to find an optimal assignment for one time-step of the HARO(w, T) method.

	$T = 5$	$T = 10$	$T = 15$
$w = 10$	$1.21\,s$	$6.75\,s$	$10.52\,s$
$w = 20$	$2.2\,s$	$17.96\,s$	$28.4\,s$
$w = 30$	$5.17\,s$	$27.33\,s$	$43.97\,s$

7.3. Sensitivity of the HARO approach with respect to the number of time-steps

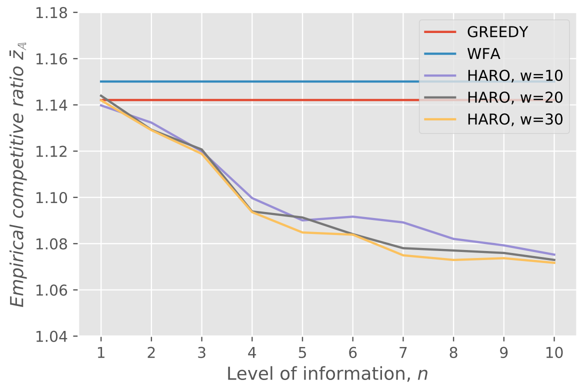
In this experiment, we demonstrate the influence of parameters w and T on the performance of the HARO(w, T) method, which are equal to the number of time-steps in WFA and AARO blocks, respectively. Typically, performance of the HARO(w, T) method remains at least at the same level and often improves as more information about the future and the past (provided by larger values of w and T) is incorporated to the formulation. However, we also discuss a trade-off between the efficiency of the HARO(w, T) algorithm and the computational time needed to find optimal assignments.

We consider continuous metric space S with $K = 5$ servers with a uniform distribution of requests and standard uncertainty sets (21). Computational results presented in Figure 5 imply that for any fixed value of T increasing the window size w often improves the performance of the HARO(w, T) algorithm, particularly for medium values of information level $4 \leq n \leq 7$. Furthermore, there is a substantial improvement in the efficiency of the HARO(w, T) algorithm when length of an adaptive window grows from $T = 5$ to $T = 10$. First, when $T = 10$ the method on average has a steeper slope for $1 \leq n \leq 4$, and it converges to lower empirical competitive ratio 1.065, rather than 1.075 for HARO(w, T) with $T = 5$. At the same time, there is no observable change when parameter T is increased from 10 to 15. Probably, for $K = 5$ servers knowing the approximate locations of the next $T = 10$ requests is already sufficient.

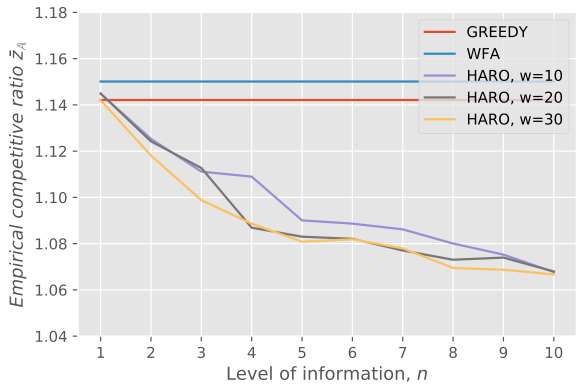
It is worth mentioning, that the vast majority of variables in formulation (16) are auxiliary, and only vector $\mathbf{y}^*(\tau) = \{(y_k^\tau)^* \mid k = 1, \ldots, K\}$ determines which server $k = 1, \ldots, K$ should be assigned at the current time-step τ. A typical behavior of the solution progress is that the optimal assignment values $\mathbf{y}^*(\tau)$ are obtained by Gurobi solver in seconds and remain unchanged after this, while it takes from minutes to tens of minutes (depending on the instance size) to optimize over auxiliary variables and prove the optimality of vector $\mathbf{y}^*(\tau)$. Table 1 displays average computational times needed to find the optimal values of binary assignment variables $\mathbf{y}^*(\tau)$ for each of the variants of HARO(w, T) approach. As a result, we infer that the HARO(w, T) algorithm provides the decision maker with a high-quality recommendation within seconds for instances of practical size. (The online algorithms GREEDY and WFA generated assignment recommendations in less than 1 second.)

7.4. Performance of the HARO approach in various settings

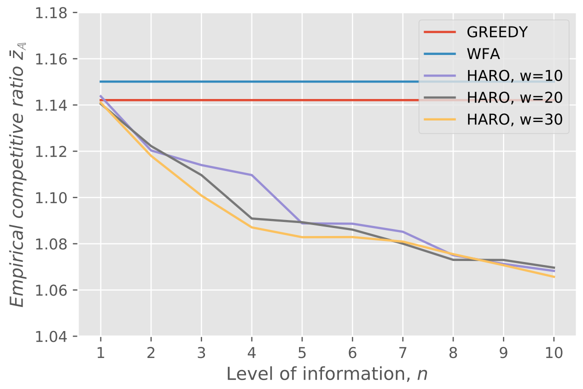
In this experiment, we demonstrate a substantial advantage of the HARO(w, T) algorithm over existing online methods (GREEDY, WFA) in various scenarios that differ by a number of servers K, locations L and distribution of requests in space S. In some sense, we extend experiment from Section 7.1 for finite metric space S to a large number of possible combinations of K and L, but present results in a more concise form. Instead of reporting a vector of length 12 describing the



(a) Performance of HARO with $T = 5$ and $w \in \{10, 20, 30\}$



(b) Performance of HARO with $T = 10$ and $w \in \{10, 20, 30\}$



(c) Performance of HARO with $T = 15$ and $w \in \{10, 20, 30\}$

Figure 5: Sensitivity of the HARO(w,T) approach with respect to w and T

performance of the HARO(w, T) method for various $n = 1, \ldots, 12$, we present in Table 2 and 3 a scalar which is equal to the average empirical competitive ratio of HARO approach:

$$\text{HARO (mean)} = \frac{1}{12} \sum_{n=1}^{12} \frac{\text{Cost of HARO}(w, T) \text{ with information level } n}{\text{Cost of OPT}}. \tag{24}$$

Having fixed optimization window sizes $w = 15$ and $T = 5$, and parameter $I = 1$, we infer from Monte Carlo simulations that on average empirical competitive ratio of all methods (including HARO(w, T)) increases when number of servers K grows or number of locations L decreases. The empirical competitive ratios of the HARO(w, T) algorithm are higher when the distribution of requests $\hat{\boldsymbol{\sigma}}$ is non-uniform (Table 3) for large enough values of L, when compared to the corresponding experiments with uniformly distributed requests.

In all tested scenarios the HARO(w, T) algorithm has a similar behavior to one presented in Section 7.1 and it almost uniformly outperforms existing online methods. The mean advantage

Table 2: Empirical competitive ratio and average computational time of the methods (uniform request distribution).

Number of locations, L	15			25			40		
Number of servers, K	3	5	10	3	5	10	3	5	10
GREEDY	1.23	1.41	1.58	1.17	1.37	1.55	1.15	1.23	1.33
WFA	1.24	1.43	1.51	1.24	1.45	1.48	1.21	1.27	1.39
HARO (mean)	1.17	1.35	1.40	1.15	1.29	1.31	1.11	1.21	1.30
Aver. comp. time /HARO/	1.22 s	1.57 s	2.01 s	1.9 s	2.11 s	2.56 s	1.92 s	3.26 s	4.04 s

Table 3: Empirical competitive ratio and average computational time of the methods (non-uniform request distribution).

Number of locations, L	15			25			40		
Number of servers, K	3	5	10	3	5	10	3	5	10
GREEDY	1.18	1.36	1.56	1.32	1.46	1.69	1.26	1.41	1.43
WFA	1.19	1.34	1.39	1.22	1.47	1.67	1.23	1.39	1.44
HARO (mean)	1.15	1.33	1.30	1.16	1.39	1.53	1.16	1.28	1.36
Aver. comp. time /HARO/	1.17 s	1.58 s	1.82 s	2.03 s	2.64 s	2.56 s	2.1 s	3.4 s	4.13 s

of the HARO(w, T) method defined in (24) varies from 2% ($K = 3, L = 25$, Table 2) to 26% ($K = 10$, $L = 15$, Table 3). Average time needed for the HARO method to find the optimal server assignment (Table 2 and 3) demonstrates the computational tractability of the algorithm for online applications.

8. Conclusion

In this paper, we have considered the K-server problem from the perspective of mixed integer, robust and adaptive optimization. We have designed new MIO formulations for the offline version of the problem and for the online WFA. We have also introduced new algorithms that employ robust optimization and include into consideration assumptions about future locations of requests.

We have combined ideas behind an efficient online algorithm (WFA) and AARO techniques and designed a new holistic algorithm (HARO) that simultaneously incorporates data describing the past and information about the future. We have empirically demonstrated that HARO method is tractable and almost uniformly yields lower objective cost than existing heuristic algorithms and standard RO approaches. In most of the settings defined by the level of information about the future, the metric space structure, the distribution of requests and number of servers, the HARO algorithm has led to lower empirical competitive ratios than other methods. Finally, we have shown that the HARO method is stable with respect to potential noise in the uncertainty sets associated with future requests.

Acknowledgments

This research was partially supported by ONR grants N00014-12-1-0999 and N00014-16-1-2786. We thank the anonymous reviewers for their careful reading of our manuscript and their many

insightful recommendations and comments.

References

Bandi, C., & Bertsimas, D. (2012). Tractable stochastic analysis in high dimensions via robust optimization. *Mathematical programming*, *134*, 23–70.

Bartal, Y., & Grove, E. (2000). The harmonic k-server algorithm is competitive. *Journal of the ACM (JACM)*, *47*, 1–15.

Baumgartner, A., Rudec, T., & Manger, R. (2012). The design and analysis of a modified work function algorithm for solving the on-line k-server problem. *Computing and informatics*, *29*, 681–700.

Bazaraa, M. S., Jarvis, J. J., & Sherali, H. D. (2011). *Linear programming and network flows*. John Wiley & Sons.

Ben-Tal, A., Goryashko, A., Guslitzer, E., & Nemirovski, A. (2004). Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, *99*, 351–376.

Ben-Tal, A., & Nemirovski, A. (2008). Selected topics in robust convex optimization. *Mathematical Programming*, *112*, 125–158.

Bertsimas, D., Brown, D. B., & Caramanis, C. (2011). Theory and applications of robust optimization. *SIAM review*, *53*, 464–501.

Bertsimas, D., Gupta, V., & Kallus, N. (2018). Data-driven robust optimization. *Mathematical Programming*, *167*, 235–292.

Borodin, A., & El-Yaniv, R. (2005). *Online computation and competitive analysis*. Cambridge university press.

Borodin, A., Linial, N., & Saks, M. E. (1992). An optimal on-line algorithm for metrical task system. *Journal of the ACM (JACM)*, *39*, 745–763.

Burley, W. R. (1996). Traversing layered graphs using the work function algorithm. *Journal of Algorithms*, *20*, 479–511.

Chrobak, M., Karloof, H., Payne, T., & Vishwnathan, S. (1991). New ressults on server problems. *SIAM Journal on Discrete Mathematics*, *4*, 172–181.

Du, D.-Z., & Hwang, F. K. (1993). Competitive group testing. *Discrete Applied Mathematics*, *45*, 221–232.

Floratos, A., & Boppana, R. (1997). The on-line k-server problem. *TR1997-732, NYU, CS Department*, (pp. 1–39).

Hanasusanto, G. A., Kuhn, D., Wallace, S. W., & Zymler, S. (2015). Distributionally robust multi-item newsvendor problems with multimodal demand distributions. *Mathematical Programming*, *152*, 1–32.

Koutsoupias, E. (1999). Weak adversaries for the k-server problem. In *Foundations of Computer Science, 1999. 40th Annual Symposium on* (pp. 444–449). IEEE.

Koutsoupias, E. (2009). The k-server problem. *Computer Science Review*, *3*, 105–118.

Koutsoupias, E., & Papadimitriou, C. H. (1995). On the k-server conjecture. *Journal of the ACM (JACM)*, *42*, 971–983.

Koutsoupias, E., & Taylor, D. S. (2000). The CNN problem and other k-server variants. In *Annual Symposium on Theoretical Aspects of Computer Science* (pp. 581–592). Springer.

Lubin, M., & Dunning, I. (2015). Computing in operations research using Julia. *INFORMS Journal on Computing*, *27*, 238–248.

Manasse, M. S., McGeoch, L. A., & Sleator, D. D. (1990). Competitive algorithms for server problems. *Journal of Algorithms*, *11*, 208–230.

Raghavan, P. (1992). A statistical adversary for on-line algorithms. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, *7*, 79–83.

Raghavan, P., & Snir, M. (1989). Memory versus randomization in on-line algorithms. In *International Colloquium on Automata, Languages, and Programming* (pp. 687–703). Springer.

Rudec, T., Baumgartner, A., & Manger, R. (2013). A fast work function algorithm for solving the k-server problem. *Central European Journal of Operations Research*, *21*, 187–205.

Sleator, D. D., & Tarjan, R. E. (1985). Amortized efficiency of list update and paging rules. *Communications of the ACM*, *28*, 202–208.