

MaxHedge: Maximizing a Maximum Online



Stephen Pasteris (UK-UCL), Fabio Vitale (FRA-INRIA),
Kevin Chan (US-ARL), Shiqiang Wang (US-IBM),
Mark Herbster (UK-UCL)

[dstl]



MaxHedge: Maximizing a Maximum Online

Stephen Pasteris, Fabio Vitale, Kevin Chan, Shiqiang Wang, Mark Herbster

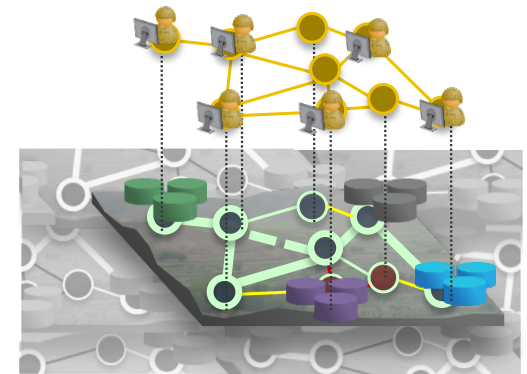
- Paper published in AISTATS 2019
- Service (server) placement problems well studied in the **batch** (static) case
- We present a novel **online** framework based on **online machine learning**
- Framework is very general in that it has different problems as subcases
- We give an efficient algorithm, MaxHedge, with a performance guarantee



Motivation

2

- The objective of distributed analytics is to support coalition operations in a wide range of services and applications
- Available resources are severely limited due to
 - Tactical systems – limited storage, compute and transmission capabilities
 - Dynamic and hostile operational environment – limited bandwidth and communications
 - Existing policies – coalitions may restrict running analytics by entity or location
- Current approaches to analytics are based on centrally located central services for execution and then results are disseminated, using significant amounts of computing and networking resources.
- Distributed analytics can significantly enhance coalition operations at the tactical edge, providing situation awareness for a variety of applications (e.g., ISR, C2).

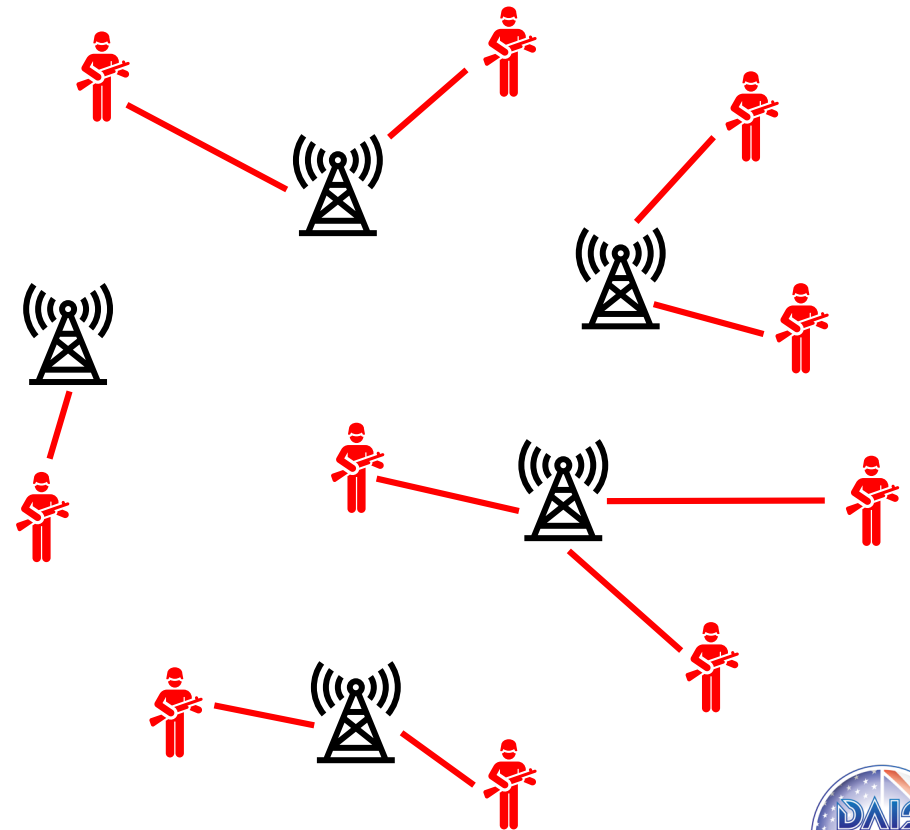


Static Server Placement Problem

3

- Given N users
- We place K servers
- $\delta(u, k)$ is the distance from user u to site k
- $\rho_u(\cdot)$ is the decreasing reward function
- **Objective:** place servers to maximize

sum of rewards:
$$\sum_u \rho_u(\min_k \delta(u, k))$$

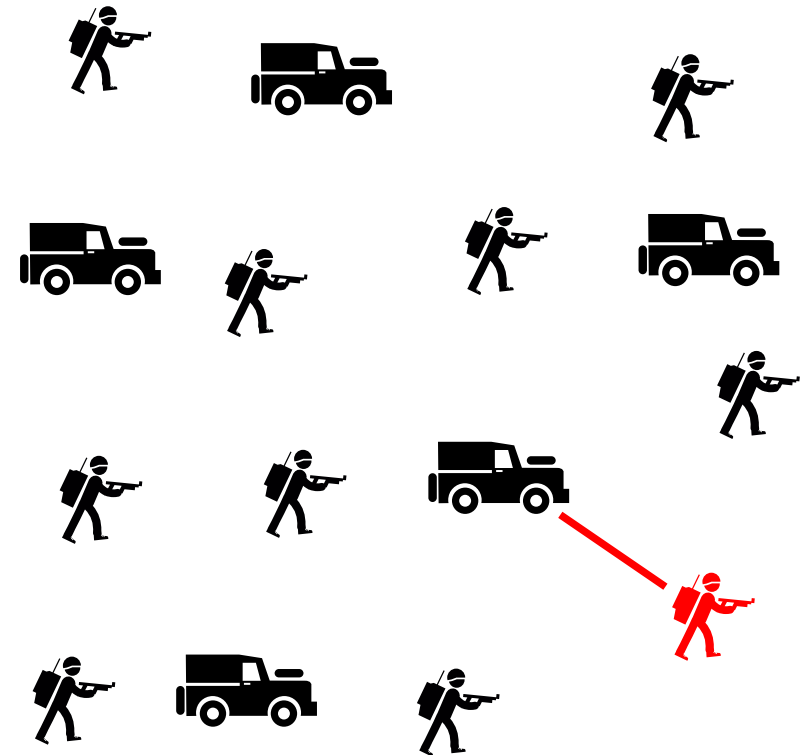


Online Learning: *On the move* ...

4

- Users are **moving**
- User requests come **one at a time**
- We now have K **mobile** servers
- We **don't know** the location of the next user request
- Given a user u request we want

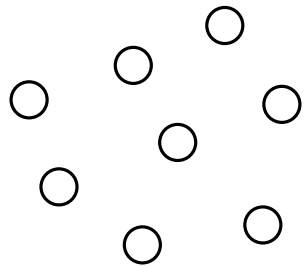
$\rho_u(\min_k \delta(u, k))$
to be large



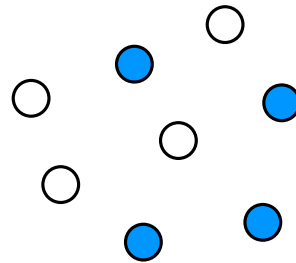
The online protocol

5

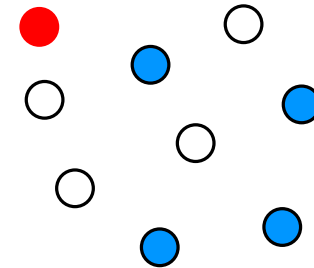
- There is an *online* stream of user requests



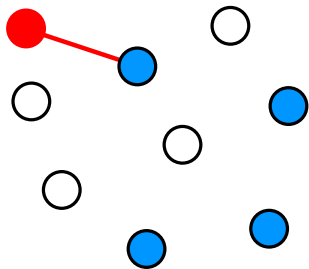
1) Given sites.



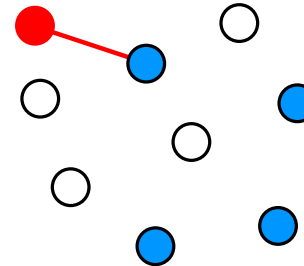
2) We **place** K servers (blue)



3) Next user (red) **requests**



4) User **connects** to server



5) Receive **reward** based on distance

K-servers: a simple special case

6

- We have a set of N sites
- Learning proceeds in T trials. On trial t :
 - For every site k we have an unknown potential reward r_k^t
 - We choose a set X^t of K sites on which to place servers
 - The potential rewards r_k^t are revealed to us
 - We receive a reward of $\max_{k \in X^t} r_k^t$
- The goal is to maximize the cumulative reward:

$$r_k^t = \rho_u(\delta(u, k))$$

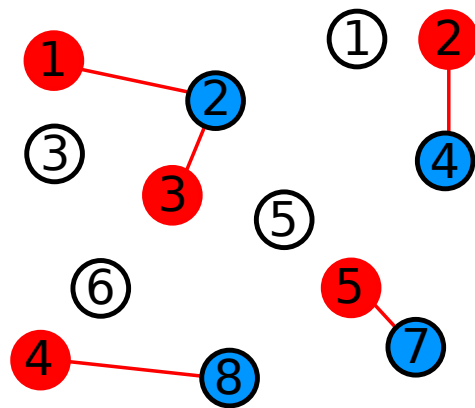
for user u at trial t

$$\sum_{t=1}^T \max_{k \in X^t} r_k^t$$



Performance of MaxHedge on K-servers: best fixed set 7

- On any trial t and any set S of K sites we define $\mu^t(S) = \max_{k \in S} r_k^t$
- $\mu^t(S)$ is the reward we would obtain on trial t if we selected $X^t = S$



Sites in S : blue
Sites not in S : white
Users: red

$$\mu^1(S) = r_2^1, \mu^2(S) = r_4^2, \mu^3(S) = r_2^3, \mu^4(S) = r_8^4, \mu^5(S) = r_7^5$$

Performance of MaxHedge on K-servers

8

- The total static reward from choosing set S for all trials:

$$\sum_{t=1}^T \mu^t(S)$$

- For any set S of K sites the expected total reward of MaxHedge is at least:

$$(1 - e^{-1}) \sum_{t=1}^T \mu^t(S) - \mathcal{O}(N\sqrt{T})$$

- **Expected reward** is **>62%** of the best possible static reward minus a small term
- **MaxHedge** takes a time of only $\mathcal{O}(n \log n)$ per trial



Generalising the framework!

9

- Each site i has *additionally* an “energy requirement” z_i
- Learning proceeds in trials $t = 1, 2, 3, \dots, T$
- On each trial t and each site i we have additionally an unknown “cost” c_i^t
- On each trial t and each site i we have a unknown “potential reward” r_i^t
- On each trial t :
 - We choose a set of sites X^t with $\sum_{i \in X^t} z_i \leq 1$
 - For all sites i , c_i^t and r_i^t are revealed to us
 - We gain a profit of:

$$\mu^t(X^t) = \max_{i \in X^t} r_i^t - \sum_{i \in X^t} c_i^t$$



- We define $C = \{x \in [0,1]^n: \sum_{i \in n} z_i x_i \leq 1\}$
- **MaxHedge** maintains a vector $\omega \in C$. Let ω^t be its value on trial t
- X^t is generated from ω^t with a randomized construction
- At the end of trial t :
 - A concave function $h^t: C \rightarrow \mathbb{R}$ is constructed (from r^t and c^t)
 - We have that $h^t(\omega^t)$ is a lower bound on the expected profit
 - ω is updated via projected gradient ascent with h^t



The general problem has various special cases which are online learning variants of classic computer science problems:

- Facility location problem: $z_i = 0$ $c_i^t = (1/T)c_i$ $r_i^t \geq 0$
- Knapsack median problem: $z_i \geq 0$ $c_i^t = 0$ $r_i^t \geq 0$
- 0/1 Knapsack problem: $z_i \geq 0$ $c_i^t \leq 0$ $r_i^t = 0$

- Historically service (server) placement problems are well studied in the **batch** case
- Presented an **online machine learning** framework for **placement** problems
- General framework models multiple combinatorial placement problems
- An efficient algorithm MaxHedge is given which has performance guarantees
- The time complexity of MaxHedge is only $\mathcal{O}(n \log n)$ per trial



Acknowledgment

"This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon."

