Competitive Analysis of Online Facility Open Problem

Extended Abstract

Binghan Wu
The University of Sydney
Sydney, Australia
biwu6051@uni.sydney.edu.au

Wei Bao
The University of Sydney
Sydney, Australia
wei.bao@sydney.edu.au

Bing Zhou
The University of Sydney
Sydney, Australia
bing.zhou@sydney.edu.au

ABSTRACT

We investigate an online cost minimization problem of serving requests in a tree of facilities, referred to as the Online Facility Open Problem (Online FOP). To address this problem, we propose the Anchor-Barrier Algorithm (ABA), a threshold-based algorithm applicable to any tree and any cost assignment, which can work in a distributed manner for scalability. We conduct the competitive analysis and show that ABA's achieves the optimal competitive ratio Height + 2, where Height is the height of the facility tree.

KEYWORDS
Online algorithms; Competitive analysis; Facility location problem

ACM Reference Format:

1 INTRODUCTION

We study an online cost minimization problem for a service system of facilities arranged in a tree topology. All nodes/facilities (used interchangeably) are initially closed and can be opened at any time by paying an opening cost. Each edge has an associated conveyance cost. Requests arrive at leaf nodes sequentially, and all requests must be served, with only open facilities able to do so. When an open facility receives a request, it serves the request without any additional cost. If a closed facility receives a request, it must forward the request to its parent, incurring the associated conveyance cost on the passing edge. If the parent is also closed, the request is recursively forwarded. If the request reaches the root and it is still closed, the request is sent to a faraway always-on facility (outside the tree) at a long-distance conveyance cost. In this paper, we consider a fully heterogeneous setting, where the opening costs/conveyance costs for different facilities/edges can be arbitrary non-negative numbers. The goal is to minimize the total cost without knowledge of whether and where the next request arrives. The challenge lies in deciding when, where and how many facilities to open so as to balance the open costs with the benefits of reduced conveyance costs. We refer to this problem as the Online Facility Open Problem (Online FOP).

2 PROBLEM FORMULATION

Our problem is formulated by a rooted tree \( G = (V, E) \), a conveyance cost vector \( \lambda \), an open cost vector \( \sigma \), a sequence of requests \( R \), where

- \( V = \{v_1, v_2, v_3, \ldots\} \) is the set of all nodes/facilities considered in the problem.
- W.l.o.g. \( v_1 \in V \) denotes the root of \( G \).
- \( E \) is a set of undirected edges.
- \( A \subseteq V \) is the set of all leaf nodes in \( G \). We use the term leaf nodes and access facilities interchangeably in this paper.

Initially, all facilities are off and can be opened by paying a one-shot open cost \( \sigma_i \), \( \forall v_i \in V \). Let \( R \) denote the sequence of requests arrive at access facilities. Only open facilities can serve requests, and no extra cost is charged. If a request arrives at a closed facility \( v_i \), \( v_j \) can only recursively forward this request to its parent by paying the conveyance cost until an open facility is reached. The conveyance cost of forwarding a request from \( v_i \) to its parent is \( \lambda_i \).

Define \( \rho(v) \) as the parent facility of \( v_i \). Define \( \mathcal{P}_{i,k} \) as the sequence of bottom-up connected facilities from \( v_j \) to \( v_k \), and define \( \mathcal{P}_i \) as the bottom-up sequence from \( v_i \) to the root. Define the height of the tree \( G \) as \( \text{Height}(G) = \max_{v'_i} |\mathcal{P}_i| - 1 \), \( \forall v_i \in A \), which is the number of edges from the leaf node to the root in the longest path.

Let \( OPT(G, \lambda, \sigma, R) \) be the total cost of the optimal offline algorithm that knows all requests \( R \) in advance given \( G \), \( \lambda \) and \( \sigma \). Let \( \beta_i \) be the indicator of facility \( v_i \) being open (\( \beta_i = 1 \)) or closed (\( \beta_i = 0 \)), and let \( \alpha_{i,t} \) be the indicator of facility \( v_i \) forwarding request \( r_t \) (\( \alpha_{i,t} = 1 \)) or not (\( \alpha_{i,t} = 0 \)) for all \( v_i \in V \) and \( r_t \in R \).
Algorithm 1 Anchor-Barrier Algorithm (ABA)

Require: \( G, \lambda, \sigma, R, \omega \)
1: \( \theta_0 \leftarrow 0, \forall v \in V \) initially
2: loop
3: Upon the arrival of a new request on \( v_j \)
4: referring\((v_j, \omega, v_j)\)
5: end loop
6: function referring\((v_j, \delta, v_j)\)
7: Input: current facility, residual budget, request source
8: Output: anchor/open facility, barrier, served
9: if \( v_i \) is open then
10: return \([v_i, 0, \text{True}]\)
11: else
12: \( \delta_{i,j} \leftarrow \min(\delta, \omega_i - \theta_i) \)
13: if \( \delta_{i,j} \geq \lambda_i \) then \( \rightarrow \) forward to the parent \( \rho(v_i) \)
14: \( v_a, b, s \leftarrow \text{referring}((\rho(v_i), \delta_{i,j} - \lambda_i, v_j)) \)
15: \( \theta_i \leftarrow \theta_i + \lambda_i + b \)
16: if \( v_s \) and \( \theta_i \geq \omega_i \) then
17: open \( v_i \)
18: return \([v_a, b, \text{True}]\)
19: else
20: return \([v_a, b, \text{False}]\)
21: end if
22: else \( \rightarrow \) touch barrier, request can not forward
23: \( \theta_i \leftarrow \theta_i + \delta_{i,j} \) \( \rightarrow \) find barrier \( b = \delta_{i,j} \)
24: if \( \theta_i \geq \omega_i \) then
25: open \( v_i \)
26: return \([v_i, \delta_{i,j}, \text{True}]\)
27: else
28: return \([v_i, \delta_{i,j}, \text{False}]\)
29: end if
30: end if
31: end if
32: end if
33: end function

The offline objective \( OPT \) is the solution to the following linear program:

\[
\min \sum_{i=1}^{\lfloor V \rfloor} \beta_i \sigma_i + \sum_{i=1}^{\lfloor V \rfloor} \sum_{r=1}^{\lfloor R \rfloor} \alpha_{i,r} \lambda_i \]

s.t. \( \alpha_{k,l} + \sum_{m \in P_{i,k}} \beta_m \geq 1, \)

\( \forall r \in R, \forall v \in P_{i,k}, \) where \( v_i \) = \( r \in R \).

Let \( ALG(G, \lambda, \sigma, R) \) be the total cost of the online algorithm which only knows the request upon its arrival. The competitive ratio is defined as the following

\[
R^* = \max_{G, \lambda, \sigma, R} \frac{ALG(G, \lambda, \sigma, R)}{OPT(G, \lambda, \sigma, R)}.
\]

Our goal is to design online algorithms that have the lowest \( R^* \), meaning that the online algorithm is as close to optimal as possible in the worst-case scenario.

3 ALGORITHM
ABA is a threshold based algorithm. In essence, each facility \( v_i \) holds an threshold \( \omega_i \). Whenever the total cost of forwarding the request \( \theta_i \) from a facility exceeds its threshold, then we open this facility. Upon the arrival of a request on \( v_j \), our online algorithm starts a distributed referring procedure at the request’s arrival facility \( v_j \) and recursively sends a reference message to its parent facility. The reference carries the residual budget \( \delta \) which is the maximum conveyance cost such that the cumulative reference cost of passed facilities does not exceed their threshold. When a facility receives a reference, it decides if it is an anchor facility using the residual budget, then there are 3 possible cases. (1) it is an open facility. This facility is the serving facility for the request, and returns the reference. (2) it is an anchor facility because \( \delta_{i,j} < \lambda_i \). Then it computes the location of the barrier, and returns the reference. The facility linking to the barrier will open for this request. (3) otherwise, it forwards the reference to its parent.

4 COMPETITIVE ANALYSIS
The performance of ABA is tied to the threshold. We have the following three Theorems to show the optimal threshold and the performance of ABA. Theorem 1 helps to show the upper bounds of the competitive ratio, and how the competitive ratio degrades as \( \gamma \) changes.

**Theorem 1.** Let \( \omega_i = \gamma \sigma_i, \forall v_i \in V, \gamma > 0 \). The competitive ratio of Algorithm 1 is bounded by \( R^* \leq \frac{1}{\gamma}(H_G + 2) \) if \( \gamma \leq 1 \), otherwise \( R^* \leq \gamma(H_G + 2) \).

Next, Theorem 2 shows that \( H_G + 2 \) is an optimal and reachable upper bound for a constructive problem instance.

**Theorem 2.** \( H_G + 2 \) is the optimal tight bound of the competitive ratio, and this bound is uniquely achieved by applying \( \gamma = 1 \).

Finally, we have theorem 3 show the optimality of \( H_G + 2 \).

**Theorem 3.** \( H_G + 2 \) is the best possible competitive ratio for any deterministic threshold design.

Given the three theorems, we have proved that ABA is \( H_G + 2 \)-competitive, and this competitive ratio is optimal.

5 CONCLUSION AND FUTURE WORK
In this paper, we study the Online FOP, which aims to minimize total cost by determining where and how much to invest in facilities. We address the Online FOP through ABA, which is a general framework that works on arbitrary tree topologies and cost settings. Moreover, ABA is distributed and scalable, making it suitable for large trees. We perform a competitive analysis and show that ABA is \( H_G + 2 \)-competitive, and this competitive ratio is optimal.

In future work, we propose to extend the scope to a more generalized setting where a currently open facility can be closed and a portion of the open costs can be refunded. Our goal is to capture the concept of divestment of non-performing assets in industrial scenarios. Furthermore, it is also interesting to explore scenarios where references can be shared between siblings, for example in edge collaborative caching, allowing direct communication between edge servers (access facilities).
REFERENCES


2560