New Results on Multi-Level Aggregation

Marcin Bienkowski[a,*], Martin Böhm[b,h,a], Jaroslaw Byrka[a], Marek Chrobak[d], Christoph Dürr[e], Lukáš Folwarczný[b,c], Łukasz Jeż[a], Jiří Sgall[b], Nguyen Kim Thang[f], Pavel Veselý[b,g]

[a] *Institute of Computer Science, University of Wrocław, Poland*
[b] *Computer Science Institute, Charles University, Czech Republic*
[c] *Institute of Mathematics, Czech Academy of Sciences, Czech Republic*
[d] *Department of Computer Science, University of California at Riverside, USA*
[e] *Sorbonne University, CNRS, Laboratoire d'informatique de Paris 6, LIP6, France*
[f] *IBISC, Univ. Evry. University Paris-Saclay, France*
[g] *Department of Computer Science, University of Warwick, Coventry, UK*
[h] *FB3: Mathematik/Informatik, University of Bremen, Germany*

Abstract

In the *Multi-Level Aggregation Problem* (MLAP), requests for service arrive at the nodes of an edge-weighted rooted tree $\mathcal{T}$. Each service is represented by a subtree X of $\mathcal{T}$ that contains its root. This subtree X serves all requests that are pending in the nodes of X, and the cost of this service is equal to the total weight of X. Each request also incurs a waiting cost between its arrival and service time. The objective is to minimize the total waiting cost of all requests plus the total cost of all service subtrees.

The currently best online algorithms for the MLAP achieve competitive ratios polynomial in the tree depth, while the best lower bound is only 3.618. In this paper, we report some progress towards closing this gap, by improving this lower bound and providing several tight bounds for restricted variants of MLAP: (1) We first study a Single-Phase variant of MLAP where all requests are released at the beginning and expire at some unknown time θ, for which we provide an online algorithm with optimal competitive ratio of 4. (2) We prove a lower bound of 4 on the competitive ratio for MLAP, even when the tree is a path. We complement this with a matching upper bound for the deadline variant of MLAP on paths.

Additionally, we provide two results for the offline case: (3) We prove that the Single-Phase variant can be solved optimally in polynomial time, and (4) we give a simple 2-approximation algorithm for offline MLAP with deadlines.

Keywords: algorithmic aspects of networks, online algorithms, scheduling and resource allocation
2000 MSC: 68W01, 68W05, 68W40, 68Q25

1. Introduction

In the *Multi-Level Aggregation Problem* (MLAP) introduced (under a different name) by Brito et al. [1], a tree $\mathcal{T}$ with positive weights assigned to edges is given, and requests from a set $\mathcal{R}$ arrive in the nodes

*Corresponding author, email: marcin.bienkowski@cs.uni.wroc.pl, tel.: +48 71 375 7838, fax: +48 71 375 7801

of $\mathcal{T}$ over time. These requests are served by subtrees rooted at the root of $\mathcal{T}$. Such a subtree X serves all requests pending at the nodes of X at cost equal to the total weight of X. Each request incurs a waiting cost, defined by a non-negative and non-decreasing function of time, which may be different for each request. The objective is to minimize the sum of the total service and waiting costs.

The MLAP model is designed to capture the dilemma of decision makers dealing with tasks that arrive over time. Performing these tasks in batches can reduce overhead associated with setup, retooling, transportation, etc. On the other hand, delaying each task's execution also incurs some cost, for example the cost of storage. In order to be processed in a batch, tasks that form a batch need to be compatible, and the MLAP model focuses specifically on tasks where this compatibility relation forms a hierarchy that can be represented by a tree. We mention here three practical scenarios that can be modeled by MLAP:

- Sensor networks periodically perform data gathering, where data collected in the sensors need to be transmitted to the main station along links forming a spanning tree (see, e.g., [2]). Sensors are battery operated, so reducing energy expenditure required for transmissions is critical. This can be accomplished by having each intermediate node in this tree gather the messages from its children and forward them all together, compressed into a single message, to its parent.

- In supply-chain management, retailers place orders for goods that need to be shipped from the factory in a timely manner. The underlying transportation network forms a tree, with the retailers in its leaves, the factory in the root, and with its intermediate nodes representing warehouses where goods can be temporarily stored while in transit. Goods that realize orders from some retailers can be shipped together from the factory and then redistributed in the intermediate nodes along the path to their destination (see [3], for example).

- Another example involves an organization hierarchy with workers at the leaves, managers at intermediate nodes, and the executive at the root (see [4] and the references therein). Here, the objective is to efficiently manage decision making and information flow along this tree.

Some earlier papers studied particular variants of waiting costs. In the MLAP-L variant, each waiting cost function is linear, that is, it is assumed to be simply the delay between the times when a request arrives and when it is served. In the MLAP-D variant (called the *deadline variant*), each request is given a certain deadline, has to be served before or at its deadline, and there is no penalty associated with waiting. This can be modeled by the waiting cost function that is 0 up to the deadline and $+\infty$ afterwards.

In this paper, we mostly focus on the online version of MLAP, where an algorithm needs to produce a schedule in response to requests that arrive over time. When a request appears, its waiting cost function is also revealed, but the algorithm has no information about the requests that arrive in the future. At each time t, the online algorithm needs to decide whether to generate a service tree at this time, and if it does,

which nodes should be included in this tree. We use the competitive ratio as our performance measure for online algorithms, that is the ratio between the total cost of an online algorithm and the cost of an optimal offline solution, maximized over all possible instances. (See Section 2 for formal definitions.)

1.1. Previous Work on Online Algorithms

The first competitive solution for arbitrary trees of depth D was given by Bienkowski et al. [5, 6]; the competitive ratio of their algorithm is $O(D^4 \cdot 2^D)$. They also gave a slightly improved, $O(D^2 \cdot 2^D)$-competitive algorithm for the MLAP-D variant. The result for the deadline variant was subsequently improved to $O(D)$ by Buchbinder et al. [7]. Azar et al.studied a more general problem called Online Service with Delays and gave $O(D^2)$-competitive algorithm for MLAP [8, 9]. The best known lower bound is 3.618 [10], and for arbitrary trees the existence of an algorithm with competitive ratio independent of tree depth remains open.

Better algorithms are known for specific tree topologies. When the tree comprises a single edge connecting the leaf to the root, MLAP is equivalent to the TCP Acknowledgement Problem (known also as the Lot Sizing Problem in the operations research community). In this variant, the decisions involve only choosing appropriate times at which the requests pending at the leaf are served. The complexity of this case is fully resolved: the optimal competitive ratio is 2 in the deterministic case [11] and $\mathrm{e}/(\mathrm{e}-1) \approx 1.582$ in the randomized case [12, 13].

When the tree has two levels, MLAP becomes equivalent to the extensively studied Joint Replenishment Problem. The currently best 3-competitive primal-dual algorithm was given by Buchbinder et al. [3] and the currently best lower bound of 2.754 is due to Bienkowski et al. [14]. The optimal competitive ratio for the deadline variant is 2 [14].

Another special case is when the tree is a path (has no branches). For this variant, Brito et al. [1] gave an 8-competitive algorithm. This result was improved by Bienkowski et al. [10] who showed that the competitive ratio of this problem is between 3.618 and 5.

For a gentle introduction to aggregation problems and techniques used for designing online algorithms, we refer the reader to the survey by Chrobak [15].

1.2. Previous Work on Offline Algorithms

When the tree is a single edge, the offline variant of MLAP can be solved in time $O(n \log n)$, where n is the number of requests [16]. The case of paths can also be solved in polynomial time [10]. However, when the tree has branches then already for two-level trees the problem becomes NP-hard [17] and even APX-hard [18, 19]. The currently best approximation, due to Bienkowski et al. [14], achieves a factor of 1.791, improving on earlier work by Levi et al. [20, 21, 22]. A better approximation ratio of 1.574 can be obtained in the deadline variant [19].

	MLAP and MLAP-L		MLAP-D	
	upper	lower	upper	lower
depth 1	2 [11]	2 [11]	1	1
rand. alg. for depth 1	$e/(e-1) \approx 1.582$ [12]	$e/(e-1) \approx 1.582$ [13]	1	1
depth 2	3 [3]	2.754 [14]	2 [14]	2 [14]
fixed depth $D \geq 2$	$O(D^2)$ [9]	2.754	$O(D)$ [7]	2
paths of arbitrary depth	5 [10]	3.618 [10], **4**	**4**	**4**

Table 1: Previous and current bounds on the competitive ratios for MLAP for trees of various depths. Ratios written in bold are shown in this paper. Unreferenced results are either immediate consequences of other entries in the table or trivial observations. Some of the results were originally stated for the MLAP-L variant, but can be extended in a straightforward way to the general MLAP.

For arbitrary depth trees, the problem becomes NP-hard already for the deadline variant [23]. For this case, Becchetti et al. [23] gave a 2-approximation algorithm. For general waiting cost functions, Pedrosa [24] showed, adapting an algorithm of Levi et al. for the multi-stage assembly problem [25], that there is a $(2+\varepsilon)$-approximation for MLAP where ε can be made arbitrarily small.

1.3. Our Contributions

In Section 3, we study a version of MLAP that we refer to as *Single-Phase* MLAP (or 1P-MLAP), in which all requests arrive at the beginning, but they also have a common *expiration time* that we denote by θ. Any request not served by time θ pays waiting cost at time θ and does not need to be served anymore. In spite of the expiration-date feature, it can be shown that 1P-MLAP can be represented as a special case of MLAP. 1P-MLAP is a crucial tool in all lower bound proofs in the literature for competitive ratios of MLAP, including those in [3, 10, 14], as well as in our lower bounds in Section 4. It also has a natural interpretation in the context of supply-chain management if we allow all orders to be canceled, say, due to changed market circumstances.

In the online variant of 1P-MLAP all requests are known at the beginning, but the expiration time θ is unknown. For this version, we give an online algorithm with competitive ratio 4. Since 1P-MLAP can be expressed as a special case of MLAP, our result implies that the techniques from [3, 10, 14] cannot be used to prove a lower bound larger than 4 on the competitive ratio for MLAP, and any study of the dependence of the competitive ratio on the depth D will require new insights and techniques.

In Section 4 we consider MLAP on paths. For this case, we give a 4-competitive algorithm for MLAP-D and we provide a matching lower bound. We show that the lower bound of 4 applies to MLAP-L as well, improving the previous lower bound of 3.618 from [10]. A summary of old and new results on online algorithms for various tree depths is given in Table 1.

In addition, we provide two results on offline algorithms (for arbitrary trees). In Section 5, we provide a 2-approximation algorithm for MLAP-D, significantly simpler than the LP-rounding algorithm by Becchetti et al. [23] with the same ratio. In Section 3.3, we give a polynomial time algorithm that computes optimal solutions for 1P-MLAP.

2. Preliminaries

Weighted trees. Let $\mathcal{T}$ be a tree with root r. For any set of nodes $Z \subseteq \mathcal{T}$ and a node x, set Z_x contains x and all its descendants in Z; in particular, $\mathcal{T}_x$ is the *induced subtree* of $\mathcal{T}$ rooted at x. The parent of a node x is denoted $parent(x)$.

For a node $x \neq r$, by ℓ_x or $\ell(x)$ we denote the positive weight of the edge connecting node x to its parent. For the sake of convenience, we often refer to ℓ_x as the weight of node x. We extend this notation to r by setting $\ell_r = 0$. If Z is any set of nodes of $\mathcal{T}$, then the weight of Z is $\ell(Z) = \sum_{x \in Z} \ell_x$.

Definition of MLAP. A *request* ρ is specified by a triple $\rho = (\sigma_\rho, a_\rho, \omega_\rho)$, where σ_ρ is the node of $\mathcal{T}$ at which ρ is issued, a_ρ is the non-negative *arrival time* of ρ, and ω_ρ is the waiting cost function of ρ. We assume that $\omega_\rho(t) = 0$ for $t \leq a_\rho$ and $\omega_\rho(t)$ is non-decreasing for $t \geq a_\rho$. MLAP-L is the variant of MLAP with linear waiting costs; that is, for each request ρ we have $\omega_\rho(t) = t - a_\rho$, for $t \geq a_\rho$. In MLAP-D, the variant with deadlines, we have $\omega_\rho(t) = 0$ for $t \leq d_\rho$ and $\omega_\rho(t) = \infty$ for $t > d_\rho$, where d_ρ is called the *deadline* of request ρ.

A *service* is a pair (X, t), where X is a subtree of $\mathcal{T}$ rooted at r (an arbitrary connected subset of vertices containing r) and t is the time of this service. We occasionally refer to X as the service tree (or just service) at time t, or even omit t altogether if it is understood from context.

An instance $\mathcal{J} = \langle \mathcal{T}, \mathcal{R} \rangle$ of the *Multi-Level Aggregation Problem* (MLAP) consists of a weighted tree $\mathcal{T}$ and a set $\mathcal{R}$ of requests arriving at the nodes of $\mathcal{T}$. A *schedule* is a set S of services. For a request ρ, let (X, t) be the service in S with minimal t such that $\sigma_\rho \in X$ and $t \geq a_\rho$. We then say that (X, t) *serves* ρ and the *waiting cost* of ρ in S is defined as $\mathsf{wcost}(\rho, \mathsf{S}) = \omega_\rho(t)$. Furthermore, request ρ is called *pending* at all times in the interval $[a_\rho, t]$. Schedule S is called *feasible* if all requests in $\mathcal{R}$ are served by S.

The cost of a feasible schedule S, denoted $\mathsf{cost}(\mathsf{S})$, is defined by

$$\mathsf{cost}(\mathsf{S}) = \mathsf{scost}(\mathsf{S}) + \mathsf{wcost}(\mathsf{S}),$$

where $\mathsf{scost}(\mathsf{S})$ is the total service cost and $\mathsf{wcost}(\mathsf{S})$ is the total waiting cost, that is

$$\mathsf{scost}(\mathsf{S}) = \sum_{(X,t) \in \mathsf{S}} \ell(X) \quad \text{and} \quad \mathsf{wcost}(\mathsf{S}) = \sum_{\rho \in \mathcal{R}} \mathsf{wcost}(\rho, \mathsf{S}).$$

The objective of MLAP is to compute a feasible schedule S for $\mathcal{J}$ with minimum $\mathsf{cost}(\mathsf{S})$. An example instance with a feasible schedule is depicted in Figure 1.

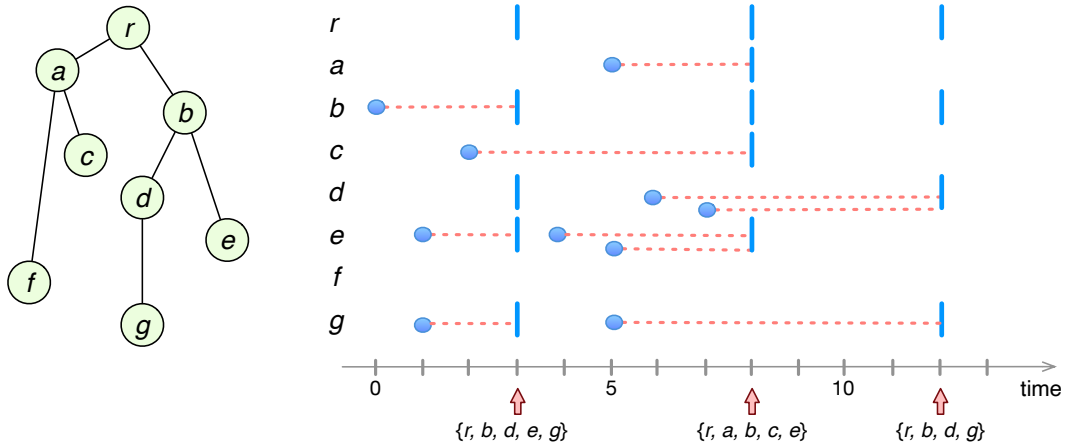

Figure 1: An example instance of MLAP and a feasible schedule. Blue dots denote arrivals of requests at given tree nodes. The depicted schedule consists of three services: $(\{r, b, d, e, g\}, 3)$, $(\{r, a, b, c, e\}, 8)$ and $(\{r, b, d, g\}, 12)$; the affected nodes are marked with blue vertical lines. In particular, the second service serves four requests: one at a, one at c and two at e, and the associated service cost is $\ell(\{r, a, b, c, e\})$. The waiting periods of requests are shown as red dashed lines. For example, the only request at node c, specified by triple $\rho = (c, 2, \omega_\rho)$, is served by the second service (at time 8). Hence, the associated waiting cost is equal to $\omega_\rho(8)$, which would be equal to $\omega_\rho(8) = 8 - 2 = 6$ for linear waiting costs.

Online algorithms. We assume the continuous time model. The computation starts at time 0 and from then on the time gradually progresses. At any time t new requests can arrive. If the current time is t, the algorithm has complete information about the requests that arrived up until time t, but has no information about any requests whose arrival times are after time t. For an online algorithm $\mathcal{A}$, we say that $\mathcal{A}$ is R-*competitive*[1] if $\mathsf{cost}(\mathsf{S}) \leq R \cdot \mathsf{opt}(\mathcal{J})$ for any instance $\mathcal{J}$ of MLAP, where S is the schedule computed by $\mathcal{A}$ on $\mathcal{J}$ and $\mathsf{opt}(\mathcal{J})$ is the optimum cost for $\mathcal{J}$.

3. Single-Phase Variant

We now consider a restricted variant of MLAP that we refer to as *Single-Phase* MLAP, or 1P-MLAP. In 1P-MLAP all requests arrive at the beginning, at time 0. The instance also includes an *expiration time* θ, common for all requests. We do not require that all requests are served. Any unserved request pays only the cost of waiting until θ.

In the online variant of 1P-MLAP, all requests, including their waiting cost functions, are known to the online algorithm at time 0. The only unknown is the expiration time θ.

Although not explicitly named, variants of 1P-MLAP have been considered in [3, 10, 14], where they were used to show lower bounds on competitive ratios for MLAP. These proofs consist of two steps, first

[1] Definitions of competitiveness in the literature often allow an additive error term, independent of the request sequence. For our algorithms, this additive term is not needed. Our lower bound proofs can be easily modified (essentially, by iterating the adversary strategy) to remain valid if an additive term is allowed, even if it is a function of $\mathcal{T}$.

showing a lower bound for online 1P-MLAP and then arguing that, in the online scenario, 1P-MLAP can be expressed as a special case of MLAP. (The corresponding property holds trivially in the offline case as well.) We use the same general approach in Section 4 to show our lower bounds.

To see that (in spite of the expiration feature) 1P-MLAP can be thought of as a special case of MLAP, we map an instance $\mathcal{J}$ of 1P-MLAP into the instance $\mathcal{J}'$ of MLAP with the property that any R-competitive algorithm for $\mathcal{J}'$ can be converted into an R-competitive algorithm for $\mathcal{J}$. We will explain the general idea when the cost function is linear; the construction for arbitrary cost functions is based on the same idea, but it involves some minor technical obstacles. Let θ be the expiration time from $\mathcal{J}$. Choose some large integers K and M. The constructed instance $\mathcal{J}'$ consists of K "nested" and "compressed" copies of $\mathcal{J}$, that we also refer to as *phases*. In the i-th phase we multiply the waiting cost function of each node by M^i. We let this phase start at time $(1 - M^{-i})\theta$ (that is, at this time the requests from this phase are released) and end at time θ. Thus the length of phase i is $M^{-i}\theta$. The main trick is that, in $\mathcal{J}'$, at time θ an optimal algorithm can serve all pending requests (from all phases) at the cost that is independent of K, so the contribution of this service cost to the cost of each phase is negligibly small. Following this idea, any R-competitive algorithm for $\mathcal{J}'$ can be converted into an R-competitive algorithm for $\mathcal{J}$, except for some vanishing additive constant. (See [3, 10, 14] for more details.)

3.1. Characterizing Optimal Solutions

Let θ be the expiration value. Then the optimal solution is to serve some subtree X (rooted at r) already at time 0 and wait until time θ with the remaining requests contained in $\overline{X} = \mathcal{T} - X$. So now we consider schedules that consist only of one service subtree $X \subseteq \mathcal{T}$ at time 0. The cost of such schedule (that we identify with X itself) is

$$\mathsf{cost}(X, \theta) = \ell(X) + \omega(\overline{X}, \theta),$$

where, for any set $U \subseteq \mathcal{T}$ and time t, we use $\omega(U, t) = \sum_\rho \omega_\rho(U, t)$ to denote the waiting cost of all requests in U at time t.

Our first objective is to characterize those subtrees X that are optimal for expiration time t. This characterization will play a critical role in our online algorithm for 1P-MLAP, provided later in this section and it also leads to an offline polynomial-time algorithm for computing optimal solutions, given in Section 3.3.

The lemma below can be derived by expressing 1P-MLAP as a linear program and using strong duality. We provide instead a simple combinatorial proof. For each subtree Z of $\mathcal{T}$, we denote its root by r_Z. (Also, recall that Z_v is the induced subtree of Z rooted at v, that is, Z_v contains all descendants of v in Z.)

Lemma 1. *A service X is optimal for an expiration time t if and only if it satisfies the following two conditions:*

(a) $\omega(X_v, t) \geq \ell(X_v)$ *for each* $v \in X$, *and*

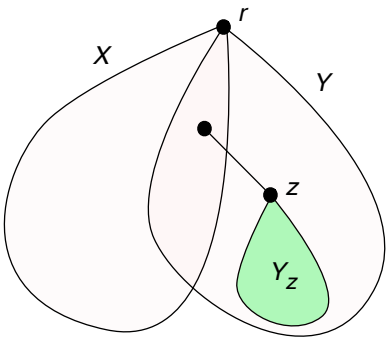

Figure 2: Illustration of the sufficiency proof for Lemma 1.

(b) $\omega(Z,t) \leq \ell(Z)$ *for each subtree* Z*, disjoint with* X*, such that* $\mathit{parent}(r_Z) \in X$*.*

Proof. ($\Rightarrow$) We begin by proving that (a) and (b) are necessary conditions for optimality of X.

(a) Suppose that there is a $v \in X$ for which $\omega(X_v,t) < \ell(X_v)$. Let $Y = X - X_v$. Then Y is a service tree (empty if $v = r$), and then

$$\begin{aligned}\mathsf{cost}(Y,t) &= \ell(Y) + \omega(\overline{Y},t)\\ &= \ell(X) - \ell(X_v) + \omega(\overline{X},t) + \omega(X_v,t)\\ &< \ell(X) + \omega(\overline{X},t) = \mathsf{cost}(X,t),\end{aligned}$$

contradicting the optimality of X.

(b) Suppose that there is a subtree Z that violates condition (b), that is $Z \cap X = \emptyset$, $\mathit{parent}(r_Z) \in X$, but $\omega(Z,t) > \ell(Z)$. Let $Y = X \cup Z$. Then Y is a service tree and

$$\begin{aligned}\mathsf{cost}(Y,t) &= \ell(Y) + \omega(\overline{Y},t)\\ &= \ell(X) + \ell(Z) + \omega(\overline{X},t) - \omega(Z,t)\\ &< \ell(X) + \omega(\overline{X},t) = \mathsf{cost}(X,t),\end{aligned}$$

contradicting the optimality of X.

($\Leftarrow$) We now prove sufficiency of conditions (a) and (b). Suppose that X satisfies (a) and (b), and let Y be any other service subtree of $\mathcal{T}$. From (b), for any node $z \in Y - X$ with $\mathit{parent}(z) \in X \cap Y$ we have $\omega(Y_z,t) \leq \ell(Y_z)$. Since both X and Y are rooted at r, any node in $Y - X$ is in some induced subtree Y_z, for some z such that $\mathit{parent}(z) \in X \cap Y$ (see Figure 2). This implies that $\omega(Y - X,t) \leq \ell(Y - X)$. Similarly, from (a), for any node $v \in X - Y$ with $\mathit{parent}(v) \in X \cap Y$ we have $\omega(X_v,t) \geq \ell(X_v)$. This implies that

$\omega(X - Y, t) \geq \ell(X - Y)$. These inequalities yield

$$\begin{aligned}\mathsf{cost}(Y,t) &= \ell(Y) + \omega(\overline{Y}, t) \\ &= \ell(X) + \omega(\overline{X}, t) + [\,\ell(Y - X) - \omega(Y - X, t)\,] - [\,\ell(X - Y) - \omega(X - Y, t)\,] \\ &\geq \mathsf{cost}(X,t),\end{aligned}$$

proving the optimality of X. □

For a time t, a subtree Z of $\mathcal{T}$ (not necessarily rooted at r) is called *t-mature* if $\omega(Z,t) \geq \ell(Z)$. We say that Z is *t-covered* if each induced subtree Z_x, for $x \neq r_Z$, is t-mature. (Note that in this definition Z itself is not required to be t-mature.) We now make two observations. First, if Z is t-covered, then each induced subtree Z_v of Z is t-covered as well. Second, if $Z = \{r_Z\}$, that is if Z consists of only one node, then Z is vacuously t-covered; thus any subtree Z of $\mathcal{T}$ has a t-covered subtree rooted at r_Z (which may not be an induced subtree).

Lemma 2. *If X and Y are t-covered service subtrees of $\mathcal{T}$ then the service subtree $X \cup Y$ is also t-covered.*

Proof. If $X = Y$ the lemma is trivial, so assume $X \neq Y$. Choose any $z \in (X - Y) \cup (Y - X)$ with $\mathit{parent}(z) \in X \cap Y$. Without loss of generality, we can assume that $z \in X - Y$. As $z \neq r$, subtree X_z is t-mature, and by its definition, X_z is disjoint with Y.

Take $Q = Y \cup X_z$. Set Q is a service subtree of $\mathcal{T}$. We claim that Q is t-covered. To justify this claim, we choose any $v \in Q - \{r\}$. Node v might be either in X_z, or in Y on the path from r to z, or in Y but not on this path. We consider these three cases below.

- If $v \in X_z = Q_z$, then Q_v is t-mature because $Q_v = X_v$.
- If $v \in Y$ and $z \notin Q_v$, then Q_v is t-mature because $Q_v = Y_v$.
- If $v \in Y$ and $z \in Q_v$, then $\omega(Q_v, t) = \omega(Y_v, t) + \omega(X_z, t) \geq \ell(Y_v) + \ell(X_z) = \ell(Q_v)$, so Q_v is t-mature in this case as well.

Thus indeed Q is t-covered, as claimed.

We can now update Y by setting $Y = Q$ and applying the above argument again. By repeating this process, we end up with $X = Y$, completing the proof. □

Choose O^t to be the inclusion-maximal t-covered service subtree of $\mathcal{T}$ (that is, a subtree rooted at r). By Lemma 2, O^t is well-defined and unique. Also, from Lemma 1 we obtain that O^t is optimal for expiration time t. Thus, the optimal cost for expiration time t is equal to

$$\mathsf{opt}(t) = \mathsf{cost}(O^t, t) = \ell(O^t) + \omega(\overline{O}^t, t).$$

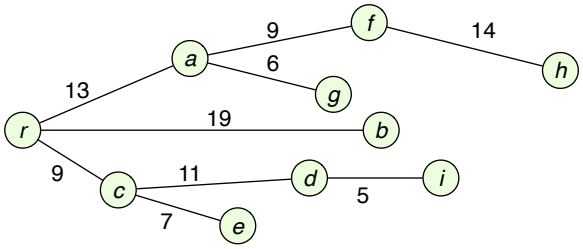

Figure 3: An example of an instance of 1P-MLAP, with one request at each node issued at time 0 and linear waiting cost function. For $\theta = 7$, the optimal service subtree is $O^7 = \{r\}$, and its cost is $\ell(O^7) + \omega(\overline{O}^7, \theta) = 0 + 9 \cdot \theta = 63$. For $\theta = 8$, the optimal service subtree is $O^8 = \{r, c, d, e, i\}$, and its cost is $\ell(O^8) + \omega(\overline{O}^8, \theta) = 32 + 5 \cdot \theta = 72$. For $\theta = 10$, the optimal service subtree is $O^{10} = \{r, c, d, e, i, a, g, f\}$, i.e., $O^{10} = \mathcal{T} \setminus \{b, h\}$, and its cost is $\ell(O^{10}) + \omega(\overline{O}^{10}, \theta) = 60 + 2 \cdot \theta = 80$. For $\theta = 19$, the optimal service subtree O^{19} is the whole tree $\mathcal{T}$, and its cost is $\ell(O^{19}) + \omega(\overline{O}^{19}, \theta) = \ell(\mathcal{T}) + 0 \cdot \theta = 93$.

An example of optimal service subtrees is given in Figure 3.

Assume that a subtree Z is t-mature and $t \leq t'$. As the waiting costs are non-decreasing, Z is t'-mature as well. This implies the following corollary.

Corollary 3. *For every $t \leq t'$, it holds that $O^t \subseteq O^{t'}$.*

3.2. An Online Competitive Algorithm

In our algorithm, we assume that all waiting cost functions are continuous with respect to t. This is only for technical convenience as the reduction from [6] implies that our result can be extended to right-continuous waiting functions, deadlines and the discrete-time model.

Without loss of generality, we can assume that $\min_{v \in \mathcal{T} - \{r\}} \ell_v > 1$; otherwise the distances together with the waiting costs can be rescaled to satisfy this property.

> **Algorithm** OnlDoubling: For any integer $i \geq 0$, define $t_i := \sup\{t \geq 0 \mid \mathsf{opt}(t) \leq 2^i\}$. At each time $t \geq 0$, if $t = t_i$, serve $O^{t_{i+1}}$, otherwise do nothing.

Notice that in any fixed instance $\mathsf{opt}(t)$ is bounded (it never exceeds $\ell(\mathcal{T})$) and continuous in t, so there exists a maximum i_0 such that for all $i \leq i_0$ we have $\mathsf{opt}(t_i) = 2^i$ whereas for all $i > i_0$ it holds that $t_i = \infty$ and $O^{t_i} = O^\infty$ serves all the requests in the instance. Thus, regardless of θ, the algorithm never makes a service after time t_{i_0}. Additionally, we have $\mathsf{opt}(\infty) \leq 2 \cdot \mathsf{opt}(t_{i_0})$.

Algorithm OnlDoubling is in essence a doubling algorithm [26]. However, although obtaining *some* constant ratio using doubling is not difficult, the formulation that achieves the optimal factor of 4 relies critically on the structure of optimal solutions that we elucidated earlier in this section. For example, note that the sequence of service costs of the algorithm does not necessarily grow exponentially.

Theorem 4. OnlDoubling *is 4-competitive for the Single-Phase MLAP.*

Proof. By our assumption that $\min_{v\in\mathcal{T}-\{r\}} \ell_v > 1$, we have $O^{t_0} = \{r\}$; that is, if $\theta \le t_0$, the optimum solution does not make any services and only pays the waiting cost. The definition of t_0 implies that $\omega(O^{t_0}, t_0) \le 1$.

We now estimate the cost of Algorithm OnlDoubling, for a given expiration time θ. Suppose first that $\theta = t_k < \infty$ (thus, $k \le i_0$), i.e., the expiration is right after the algorithm's service at time t_k. The total service cost of the algorithm is then $\sum_{i=0}^{k} \ell(O^{t_{i+1}})$. To estimate the waiting cost, consider some node v. If $v \in O^{t_0}$, then the waiting cost of v is $\omega(v, t_0)$. If $v \in O^{t_{i+1}} - O^{t_i}$, for some $i = 0, ..., k$, then the waiting cost of v is $\omega(v, t_i)$. If $v \notin O^{t_{k+1}}$, then the waiting cost of v is $\omega(v, \theta) = \omega(v, t_k)$. Thus, the total cost of OnlDoubling is

$$\begin{aligned}
\textsc{OnlDoubling}(t_k) &= \sum_{i=0}^{k} \ell(O^{t_{i+1}}) + \omega(O^{t_0}, t_0) + \sum_{i=0}^{k-1} \omega(O^{t_{i+1}} - O^{t_i}, t_i) + \omega(O^{t_{k+1}} - O^{t_k}, t_k) + \omega(\overline{O}^{t_{k+1}}, t_k) \\
&\le \sum_{i=0}^{k} \ell(O^{t_{i+1}}) + \omega(O^{t_0}, t_0) + \sum_{i=0}^{k-1} \omega(\overline{O}^{t_i}, t_i) + \omega(\overline{O}^{t_k}, t_k) \\
&= \sum_{i=0}^{k+1} \big[\, \ell(O^{t_i}) + \omega(\overline{O}^{t_i}, t_i) \,\big] + \omega(O^{t_0}, t_0) \\
&\le \sum_{i=0}^{k+1} \mathsf{opt}(t_i) + 1 \le \sum_{i=0}^{k+1} 2^i + 1 = 2^{k+2} = 4 \cdot \mathsf{opt}(t_k),
\end{aligned}$$

where in the last line we use $k \le i_0$, which implies $\mathsf{opt}(t_i) = 2^i$ for all $i \le k$ and $\mathsf{opt}(t_{k+1}) \le 2^{k+1}$.

Next, suppose that θ is between two service times, say $t_k \le \theta < t_{k+1}$, in which case again $k \le i_0$. The optimality of O^{t_k} at time t_k implies that $\mathsf{opt}(t_k) = \ell(O^{t_k}) + \omega(\overline{O}^{t_k}, t_k) \le \ell(O^{\theta}) + \omega(\overline{O}^{\theta}, t_k)$. By Corollary 3, we have $O^{t_k} \subseteq O^{\theta} \subseteq O^{t_{k+1}}$. Then, the increase of the optimum cost from time t_k to time θ can be estimated as

$$\begin{aligned}
\mathsf{opt}(\theta) - \mathsf{opt}(t_k) &\ge \big[\, \ell(O^{\theta}) + \omega(\overline{O}^{\theta}, \theta) \,\big] - \big[\, \ell(O^{\theta}) + \omega(\overline{O}^{\theta}, t_k) \,\big] \\
&= \omega(\overline{O}^{\theta}, \theta) - \omega(\overline{O}^{\theta}, t_k) \ge \omega(\overline{O}^{t_{k+1}}, \theta) - \omega(\overline{O}^{t_{k+1}}, t_k),
\end{aligned}$$

where the last expression is the increase in Algorithm OnlDoubling's cost from time t_k to time θ. This implies that the ratio at expiration time θ cannot be larger than the ratio at expiration time t_k.

The final case is when $0 \le \theta < t_0$. Then $\mathsf{opt}(\theta) < 1$. By our assumption, all weights are greater than 1, and this implies $\mathsf{opt}(\theta) = \omega(\mathcal{T}, \theta) = \textsc{OnlDoubling}(\theta)$. □

3.3. An Offline Linear-Time Algorithm

The offline algorithm for computing the optimal solutions is based on the above-established properties of optimal sets O^t.

Theorem 5. *The optimal offline solution to 1P-MLAP can be computed in linear time.*

Proof. Our algorithm proceeds bottom up, starting at the leaves, and pruning out subtrees that are not t-covered. The pseudo-code of our algorithm is shown below.

Algorithm 1 CovSubT (v, t)

$A_v \leftarrow \{v\}$
$\delta_v \leftarrow \omega(v, t)$
for each child u of v **do**
  $(A_u, \delta_u) \leftarrow$ CovSubT(u, t)
  if $\delta_u \geq \ell_u$ **then**
    $A_v \leftarrow A_v \cup A_u$
    $\delta_v \leftarrow \delta_v + \delta_u - \ell_u$
return (A_v, δ_v)

For each node v the algorithm outputs a pair (A_v, δ_v), where A_v denotes the maximal (equivalently w.r.t. inclusion or cardinality) t-covered subtree of $\mathcal{T}$ rooted at v, and $\delta_v = \omega(A_v, t) - \ell(A_v - \{v\})$. That is, δ_v is the "surplus" waiting cost of A_v at time t. (Note that we do not account for ℓ_v in this formula.) To compute O^t, the algorithm computes $(A_r, \delta_r) =$ CovSubT(r, t) and returns A_r.

By a routine argument, the running time of Algorithm CovSubT is $O(N)$, where N is the size of the instance (that is, the number of nodes in $\mathcal{T}$ plus the number of requests). Here, we assume that the values $\omega(v, t)$ can be computed in time proportional to the number of requests in v. □

4. MLAP on Paths

We now consider the case when the tree is just a path. For simplicity, we assume a generalization to the continuous case, that we refer to as *the MLAP problem on the line*, when the path is represented by the half-line $[0, \infty)$ and the requests can occur at any point $x \in [0, \infty)$. The point 0 corresponds to the root, each point $x \in [0, \infty)$ is a node, and each service is an interval of the form $[0, x]$. We say that an algorithm *delivers from* x if it serves the interval $[0, x]$.

4.1. Optimal Solution for MLAP-D on Paths

We first prove that the competitive ratio of MLAP-D (the variant with deadlines) on the line is at most 4. By the lower bound we present next, this bound is optimal.

> **Algorithm** OnlLine: Create a service only when a deadline of a pending request is reached. If a deadline of a request at x is reached, deliver from $2x$.

Theorem 6. *Algorithm* OnlLine *is 4-competitive for MLAP-D on the line.*

Proof. The proof uses a charging strategy. We represent each adversary's service, say when the adversary delivers from a point y, by an interval $[0, y]$. The cost of each service of OnlLine is then charged to a segment of one of those adversary's service intervals.

Consider a service triggered by a deadline t of a request ρ at some point x. When serving ρ, OnlLine delivered from $2x$. Fix the last service of the adversary delivered from a point $x' \geq x$ at a time $t' \leq t$. (Such service exists, because the adversary must have served ρ between its arrival time and its deadline t.) We charge the cost $2x$ of the algorithm's service to the segment $[x/2, x]$ of the adversary's service interval $[0, x']$ at time t'.

We now claim that no part of the adversary's service is charged twice. To justify this claim, suppose that there are two services of OnlLine, at times $t_1 < t_2$, triggered by requests from points x_1 and x_2, respectively, that both charge to an adversary's service from x' at time $t' \leq t_1$. By the definition of charging, the request at x_2 is serviced by the adversary no later than at time t', so its arrival time is no larger than t'. As x_2 was not served by OnlLine's service at t_1, it means that $x_2 > 2x_1$, and thus the charged segments $[x_1/2, x_1]$ and $[x_2/2, x_2]$ of the adversary's service interval at time t' are disjoint.

Summarizing, for any adversary's service interval $[0, y]$, its charged segments are disjoint. Any charged segment receives the charge equal to 4 times its length. Thus this interval receives the total charge at most $4y$. This implies that the competitive ratio is at most 4. □

4.2. Lower Bounds

We now show lower bounds of 4 for MLAP-D and MLAP-L on the line. In both proofs we show the bound for the corresponding variant of 1P-MLAP. For the latter, we use a reduction from the online bidding problem [26, 27]. Roughly speaking, in online bidding, for a given universe $\mathcal{U}$ of real numbers, the adversary chooses a secret value $u \in \mathcal{U}$ and the goal of the algorithm is to find an upper-bound on u. To this end, the algorithm outputs an increasing sequence of numbers $x_1, x_2, x_3, \ldots$. The game is stopped after the first x_k that is at least u and the bidding ratio is then defined as $\sum_{i=1}^{k} x_i/u$.

Chrobak et al. [27] proved that the optimal bidding ratio is exactly 4, even if it is restricted to sets $\mathcal{U}$ of the form $\{1, 2, \ldots, B\}$, for some integer B. More precisely, they proved the following result.

Lemma 7. *[27] For any $R < 4$, there exists an integer $B > 0$, such that any sequence of integers $0 = x_0 < x_1 < x_2 < \ldots < x_{m-1} < x_m = B$ has an index $k \geq 1$ with $\sum_{i=0}^{k} x_i > R \cdot (x_{k-1} + 1)$.*

Theorem 8. *There is no online algorithm for MLAP-D on the line with competitive ratio smaller than* 4.

Proof. We show that no online algorithm for 1P-MLAP-D (the deadline variant of 1P-MLAP) on the line can attain competitive ratio smaller than 4. Suppose the contrary, i.e., that there exists an R-competitive deterministic algorithm Alg with $R < 4$. Let B be the integer whose existence is guaranteed by Lemma 7.

We create an instance of 1P-MLAP-D, where, at time 0, for every $x \in \{1, \ldots, B\}$ there is a request at x with deadline x.

Without loss of generality, ALG issues services only at integer times $1, 2, ..., B$. The strategy of ALG can be now defined as a sequence of services at times $t_1 < t_2 < \ldots < t_m$, where at time t_i it delivers from $x_i \in \{t_i, t_i + 1, ..., B\}$. Without loss of generality, $x_1 < x_2 < \ldots < x_m$. We may assume that $x_m = B$ (otherwise the algorithm is not competitive at all); we also add a dummy service from $x_0 = 0$ at time $t_0 = 0$.

The adversary now chooses some $k \geq 1$ and stops the game at the expiration time θ that is right after the algorithm's k-th service, say $\theta = t_k + 1/2$. ALG's cost is then $\sum_{i=0}^{k} x_i$. The request at $x_{k-1} + 1$ is not served at time t_{k-1}, so, to meet the deadline of this request, the schedule of ALG must satisfy $t_k \leq x_{k-1} + 1$. This implies that $\theta < x_{k-1} + 2$, that is, all requests at points $x_{k-1} + 2, x_{k-1} + 3, ..., B$ expire before their deadlines and do not need to be served. Therefore, to serve this instance, the optimal solution may simply deliver from $x_{k-1} + 1$ at time 0. Hence, the competitive ratio of ALG is at least $\sum_{i=0}^{k} x_i/(x_{k-1} + 1)$. By Lemma 7, it is possible to choose k such that this ratio is strictly greater than R, a contradiction with R-competitiveness of ALG. □

Next, we show that the same lower bound applies to MLAP-L, the version of MLAP where the waiting cost function is linear. This improves the lower bound of 3.618 from [10].

Theorem 9. *There is no online algorithm for MLAP-L on the line with competitive ratio smaller than* 4.

Proof. Similarly to the proof of Theorem 8, we create an instance of 1P-MLAP-L (the variant of 1P-MLAP with linear waiting cost functions) that does not allow a better than 4-competitive online algorithm. Fix any online algorithm ALG for 1P-MLAP-L and, towards a contradiction, suppose that it is R-competitive, for some $R < 4$. Again, let B be the integer whose existence is guaranteed by Lemma 7. In our instance of 1P-MLAP-L, there are 6^{B-x} requests at x for any $x \in \{1, 2, \ldots, B\}$.

Without loss of generality, we make the same assumptions as in the proof of Theorem 8: algorithm ALG is defined by a sequence of services at times $0 = t_0 < t_1 < t_2 < \ldots < t_m$, where at each time t_i it delivers from point x_i, where $0 = x_0 < x_1 < \ldots < x_m = B$.

Again, the strategy of the adversary is to stop the game at some expiration time θ that is right after some time t_k, say $\theta = t_k + \epsilon$, for some small $\epsilon > 0$. The algorithm pays $\sum_{i=0}^{k} x_i$ for serving the requests. The requests at $x_{k-1} + 1$ waited for time t_k in ALG's schedule and hence ALG's waiting cost is at least $6^{B-x_{k-1}-1} \cdot t_k$.

The adversary delivers from point $x_{k-1} + 1$ at time 0. Each of the remaining, unserved requests at points $x_{k-1} + 2, x_{k-1} + 3, \ldots, B$ pay θ for waiting. There are $\sum_{j=x_{k-1}+2}^{B} 6^{B-j} \leq (1/5) \cdot 6^{B-x_{k-1}-1}$ such requests and hence the adversary's waiting cost is at most $(1/5) \cdot 6^{B-x_{k-1}-1} \cdot (t_k + \epsilon)$.

Therefore, the algorithm-to-adversary ratio on the waiting costs is at least $5t_k/(t_k + \epsilon)$. For any k we can choose a sufficiently small ϵ so that this ratio is larger than 4. The ratio on the servicing cost is

$\sum_{i=0}^{k} x_i/(x_{k-1}+1)$, and by Lemma 7, it is possible to choose k for which this ratio is strictly greater than R. This yields a contradiction to the R-competitiveness of ALG. □

We point out that the analysis in the proof above gives some insight into the behavior of any 4-competitive algorithm for 1P-MLAP-L (we know such an algorithm exists, by the results in Section 3), namely that, for the type of instances used in the above proof, its waiting cost must be negligible compared to the service cost.

5. An Offline 2-Approximation Algorithm for MLAP-D

In this section we consider the offline version of MLAP-D, for which Becchetti et al. [23] gave a polynomial-time 2-approximation algorithm based on LP-rounding. We give a much simpler argument that does not rely on linear programming.

We use an alternative specification of schedules that is easier to reason about in the context of offline approximations. If S is a schedule, for each node $x \in \mathcal{T}$ we can specify the set S_x of times t for which S contains a service (X, t) with $x \in X$. Then the set $\{\mathsf{S}_x\}_{x\in\mathcal{T}}$ uniquely determines S. Note that we have $\mathsf{S}_x \subseteq \mathsf{S}_y$ whenever y is an ancestor of x. We can now write the service cost as $\mathsf{scost}(\mathsf{S}) = \sum_{x\in\mathcal{T}} \ell_x \cdot |\mathsf{S}_x|$ (recall that $\ell_r = 0$). Without loss of generality, we assume that in an offline schedule, each service time is equal to some deadline.

Let $\mathcal{R}$ be the set of requests from the given instance. For each node v, define $\mathcal{R}_v$ to be the set of all intervals $[a_\rho, d_\rho]$ corresponding to requests ρ issued in $\mathcal{T}_v$, the subtree of $\mathcal{T}$ rooted at v.

Algorithm OFFLBYL: We proceed level by level, starting at the root and in order of increasing depth, computing the service times S_v for all nodes $v \in \mathcal{T}$.

- For the root r, S_r is the set of the deadlines of all requests.
- Consider now some node v with parent u for which S_u has already been computed. Using the standard earliest-deadline algorithm, compute S_v as the minimum cardinality subset of S_u that intersects all intervals in $\mathcal{R}_v$.

  More precisely, start with S_v empty. Repeat the following steps until $\mathcal{R}_v$ is empty:
  - find $[a, d] \in \mathcal{R}_v$ with the minimal d;
  - find $\bar{d} \in \mathsf{S}_u$ maximal such that $\bar{d} \le d$;
  - add $\bar{d}$ to S_v;
  - remove all intervals $[a_\rho, d_\rho]$ containing $\bar{d}$ from $\mathcal{R}_v$.

Theorem 10. *Algorithm* OFFLBYL *is a polynomial-time* 2*-approximation for MLAP-D.*

Proof. We first show that the approximation ratio of Algorithm OffLByL is at most 2. Denote an optimal schedule by S^*. According to our convention, S^*_v is then the set of times when v is served in S^*. As $\mathsf{cost}(\mathsf{S}) = \sum_v \ell_v \cdot |\mathsf{S}_v|$ and $\mathsf{cost}(\mathsf{S}^*) = \sum_v \ell_v \cdot |\mathsf{S}^*_v|$, it is sufficient to show that $|\mathsf{S}_v| \leq 2 \cdot |\mathsf{S}^*_v|$ for each $v \neq r$. Note that as $\ell_r = 0$, setting S_r to be the set of all deadlines incurs no cost.

To this end, let u be the parent of v. The set S_u intersects all intervals in $\mathcal{R}_v$ (because it intersects all intervals of $\mathcal{R}_u$, a superset of $\mathcal{R}_v$). We construct $\mathsf{S}'_v \subseteq \mathsf{S}_u$ as follows. For each $t \in \mathsf{S}^*_v$, choose the maximal $t^- \in \mathsf{S}_u$ such that $t^- \leq t$, and the minimal $t^+ \in \mathsf{S}_u$ such that $t^+ \geq t$. Add t^-, t^+ to S'_v. (More precisely, each of them is added only if it is defined.) Then $\mathsf{S}'_v \subseteq \mathsf{S}_u$ and $|\mathsf{S}'_v| \leq 2 \cdot |\mathsf{S}^*_v|$. Furthermore, any interval $[a_\rho, d_\rho] \in \mathcal{R}_v$ contains some $t \in \mathsf{S}^*_v$ and intersects S_u, so it also must contain either t^- or t^+. Therefore S'_v intersects all intervals in $\mathcal{R}_v$. Since we pick S_v optimally from S_u, we have $|\mathsf{S}_v| \leq |\mathsf{S}'_v| \leq 2 \cdot |\mathsf{S}^*_v|$, completing the proof.

Clearly OffLByL runs in polynomial time. In fact, it can be implemented in quadratic time as follows. Let N be the number of nodes in $\mathcal{T}$ plus the number of requests. We first prepare two sorted lists of requests, one sorted by non-decreasing arrival times and the other by non-decreasing deadlines, using $O(N \log N)$ operations. Now the earliest-deadline algorithm can compute each S_v in $O(N)$ operations by sweeping the lists. Altogether we need to process $O(N)$ vertices, a total of $O(N^2)$ operations including preprocessing. The only operations we need are comparisons of numbers that appear on the input. □

It is easy to show that the analysis above is tight. Consider a tree of three nodes: the root r, the intermediate node a, and the leaf b, where $\ell_a = \epsilon$ and $\ell_b = 1$. The input contains of three requests: ρ_0 at a, with arrival time 0 and deadline 2, ρ_1 at b, with arrival time 1 and deadline 4, and ρ_2 at b, with arrival time 3 and deadline 5. An optimal solution could choose sets $\mathsf{S}^*_r = \{2, 4, 5\}$, $\mathsf{S}^*_a = \{2, 4\}$ and $\mathsf{S}^*_b = \{4\}$, of total cost $1 + 2\epsilon$. On the other hand, the choices made by OffLByL at node a cause possible choices at node b to be sub-optimal. That is, it chooses $\mathsf{S}_r = \{2, 4, 5\}$, $\mathsf{S}^*_a = \{2, 5\}$ and $\mathsf{S}^*_b = \{2, 5\}$ of total cost $2 + 2\epsilon$. As ϵ can be arbitrarily small, the approximation ratio can be arbitrarily close to 2.

Acknowledgements

The presented results were included in their preliminary form in the paper that appeared in the proceedings of 24th Annual European Symposium on Algorithms (ESA'16) [5]. Other results of that paper appeared subsequently in Operations Research [6].

Research was supported by NSF grants CCF-1536026, CCF-1217314 and OISE-1157129, Polish National Science Centre grants 2016/21/D/ST6/02402, 2016/22/E/ST6/00499 and 2015/18/E/ST6/00456, projects 17-09142S and 19-27871X of GA ČR, GAUK project 634217, European Research Council grant ERC-2014-CoG 647557, FMJH PGMO, ANR OATA, and RFSI. L. Folwarczný was supported by the grant SVV-2020-260578.

References

[1] C. Brito, E. Koutsoupias, S. Vaya, Competitive analysis of organization networks or multicast acknowledgement: how much to wait?, Algorithmica 64 (4) (2012) 584–605.

[2] W. Yuan, S. V. Krishnamurthy, S. K. Tripathi, Synchronization of multiple levels of data fusion in wireless sensor networks, in: Proc. Global Telecommunications Conference (GLOBECOM), 221–225, 2003.

[3] N. Buchbinder, T. Kimbrel, R. Levi, K. Makarychev, M. Sviridenko, Online make-to-order joint replenishment model: primal-dual competitive algorithms, in: Proc. 19th ACM-SIAM Symp. on Discrete Algorithms (SODA), 952–961, 2008.

[4] C. Papadimitriou, Computational aspects of organization theory, in: Proc. 4th European Symp. on Algorithms (ESA), 559–564, 1996.

[5] M. Bienkowski, M. Böhm, J. Byrka, M. Chrobak, C. Dürr, L. Folwarczný, L. Jez, J. Sgall, N. K. Thang, P. Veselý, Online algorithms for multi-level aggregation, in: Proc. 24th European Symp. on Algorithms (ESA), 12:1–12:17, 2016.

[6] M. Bienkowski, M. Böhm, J. Byrka, M. Chrobak, C. Dürr, L. Folwarczný, L. Jez, J. Sgall, N. K. Thang, P. Veselý, Online Algorithms for Multilevel Aggregation, Oper. Res. 68 (1) (2020) 214–232.

[7] N. Buchbinder, M. Feldman, J. S. Naor, O. Talmon, O(depth)-competitive algorithm for online multi-level aggregation, in: Proc. 28th ACM-SIAM Symp. on Discrete Algorithms (SODA), 1235–1244, 2017.

[8] Y. Azar, A. Ganesh, R. Ge, D. Panigrahi, Online service with delay, in: Proc. 49th ACM Symp. on Theory of Computing (STOC), 551–563, 2017.

[9] Y. Azar, N. Touitou, General Framework for Metric Optimization Problems with Delay or with Deadlines, in: Proc. 60th IEEE Symp. on Foundations of Computer Science (FOCS), 60–71, 2019.

[10] M. Bienkowski, J. Byrka, M. Chrobak, Ł. Jeż, J. Sgall, G. Stachowiak, Online control message aggregation in chain networks, in: Proc. 13th Int. Workshop on Algorithms and Data Structures (WADS), 133–145, 2013.

[11] D. R. Dooly, S. A. Goldman, S. D. Scott, On-line analysis of the TCP acknowledgment delay problem, Journal of the ACM 48 (2) (2001) 243–273.

[12] A. R. Karlin, C. Kenyon, D. Randall, Dynamic TCP acknowledgement and other stories about e/(e - 1), Algorithmica 36 (3) (2003) 209–224.

[13] S. S. Seiden, A guessing game and randomized online algorithms, in: Proc. 32nd ACM Symp. on Theory of Computing (STOC), 592–601, 2000.

[14] M. Bienkowski, J. Byrka, M. Chrobak, Ł. Jeż, D. Nogneng, J. Sgall, Better approximation bounds for the joint replenishment problem, in: Proc. 25th ACM-SIAM Symp. on Discrete Algorithms (SODA), 42–54, 2014.

[15] M. Chrobak, Online aggregation problems, SIGACT News 45 (1) (2014) 91–102.

[16] A. Aggarwal, J. K. Park, Improved algorithms for economic lot sizing problems, Operations Research 41 (1993) 549–571.

[17] E. Arkin, D. Joneja, R. Roundy, Computational complexity of uncapacitated multi-echelon production planning problems, Operations Research Letters 8 (2) (1989) 61–66.

[18] T. Nonner, A. Souza, Approximating the joint replenishment problem with deadlines, Discrete Mathematics, Algorithms and Applications 1 (2) (2009) 153–174.

[19] M. Bienkowski, J. Byrka, M. Chrobak, N. B. Dobbs, T. Nowicki, M. Sviridenko, G. Swirszcz, N. E. Young, Approximation algorithms for the joint replenishment problem with deadlines, Journal of Scheduling 18 (6) (2015) 545–560.

[20] R. Levi, R. Roundy, D. B. Shmoys, A constant approximation algorithm for the one-warehouse multi-retailer problem, in: Proc. 16th ACM-SIAM Symp. on Discrete Algorithms (SODA), 365–374, 2005.

[21] R. Levi, R. Roundy, D. B. Shmoys, M. Sviridenko, A constant approximation algorithm for the one-warehouse multiretailer problem, Management Science 54 (4) (2008) 763–776.

[22] R. Levi, M. Sviridenko, Improved approximation algorithm for the one-warehouse multi-retailer problem, in: Proc. 9th Int. Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX), 188–199, 2006.

[23] L. Becchetti, A. Marchetti-Spaccamela, A. Vitaletti, P. Korteweg, M. Skutella, L. Stougie, Latency-constrained aggregation in sensor networks, ACM Transactions on Algorithms 6 (1) (2009) 13:1–13:20.

[24] L. L. C. Pedrosa, private communication, 2013.

[25] R. Levi, R. Roundy, D. B. Shmoys, Primal-dual algorithms for deterministic inventory problems, Mathematics of Operations Research 31 (2) (2006) 267–284.

[26] M. Chrobak, C. Kenyon-Mathieu, SIGACT news online algorithms column 10: competitiveness via doubling, SIGACT News 37 (4) (2006) 115–126.

[27] M. Chrobak, C. Kenyon, J. Noga, N. E. Young, Incremental medians via online bidding, Algorithmica 50 (4) (2008) 455–478.