Acta Univ. Sapientiae, Informatica, **6,** 2 (2014) 163–177

DOI: 10.1515/ausi-2015-0001

A new model for the linear 1-dimensional online clustering problem

András LONDON
University of Szeged
Institute of Informatics
email: london@inf.u-szeged.hu

József NÉMETH
University of Szeged
Institute of Informatics
email: nemjozs@inf.u-szeged.hu

Tamás NÉMETH
University of Szeged
Institute of Informatics
email: tnemeth@inf.u-szeged.hu

Áron PELYHE
University of Szeged
Institute of Informatics
email: pelyhe@inf.u-szeged.hu

Abstract. In this study, a mathematical model is presented for an online data clustering problem. Data clustering plays an important role in many applications like handling the data acknowledgment problem and data stream management in real-time locating systems. The inputs in these problems are data sequences, each containing several data elements. Each data element has an arrival time and a weight that reflects its importance. The arrival times are not known in advance, and some data elements never arrive. Hence the system should decide which moment is optimal for forwarding the collected data for processing. This requires finding a good trade-off between the amount of collected information and the waiting time, which may be regarded as a minimization problem. Here, we investigate several online algorithms and present their competitive analysis and average case studies. Experimental results, based on simulations using artificially generated data, are also presented and they confirm the efficiency of our methods.

Computing Classification System 1998: F.1.2
Mathematics Subject Classification 2010: 68W27
Key words and phrases: data clustering, online algorithms, learning algorithms, real-time locating systems

1 Introduction

Online optimization is concerned with the type of problems where decisions should be made without knowing the whole input data. These class of problems are usually called online problems (in contrary, the offline problems are those problems where the whole input is available when the decision is made). In several parts of computer science, economics and operational research many problems can be solved only in an online manner. For some good discussions of this see, for instance [2, 12, 14].

In online clustering problems the goal is the classification of points into sets in an online fashion such that a given objective function, which depends on the distance between any two points in the same cluster, is minimized. Points that arrive consecutively have to be assigned to clusters at the time of arrival. Previous results on the online data clustering problem for data sequences can be found, for example, in [5, 11], with unit sized clusters and in [7, 8, 9] with variable sized clusters.

The problem that is closely related to the data clustering problem examined here first emerged during the study of a real-time locating system developed by the Fraunhofer IIS and investigated and generalized by Németh et al. in [17]. Below, we apply a different mathematical model for the problem, which is a minimizing problem in contrast to the maximizing one defined in [17].

A real-time locating system (RTLS) serves to determine positions of objects with high precision. It has various applications in transport and logistic, ambient assisted living, emergency mission support and also in sports, such as in the so-called Chip-in-the-Ball technologies. As an example, the locating system developed by the Fraunhofer IIS is a radio-based system operating with a local positioning infrastructure using two measurements (the "angle of arrival" and the "round trip time") to detect the position of objects. For a detailed description of RTLS systems see [3] and [4].

In such a system, an object (like the ball of a football game or the shin pad of a player) is equipped with a tag which periodically broadcasts radio signals to infrastructure nodes (such as receiver devices in the stadium). Besides the positioning data, a radio signal also carries the user data and an ID. The signals having the same ID belong to the same locating cycle, which we call a burst. The measured parameters from a given burst data set are distributed over the infrastructure nodes. After a measurement is made for the individual parts of the burst data set, the data elements are forwarded to the central positioning server (via a transport protocol). After receiving sufficient data (ideally the total burst data set), the positioning server can determine the positions of the objects by going through the following algorithmic steps in the server: (1) data filtering, (2) data clustering (3) burst filtering (4) position calculation (5) position filtering.

In this study, we focus on the second step. Our aim is to forward the data as quickly as possible for burst filtering, and also minimize the number of unprocessed data elements to get more precise results in position calculations. The nature of the problem requires that we work in an online manner. The algorithm we designed also has to handle transmission errors and disturbances like provisional coverage (screening) of an infrastructure node, network packet losses and processing delays. Therefore, the data clustering algorithm cannot simply wait for all elements of a burst; a more sophisticated approach is needed! Evidently, the crucial point of the algorithm is to determine the time when the collected data items have to be sent to the server for calculating the positions. Two different objectives should be considered at the same time: (i) the positioning server should receive as much available data from the infrastructure nodes as possible, and (ii) the system is not allowed to wait too long for the incoming data, since the long delays decrease the relevance of the calculated position. The usual principle in a real-time calculation of positions is that a fairly-good position is better than a more accurate position determined too late. We will define a minimization problem that takes into account both goals. The second goal is considered directly, while the first goal appears in the objective function in an indirect way.

Here, collecting the data and calculating the positions of several tags can be done in parallel and independently of each other, hence we will assume that there is only one tag in the system and the infrastructure nodes collect the data only from this tag. We should also mention that this problem is similar to the online data acknowledgment problem (see e.g. [15, 1, 10, 16, 18]). In a computer network, from a communication aspect, data is sent by packets and in the data acknowledgment problem, given a sequence X of packet arrival times, the goal is to partition X into subsequences, where the end of a subsequence is defined by an acknowledgment. One acknowledgment may acknowledge many packets, but if an algorithm waits too long, the sender might resend the packets and this would result in congestion of the packets in the network. Here, our methods may also be applied to handle the data acknowledgment task.

Below, we present a mathematical model and describe the corresponding objective function for the problem. In Section 3, we apply the method of competitive analysis, which is often used to evaluate the quality of online algorithms (see [14] for a nice summary on competitive analysis). We will prove a result which states that there is no competitive algorithm for the problem.

The average case study is also presented for real-world situations, where arrival times of the input elements follow a certain probability distribution. For some discussions on the probabilistic analysis of algorithms see [6] and [13]. Lastly in Section 4, we present experimental results got from using different algorithms with artificially generated data which we used to verify the efficiency of our algorithms.

2 The proposed mathematical model

The input of the data clustering problem is a sequence of cycles $X = (X^1, X^2, \dots)$, where each cycle X^k is a sequence of data elements $(x_1^k, \dots, x_m^k)$ where m is the number of all possible incoming data elements (and it is equal to the number of infrastructure nodes) in a cycle. The reception time of data x_i^k is denoted by t_i^k $(i = 1, 2, \dots, m;\ k = 1, 2, \dots)$. Since there is no guarantee that all data elements will arrive in each cycle, let $t_i^k = \infty$ if x_i^k does not arrive. The bursts will be denoted by $B^1, B^2, \dots$ where $B^k = \{x_i^k \in X^k : t_i^k < \infty\}$. The difference between the reception times of the last and first data elements of the same burst is called the length of the burst.

The exact time when an algorithm decides to send the collected data in burst B^k to the server will be denoted by p_k $(k = 1, 2, \dots)$ and it is called the positioning time. Let $\hat{B}^k(t) = \{x_i^k : t_i^k < t\}$; thus, $\hat{B}^k(p_k) = \hat{B}^k$ is the subset of data elements in burst B^k which is sent to the computer for positioning.

Usually, the infrastructure nodes (wireless smart items, goniometers) have different technical properties (type, position, accuracy), hence the data collected may not all have the same importance. In our model, we assign a weight w_i to data x_i^k $(k = 1, 2, \dots)$, which denotes the importance of the data with respect to the infrastructure node that collected it. Without loss of generality,

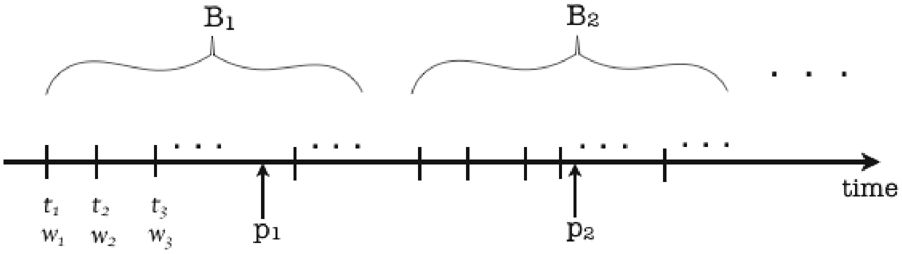

Figure 1: An example for the clustering of the input data signals

we may assume that $\sum_{i=1}^{m} w_i = 1$. Figure 1 shows an example of the input and the notations. We should add that other attributes (i.e. more information) are available in real-time locating systems like the type of the measurements (see the AoA value and RTT value in [4]), which are essential for calculating the positions, but are not needed in calculations using data clustering algorithms.

To evaluate the performance of the algorithms that we are going to devise, first we have to define an objective function which measures their efficiency. In our definition, we simultaneously take into account two objectives of the algorithm, namely (i) the waiting time for the first input data element (which is the time elapsed between the starting time of the burst and the positioning time) should not be too long, and (ii) the amount of data that could be lost (that is, $|\{x_i^k \in B^k \setminus \hat{B}^k : i = 1, 2, \dots, m\}|$ for burst B^k) should be small. In addition, the second objective is integrated over the time-dependency, meaning that a data element arriving later carries less weight in the cost function. Let $r_k = \min_i\{t_i^k : x_i^k \in B^k\}$ be the reception time of the first incoming data in burst B^k. For a given k, we will define the objective function f_k for burst B^k, which has to be minimized, as

$$f_k(t) = \lambda(t - r_k) + (1 - \lambda) \sum_{i:x_i^k \in B^k \setminus \hat{B}^k(t)} \frac{1}{1 + t_i^k} w_i, \tag{1}$$

where $\lambda \in [0, 1]$ is a constant parameter which measures the unit latency. Since the functions $f_1, f_2, \dots$ are independent (because each data element belongs to exactly one burst), by summing them up, we can get the overall cost; that is,

$$F(p_1, p_2, \dots) = \sum_k f_k(p_k), \tag{2}$$

where p_k is the positioning time of burst B^k. Since minimizing F is equivalent to minimizing all its terms, it is sufficient to consider a single term and solve the problem for it, i.e. just consider burst B. Let us assume that the reception time of the first input data element $x_1$is $t_1 = 0$ and denote the positioning time in burst B by p. What we would like to do is to minimize the function f with the form

$$f(p) = \lambda p + (1 - \lambda) \sum_{i:x_i \in B \setminus \hat{B}(p)} \frac{1}{1 + t_i} w_i. \tag{3}$$

Here, the first term is viewed as the loss of latency of the first data element to arrive (where $\lambda \in [0, 1]$ is the cost of the unit latency), while the second term is the sum of the weights of all data arriving after positioning. We shall assume that the longer we have to wait for a data, the less important it is.

3 Analysis of the model

3.1 Competitive analysis

An online algorithm for a minimization problem is said to be c-competitive, if the value of the cost function calculated by using this online algorithm is not more than c times the optimum value of the cost function (obtained by using the offline algorithm) and this holds for all possible input data streams. Formally, for an arbitrary algorithm $\mathcal{A}$ and an input sequence X the value of the cost function in the solution obtained by using $\mathcal{A}$ is $\mathcal{A}(X)$. Let $\mathsf{OPT}(X)$ denote the optimal value of the cost function (offline optimum) for the input X. The online algorithm $\mathcal{A}$ is c-competitive if $\mathcal{A}(X) \leq c \cdot \mathsf{OPT}(X)$.

3.1.1 Analysis without constraints

First, we will assume that there is no any restriction on the input data sequence X. We will show that there exists no constant competitive algorithm for the problem, as the following theorem states.

Theorem 1 *There is no competitive online algorithm for the online data clustering problem that uses the objective function defined by* (3). *More precisely, for every constant* K *there exists an input sequence* X *such that the competitive ratio is larger than* K.

Proof. Let us consider the following input sequence. Let x_1 be the first data element of burst B, which arrives at the infrastructure node 1 at time $t_1 = 0$. If the online algorithm chooses $p > 0$ as the positioning time and if there are no more data elements, the value of the cost function expressed in terms of p is positive, while the offline optima is 0; thus the algorithm is not competitive. If the online algorithm sends the first data element of the burst for positioning immediately after it arrives (i.e. x_1 in t_1; we will call it No Waiting Time Algorithm (NWT)), then in worst case, each other element arrives at time $\delta > 0$ and hence the competitive ratio of the NWT is

$$\frac{(1-\lambda)\displaystyle\sum_{i:x_i\in B\setminus\hat{B}(0)}\tfrac{1}{1+\delta}w_i}{\lambda\delta} = \frac{1}{\delta(1+\delta)}\frac{1-\lambda}{\lambda}\sum_{i:x_i\in B\setminus\hat{B}(0)} w_i \approx \frac{(1-\lambda)}{\lambda\delta^2},$$

which can be an arbitrary large constant if δ is set close to 0. □

3.2 Average case study

We have just found a negative result which states that there is no competitive algorithm, meaning that any algorithm can be easily fooled by a “malicious”' input data sequence. Although in general the input data sequences of real-time positioning systems (or of a data acknowledgment problem) are not like the worst case examples applied in the proof of Theorem 1, they can be modeled by a series of reception times that follow a certain probability distribution. Here, we will assume that the reception time t_i of $x_i \in X$ is a random variable with a given probability distribution F and we also assume that $\{t_i : x_i \in B\}$, $i = 1, \ldots, m$ are independent. Furthermore we will assume that t_i and w_i are independent of each other. Since the sum of the weights (of all data that may ideally arrive) is 1, the expected value of w_i is $1/m$ ($i = 1, 2, \ldots, m$) by using the linearity property of the expected value.

3.2.1 Constant waiting time algorithm

Our goal is to minimize the expected cost of the online algorithm if the distribution of t_i's is given in advance. As before, let p be the positioning time in burst B. Let $K_i = w_i/(1 + t_i)$ if $t_i > p$ and $x_i \in B$, and let $K_i = 0$ otherwise. Using this notation, we get the objective function f with the form

$$f(p) = \lambda p + (1 - \lambda) \sum_{i=1}^{m} K_i. \tag{4}$$

The expected value of K_i is

$$\begin{aligned} \mathrm{E}[K_i] &= \mathrm{E}\Big[\frac{1}{1+t_i} w_i\Big] \Pr(t_i > p) = \\ &= \mathrm{E}[w_i]\mathrm{E}\Big[\frac{1}{1+t_i}\Big] \Pr(t_i > p) = \frac{1}{m}\mathrm{E}\Big[\frac{1}{1+t_i}\Big] \Pr(t_i > p), \end{aligned} \tag{5}$$

obtained by using the independency property of t_i and w_i. By using the linearity property of the expected value and (5), we get that

$$\begin{aligned} \mathrm{E}[f(p)] &= \lambda p + (1 - \lambda)\mathrm{E}\Big[\sum_{i=1}^{m} K_i\Big] = \\ &= \lambda p + (1 - \lambda) m \mathrm{E}[K] = \\ &= \lambda p + (1 - \lambda) \int_{p}^{\infty} g_{\mathcal{F}}(t) dt \int_{0}^{\infty} \frac{1}{1+t} g_{\mathcal{F}}(t) dt. \end{aligned} \tag{6}$$

Here, $g_{\mathcal{F}}$ is the probability density function of the distribution $\mathcal{F}$ and t is a random variable with distribution $\mathcal{F}$. If $\mathcal{F}$ is given, then this formula can be calculated numerically and then it can be optimized with respect to p in a minimization problem.

3.2.2 Constant waiting time algorithm for data streams from normal distribution

Usually, the reception times of a data stream of a real-time positioning system are considered to follow (or at least can be approximated by) a normal distribution, since in such systems, the probability of having large differences between the arrival times among the data elements is small in general. Now, let t be a random variable with a normal distribution ($t \sim \mathcal{N}(\mu, \sigma^2)$) having mean μ and variance σ^2. Then the expected cost of f can be calculated as

$$\begin{aligned} E[f(p), t \sim \mathcal{N}(\mu, \sigma^2)] = \\ &= \lambda p + (1-\lambda)\frac{1}{2\pi\sigma^2} \cdot \\ &\cdot \int_p^\infty \exp\Big[-\frac{(t-\mu)^2}{2\sigma^2}\Big] dt \int_0^\infty \frac{1}{1+t} \exp\Big[-\frac{(t-\mu)^2}{2\sigma^2}\Big] dt \quad = \\ &= \lambda p + c(1-\lambda)\frac{1}{2\pi\sigma}\sqrt{\frac{\pi}{2}}\mathrm{Erf}\Big[\frac{\mu-p}{\sqrt{2}\sigma}\Big], \end{aligned} \tag{7}$$

where

$$c = \int_p^\infty \frac{1}{1+t} \exp\Big[-\frac{(t-\mu)^2}{2\sigma^2}\Big] dt \in [0,1]$$

is a numerically computable constant and

$$\mathrm{Erf}[x] = \frac{2}{\sqrt{\pi}} \int_0^x \exp[-x] dx = 1 - \frac{2}{\sqrt{\pi}} \int_x^\infty \exp[-x] dx.$$

By differentiating wrt p we find that the expression (7) achieves its maximum value if the positioning time p is

$$p = \mu + \sqrt{2\log\Big[\frac{\lambda 2\pi\sigma^2}{c(1-\lambda)}\Big]}\sigma. \tag{8}$$

In the case where we know (or we can estimate) the parameters of the normal distribution (for each burst) of the arrival times, we get a constant waiting time method (CWT), where the positioning time of a burst is given by (8).

4 Experimental evaluation

We saw above that in the worst case there is no efficient algorithm for positioning. Then we showed that in the average case (considering an arbitrary probability distribution $\mathcal{F}$ of the reception times) a simple constant waiting time algorithm can achieve the best possible (minimal) expected cost, but it strongly depends on the expected value and variance of $\mathcal{F}$, which is usually not known. Below, we will define a more sophisticated algorithm that tries to learn a fairly good value of the positioning time p (which is close to the online optimum) by using the optimal values of the previous bursts. We should mention here that similar parameter learning algorithms have also been designed for the data acknowledgment problem and they are described in [16] and [18].

4.1 Variable waiting time algorithm

Now we will describe a variable waiting time algorithm for the data clustering problem. In this algorithm, each burst B^k has a starting time r_k, which is the reception time of the first data element having burst ID k. As in the description of the model, $\hat{B}^k(p) \in B^k$ is the set of those data elements in B^k that is sent to the computer for positioning. The set of data elements that have arrived before the time $\hat{t}$ in burst B^k, is denoted by $\hat{B}^k(\hat{t})$. The algorithm uses a variable t that denotes the waiting time for the data in each burst and it tries to learn the best possible value of t. Let $\mathtt{opt}(k)$ be the online optimal value of the cost function f for burst B^k calculated as follows: whenever a data in burst k arrives at time t^k, we calculate

$$f^{\text{online}}(t_i^k) = \lambda(t_i^k - r_k) + (1-\lambda)\Big[\sum_{i:x_i^k \in B^k(t^k)} \frac{1}{1+t_i^k} w_i + \frac{1}{1+t^k} \sum_{i:x_i^k \in X \setminus B^k(t^k)} w_i\Big], \tag{9}$$

for all $t_i^k \leq t^k$ by considering the worst case that can happen; that is, if all other possible data elements arrive just after t^k. Then, $\mathtt{opt}(k) = \min\{f^{\text{online}}(t^k) : t_i^k \leq t^k\}$, which is the online optimum calculated by using the data elements that had already arrived. Thus, $p_k = \text{argmin}\{\mathtt{opt}(k)\}$ is the optimal positioning time for the online algorithm. After, let $\hat{p}_k = p_k - r_k$. We note here that the online optimum becomes equal to the offline optima when a burst is ended. The

Algorithm 1: Variable waiting time algorithm (VWT)

```
Data: sequence of burst $B^1, B^2 \dots$
Result: positioning time $p^k$ for each burst $B^k$ $(j = 1, 2, \dots)$
Initialize $p_k = 0$ $(k = 1, 2, \dots)$;
foreach data element $x$ with arrival time $t$ do
    if $x \in B^k$ then
        $\mathrm{opt}(k) = \mathrm{opt}(B^k(t))$;
    end
    if $t - r_k \geq (\hat{p}_{j-1} + \cdots + \hat{p}_{j-\ell})/k$ && $p_k \neq 0$ then
        $p_k = r_k + (\hat{p}_{k-1} + \cdots + \hat{p}_{j-\ell})/\ell$ ;
    end
end
```

simple idea behind the construction of the algorithm is to use the average of the previously (and simultaneously) calculated positioning times for the actual burst that we are optimizing. Algorithm 1 shows the details of the learning process.

4.2 Empirical results

To analyze the performance of our algorithms, we generated the following input data stream. The number of bursts is 1000, the arriving times in each burst coming from a normal distribution with an expected value between 5 and 20 and a variance between 2 and 8. The input data can be found in `http://www.inf.u-szeged.hu/~london/1000burst_input.txt`. Figure 2 shows a simple example of the construction of the input. The bursts follow each other consecutively, such that the overlap between two bursts varies between 0 and 30 percent of the latter one. The average burst length is 20. In a burst, optimally the number of data elements is 40 (which is the number of infrastructure nodes), but we randomly delete each data element with probability 0.1 (in reality it may happen that data does not arrive at an infrastructure node). If a data item is deleted in a burst, the probability that it appears in the next one is 0.3. The weights of the data elements (related to the importance of the infrastructure nodes) lie between 0 and 50, assigned to each one with a uniform probability at the beginning.

We also devised three constant waiting time algorithms (CWT) to handle the data stream management task. CWT1 uses the time $r_k + 5$ for positioning (in the jth burst), which is generally less than the middle of the burst (i.e.

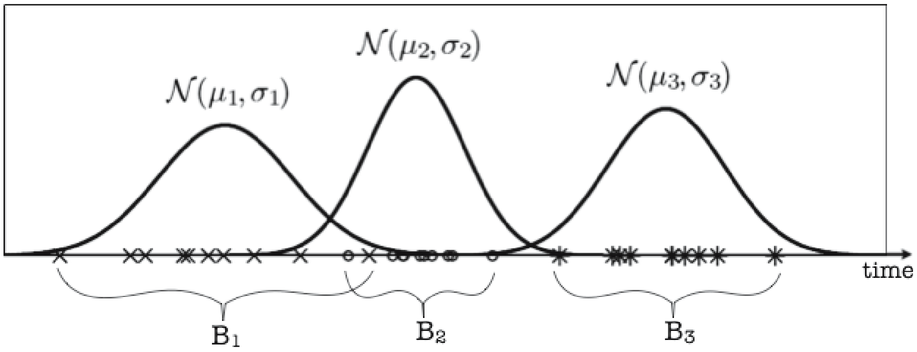
$\mathcal{N}(\mu_1, \sigma_1)$
$\mathcal{N}(\mu_2, \sigma_2)$
$\mathcal{N}(\mu_3, \sigma_3)$
time
B_1
B_2
B_3

Figure 2: The structure of the input data sequence generated from a normal distribution

the starting time plus the expected value of the distribution that generates the arrival times of elements in the burst). CWT2 uses $r_k + 10$ which is close to the middle of the burst in most cases and CWT3 uses $r_k + 15$, which is generally close the the end of the burst. The learning (VWT) algorithm uses the positioning times of the last 50 bursts to calculate the waiting time for the current one. Figure 3 shows the aggregated cost of the different algorithms after a given number of cycles. As can be seen, the calculated cost using the learning algorithm approaches the offline optima after just a few cycles and remains close to it, in contrast to the constant waiting time algorithms where the aggregated costs progressively diverge farther from the offline optima.

Table 1 shows the performances values of the different algorithms, obtained by dividing the value of the total cost function of the different algorithms by the optimum value of the total cost function. The test results tell us that the learning algorithm performs well in general and remains very close to the offline optima regardless of the choice of λ (see Table 1 and Figure 4). In contrast, the efficiency of the CWT algorithms strongly depends on the choice of λ and the average length of the burst. It is not surprising that the higher the value of λ, the better the constant time algorithm will be, which chooses an earlier time for positioning, since if λ is bigger, the cost resulting from latency is also higher. This observation also holds in the reverse case. The NWT algorithm (which sends the first element that arrives in a burst for positioning), in practice, performs poorly on data streams which are like those that in real-world cases just as expected. Similar to CWT1, it only gives an acceptable result when λ is high.

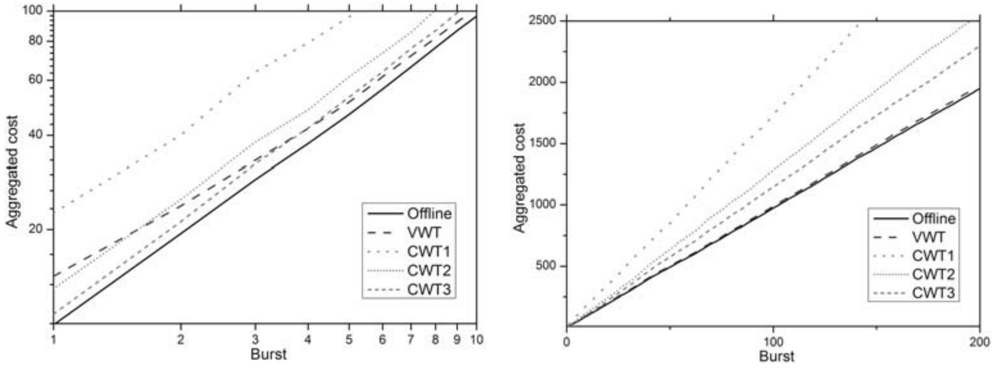

Figure 3: The aggregated cost of the different algorithms after 10 (left) and 200 (right) cycles

	$\lambda = 0.1$	$\lambda = 0.3$	$\lambda = 0.5$	$\lambda = 0.7$	$\lambda = 0.9$
NWT	31.693	8.715	4.225	2.398	1.377
CWT1	11.482	3.346	1.787	1.232	1.333
CWT2	6.127	2.042	1.301	1.151	1.844
CWT3	3.233	1.425	1.159	1.277	2.512
VWT	1.007	1.005	1.016	1.091	1.317

Table 1: The test results

It may be concluded that the VWT algorithm is useful in general and the output values of it are close to the optimal values, even when the burst length and the parameters of the distribution of the arriving times are unknown. However, the efficiency of a constant time algorithm strongly depends on the λ and time parameters of the objective function, the average burst length and also on the distribution of the arrival times.

5 Summary

In this paper we defined an online optimization model for the data stream management problem, which arises in real-time locating systems and in a similar form in the data acknowledgment problems. We constructed different algorithms to solve the problem and analyzed them with the tool of competi-

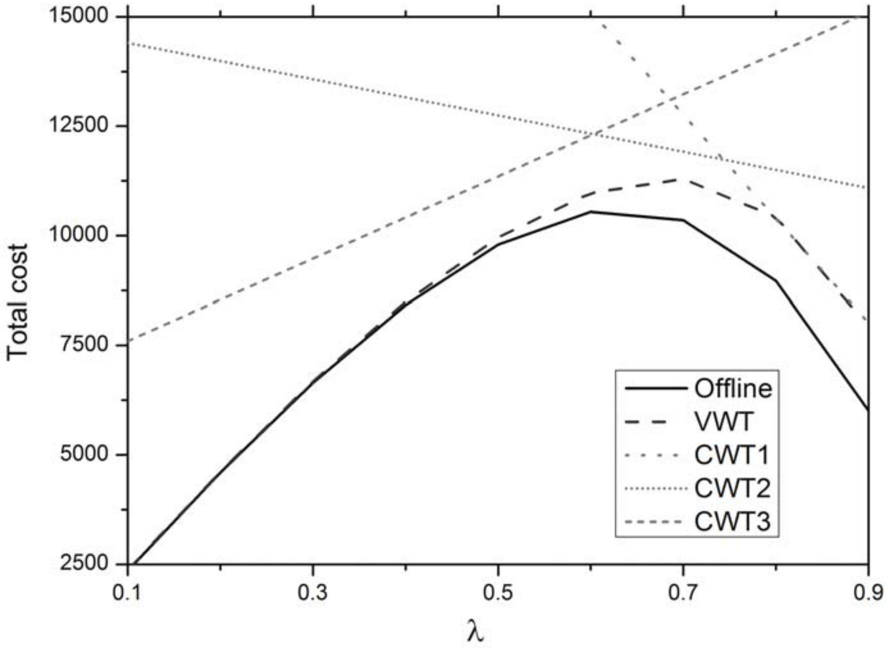

Figure 4: Total cost of the different algorithms after 1000 cycles for different λ values

tive analysis and with expected value analysis. Although we showed, by using the worst case study, that there is no competitive algorithm for this model, the average case study confirmed the validity and applicability of our model and algorithms for more realistic data streams. A more sophisticated variable waiting time algorithm that uses the average of the optimal positional times of some of the previous bursts was also created for position calculations of the subsequent bursts. We showed empirically that this learning method is useful and close to the global optimum, even when we have no a priori information about the input data stream. Some outstanding questions remain that deserve further examination. For instance if we use a more general objective function like $f(p) = \lambda g(p) + (1 - \lambda) \sum_{i:t_i>p} h(t_i) w_i$ with any g and h, such that g is non-decreasing and h is non-increasing, will we get better results? The analytical study that we have already done applies for the general case, but it would be interesting to see more applications (besides the data stream management and data acknowledgment problems) where we can apply this model with different suitable g and h functions. It is also an open question (mentioned by the authors of [17]) of how should go about solving the problem when we do not receive burst ID information. Lastly, it would be also interesting to use and analyze this objective function in the data acknowledgment problem by considering various probability distributions of the arrival times and find out whether it will lead to good performance values.

Acknowledgement

This work was supported by the European Union and the European Social Fund through project Telemedicina (Grant no.: TÁMOP-4.2.2.A-11/1/KONV-2012-0073).

András London was supported by the European Union and the State of Hungary, co-financed by the European Social Fund in the framework of TÁMOP-4.2.4.A/2-11-1-2012-0001 'National Excellence Program'.

The authors would like to thank Csanád Imreh for providing helpful comments.

References

[1] S. Albers, H. Bals, Dynamic TCP acknowledgment: Penalizing long delays, *SIAM J. Discrete Math.* **19,** 4 (2005) 938–951. ⇒165

[2] A. Borodin, R. El-Yaniv, *Online Computation and Competitive Analysis*, Cambridge University Press, 1998. ⇒164

[3] M. Brugger, T. Christ, F. Kemeth, S. Nagy, M. Schaefer, M. M.Pietrzyk, The FMCW technology-based indoor localization system, in: *Ubiquitous Positioning Indoor Navigation and Location Based Services*, 2010, pp. 1–6. ⇒164

[4] M. Brugger, F. Kemeth, Locating rate adaptation by evaluating movement specific parameters, in: *Adaptive Hardware and Systems*, 2010, pp. 127–131. ⇒164, 167

[5] T. M. Chan, H. Zarrabi-Zadeh, A randomized algorithm for online unit clustering, in: *Approximation and Online Algorithms* 2007, pp. 121–131. ⇒164

[6] E. G. Coffman, G. S. Lueker, *Probabilistic Analysis of Packing and Partitioning Algorithms*, Wiley, New York, 1991. ⇒166

[7] J. Csirik, L. Epstein, C. Imreh, A. Levin, Online clustering with variable sized clusters *Algoritmica* **65,** 2 (2013) 251–274. ⇒164

[8] G. Divéki, Online clustering on the line with square cost variable sized clusters *Acta Cybernetica* **21,** 1 (2013) 75–88. ⇒164

[9] G. Divéki, C. Imreh, Online facility location with facility movements, *CEJOR Cent. Eur. J. Oper. Res.* **19,** 2 (2011) 191–200. ⇒164

[10] D. R. Dooley, S. A. Goldman, S. D. Scott, On-line analysis of the TCP acknowledgment delay problem, *Journal of the ACM* **48,** 2 (2001) 243–273. ⇒ 165

[11] L. Epstein, R. Van Stee, On the online unit clustering problem, in: *Approximation and Online Algorithms*, 2008, pp. 193–206. ⇒164

[12] A. Fiat, G. Woeginger, *Online Algorithms: The State of the Art*, Springer, Heidelberg, 1998. ⇒164

[13] M. Hofri, *Probabilistic Analysis of Algorithms: On Computing Methodologies for Computer Algorithms Performance Evaluation*, Springer-Verlag New York, 1987. ⇒166

[14] C. Imreh, Competitive analysis, in: *Algorithms of Informatics, Vol. 1. Foundations* (ed. A. Iványi), mondAt Kiadó, Budapest, 2007, pp. 395–428. ⇒ 164, 165

[15] C. Imreh, T. Németh, On time lookahead algorithms for the online data acknowledgement problem, in: *Mathematical Foundations of Computer Science*, 2007, pp. 288–297. ⇒165

[16] C. Imreh, T. Németh, Parameter learning algorithm for the online data acknowledgment problem, *Optimization Methods and Software* **26,** 3 (2011) 397–404. ⇒ 165, 171

[17] T. Németh, S. Nagy, C. Imreh, Online data clustering algorithms in an RTLS system, *Acta Univ. Sapientiae Informatica*, **5,** 1 (2013) 5–15. ⇒164, 176

[18] T. Németh, B. Gyekiczki, C. Imreh, Parameter learning in lookahead online algorithms for data acknowledgment, *Proc. 3th IEEE Symposium on Logistics and Industrial Informatics*, 2011, pp. 195–198. ⇒165, 171

Received: September 20, 2014 • Revised: October 26, 2014