On the optimization models of congestion control

Cs. Imreh* V. Bilicki*

Abstract

In this paper we investigate an optimization model of congestion control. We extend the model of [8] in two ways. We define the dynamic model in the case of the gentle cost function. We analyze the MIMD (multiplicatively increase, multiplicatively decrease) strategy in this model. In the static model we define and investigate a multidimensional version of the problem. In the second part of the paper we use simulation results for investigating the efficiency of the algorithms considered.

1 Introduction

When a computer network is congested the packets in the system have long delays and many packets will be lost. Therefore it is a basic and important problem to control the traffic and avoid congestion. In the Internet, congestion control is part of the TCP protocol. The basic idea of the congestion control mechanism builded into the TCP protocol is the additively increasing multiplicatively decreasing (AIMD) strategy, after some starting phase the protocol increases additively its sending rate if it can send the packets and it decreases the sent amount by a multiplicative factor two if it suffers some packet loss. The TCP protocol and its different versions are widely investigated mainly by empirical studies (see [4], [12]). Recently this question was considered as an optimization model.

*Department of Informatics, University of Szeged, Árpád tér 2, H-6720 Szeged, Hungary

In [8] the authors give an optimization model of the problem of congestion control. In this model an unicast flow between two hosts (a source A and a sink B) is considered. Time is divided into successive disjoint time periods. In each period t there is an unknown available bandwidth u_t and the source is allowed to send some amount x_t of packets. All the feedback information which is received back is whether the available bandwidth is violated or not. The goal is to find the available bandwidth by minimal guessing cost. If $x_t < u_t$ then the cost of the period is the possible amount which we do not use, this cost is $u_t - x_t$. If $x_t > u_t$ then two cost functions are defined. In the case of severe cost function the cost of the period is u_t in the case of gentle cost function this cost is $\alpha(x_t - u_t)$.

In [8] two models are considered. In the static model the available bandwidth is fixed, $u_t = u$ for every period. Then the problem leads to a guessing game. The algorithms for the solution of this guessing game can be described by the pinning intervals. In each step of this game an interval of pinning is given which contains u, the game is finished when the interval contains only one element. In the case of gentle cost function the authors found an optimal algorithm, it divides in each step the pinning interval into two parts whose size are in the ratio $1 : \sqrt{\alpha}$. In the case of severe cost function many algorithms were investigated.

A more realistic model is the dynamic one, here the available bandwidth is changing over time. In [8] it is supposed that the bandwidth is generated by an adversary. The algorithm tries to estimate the sequence of available bandwidths without having any information about it. Therefore in this case the algorithms which can be used are online algorithms. The efficiency of the online algorithms in adversarial models is usually evaluated by the competitive analysis (see [3]). To be able to use this type of analysis the problem was transformed into a maximization problem. To find constant competitive algorithms restricted adversaries were considered, the authors examined three types of restricted adversary, (restricted to an interval, restricted by a multiplicative factor, restricted by an additive factor). In the dynamic model only the severe cost function was investigated, the authors found optimal algorithms or optimal algorithms up to a constant factor in every cases. In the case of the adversary restricted to a fixed interval an optimal randomized algorithm was find as well.

In [1] randomized online algorithms are investigated. The authors consider the dynamic model with severe cost function against an adversary restricted by a multiplicative factor. The results of [8] were extended to the area of machine learning. In [11] the authors consider a generalized version of the problem where the cost of the algorithm is compared to the best result from the set of some constant predictors.

In this paper we extend in two different ways the model presented in [8]. First in Section 2 we define the dynamic model also in the case of the gentle cost function. We consider the adversary restricted to an interval, and the adversary restricted by a multiplicative factor. In the first case we find a simple optimal algorithm in the second case we investigate the MIMD (multiplicatively increase, multiplicatvely decrease) strategy. This strategy and its randomized version was investigated in [8] and [1] for the case of severe cost function. The MIMD strategy was also investigated in [6]. Then, in Section 3 we define a generalized version of the static problem the multidimensional version. This version models the case where more users have to find the unknown bandwidth allocation. In Section 4 we use simulation results for the evaluation of the algorithms. We use the simulation tool OMNET++ [14] to simulate the available bandwidths based on different distributions (uniform, lognormal, weibull). We also consider the efficiency of the algorithms in a more realistic model, where more users are competing for the bandwidths.

2 Dynamic case, gentle cost function

In this section we investigate the gentle cost function in the dynamic case. In the dynamic model the available bandwidth is changing over time, it is generated by an adversary for each period t. This leads to an online problem. Usually in adversarial models the efficiency of the online algorithms are analyzed by the competitive analysis (see [3]) where the cost or benefit of the examined online algorithm is compared to the cost or benefit of the optimal offline algorithm which has full information about the behavior of the adversary. Since the optimal algorithm with full information has cost 0, (it knows u_t therefore it can send exactly the amount u_t every time), thus investigating the cost function in this case is meaningless. Therefore we change the problem into a maximization problem, instead of minimizing the cost the objective is to maximize the amount of the sent packets. In the case of gentle cost function we can define the objective function as follows.

At the time period t, the adversary generates the unknown amount of available bandwidth u_t the algorithm tries to send x_t packets. The algorithm receives back only the information whether the bandwidth is violated or not. The periods where the bandwidth is not violated ($x_t \leq u_t$) are called successful periods, the periods where the bandwidth is violated ($x_t > u_t$) are called unsuccessful periods. The benefit of the optimal offline algorithm which has full information is u_t, it can use the available bandwidth. The benefit of the online algorithm is

$$g(x_t, u_t) = \begin{cases} x_t, & \text{if } x_t \leq u_t, \\ \max\{0, u_t - \alpha(x_t - u_t)\} & \text{if } x_t > u_t. \end{cases}$$

The difference to the severe cost function is that in the severe cost function unsuccessful periods have gain 0. Therefore if we increase the value of α the gentle function will tend to the severe cost function. The total benefit of an algorithm A on an input sequence $I = \{u_1, \ldots, u_T\}$ is $A(I) = \sum_{t=1}^{T} g(x_t, u_t)$, the optimal offline total benefit on the sequence $OPT(I) = \sum_{t=1}^{T} u_t$. An algorithm is called c-competitive if for every input the optimal benefit is at most c times more then the benefit achieved by the algorithm. Formally an algorithm A is c-competitive if $OPT(I) \leq cA(I)$ for every input instance I. To use this type of definition we assume the technical assumption $u_1 = x_1$.

If we allow the adversary to choose arbitrary u_t in every step then no algorithm can be constant competitive, therefore we must restrict the adversary. In [8] three restrictions were investigated, here we consider only the adversary restricted to a fixed interval and the adversary restricted by a multiplicative factor.

2.1 Adversary restricted to an interval

In this part we suppose that the adversary is only allowed to choose the available bandwidth from a given interval, this means that $u_t \in [a, b]$ is valid for every t. Then the following result holds.

Theorem 1 *The competitive ratio of the optimal online algorithm againts the adversary restricted to the interval* $[a, b]$ *is* $\frac{a+\alpha b}{a+\alpha a}$.

Proof: First we show that no online algorithm can have smaller competitive ratio than $\frac{a+\alpha b}{a+\alpha a}$. Consider an arbitrary online algorithm. Let us suppose that it sends the amount x_t at time period t. First, observe that in the cases where $a - \alpha(x_t - a) \leq 0$, the adversary can use $u_t = a$, and it has a positive gain, but the algorithm has gain 0, thus we can suppose that $x_t \leq \frac{1+\alpha}{\alpha}a$. If $u_t \geq x_t$ the adversary has gain u_t the algorithm has gain x_t, the ratio of the gains is $\frac{u_t}{x_t}$. This function is monotone increasing in u_t, therefore in this case the adversary chooses $u_t = b$. If $u_t < x_t$ the adversary has gain u_t, the algorithm has gain $u_t - \alpha(x_t - u_t)$. The ratio of the gains is decreasing in u_t, thus the adversary chooses $u_t = a$ in this case. Therefore the adversary can reach a ratio of gains

$$\max\{\frac{b}{x_t}, \frac{a}{a - \alpha(x_t - a)}\}.$$

Since the first part is decreasing in x_t, the second is increasing, the maximum will be minimal for the x_t where the two parts are equal. This yields $x_t = \frac{ab(\alpha+1)}{a+\alpha b}$. Substituting this value we obtain that the maximum is $\frac{a+\alpha b}{a+\alpha a}$, and it follows that no algorithm can have smaller competitive ratio.

On the other hand the algorithm which uses $x_t = \frac{ab(\alpha+1)}{a+\alpha b}$ for each t is $\frac{a+\alpha b}{a+\alpha a}$ - competitive, since as above shown then the adversary cannot reach larger ratio of the gains.

2.2 Adversary restricted by a multiplicative factor

In this case the adversary is allowed to change the available bandwidth in every step by a multiplicative factor μ. This means that $u_{t+1} \in [u_t/\mu, \mu u_t]$ is valid for every t. We analyze here the following algorithm MIMD, which is based on the same idea as the algorithm which is used against this type of adversary in case of severe cost function in [8].

Algorithm MIMD

- If there is a successful period ($x_t \leq u_t$), then we increase the sent amount by a multiplicative factor μ, $x_{t+1} = \mu x_t$.

- If there is an unsuccessful period ($x_t \leq u_t$), then we decrease he sent amount by a multiplicative factor 2, $x_{t+1} = x_t/2$.

Theorem 2 *MIMD is* $\mu(2 + 2\frac{\alpha}{\alpha+1})$ *- competitive.*

Proof: First let us observe that $u_t \leq 2\mu x_t$ is valid for each t. One can prove this observation by induction. At the beginning $u_1 = x_1$. Suppose that the statement is true for t. If $x_t \leq u_t$ holds, then in the next step $x_{t+1} = \mu x_t$

and $u_{t+1} \leq \mu u_t$, therefore $u_{t+1} \leq 2\mu x_{t+1}$ will be valid. If $x_t \geq u_t$, then $x_{t+1} = x_t/2$, $u_{t+1} \leq \mu u_t$, and $u_{t+1} \leq 2\mu x_{t+1}$ follows.

We divide the time for phases. A phase starts with a successful time period and besides the starting period it contains a (possible empty) sequence unsuccessful periods, it ends at the first time period t, where $x_{t+1} \leq u_{t+1}$ holds and then at time period $t+1$ a new phase is started.

Consider an arbitrary phase. Suppose it starts at time period t and it contains i unsuccessful periods $t+1, \ldots, t+i$. We show that $\sum_{j=0}^{i} u_{t+j} \leq \mu(2 + 2\frac{\alpha}{\alpha+1}) \sum_{j=0}^{i} g(x_{t+j}, u_{t+j})$ and the statement of the theorem follows.

For time period t, $u_t \leq 2\mu x_t$. In the other periods of the phase we have unsuccessful periods therefore $x_{t+j} = \mu x_t/2^{j-1}$ and $g(x_{t+j}, u_{t+j}) = \max\{0, u_{t+j} - \alpha(x_{t+j} - u_{t+j})\}$.

If the adversary chooses $u_{t+j} = \frac{1+\alpha}{\alpha} x_{t+j}$, then the algorithm has gain 0 in the unsuccessful periods. Let us observe that if the adversary chooses a larger value then the gain of the algorithm has $1+\alpha$ larger increase then the optimal gain, therefore the ratio of the gains (which is larger then 1) would decrease. This means that the adversary maximizes the ratio if it chooses $u_{t+j} = \frac{1+\alpha}{\alpha} x_{t+j}$ for every $j = 1, \ldots, i$. In this case the total gain of the algorithm in the phase is the gain achieved at the successful period which is x_t. The total gain achieved by the adversary

$$\sum_{j=0}^{i} u_{t+j} \leq 2\mu x_t + \sum_{j=1}^{i} \frac{1+\alpha}{\alpha} x_{t+j} =$$

$$= 2\mu x_t + \sum_{j=1}^{i} \frac{1+\alpha}{\alpha} \mu x_t/2^{j-1} \leq \mu(2 + 2\frac{\alpha}{\alpha+1}) x_t,$$

which yields the desired statement.

3 The multidimensional static model

In the static version the available bandwidth is not changed over time it is a fixed integer constant u. We have an a priori knowledge that the available bandwidth is not larger than n. The goal is to find this bandwidth by sending different amounts in the time periods, and getting back the answer whether $x_t > u$ or not. Here we consider cost functions instead of gain. Two different cost functions were introduced, the severe cost function is

$$\begin{cases} u - x_t, & \text{if } x_t \le u_t, \\ u & \text{if } x_t > u, \end{cases}$$

and the gentle cost function is

$$\begin{cases} u - x_t, & \text{if } x_t \le u, \\ \alpha(x_t - u) & \text{if } x_t > u. \end{cases}.$$

We are interested in algorithms which finds the element u in the worst case with minimal cost. We denote this worst case cost of an algorithm A by $MAXCOST(A)$.

In this section we consider the multidimensional generalization of the problem. In this version there is an unknown vector of dimension k, which shows the bandwidth allocation among the users. Each component contains the available bandwidth of a user, the sum of the components gives the total bandwidth which is at most n. The users try to find out their bandwidth independently. The goal is to minimize the maximal total cost of the users. We denote this maximal worst case cost of an algorithm A by $MMAXCOST(A)$. We analyze some strategies from [8], and investigate how the worst case costs are changed due the increased number of users. We will see that in the case of severe cost function the number of users does not play crucial role in the worst case cost, but in the case of gentle cost function this cost depends linearly on the number of users. This property of the cost functions is based on the fact that in the case of severe cost function all algorithm can determine the small bandwidths with very small cost.

In [8] the algorithm were represented by pinning intervals. A pinning interval is an interval which surely contains u, and an algorithms is a function which gives for every interval the guess which divides the interval further.

For the severe cost function we consider two algorithms. BIN is a well-known binary search algorithm in every steps it halves the pinning interval. Algorithm TCP follows the main idea of the TCP congestion control protocol, it halves its guess until it becomes less than or equal to u and then it keeps increasing the guess until it determines u. The following result is true for these algorithms ([8]).

Proposition 1 *([8]) In case of severe cost function*

$$MAXCOST(BIN) = \frac{n \log_2 n}{2} + O(n)$$

$$MAXCOST(TCP) = \frac{n^2}{8} + O(n)$$

For the gentle cost function an algorithm called G_α is considered. This algorithm divides in each steps the pinning interval into two parts whose size are in the ratio $1 : \sqrt{\alpha}$. For this algorithm the following result holds ([8]).

Proposition 2 *([8]) In case of gentle cost function*

$$MAXCOST(G_\alpha) = \sqrt{\alpha}n + O(\log n)$$

In the multidimensional model we have the following results.

Theorem 3 *In the case of severe cost function*

$$MMAXCOST(BIN) \leq n \log_2 n + O(n)$$

$$MMAXCOST(TCP) = \frac{n^2}{8} + O(n)$$

Proof: Consider the algorithm BIN. In each component the user finds the value in at most $\lceil \log_2 n \rceil$ probe. Each probe has cost at most u_i, therefore the total cost is not more then $\log_2 n \sum_{i=1}^{k} u_i + O(n) = n \log_2 n + O(n)$.

Now consider algorithm TCP. Suppose that the unknown vector is $(u_1, \ldots, u_k)$. Consider each u_i in the form $2^{l_i} + v_i$ where $0 \leq v_i \leq 2^{l_i}$. Then to find u_i the TCP algorithm of the i-th user makes $\log_2 n - l_i$ unsuccessful periods with cost u_i, and v_i successful periods with cost $v_i, \ldots, 1$. The total cost of the first type is $O(n)$. Let us observe that $\sum_{i=1}^{k} v_i \leq n/2$. This yields that the total amount of the second type of cost is at most $\sum_{j=1}^{n/2} j = \frac{n^2}{8} + O(n)$. Therefore the total cost is not more than $\frac{n^2}{8} + O(n)$. On the other hand we cannot give better bound as the example $(0, \ldots, 0, n-1)$ shows.

Theorem 4 *In case of gentle cost function*

$$MMAXCOST(G_\alpha) = \sqrt{\alpha}kn + O(\log n)$$

Proof: In the algorithm G_α, if we have a pinning interval $[i, j]$, then the guess is $i+\max\{1, \lfloor \frac{j-i}{1+\sqrt{\alpha}} \rfloor\}$. In the following parts we allow to guess fractional values, the extra cost can build into the additive constant. It follows from the proposition that the maximal cost cannot be larger since in each component the maximal cost is at most $\sqrt{\alpha}n + O(\log n)$. To show that the above cost can be necessary we calculate the cost of finding bandwidth 1. The number of periods will be $L = \log_{1+\sqrt{\alpha}} n$. The guessed bandwidths will be $1 + \frac{n}{1+\sqrt{\alpha}}$, $1 + \frac{n}{(1+\sqrt{\alpha})^2}, \ldots, 1 + \frac{n}{(1+\sqrt{\alpha})^L}$. The total cost is

$$\alpha n \sum_{i=1}^{L} \frac{1}{(1+\sqrt{\alpha})^i} = \sqrt{\alpha}n + O(1).$$

Therefore the total cost of finding the bandwidth allocation $(1, \ldots, 1)$ is $MMAXCOST(G_\alpha) = \sqrt{\alpha}kn + O(\log n)$, and our statement follows.

4 Simulation

4.1 About the statistical behavior of the TCP traffic

With the expansion of the Internet, the traffic engineering, analysis and modelling has become critical for successful device and network design. At the beginning, it was a plausible idea to use the well performing and widely accepted framework for this new area, which was used for telecommunication network modelling, too. This framework uses Markovian chains for modelling states and state transitions in a network with Poisson call arrival and exponential call hold duration probabilities. The old theory allows a tight equilibrium bound on performance variables, so it's sources are well behaving. If such a process is appropriately rescaled in time, the resulting process loses dependence, and it takes the properties of an independent identical distribution (i.i.d) random variables on with exponential smallness of rare events.

Unfortunately most of the packet switched data network traffic performance parameters can be best described with the help of heavy tailed distributions. This property can be explained with several reasons, for example with the heavy tailed nature of the Unix file system file sizes. Leland[9] described this phenomenon for the first time. Crovella and Bestavros[5] studied the Word Wide Web traffic and it's heavy tailed behavior based on the sizes of the web objects. Leland[9] and Willinger[15] went further, they presented the self-similar nature of the network traffic. They showed that large number of independent *on/off* sources with heavy tailed *on/off* period leads to self similarity in the aggregate process. Norros [10] presented, that depending on the queuing model with independent heavy tailed input sources, the queue length tail distribution could be modelled with Weibull distribution. Most of the analytical results are asymptotic and they concentrate on first-order performance measures in open-loop systems.

To be able to study the behavior of the TCP, we have to take care about the statistical properties of the controlled traffic. It is evident upon the above mentioned difficulties that the simulation, as an examination procedure is an acceptable solution. In the following, we will describe the used simulation framework.

4.2 The OMNETPP simulator framework

OMNETPP[14] is an object oriented modular discrete time event simulator. It is developed at the Technical University of Budapest by András Varga. This framework can be freely downloaded and used. It has a useful tool set for building, analyzing, and debugging the simulations. For the implementation of the proper solution, the C++ language can be used, so we can utilize the Object Oriented design patterns for object modelling, and the fast execution.

We can build our simulated system from modules and sub modules which are connected with gates and links. The communication between modules is achieved with the help of messages. Messages represent the events in this framework which are maintained in the Future Event List (FES). We can customize, rebuild, redesign every object. The time is modelled with the help of propagation delays on the links between modules.

One critical point in the simulation is to find the right random number generator. The typical random number generator never produces true random numbers. They use seeds and a deterministic algorithm to produce Pseudo Random Numbers(PRNG) and the following seeds. Starting with the same seed, the distinct simulations produce the same pseudo random values. This property is very useful because it makes the simulation repeatable. The PRNG used in OMNTEPP is based on [7] and has the following properties:

- Range:
$$1, .., 2^{31} - 2$$

- Period length:
$$2^{31} - 2$$
- Method:
$$x := (x * 7^5) mod (2^{31} - 1)$$
- Verification:
$$if\ \ x[0] = 1\ \ then\ \ x[10000] = 1,043618065$$

Mathematical transformations are used to produce the needed distribution.

4.3 The simulation

We implemented and used two kinds of components:

- Algorithm
- Queue

The algorithm tries to guess the currently available bandwidth which is generated by the queue. We consider four algorithms. The first is the above investigated MIMD algorithm which increases and decreases its sending rate by multiplicative factor. We also consider the AIAD algorithm, which increases and decreases its sending rate with an additive factor. The firth algorithm is called TCP this is a rough version which was investigated in the static part, its sending rate has multiplicative decrease and additive increase. Beside these algorithms, we added a new one too, the VTCP, which is a better model of the TCP. Instead of unbounded cycles of guessing, it has variable length cycles based on the transfer of files with length of Pareto distribution. It divides the number of outstanding messages by a constant in the decreasing periods, increments the number by multiplication before the first unsuccessful period, and after that the increment process will be additive. When it finishes the file transfer, it starts the algorithm from the beginning. Let us note here that besides the MIMD strategy none of the above algorithms has constant competitive ratio against the adversary restricted by a multiplicative factor.

We would like to measure the investigated algorithms in a close to real situation. We used several distributions for queue length to achieve this goal. The uniform distribution was selected as the simplest case. The average queue length in a typical LAN environment is around 16 kbytes, so we selected 16000 as the upper bound of the uniform distribution. The lower bound is 0. The WEB generates about 70% of the total backbone traffic [13]. So it is a plausible idea to use the distribution of the length of the http body responses as a second approach for queue length distribution. Based on the article [2] we can use the Lognormal distribution.

$$p(x) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-(lnx-\mu)^2/2\sigma^2}; \mu = 9.357, \sigma = 1.318$$

We used the Weibull distribution for queue length as the third approximation based on article [10] from Norros.

$$p(x) = \frac{bx^{b-1}}{a^b} e^{-(x/a)^b}; a = 16000, b = 2$$

Our simulator works as follows: the queue receives the hint from algorithm and if it is below the actual queue length, it sends a true response, otherwise a false one. The base unit of the hint generation is 1500 bytes long, this value is derived from the Maximal Transmission Unit on an ethernet segment. We use 10000 cycles of hint and answer generation for statistically reliable results. The width of the collected data compared to its mean is relatively narrow with this iteration number.

Because the framework supports the separated design and implementation, we implemented every algorithm as a separated module which is connected to a central node: to the Queue. An algorithm and the queue module is active during the simulation. They use messages for communicating from one side the message length and from the opposite side the result of the hint. The queue uses the previously presented random algorithms for the queue length generation.

We also used truncated distributions. This belongs to the model of the adversary restricted by a multiplicative factor. In the μ truncated case the queue length is not allowed to change more than a μ multiplicative factor. We consider two truncated cases the cases where the distribution are allowed to change by a factor 1/5 and 1/10 these situations are denoted by (5) and (10) behind the algorithms. We considered three cost functions the severe cost function SG, and two gentle cost functions $GG(0,1)$ and $GG(5)$ the gentle functions with $\alpha = 0,1$ and $\alpha = 5$.

We have the following results. In table 1 we present the optimal parameters settings. In every algorithm we had two parameters the up and down parameters (some of them are multiplicative, some of them are additive). We determined for every algorithms, cost functions and distributions (also for the truncated ones) the optimal parameter setting.

In table 2 we used the algorithms with the best parameters setting in each case. We calculated the ratio of the gain achieved by the algorithm and the optimal offline gain, which is the total available bandwidth. This ratios are contained in the table in percentages.

Table 1: Optimal up/down values

Algorithm	uniform(1,16000)			lognormal(9.36,1.32)			weibull(16000,2)		
	GG(0.1)	GG(5)	SG	GG(0.1)	GG(5)	SG	GG(0.1)	GG(5)	SG
AIAD(0)	2.7/1	1/1.1	1.2/1.6	1.6/1.1	1/1.7	1/1.7	2.8/1	1/1.6	1/1.9
AIAD(5)	2.8/1	1/1.7	1/2.6	2.8/1	1/2.3	1/2.6	2.8/1	1/2.4	1/2.7
AIAD(10)	2.9/1	1/1.7	1/2.7	2.8/1	1/2.2	1/2.8	2.9/1	1/2.3	1/2.8
MIMD(0)	1.4/1.1	1.1/1.1	1.1/1.2	1.2/1.1	1.1/1.2	1.1/1.3	1.3/1.1	1.1/1.2	1.1/1.4
MIMD(5)	1.4/1.1	1.1/1.3	1.1/1.6	1.3/1.1	1.1/1.4	1.1/1.7	1.4/1.1	1.1/1.3	1.1/1.6
MIMD(10)	1.3/1.1	1.1/1.3	1.1/1.5	1.5/1.1	1.1/1.3	1.1/1.5	1.4/1.1	1.1/1.3	1.1/1.5
TCP(0)	2.9/1.1	1/1.6	1/1.8	1.7/1.1	1/2.3	1.1/2.7	2.9/1.1	1/1.3	1/1.3
TCP(5)	1.9/1.1	1/1.6	1/1.9	2.7/1.1	1/1.6	1.1/1.7	2.8/1.1	1/1.5	1.1/1.7
TCP(10)	1.5/1.1	1/1.3	1/1.8	2.8/1.1	1/1.3	1/1.6	2.5/1.1	1/1.3	1/1.7
VTCP(0)	1.3/1.1	1/1.2	1/1.2	1.6/1.2	1/1.4	1/1.6	1.5/1.1	1.1/1.4	1.1/1.7
VTCP(5)	1.5/1.1	1.1/1.7	1.1/2.2	1.5/1.1	1.1/1.6	1.1/1.7	1.5/1.1	1.1/1.5	1.1/1.8
VTCP(10)	1.7/1.1	1.1/1.7	1.1/2	2.4/1.1	1.1/1.5	1.1/1.8	2.6/1.1	1.1/1.3	1.1/1.7

4.4 Evaluation of the results

First let us investigate the behavior of the optimal parameters setting of the algorithms by Table 1. From this results our main observation is the comparison of the objective functions. We can see that the gentle function, (mainly $GG(0,1)$) results in such an optimal parameters settings where the algorithms intend to send more packets. The up parameters are larger the down parameter are smaller then in the case of severe cost function.

More interesting is Table 2, where the efficiency of the algorithms are measured. The first observation that the algorithms are much better in

Table 2: Best results

Algorithm	uniform(1,16000)			lognormal(9.36,1.32)			weibull(16000,2)		
	GG(0.1)	GG(5)	SG	GG(0.1)	GG(5)	SG	GG(0.1)	GG(5)	SG
AIAD(0)	37,122	37,122	33,023	26,348	26,348	24,313	42,855	42,855	38,645
AIAD(5)	92,717	59,917	51	93,535	62,569	53,846	94,450	65,919	57,161
AIAD(10)	94,667	68,436	52,898	95,642	72,010	57,981	96,213	74,270	60,926
MIMD(0)	37,944	37,944	34,791	26,876	26,876	24,898	43,582	43,582	39,992
MIMD(5)	94,391	67,594	60,068	94,074	65,545	59,245	94,595	68,010	60,008
MIMD(10)	96,352	75,919	65,757	96,105	75,293	65,683	96,094	76,139	65,175
TCP(0)	37,827	37,827	34,680	27,415	27,415	25,598	43,776	43,776	40,073
TCP(5)	94,969	63,141	53,357	94,495	65,352	56,700	95,051	67,810	58,985
TCP(10)	96,448	68,644	54,513	96,380	72,526	60,046	96,442	74,803	62,300
VTCP(0)	39,243	39,243	35,279	27,300	27,300	25,449	44,069	44,069	40,466
VTCP(5)	94,737	65,368	57,956	94,082	65,951	58,544	94,592	68,062	60,697
VTCP(10)	96,349	72,658	63,103	96,235	74,421	63,996	96,324	75,338	64,746

average case than the competitive analysis shows. We must note that this is usually true for online problems. Considering the results in details we can conclude the following observations.

We can see by the simulation results the power of the restrictions on the adversary. In the cases where the randomly generated bandwidths were truncated every algorithm for every distributions and cost functions had much better results than in the unrestricted case.

Comparing the algorithms there are not very large differences. Against the uniform distribution the order of the algorithms is VTCP-MIMD-TCP-AIAD, against lognormal distribution the order is TCP-VTCP-MIMD-AIAD, and against the weibull distribution the order is MIMD-VTCP-TCP-AIAD. We can also see that the truncation of the distribution helps the most in the efficiency of MIMD, againts the truncated distribution MIMD had the best results.

Comparing the cost functions we can see by the simulation results that the gentle cost function improves the algorithms efficiency. In the case of $GG(0,1)$ we obtained very good results close the 100 percentages, the difference between the severe cost function and $GG(0,1)$ in some cases were more than 30 percentages. Let us note here that the efficiency in the cases of $GG(\alpha)$ are decreasing in α, and $GG(\infty)$ is the severe cost function. It is an interesting data that in the cases of unrestricted distribution there is only a small difference between $GG(0,1)$ and the severe cost function. The reason is that the advantage of the gentle function (in unsuccessful periods it can have positive gain) disappears if there can be very large differences between the consecutive bandwidths.

Considering the different distributions it seems that the algorithms have the best performance in the case of the weibull distribution and the worst performance in the case of lognormal distribution. Here we have the interesting observation, that the truncation of the distributions decreases the differences in the case of the truncation $\mu = 1/10$ there are only small differences between the performances.

4.5 Race for bandwidth

In previous section we saw several results and best values for the investigated algorithms. There was a queue, only one player, and no race for available bandwidth. Like in the most real life situation, there is no guarantied bandwidth in the following survey. The stronger, greedier algorithm wins. Our algorithms compete with the VTCP algorithm. We build a simulation network with 20 VTCP sources, a queue and a module for investigated algorithm to produce adequate traffic. We used the Weibull distribution for available bandwidth generation and utilized the best up/down values for every algorithm from the previous section. The queue receives the hints and if the sum of them is smaller than the available bandwidth, it sends a positive answer for each player, in an opposite case, it sends a negative answer. It is obvious from the table 3 that our best up/down values are not the optimal solution in race conditions.

Table 3: Transferred bytes

	AIAD	**MIMD**	**TCP**	**VTCP**
Transferred Bytes	798,248	13,130	3,770,730	2,218,380

How can we explain this result? The algorithms with additive increasing steps(AIAD, TCP, VTCP) have advantage because of the fixed length step (MTU). Among these algorithms the TCP has the best difference between the increase and decrease steps.

5 Summary and open questions

In this paper we extended the model of [8] in two different ways. We considered their dynamic model in the case of gentle cost function. We found optimal algorithm against the adversary restricted to a fixed interval, and we analyzed the strategy MIMD against the adversary restricted by a multiplicative factor. Concerning this part it could be an interesting further question to investigate the adversary restricted by an additive factor, or to consider randomized algorithm.

We also considered the multidimensional extension of the static model. We investigated how the algorithms from [8] are changed if we increase the number of users. Here it would be interesting question to define a dynamic multidimensional model. A further open problem is to develop the optimization model, where some feedback information is provided to the algorithms not only the fact whether the bandwidth is violated or not. In real applications the algorithms can gain some more information based on the packet delays.

Beside the theoretical results we used empirical analysis to study the efficiency of the algorithms. We used a simulation tool and generated a scenario which is close to the real traffic. We considered the weibull and lognormal distributions which describe the real internet traffic. We also investigated the scenario where the algorithms have to compete against each other to receive the bandwidths.

References

[1] S. Arora, W. Brinkman, A randomized online algorithm for bandwidth utilization, *Journal of Scheduling*, **(7)3**, 187–194, 2004.

[2] P. Barford, M. Corvella, Generating Representative Web Workloads for Network and Server Performance evaluation, In *Proc. ACM SIGMETRICS* 151–160, 1998.

[3] A. Borodin, R. El-Yaniv, *Online Computation and Competitive Analysis*, Cambridge Univesity Press, 1998.

[4] D. Chiu, R. Jain, Analysis of the increase and decrease algorithm for congestion avoidance in computer networks, *Journal of Computer Networks and ISDN Systems*, **17**, 1–14, 1989.

[5] M. Crovella and A. Bestavros, Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes, *IEEE/ACM Transactions on Networking*, **5(6)** 835–846.

[6] N. Garg, N. E. Young, Online End to End Congestion Control, *Proceedings of the 43-th IEEE Symposium on Fondation of Computer Science*, 303–312, 2002.

[7] R. Jain, *The Art of Computer Systems Performance Analysis* Wiley, New York, 1991.

[8] R. Karp, E. Koutsoupias, C. Papadimitrou, S. Shenker, Optimization Problems in Congestion Control, *Proceedings of the 41- est IEEE Symposium on Fondation of Computer Science*, 66–74, 2000.

[9] W. Leland, M. Taqqu, W. Willinger and D. Wilson, On the self-similar nature of Ethernet traffic, In *Proc. ACM SIGCOMM '93* 183–193, 1993.

[10] I. Norros, A storage model with self-similar input, *Queuing Systems*, **16** 384–396, 1994.

[11] A. Piccolboni, C. Schindelhauer, Discrete Prediction Games with arbitrary Feedback and Loss, *Proceedings of the 14th Annual Conference on Computational Learning Theory, COLT 2001 and 5th European Conference on Computational Learning Theory, EuroCOLT 2001, LNCS 2111,* 208–223, 2001

[12] W. R. Stevens, *TCP/IP Illustrated, Volum 1: The protocols*, Addison-Wesley 1994.

[13] K. Thomson, J. G. Miller, G. Claffy. The nature of the Beast: Recent Measurements from an Internet Backbone, *INET*, www.caida.org/outreach/papers/1998/Inet98/index.xml, 1998.

[14] A. Varga, OMNet++, *IEEE Network Interactive* **16(4)**.

[15] W. Willinger, M. Taqqu, R. Sherman, and D. Wilson, Self-similarity through high variability: statistical analysis of Ethernet LAN traffic at the source level, In *Proc. ACM SIGCOMM '95* 100–113, 1995.