CZ4052 Cloud Computing Assignment 1
TCP AIMD

Eugene Poh

Introduction

In this study, I aim to explore the Adaptive Increase Multiplicative Decrease (AIMD) mechanism of TCP, focusing on tuning its parameters for optimal performance in high-speed networking data center environments. The report includes an exploration of AIMD parameters, experiments, and discussions based on the results obtained.

AIMD Parameter exploration

Linear approach

First, as a reference, we plotted the common AIMD algorithm, to demonstrate the convergence towards the fairness line, as shown in Fig 1. As expected, we can see that the graph is shifting towards the fairness line. In addition, it is going above the efficiency line at the end.

However, it is possible to add bias towards a user. For instance, by increasing the multiplicative decrease and decreasing the additive increase, we can favor against user 1. In Fig 2, we manipulate the AIMD parameters, a, b to favor user 1, thereby allowing the lines converge closer to user 1. The opposite can be done as well to influence the convergence closer to user 2 in Fig 3.

Hence, this poses an interesting question, in a multi-user network, how can we add bias via AIMD algorithm towards certain users in order to attain the best overall network performance?

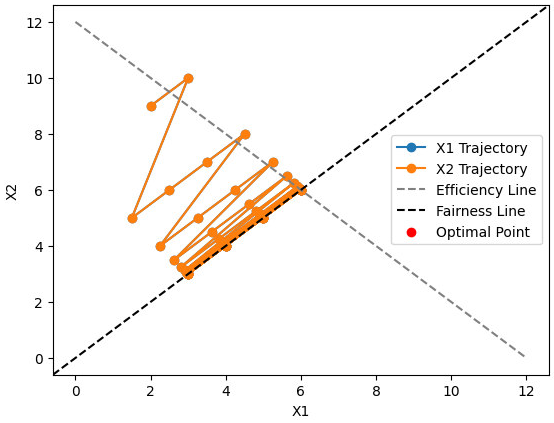

Fig 1

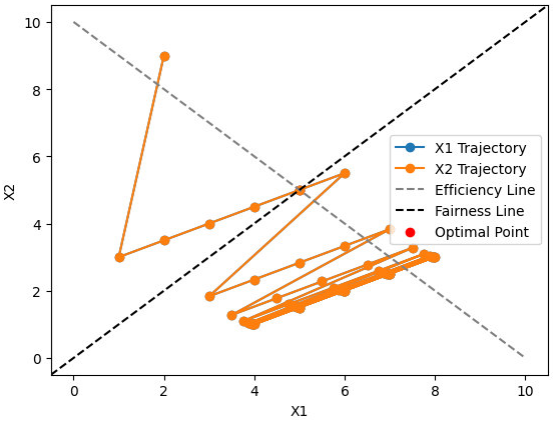

Fig 2

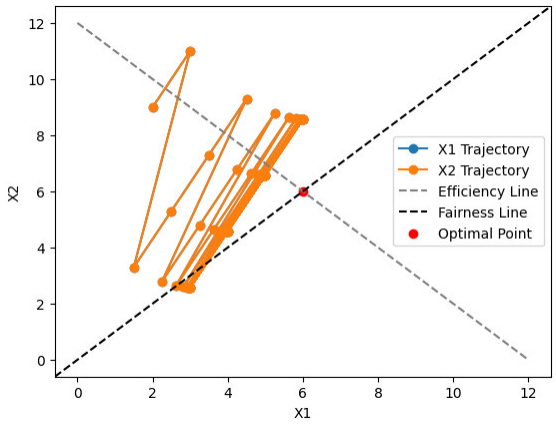

Fig 3

Experiment details of linear equations are summarized in table 1.

Figure	Number of steps	x_{goal}	$x1_a$	$x1_b$	$x2_a$	$x2_b$	$x1_{initial}$	$x2_{initial}$
1	50	12	1	0.5	1	0.5	1	10
2	50	10	1	0.5	0.5	0.33	2	9
3	50	12	1	0.5	2	0.33	1	1

Table 1

TCP Ex Machina approach

In order to pursue this question, I decided to seek TCP Ex Machina as an inspiration. In this study, a much simpler implementation is adopted. The scope of the experiment is restricted to linear AIMD algorithm. simpy library is used to simulate a simple multi-user network system. From this simulation, the network conditions, throughput, queue delay and latency are generated as features for my machine learning model.

For simplicity, latency values are random values, while throughput is controlled via bandwidth which is modelled as a shared resource with a capacity less than the number of users. In our experiment for reproducibility and simplicity, the last user always gets no bandwidth. Packet loss is influenced by the loss of throughput and random values. Then AIMD algorithm with random a.b values are randomly applied for each user.

The machine learning model uses a simple Linear regression model to predict the optimal a,b for each user in the network, in order to obtain the best overall network performance which is indicated by the sum of throughput, inverse of packet loss, inverse of latency, of all users in the network. By generating large amounts of variations of a,b parameters, the model can hopefully learn the best a,b combinations of the multi-user network.

By specifying the desired throughput, latency, packet loss goals, the model can predict the best a,b combinations.

To demonstrate, an example of network of 4 users and the following network conditions is used,

Throughput	1/PacketLoss	1/Latency

579.706664	0.205925	0.007924

Table 2

Predicted a,b Parameters for AIMD (Multi-User):

User	a	b
0	2.50597275	0.76795614
1	3.87566629	0.51438034
2	1.15410487	0.37886906
3	3.2134887	0.32137359

Table 3

Amongst the 4 users, user 1 is most favored while user 2 is the least favored. For clearer visualization, table 4 is provided.

	Total Throughput	Total Packet Loss	Total Latency
User 0	8426.852208	0	934.4844
User 1	7917.352087	0	914.9422
User 2	8228.987466	0	892.9071
User 3	30	148.5166557	908.0641

Table 4: summation of all samples of the network conditions respectively

It seems that users with poorer performance tend to have higher a parameter values and lower b parameter values. This could mean that the model is favoring better performers with less multiplicative decrease and less additive increases while for poorer performers the opposite is applied.

It should be noted that User 3 rarely has access to the bandwidth resource and has the highest packet loss. The model seems to detect this and prioritize on user 3 over user 1 and 2 in an attempt to get a better overall network performance.

Non-linear approach

Apart from linear approaches, non-linear approaches can be considered as well. For example, by comparing Fig 4 to Fig 1,2,3, it can be observed that the use of log can shorten the convergence considerably thereby indicating an advantage.

However, at the same time, the behavior of non-linear AIMD algorithms can prove to have eccentric wild variations compared to its linear counterparts as shown in Fig 5 which suggests a double edge sword situation.

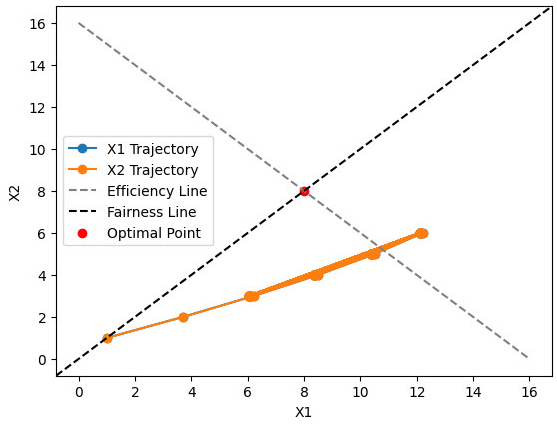

Fig 4

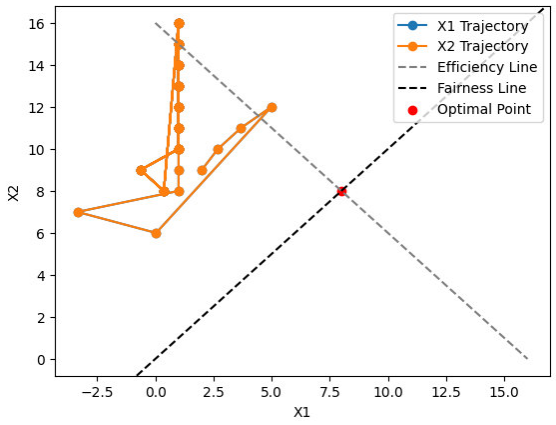

Fig 5

Non-linear experiment details are summarized in table 2.

Figure	Number of steps	x_{goal}	$x1_a$	$x1_b$	$x2_a$	$x2_b$	$x1_{initial}$	$x2_{initial}$
4	50	16	$\log(X_{goal} - x_1)$	0.5	1	0.5	1	1
5	50	10	$\log(x_1)$	e^{x_1}	1	0.5	2	9

Table 2

Conclusion

In conclusion, a simple proof of concept of leveraging machine learning to optimize network performance, inspired by TCP Ex Machina, was demonstrated. Hopefully this exploration opens avenues for future research focus on optimizing AIMD parameters for enhanced performance in high-speed networking data center environments. Through our experiments, we made brief comparisons between linear and non-linear AIMD algorithms in hopes of prompting intriguing discussions.

Appendix

https://github.com/Eugene7997/CZ4052_individual_project/