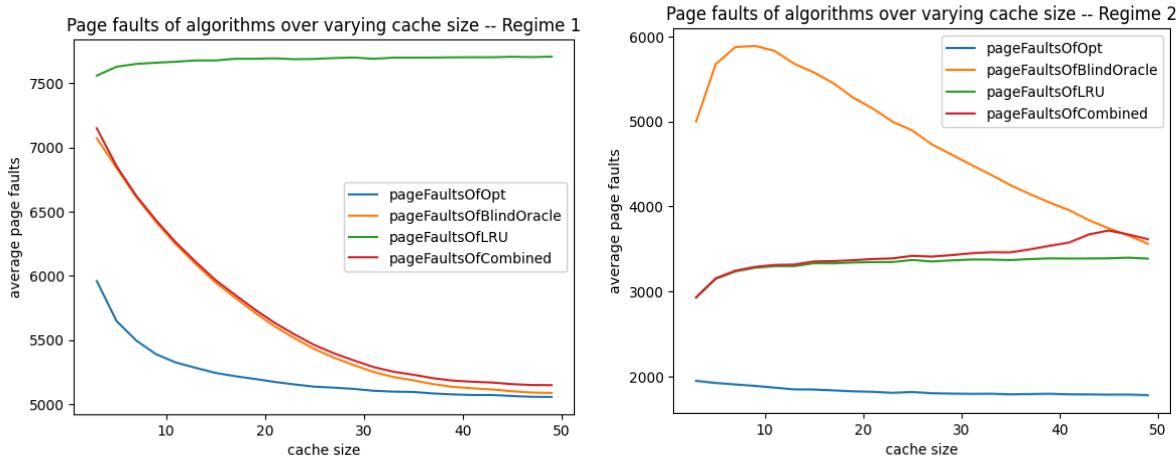


PAGING

Section 1:

Observing the trend by varying the cache size and checking the average page faults occurred when running OPT, BlindOracle, LRU and Combined algorithms:



Note: N = 10k (where every page belongs to [N] and k is cache size)

Kept the below parameters constant for every k	Regime 1	Regime 2
n (number of requests)	10000	10000
epsilon (controls the amount of locality)	0.3	0.8
tow (one of noise controlling parameters)	0.5	0.9
omega (another noise controlling parameter)	180	1000
threshold (combined alg. uses to switch b/w algorithms)	0.1	0.1

Observations for Regime 1:

I chose the above mentioned constant values by noticing that, with these values and k = 10, OPT is significantly better than BlindOracle which is significantly better than LRU. Because the values of tow and omega are not very high or not very low, OPT performs much better than BlindOracle. As we know LRU is designed to utilize the amount of locality that is present in real life when requesting pages in computer memory, when we keep the value of epsilon low, LRU performs much worse than BlindOracle with nearly average noise.

After plotting the graph, I see that, as "k" increased, pageFaults of BlindOracle started coming close to page faults of OPT. I think the reason behind this is the relatively low value of omega. We have 2 parameters to add noise to the true H values to make them look like predictions – tow and omega. Tow controls how many values are to be changed in the true H sequence. Whereas, omega controls, how much the predicted value is deviated from its true value in the H sequence. As cache size grows, the chance that both the predicted and the true H values lying within the cache increases because omega is quite low. Hence, we see the trend where BlindOracle coming close to OPT (in terms of page faults) is observed when cache size is high.

Also, the page faults in OPT decreased faster when cache size is increased from 3 to 15. After that the page faults remained almost the same from 20 to 50. To me it looks like the reason behind is the low amount of locality. When cache is increased from 3 to 15, because there is some amount of locality, we see the page

faults decreased significantly. Whereas, after that the impact of low amount of locality became negligible with increasing cache size.

One more interesting observation is, the page faults increased slightly when cache size is increased for LRU. The values of k, N and epsilon should only be responsible for this. But, I need to spend more time and see what is happening inside the cache to understand this and come to a conclusion.

Observations for Regime 2:

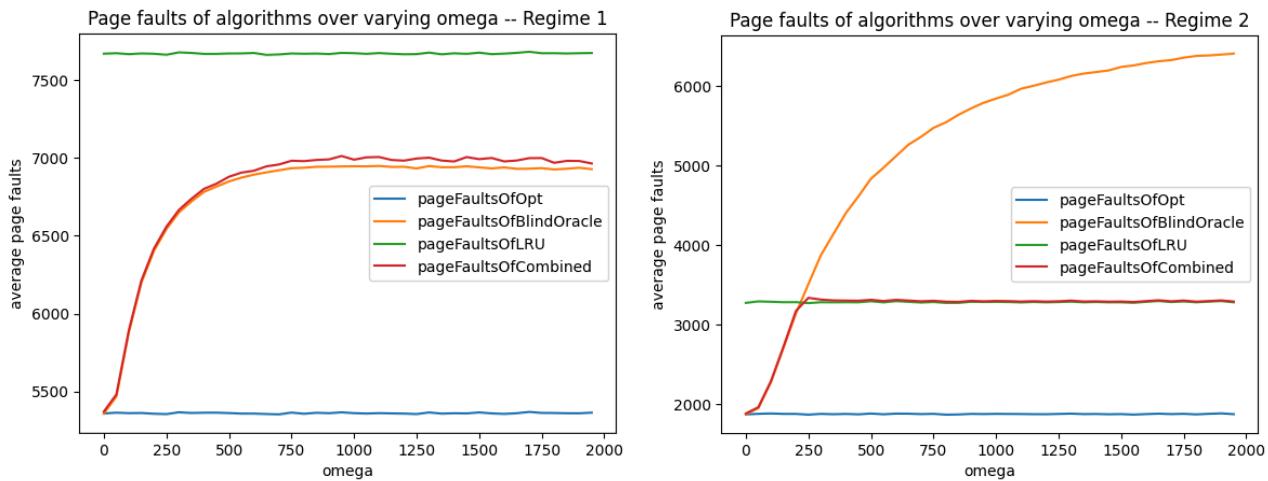
Here, the initial observation is the page faults of OPT are much lower compared to regime 1. This should be because the amount of locality is very high in regime 2. So less number of distinct page requests arrive, i.e., there is a high chance that page requests repeat a lot.

We have very high noise here, so BlindOracle has very high page faults at the beginning. As the cache size increased, the page faults of BlindOracle came close to page faults of LRU. The same explanation as before holds good here, i.e., the impact of amount of deviation caused by omega became lower and lower when cache became higher. But an interesting observation can be seen where the page faults of BlindOracle increased significantly when k is increased from 3 to 10. I can't conclude why this happened by looking at the graph and the parameters used for the algorithm, maybe I have to vary epsilon to see if this trend is related to the amount of locality.

I have not discussed much about the Combined algorithm in both regimes, because the threshold to switch is 0.1 and we see there are no intersections between the lines of LRU and BlindOracle, so the behavior of Combined is as expected. The page faults of Combined are always close to whichever algorithm performs better.

Section 2:

Observing the trend by varying omega and checking the average page faults occurred when running OPT, BlindOracle, LRU and Combined algorithms:



Kept the below parameters constant for every omega	Regime 1	Regime 2
n (number of requests)	10000	10000
epsilon (controls the amount of locality)	0.3	0.8
tow (one of noise controlling parameters)	0.5	0.9
k (cache size)	10	10
N (each request belongs to [N])	100	100
threshold (combined alg. uses to switch b/w algorithms)	0.1	0.1

Observations for Regime 1 and Regime 2:

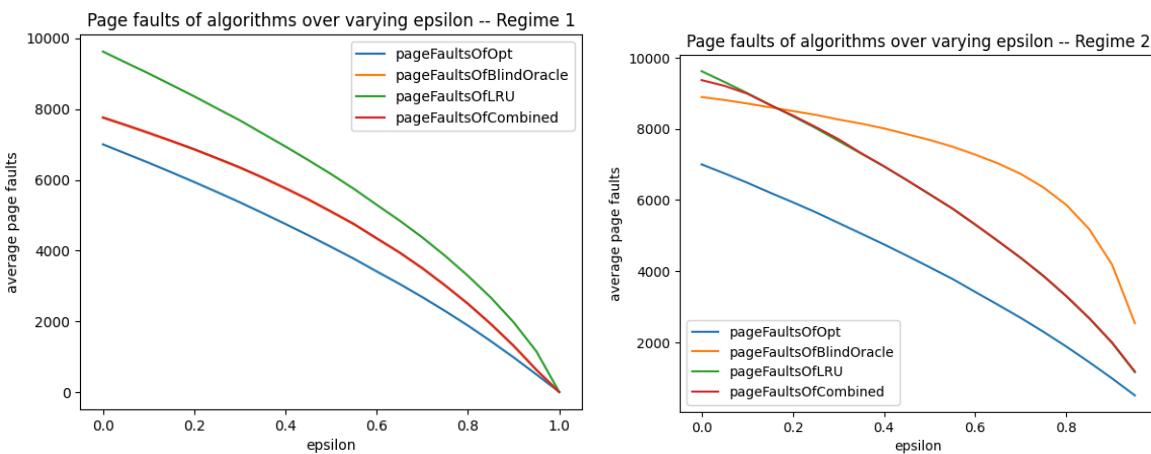
In both the regimes, when omega is 0, the H predictions and the true H sequence are the same, so number of page faults of OPT and BlindOracle are the same. Also, omega doesn't impact LRU, so we see the average page faults of LRU are the same over varying omega. We can easily observe increasing omega also increases page faults in BlindOracle, which is expected.

In regime 1, we have a small tow => BlindOracle doesn't perform too bad. Also we have small epsilon => LRU performs bad. This also reflected in the regime 1 plots. In regime 2, we have higher epsilon => so LRU performs well, and at a point where omega crosses a certain value, BlindOracle performs worse than LRU (as tow is also high). So we see they both intersect, which is expected.

Combined algorithm works almost similar to BlindOracle in regime 1, as BlindOracle performs much better than LRU in regime 1. In regime 2, Combined algorithm performs like BlindOracle until certain value of omega, after that as BlindOracle becomes worse, Combined algorithm behaves like LRU. The Combined algorithm is as expected and can be easily observed.

Section 3:

Observing the trend by varying epsilon and checking the average page faults occurred when running OPT, BlindOracle, LRU and Combined algorithms:



Kept the below parameters constant for every epsilon	Regime 1	Regime 2
n (number of requests)	10000	10000
k (cache size)	10	10
N (every page belongs to [N])	100	100
tow (one of noise controlling parameters)	0.5	0.9
omega (another noise controlling parameter)	180	1000
threshold (combined alg. uses to switch b/w algorithms)	0.1	0.1

Observations for Regime 1:

We don't clearly see the orange line (pageFaultsOfBlindOracle) in regime 1. If we look very closely it is coinciding with the red line (pageFaultsOfCombined). The reason behind this could be, regime 1 is designed in such a way that BlindOracle works significantly better than LRU. Having a lower epsilon only makes things worse for LRU. If BlindOracle is not run with such a low noise, then when epsilon becomes higher there would be a chance of intersection between the lines of BlindOracle and LRU somewhere at the end. It will be interesting to put some higher noise and test.

All the 4 algorithms are having lesser and lesser number of page faults with low epsilon. Amount of locality, in general, reduces the number of distinct pages requested, hence it helps all algorithms (helping LRU the most because it is designed only to utilize the amount of locality in real life). Also when epsilon is 1, the same page would be requested always, so the page faults is 1 for all algorithms.

Observations for Regime 2:

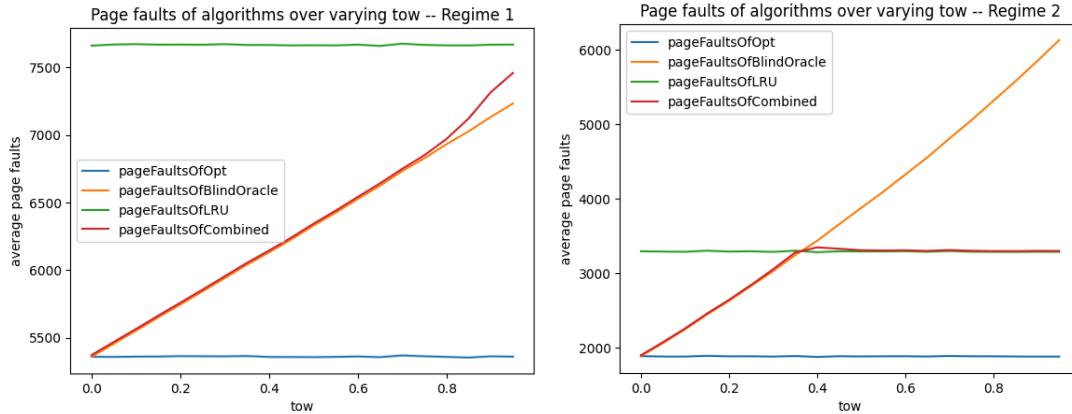
Here, there is an intersection between the BlindOracle and the LRU line some where around epsilon = 0.2. Because regime 2 has high noise for H predictions, BlindOracle shouldn't work well when compared with LRU. But when epsilon (amount of locality) is too low, LRU is behaving worse than BlindOracle with high noise. After some decent epsilon LRU started behaving better. This looks expected.

Also, from the above graph, it looks like Combined algorithm switches between LRU and BlindOracle initially, and then works similar to LRU. This is also as expected which can be derived from the above explanation.

The reduction of number of page faults with increasing epsilon is similar to regime 1 and is expected.

Section 4:

Observing the trend by varying tow and checking the average page faults occurred when running OPT, BlindOracle, LRU and Combined algorithms:



Kept the below parameters constant for every tow	Regime 1	Regime 2
n (number of requests)	10000	10000
epsilon (controls the amount of locality)	0.3	0.8
omega (one of noise controlling parameters)	180	1000
k (cache size)	10	10
N (each request belongs to [N])	100	100
threshold (combined alg. uses to switch b/w algorithms)	0.1	0.1

Observations for Regime 1 and Regime 2:

In both the regimes, when tow is 0, the H predictions and the true H sequence are the same, so number of page faults of OPT and BlindOracle are the same. Also, tow doesn't impact LRU, so we see the average page faults of LRU are the same over varying tow. We can easily observe increasing tow also increases page faults in BlindOracle, which is expected.

In regime 1, we have a small omega => BlindOracle doesn't perform too bad. Also we have small epsilon => LRU performs bad. BlindOracle should have more page faults with higher tow. This is also reflected in the regime 1 plots. In regime 2, we have higher epsilon => so LRU performs well, and at a point where tow crosses a certain value, BlindOracle performs worse than LRU (as tow is also high). So we see they both intersect, which is expected.

Combined algorithm works almost similar to BlindOracle in regime 1, as BlindOracle performs better than LRU in regime 1. In regime 2, Combined algorithm performs like BlindOracle until certain value of omega, after that as BlindOracle becomes worse, Combined algorithm behaves like LRU. The Combined algorithm is as expected and can be easily observed.

Conclusion:

I have noted all the observations with respect to each trend and regime in the above sections. Trends 2 and 4 are as expected and their plots are clearly understandable. Most of the Trend 1 is also as expected, but there are some interesting findings. Like, for a particular range of cache size, the number of page faults increased with the increasing cache size. And, page faults of LRU always increased with the increase in cache size. Most of the Trend 3 is also clear and it will be interesting to test regime 1 with a slightly higher noise as I mentioned in Section 3.

Overall, it is good to say that Combined algorithm is performing quite well. So, when the input page requests are given in online fashion, and there are H prediction values available, it is safe to go with CombinedAlg when the input size is huge. Depending on the amount of locality that we expect, maybe we can try use FIFO or some other algorithm instead of LRU in the CombinedAlg when the amount of locality is too low.

Doing this project gave me a chance to implement algorithms like LRU, BlindOracle, and CombinedAlg. This project also introduced me to online algorithms with advice before they are taught in class and helped me understand the power of advice in real life which we can clearly see in BlindOracle. The idea of CombinedAlg is well framed and I didn't expect it would work this well. Also, I got a good understanding of why these algorithms are designed this way, when thinking about them during implementation. By plotting graphs, analysing them and consolidating the results, things became more evident.