**DIBAKAR BARUA**

**GTID: 903061468**

**Emai: dbarua7@gatech.edu**

**PROJECT 1: EXPERIMENTATION**

**Objective:** To design an optimum cache with total cache storage less than 48KB and offset bits less than or equal to 6, for each of the traces provided. Also, to specify whether we would use a Victim Cache or the Strided Prefetcher if we could use only one.

**Solution:** In order to go about the experiments I have followed the approach given below:

1. ***In my first iteration I choose just my L1 cache, without the prefetcher and the victim cache and obtain an optimum S.*** Since we are given a budget of 48 KB we can safely say that our cache with parameters **C, B S, V, K** (Cache Size, Offset Bits, Associativity ($2^s$), Victim Cache Size, Degree of Prefetching) should fall within the ambit of the equation:-

   $48.1024.8 >= (64 – (C-S) + 2)* 2^{(C-B)} + (64 – B +2) * V + 2^c * 8 + 2^B * V * 8$

   Now we define the average access time for our cache as:

   Average Access Time for our cache (AAT) = Hit Time + Miss Rate * Miss Penalty

   Where Hit Time = 2 + 0.2*S

   Now, we know that as we increase the cache size ($2^c$), we will keep getting better Average Access Times and eventually the cache will saturate due to compulsory misses. Since we are allotted a maximum space of 48K, **C-max = 15** (since $2^{16}$ = 64K). **To obtain minimum possible AAT, we can safely assume that C= 15 will give best possible value even if the cache size saturated earlier.**

**Hence I choose C = 15 for all my caches. I choose B = 6 (maximum) simply as a reference to obtain my optimum S.**

***First iteration: C = 15, B = 6 (we are given budget of B to vary from 0 to 6), V = 0, K = 0 and I vary S from 0 to 9 (S = C-B for a fully associative cache) to obtain optimum S.***

2. ***In my second iteration, after obtaining my optimum value of S, I study if my block size (B = 6) may have <u>polluted</u> my cache, i.e can I obtain a better AAT for lower B?***

   ***Second Iteration: C =15, S = optimum (previous), V = 0, K = 0, and I vary B from 0 to 6.***

3. ***In my third iteration, after I have obtained my optimum B, I turn the Victim Cache on. I check my AATs by varying V from small to very large values and check if my VC size saturates on AAT, i.e I keep getting the same AAT despite increasing V (diminishing returns). I obtain the maximum possible V for my cache till now (for C = 15, B = optimum, S = optimum), and if my AAT saturates before that value, I set V. If not, I set to maximum possible value to stay within the 48 KB restriction.***

4. ***Once I know the optimum value of V, I turn the prefetcher on. I test for optimum value of S AGAIN for a low value of prefetching and a high value of prefetching***. I do this because the prefetcher could be polluting my cache and since it is a dumb prefetcher, it could be showing arbitrary behavior for a fully associative cache **(since the prefetcher WILL NOT ALWAYS evict a prefetched blocked if K >= 1 since there is no index).**

   ***Third Iteration: C =15, B =6, V = optimum, K = 4 (low), vary S from 0 to 9 and to find new optimum S.***

5. ***After this I repeat the above process for a high value of K (120 in my case) and observe the new optimum S and the average access times. If the AAT has improved, I find the optimum S, and carry on experimentation. If not, I analyze lower values of K than this.***

   ***Fourth Iteration: C =15, B =6, V = optimum, K = 120 (high), vary S from 0 to 9 and to find new optimum S.***

6. ***Once we obtain the final optimum value of S, we look at the prefetcher separately. We vary K for a large range of values, observe the trend, and if it is random (the value of AAT vacillates for increasing values of K due to its "dumb" nature and hence we consider this. After this we choose the optimum value of K for lowest possible AAT. A point to note is that there is no restriction on K and hence we can vary it as much as we want, due to its dumb nature we cannot predict its optimum value but once we see a trend that repeats itself, or increases the AAT on an average, we conclude that the minimum has to lie within the values we have considered.***

   ***Fifth Iteration: C =15, B =6, V = optimum, S = optimum, vary K over a large range to observe the trend and make a conclusion about possible lowest K.***

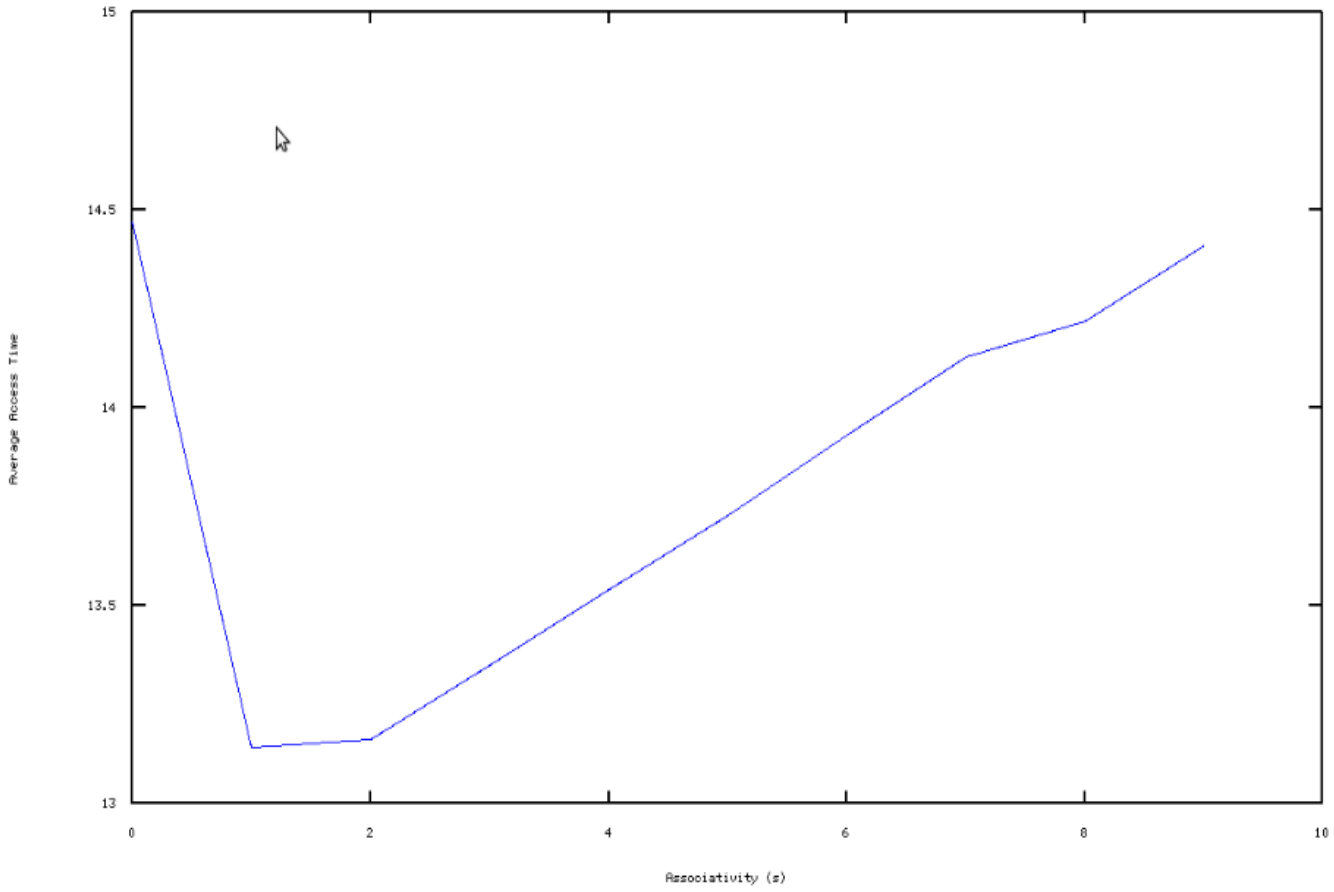
7. ***To obtain if I should prefer a Victim Cache or a Prefetcher if I had the budget of only one, I vary V for K = 0 to observe the achieved AATs, around the optimum value obtained thus far (for V), and I vary K for V = 0 for a large set of values to observe the AATs. The lower of the lowest of the achieved AATs tells me which one I should prefer.***

Now I will elucidate this process for each trace and provide my results. *I will also comment on the efficacy of my solution, how unrealistic it is in a real-life scenario, why the nature of our cache is affecting my result, how it is also unrealistic in a real life scenario etc.*:

**ASTAR.TRACE**

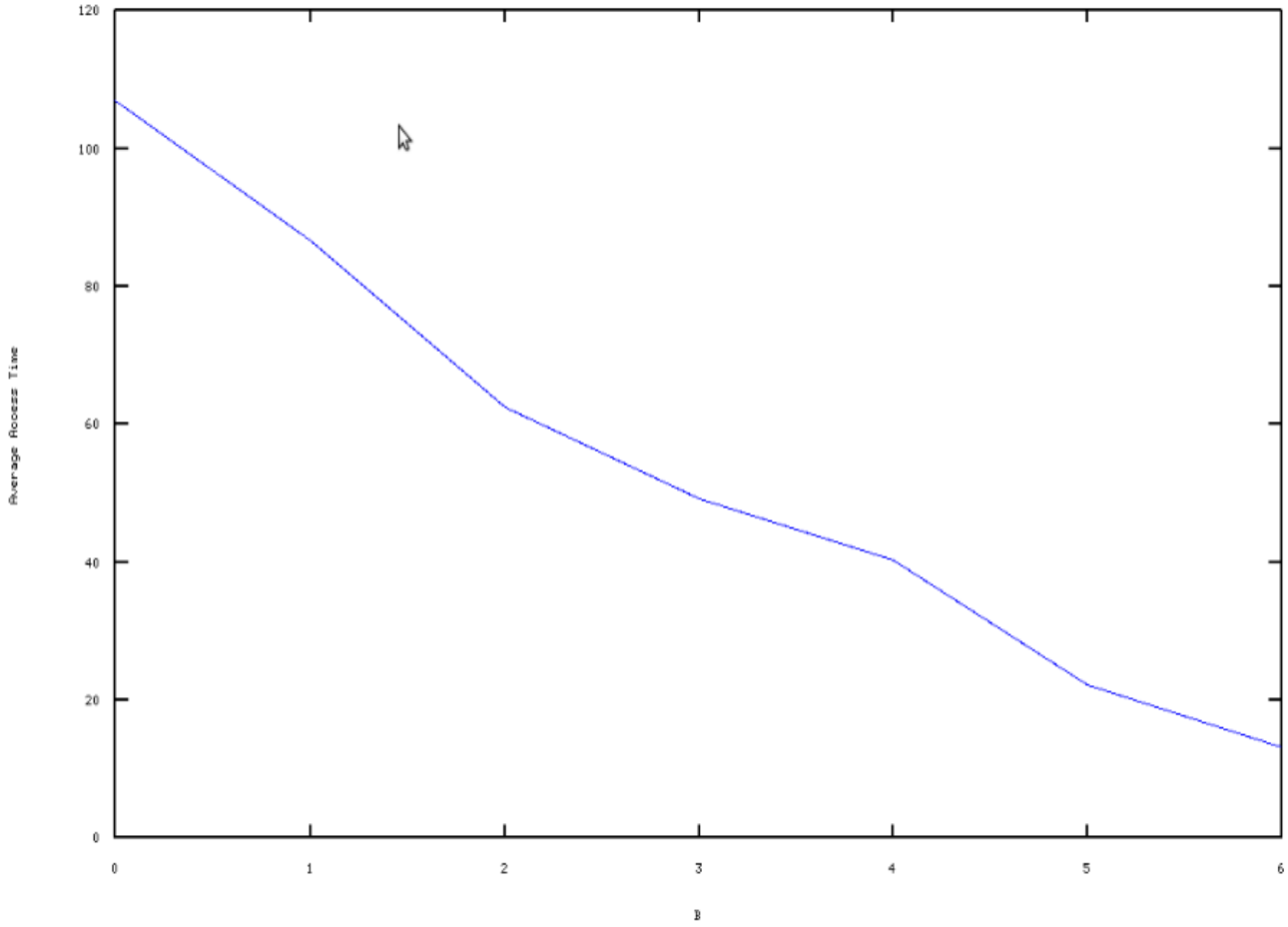
Part a.) Optimum AAT:

1. C =15 V =0 K =0 B = 6, vary S from 0 to 9: (**Note all graphs have AAT on Y-axis**)



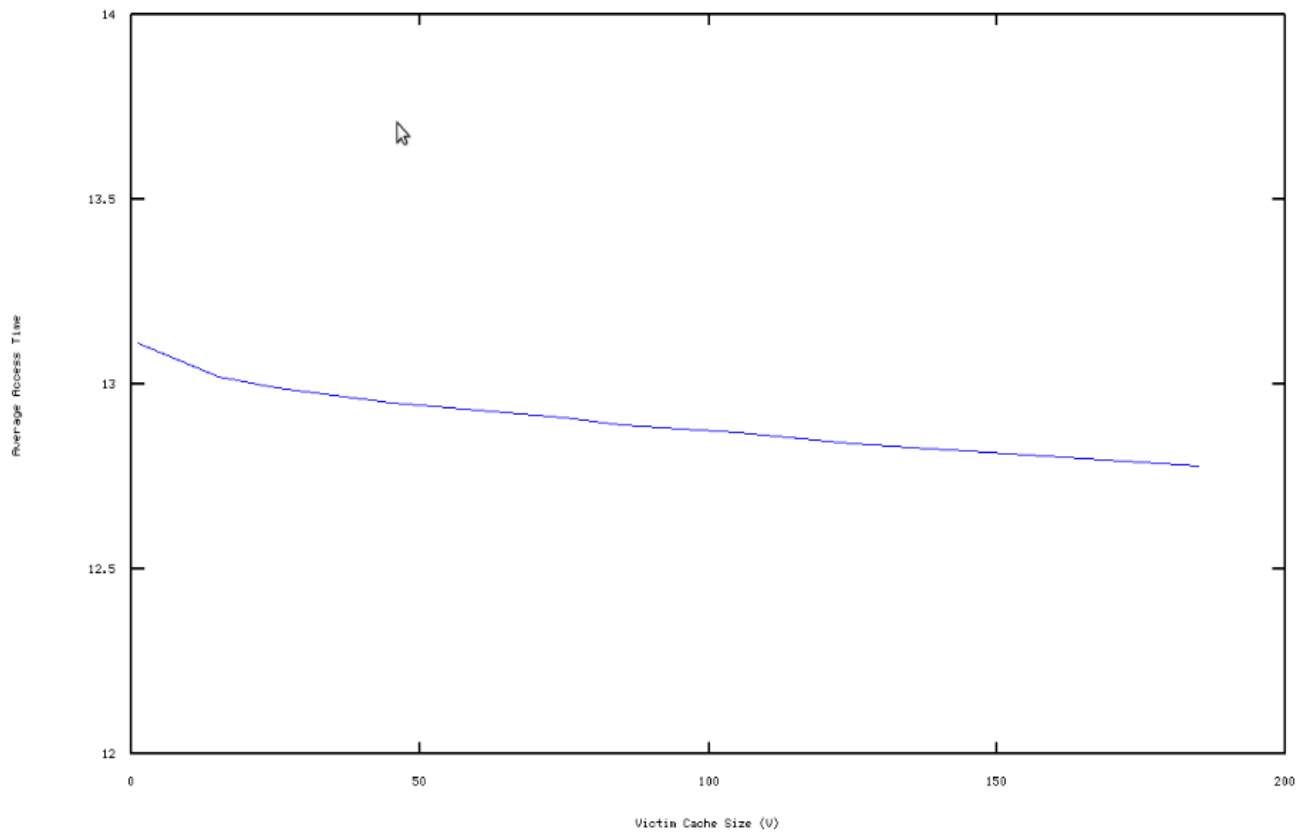

**We observe that AAT is minimum for S = 1 at with AAT min = 13.14. This is an expected trend since AAT reduces initially as miss rate reduces with increasing S, but hit time also increases and when that value takes precedence, AAT will start increasing.**

2. C =15 S = 1 V = 0 K = 0 vary B from 0 to 6: **(B on x axis)**



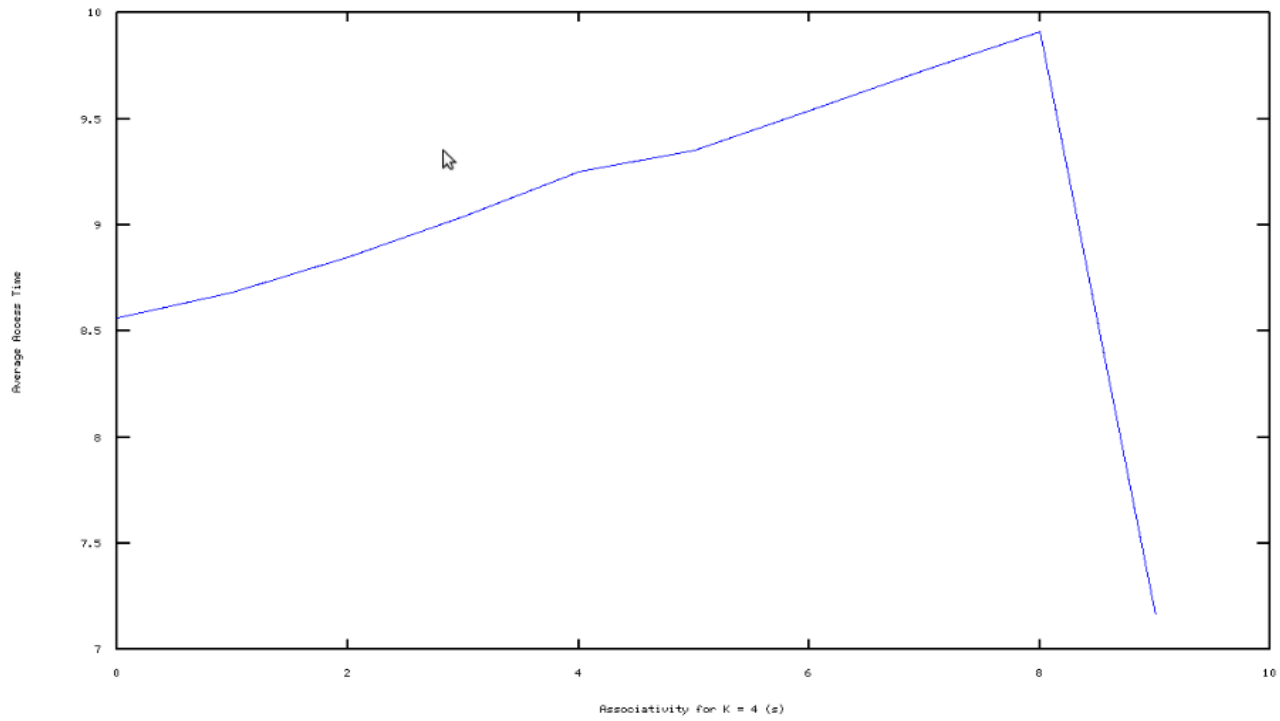

**We observe that AAT reduces uniformly with increasing B and hence the increasing block size is not polluting our cache. Hence B optimum = 6 for AAT min = 13.14.**

3. Now I vary V for values: V = (10 15 25 45 75 85 105 125 145 165 185):



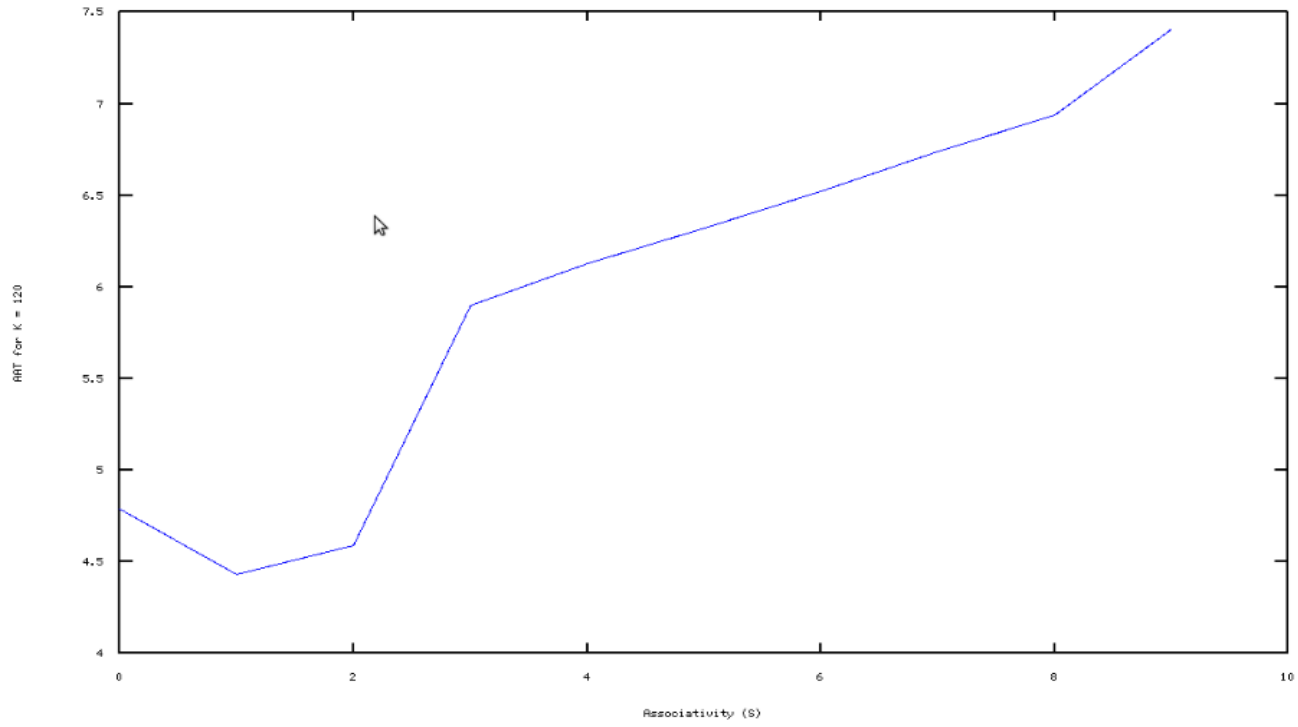

**We observe that AAT reduces with increasing V, tending to reach saturation but it does not. We know for C =15, B =6, S = 1, my cache can support maximum V as 182. Hence following the above curve, I set V =182 to obtain AAT min = 12.78.**

4. Now I turn the prefetcher on: Find optimum S for K = 4:



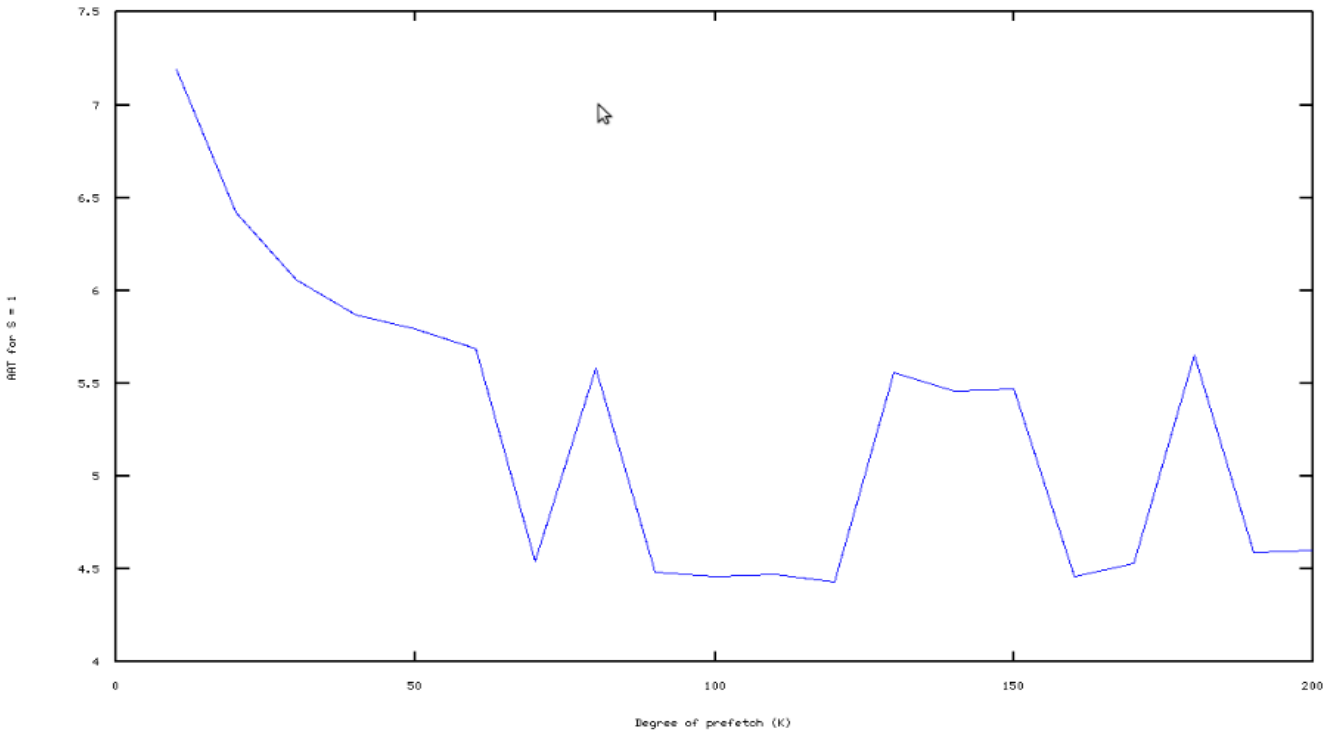

**I find that the fully associative cache (S=9) gives me optimum performance. The AATs achieved range from upwards of 7 units to just below 10 units.**

5. Test for high values of prefetching. Find optimum S for K = 120.



**Now I find that S = 1 is the optimum case and AATs have reduced to range from 4.5 to roughly 7.5 units. Hence I consider S = 1 as my final optimum value of S with AAT min = 4.43.**

6. Now that we know S = 1 is an optimum, I vary K for large values to study its pattern. Current setting is C = 15 B = 6 S = 1 V = 182 and I vary K:

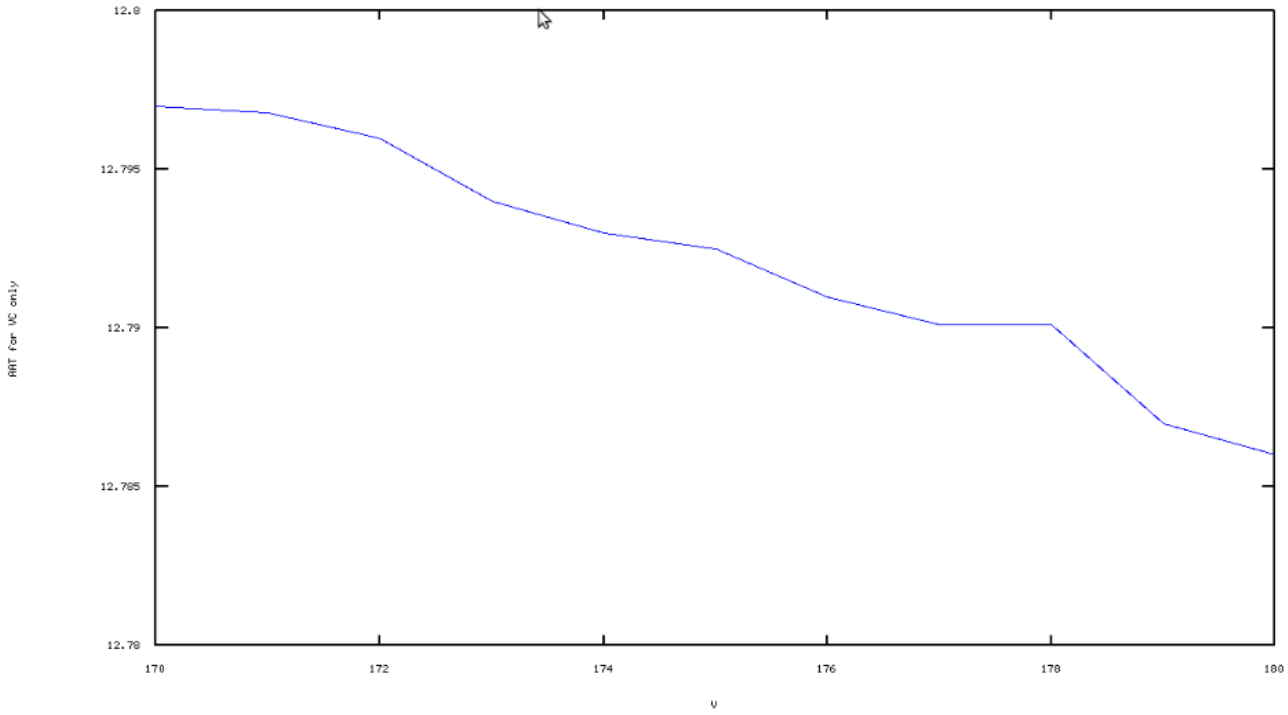



**Here I observe the vacillations I expected in AAT with increasing K. This proves the "dumb nature of the prefetcher and observing how the average value has saturated (roughly after K = 60), I assume my minimum to lie in this range, *i.e the pattern would repeat itself.* From above graph I obtain K = 120 for AAT min = 4.43 units.**

**Hence my FINAL CACHE CONFIGURATION IS: C = 15, B = 6, S = 1, V = 182, K = 120 for AAT min = 4.43 units for ASTAR.TRACE.**

Part b.) Victim Cache or Prefetcher Selection:

1. VC Only: I vary V **from 170 to 180 for K = 0** and observe AATs.




**As expected, V = 180 gives minimum value of AAT to be 12.786.**

2. Prefetcher only: Vary K from 0 to 160 for V = 0:

**The optimum AAT is achieved for K = 120 again at 5.18.**

**From above we can clearly say that we would prefer the *Strided Prefetcher* for ASTAR.TRACE due to lower optimum AAT at 5.18 units.**

**Conclusions about the Cache:**

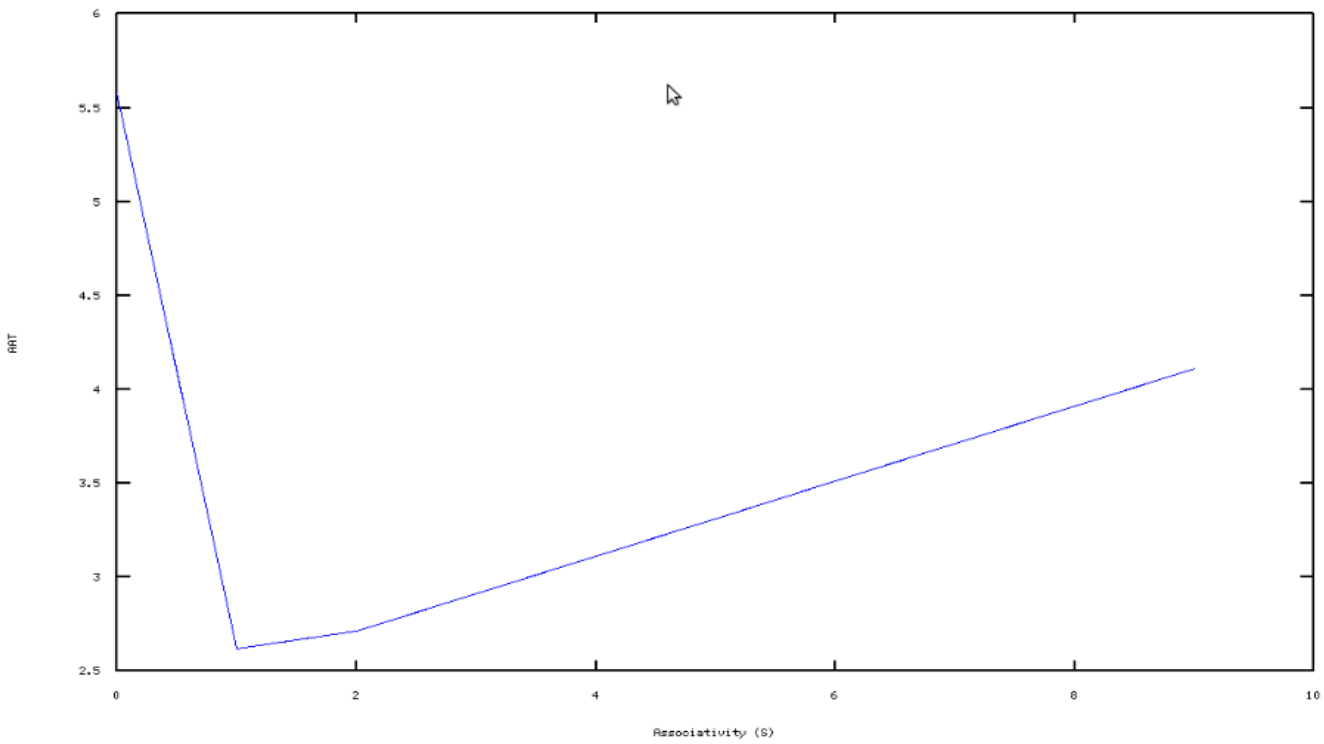
1. The value of V is unreasonably high and we would get a reasonable AAT not too different from current value at much lower values of V. But since we have no restriction on V and have to optimize AAT, we choose highest possible value of V after looking at the trend in the graph. **In a real life scenario, the Victim Cache Size would have a size restriction to adhere to, for power and area optimization.**

2. The value of K is unreasonably high. Since we have a **dumb prefetcher,** the trend of AAT v/s K is random and we need to choose a high value to optimize AAT after concluding a repetition of the given pattern in the graph. In a real life scenario, a more logical prefetcher would be used rather than a strided prefetcher that can replace prefetched blocks.

**BZIP2.TRACE**

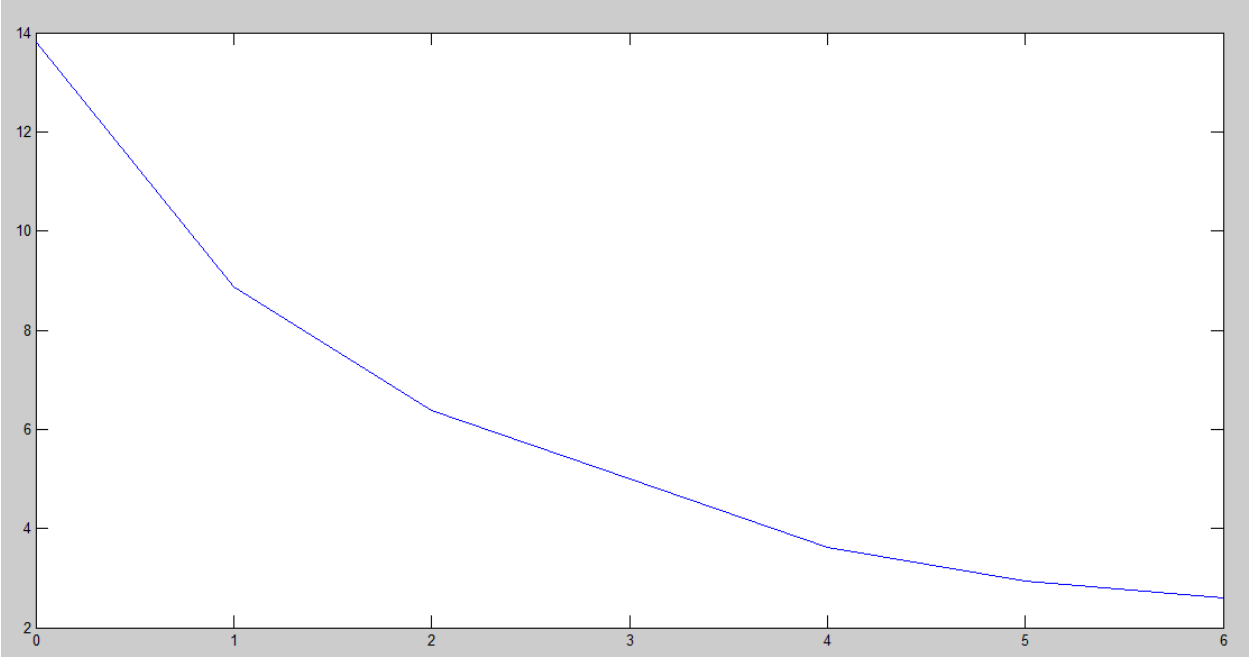
Part a.) Optimum AAT:

1. C =15 V =0 K =0 B = 6, vary S from 0 to 9: (**Note all graphs have AAT on Y-axis**)



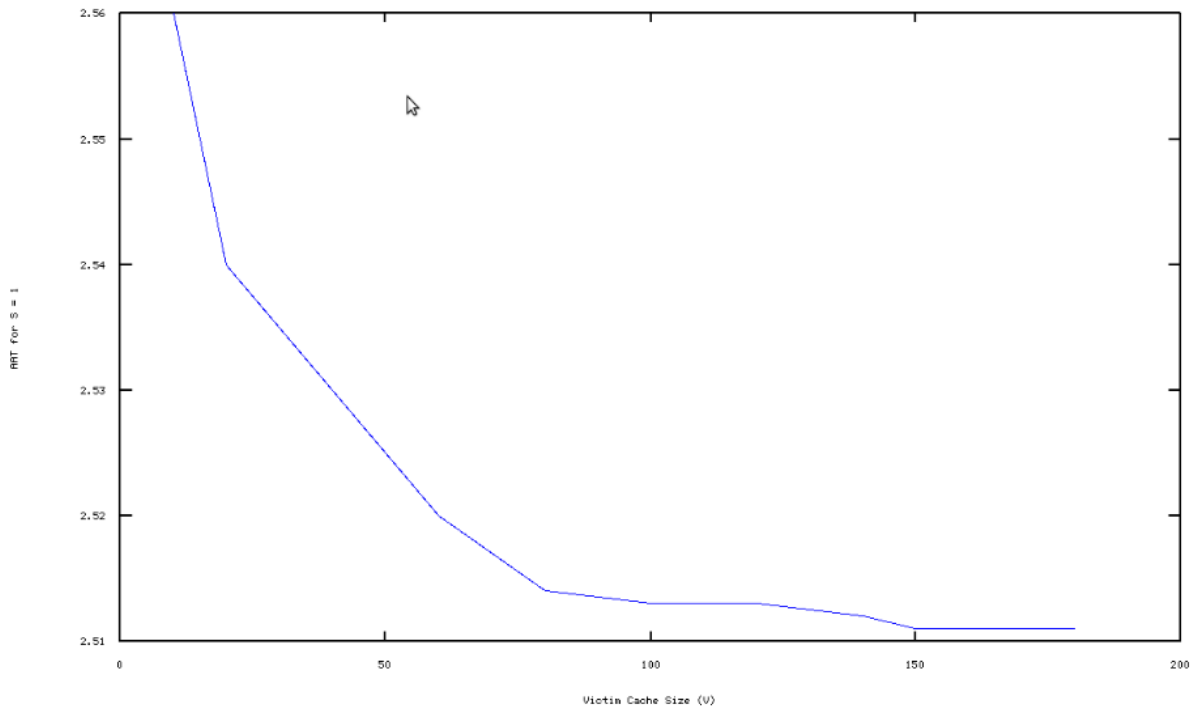

**We observe that AAT is minimum for S = 1 at with AAT min = 2.62. This is an expected trend since AAT reduces initially as miss rate reduces with increasing S, but hit time also increases and when that value takes precedence, AAT will start increasing.**

2. C =15 S = 1 V = 0 K = 0 vary B from 0 to 6: **(B on x axis)**



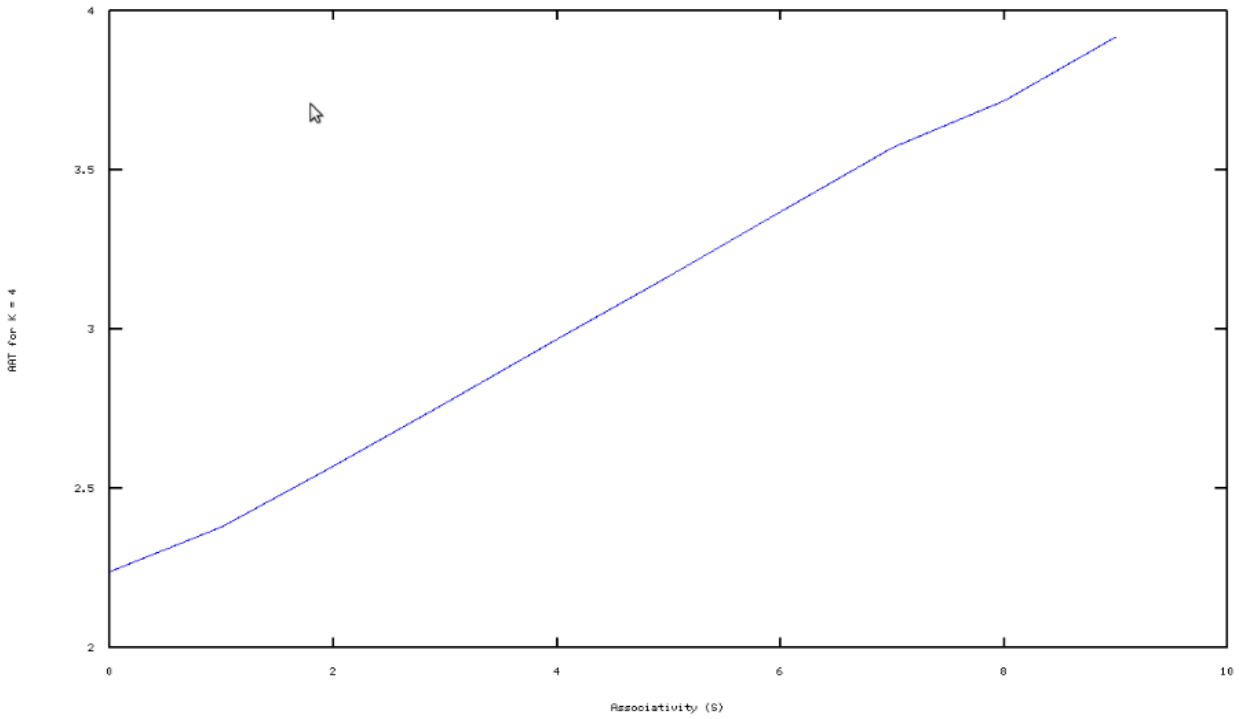

**We observe that AAT reduces uniformly with increasing B and hence the increasing block size is not polluting our cache. Hence B optimum = 6 for AAT min = 2.62.**

3. Now I vary V for values: V = (10 15 25 45 75 85 105 125 145 165 185):



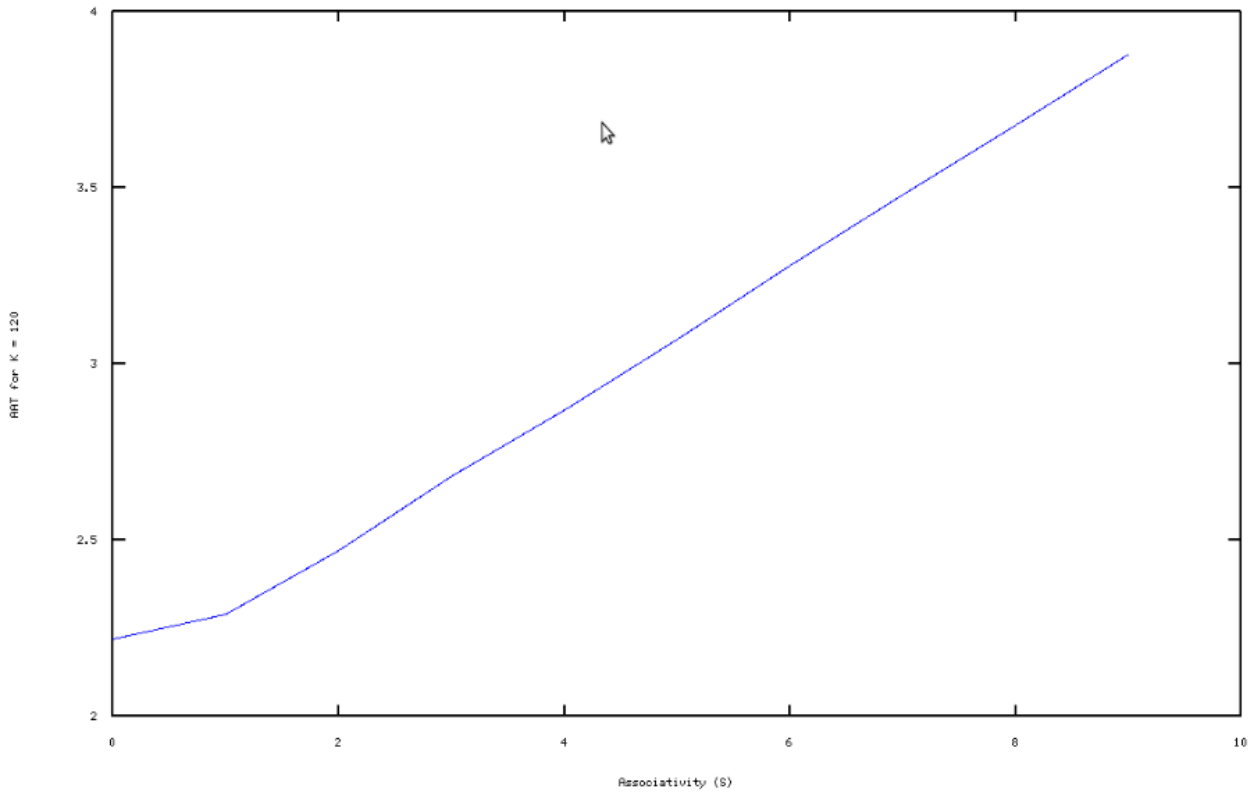
**We observe that AAT reduces with increasing V and saturates after V = 150. Hence following the above curve, I set V =150 to obtain AAT min = 2.511.**

4. Now I turn the prefetcher on: Find optimum S for K = 4:



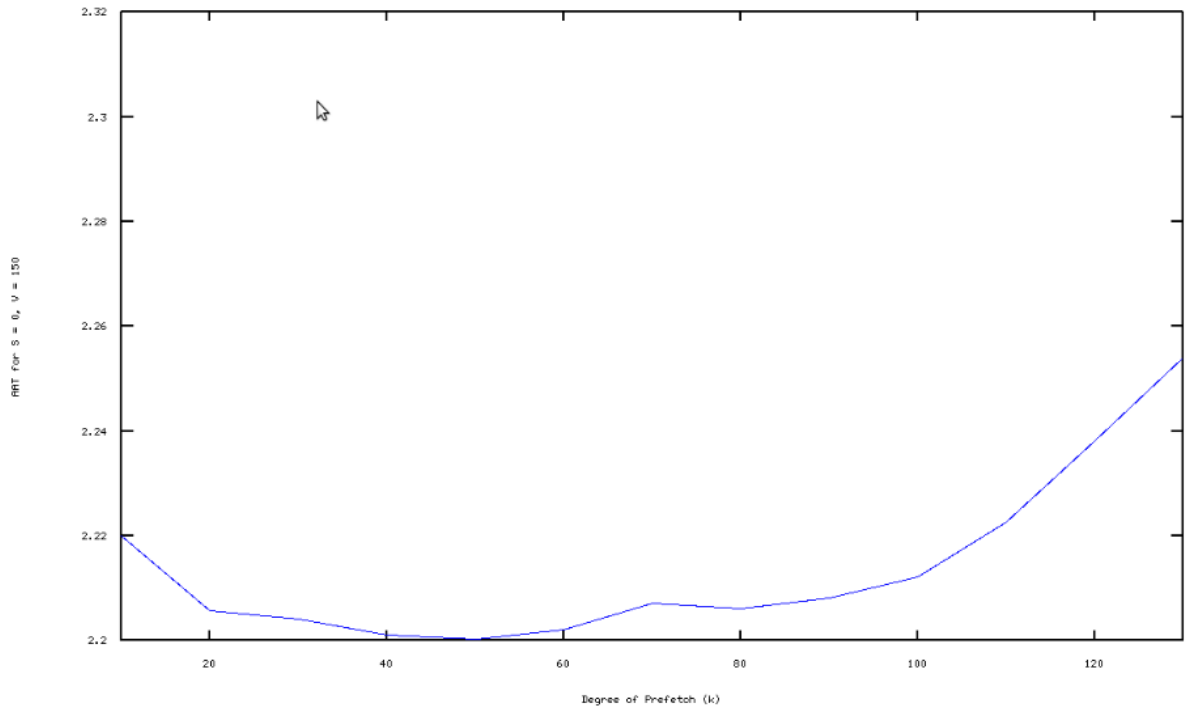

**I find that (S=0) gives me optimum performance. The AATs achieved range from upwards of 2.25 units to just below 4 units.**

5. Test for high values of prefetching. Find optimum S for K = 120.




**Now I find that S = 0 is the optimum case and AATs have reduced to range from 2.22 to roughly 4 units. Hence I consider S = 1 as my final optimum value of S with AAT min = 2.22.**

6. Now that we know S = 0 is an optimum, I vary K for large values to study its pattern. Current setting is C = 15 B = 6 S = 0 V = 150 and I vary K:

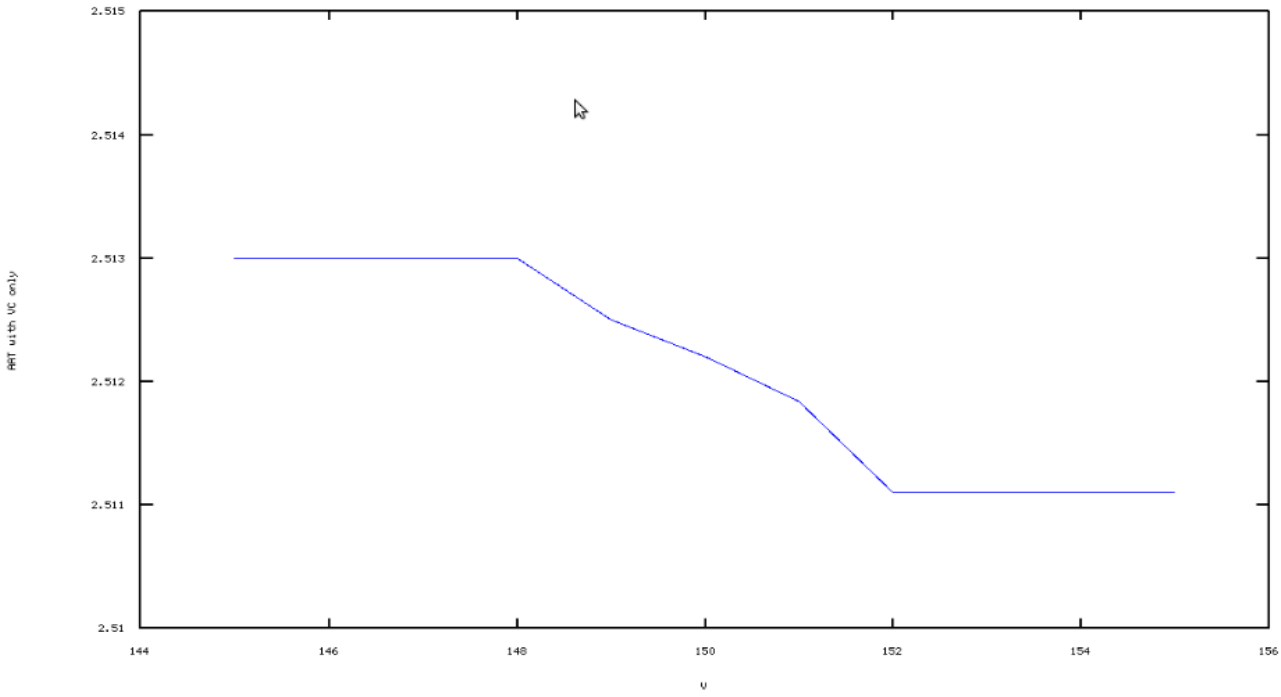



**Here I observe a gentler trend that would be expected from a normal prefetcher for increasing K. This is because I WOULD EXPECT A HIGH DEGREE OF PREFETCHING TO EVENTUALLY POLLUTE MY CACHE LEADING TO INCREASING AAT WITH VERY HIGH K. From above graph I obtain K = 50 for AAT min = 2.20018 units.**

**Hence my FINAL CACHE CONFIGURATION IS: C = 15, B = 6, S = 0, V = 150, K = 50 for AAT min = 2.20018 units for BZIP2.TRACE.**

Part b.) Victim Cache or Prefetcher Selection:

1. VC Only: I vary V **from 170 to 180 for K = 0** and observe AATs.




**As expected, V = 150 gives minimum value of AAT to be 2.511**

3. Prefetcher only: Vary K from 0 to 160 for V = 0:

**The optimum AAT is achieved for K =60 at 5.464.**

**From above we can clearly say that we would prefer the *Victim Cache* for BZIP2.TRACE due to lower optimum AAT at 2.511 units.**

**Conclusions about the Cache:**

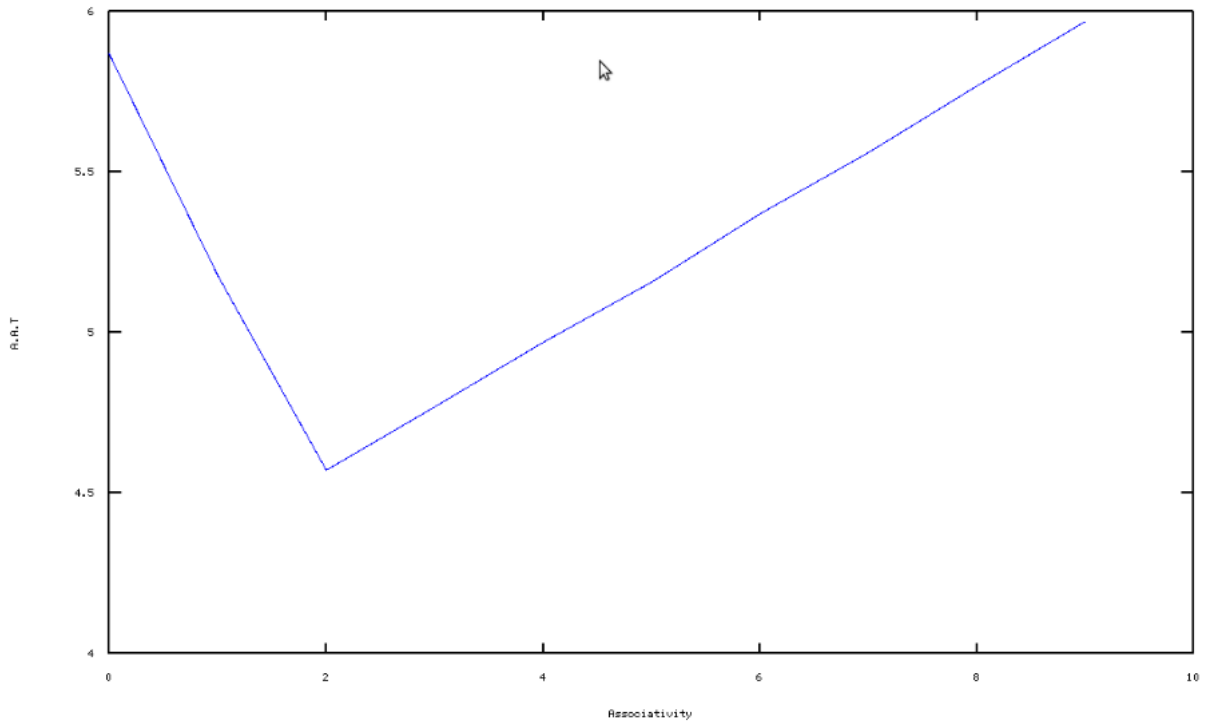
1. The value of V is unreasonably high and we would get a reasonable AAT not too different from current value at much lower values of V. But since we have no restriction on V and have to optimize AAT, we choose highest possible value of V after looking at the trend in the graph. **In a real life scenario, the Victim Cache Size would have a size restriction to adhere to, for power and area optimization.**

2. The value of K is unreasonably high. Since we have a **dumb prefetcher,** the trend of AAT v/s K is random and we need to choose a high value to optimize AAT after concluding a repetition of the given pattern in the graph. In a real life scenario, a more logical prefetcher would be used rather than a strided prefetcher that can replace prefetched blocks.

**MCF.TRACE**

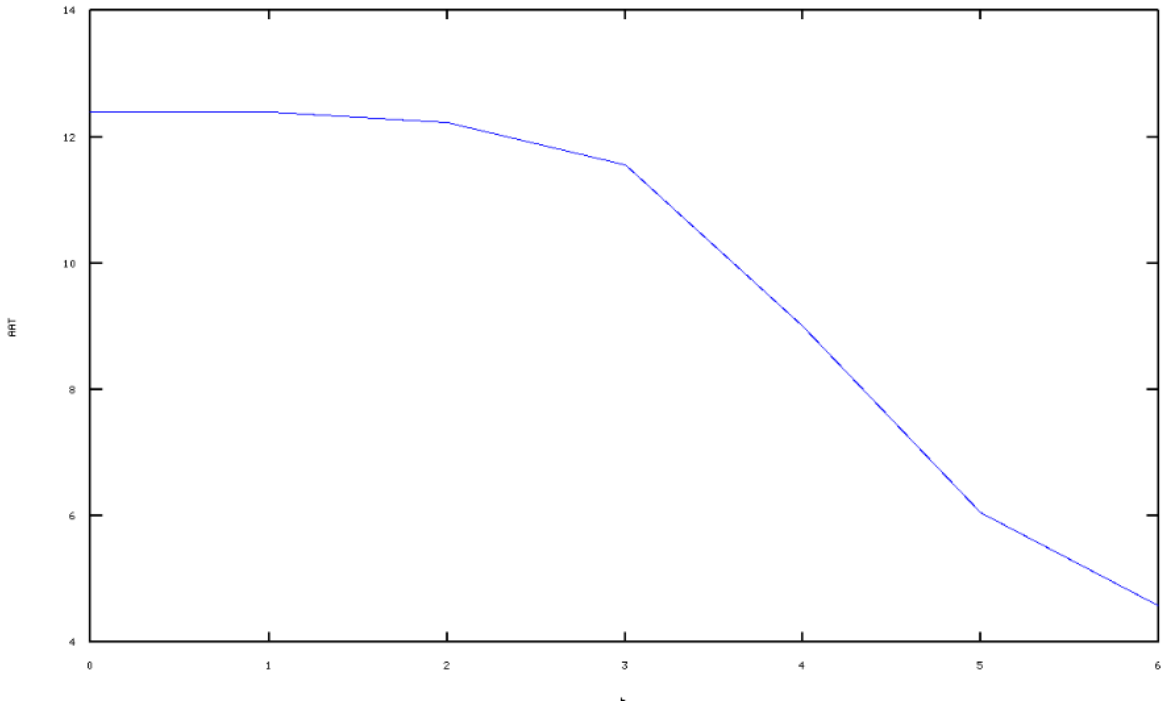
Part a.) Optimum AAT:

1. C =15 V =0 K =0 B = 6, vary S from 0 to 9: (**Note all graphs have AAT on Y-axis**)



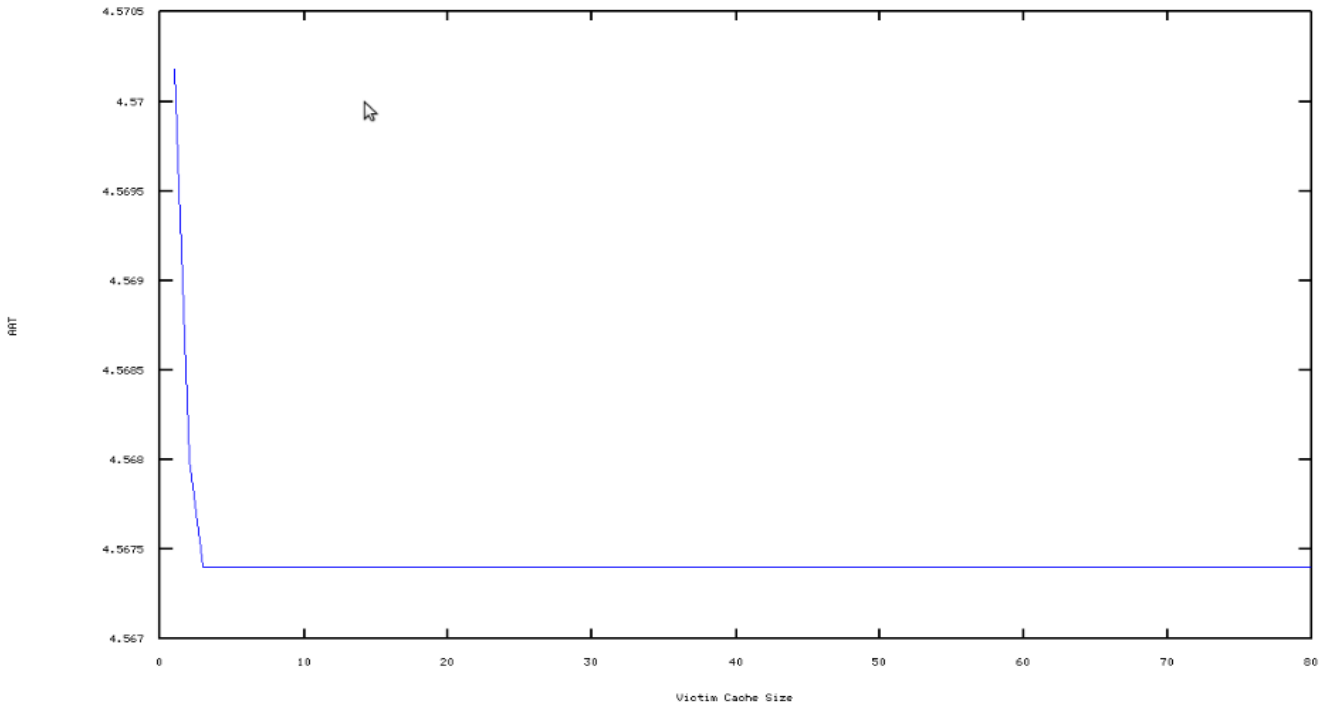

**We observe that AAT is minimum for S = 2 at with AAT min = 4.57. This is an expected trend since AAT reduces initially as miss rate reduces with increasing S, but hit time also increases and when that value takes precedence, AAT will start increasing.**

2. C =15 S = 2 V = 0 K = 0 vary B from 0 to 6: **(B on x axis)**



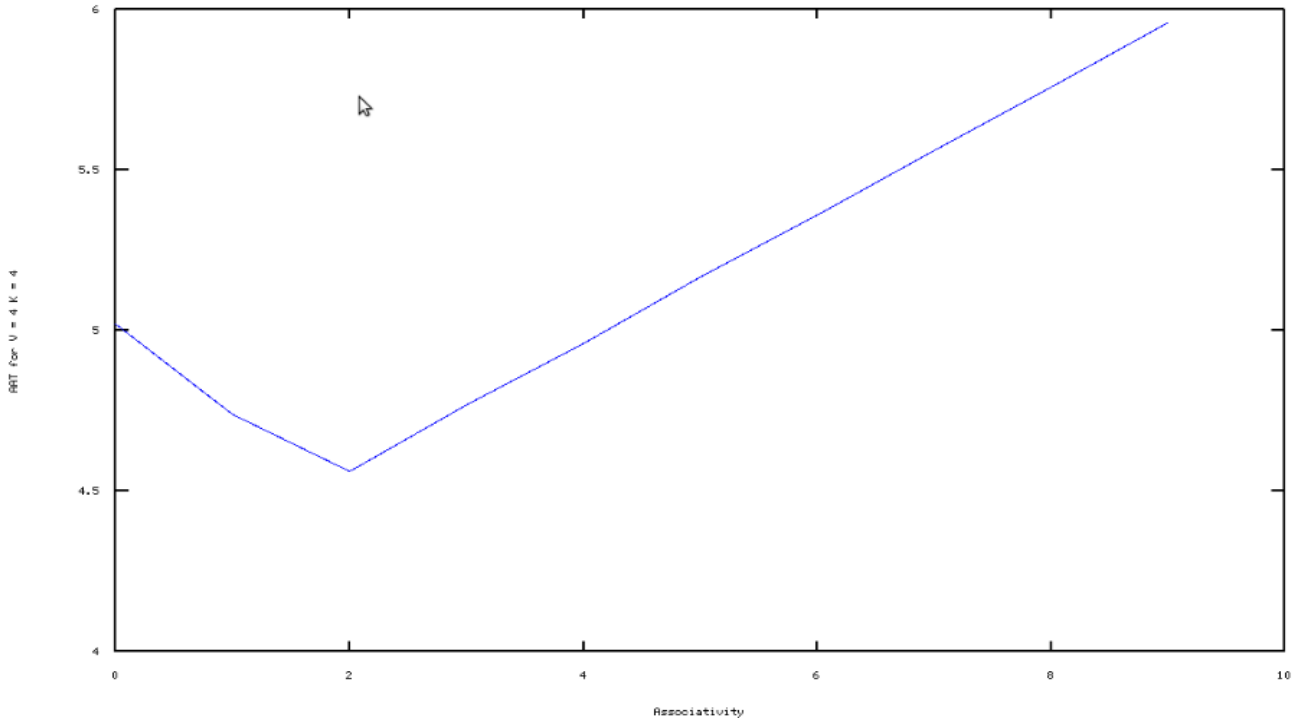

**We observe that AAT reduces uniformly with increasing B and hence the increasing block size is not polluting our cache. Hence B optimum = 6 for AAT min = 4.57.**

3. Now I vary V for values: V = (10 15 20 25 30 35 40 45 50 55 60 65 75 80 85):



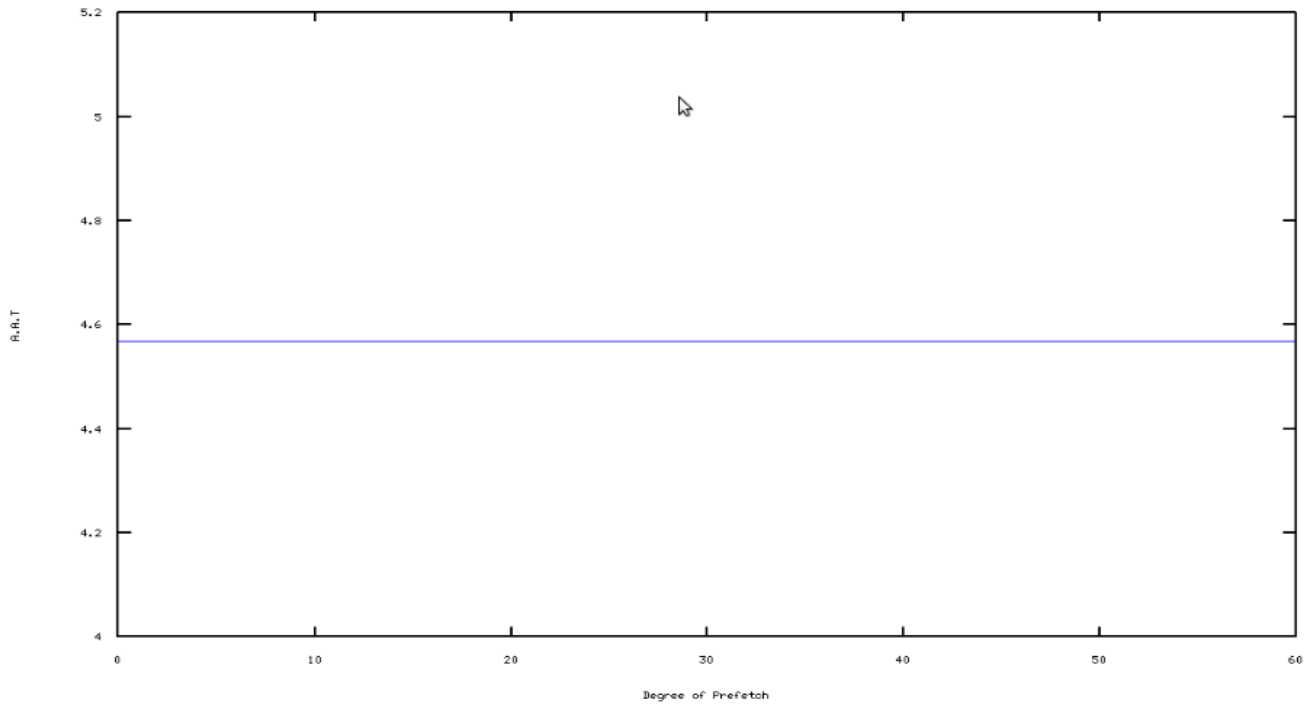
**We observe that AAT saturates at AAT min = 4.5674 for V = 4 hence we choose V = 4 as our optimum cache size. This is an expected trend since eventually the VC is expected to saturate on AAT if we keep increasing its size.**

4. Now I turn the prefetcher on: Find optimum S for K = 4:




**We observe S optimum = 2 with AAT min = 4.5674.**

5. Since the AAT remains unchanged despite turning on the prefetcher, I vary K for S = 2 to see if it does contribute to AAT on increments:




**Clearly, AAT does not change upon increasing K to any value. Hence K = 0 is optimum for our cache since prefetching is of no use.**

**Hence my FINAL CACHE CONFIGURATION IS: C = 15, B = 6, S = 2, V = 4, K = 0 for AAT min = 4.5674 units for MCF.TRACE.**

**Since, the prefetcher has no contribution to AAT for this trace, I choose a VICTIM CACHE as my solution if I could choose only one of the two optimizations.**

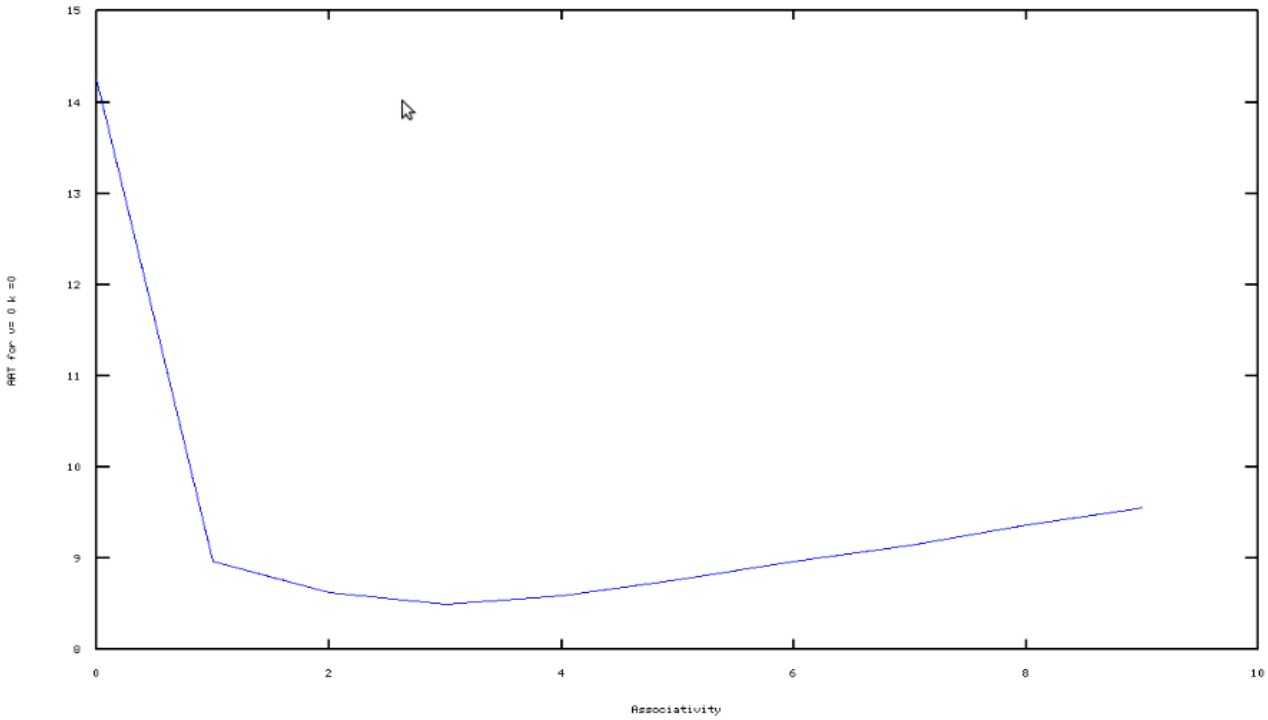
**Conclusions about the Cache:**

I find this a **REASONABLE** Cache Design since V and K don't have extremely high values. The graphs obtained showed expected trends. This was perhaps due to the much lower number of times this trace happens to satisfy a prefetch condition. Hence the "dumb prefetcher" is essentially the most unreasonable component of our Cache previously.

**PERLBENCH.TRACE**

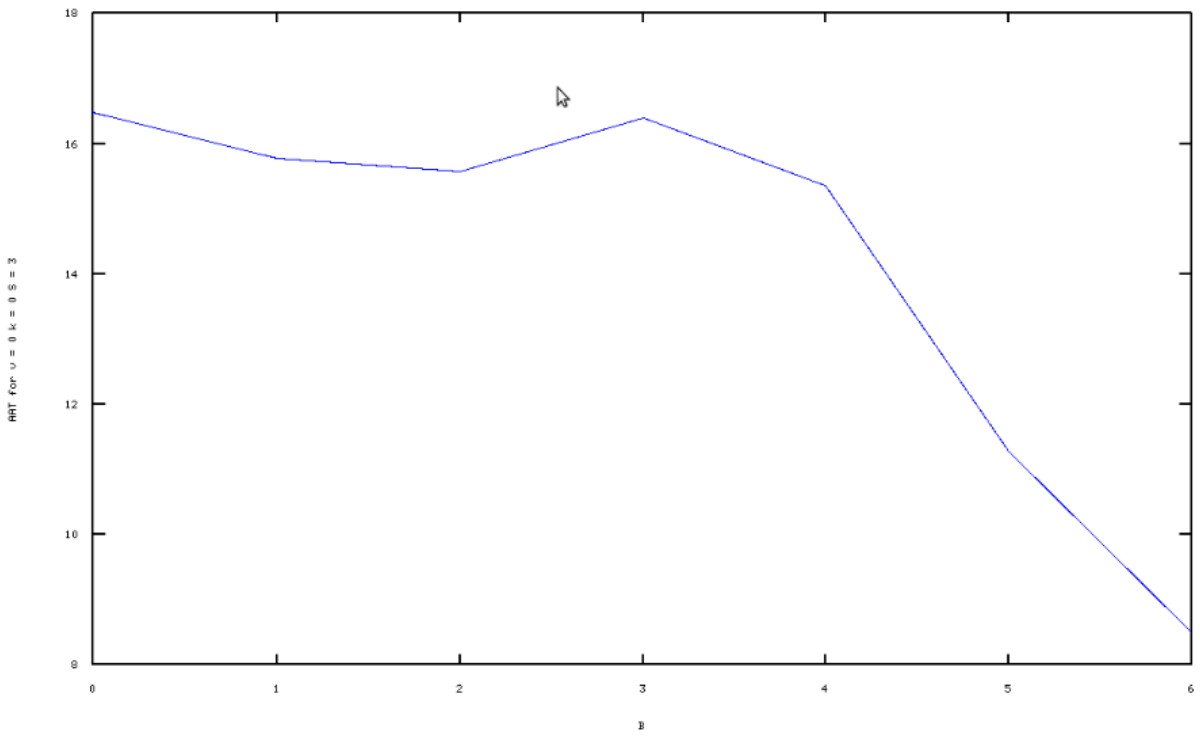
Part a.) Optimum AAT:

1. C =15 V =0 K =0 B = 6, vary S from 0 to 9: (**Note all graphs have AAT on Y-axis**)



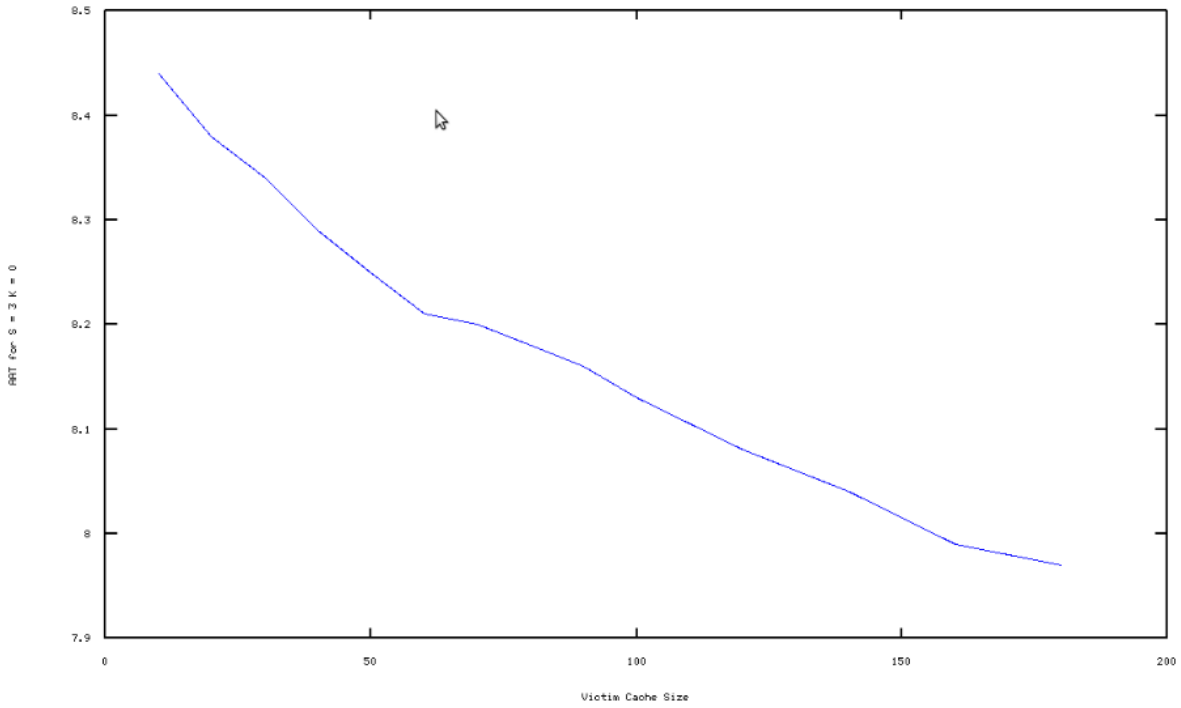

**We observe that AAT is minimum for S = 3 at with AAT min = 8.48. This is an expected trend since AAT reduces initially as miss rate reduces with increasing S, but hit time also increases and when that value takes precedence, AAT will start increasing.**

2. C =15 S = 3 V = 0 K = 0 vary B from 0 to 6: **(B on x axis)**



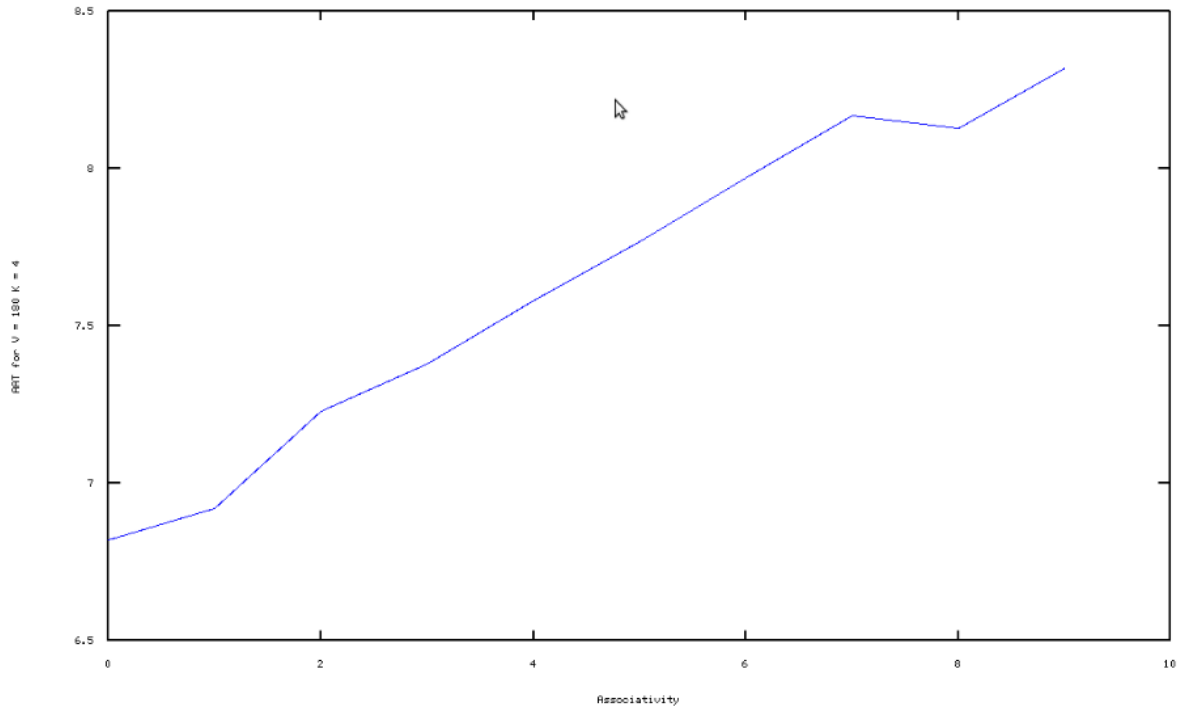

**We observe that AAT reduces uniformly with increasing B and hence the increasing block size is not polluting our cache. Hence B optimum = 6 for AAT min = 8.4874.**

3. Now I vary V for values: V = (10 15 25 45 75 85 105 125 145 165 185):



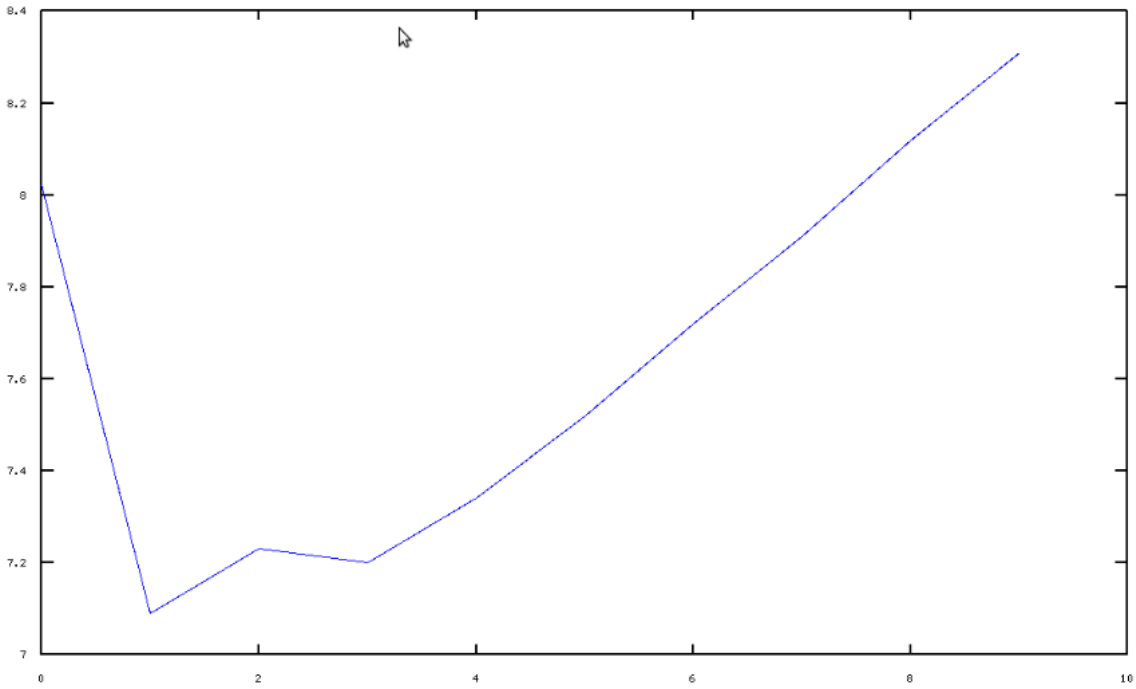

**We observe that AAT reduces with increasing V. We know for C =15, B =6, S = 2, my cache can support maximum V as 180. Hence following the above curve, I set V =180 to obtain AAT min = 7.97.**

4. Now I turn the prefetcher on: Find optimum S for K = 4:



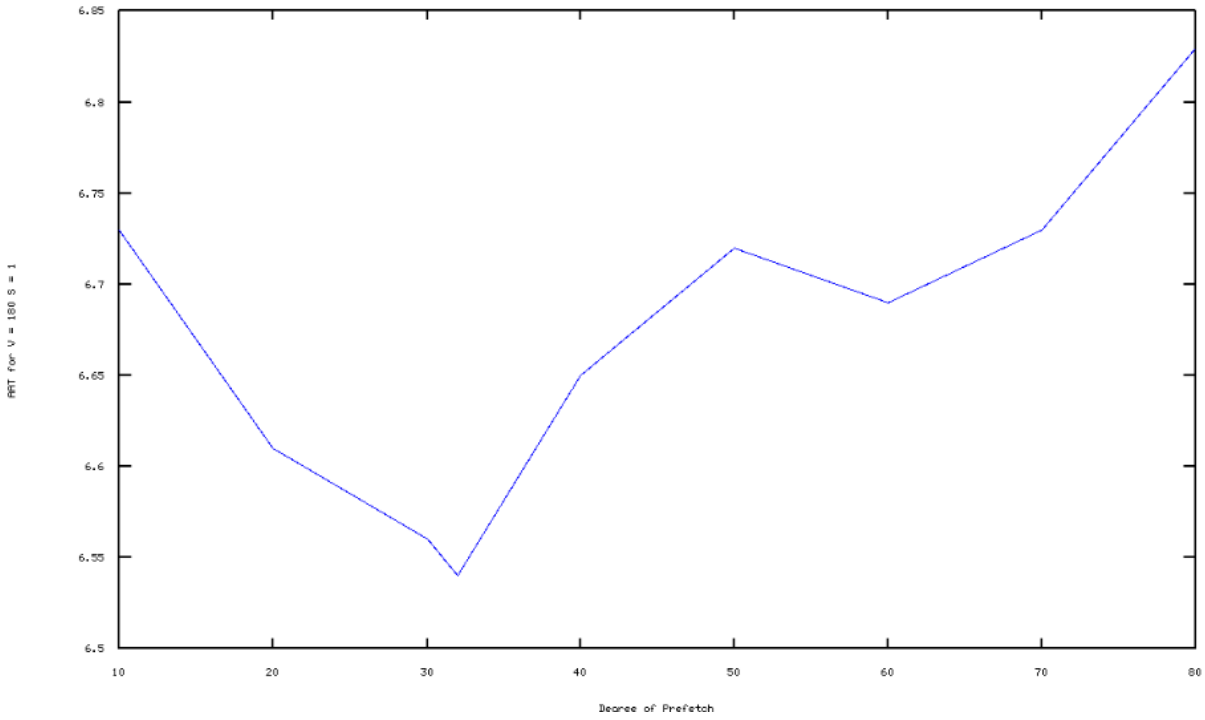
**I find that the cache (S=0) gives me optimum performance. AAT min = 6.82.**

5. Test for high values of prefetching. Find optimum S for K = 120. (S on X axis)




**I find that the cache (S=1) gives me optimum performance. AAT min = 6.82.**

6. Now that we know S = 1 is an optimum, I vary K for large values to study its pattern. Current setting is C = 15 B = 6 S = 1 V = 180 and I vary K:

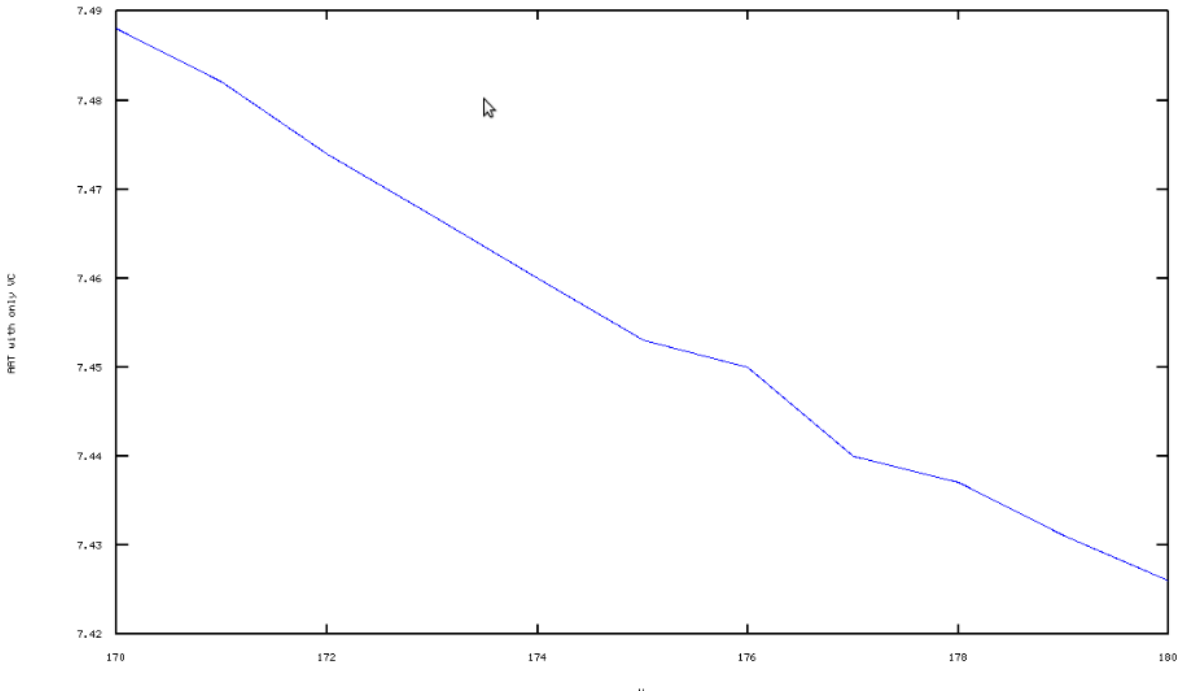



**Here I observe another gentle behavior of increasing *K polluting the cache*. From above graph I obtain K = 32 for AAT min = 6.54 units.**

**Hence my FINAL CACHE CONFIGURATION IS: C = 15, B = 6, S = 1, V = 180, K = 32 for AAT min = 6.54 units for PERLBENCH.TRACE.**

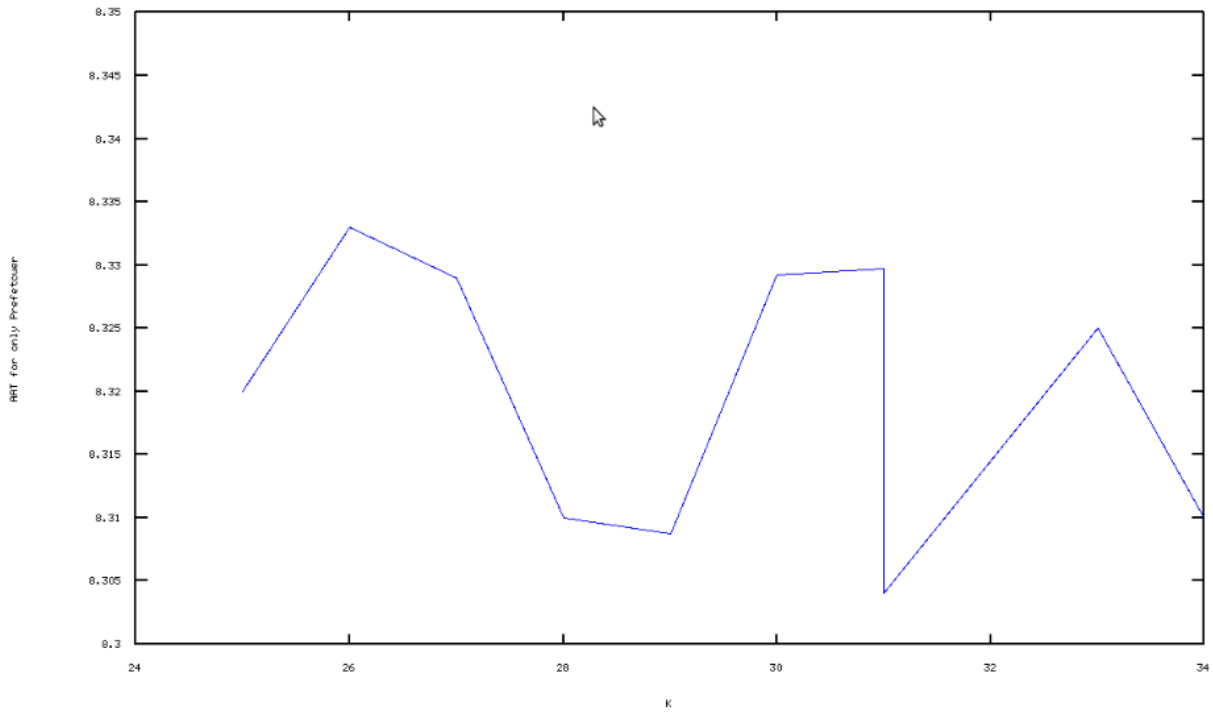
Part b.) Victim Cache or Prefetcher Selection:

4. VC Only: I vary V **from 170 to 180 for K = 0** and observe AATs.




**As expected, V = 180 gives minimum value of AAT to be 7.426.**

5. Prefetcher only: Vary K from 0 to 160 for V = 0:




**The optimum AAT is achieved for K = 31 at 8.29.**

**From above we can clearly say that we would prefer the *VICTIM CACHE* for PERLBENCHs.TRACE due to lower optimum AAT at 5.18 units.**

**Conclusions about the Cache:**

3. The value of V is unreasonably high and we would get a reasonable AAT not too different from current value at much lower values of V. But since we have no restriction on V and have to optimize AAT, we choose highest possible value of V after looking at the trend in the graph. **In a real life scenario, the Victim Cache Size would have a size restriction to adhere to, for power and area optimization.**

4. The value of K is unreasonably high. Since we have a **dumb prefetcher,** the trend of AAT v/s K is random and we need to choose a high value to optimize AAT after concluding a repetition of the given pattern in the graph. In a real life scenario, a more logical prefetcher would be used rather than a strided prefetcher that can replace prefetched blocks.