APRIL 21, 2016

OPERATING SYSTEMS MEMORY SIMULATION

PROJECT – PHASE 2

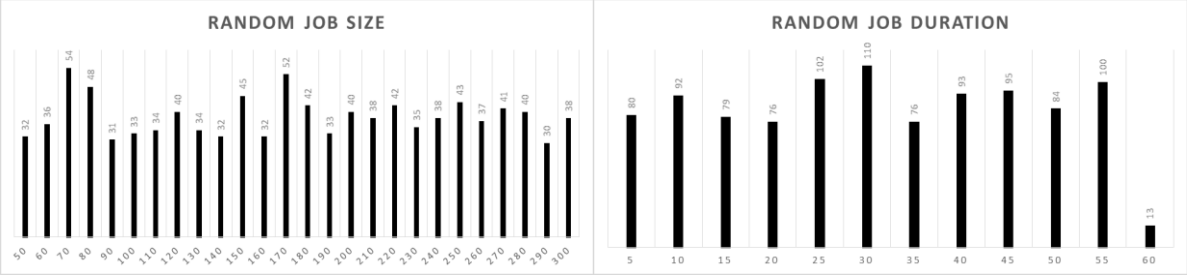
NATHAN LEA
CS4323

Table of Contents

Random Number Generator:

The same random number generator that was used in phase 1 was used in phase 2, the SecureRandom Object. This random number is a cryptographically strong random number generator which produces a nondeterministic output. This ensures that the jobs that are being produced by the simulation are truly random and cannot be guessed based on the last job.

Below is an output of the secure random number generator scaled to the correct mappings for job size and duration respectively each was over 1000 runs:



This graphs show that the random number generator is not symmetric and does not have a normalized distribution, which means that it is truly a random number generator.

Adaptions from Phase 1

Much of the project is very similar to phase 1, but there are several pieces that have been changed to fit the new guidelines for phase 2. The central idea of the simulation has not changed, job is initialized, job is attempted to be placed in memory according to a certain strategy, job is executed, and finally the job is removed from memory. The method for doing this has been altered which forced several changes in several areas in the simulation such as the simulation approach, job representation, memory manager, the addition of the pending list and the round robin scheduler, along with coalescence and compaction.

Simulation Approach

The simulation is event based with a 5 VTU increase each time through the simulation

loop because of the desired Round Robin Schedule with a time quantum of 5.

Job Representation

A job is represented by a block of memory that is equivalent to the size of that job. For easy of locating and gathering data about the job, each job has a job header represented below:

PID	Process Identifier
DURATION REMAINING	Duration left of the Job
DURATION	Duration of the Job
VTU	Time job entered memory
PID	Process Identifier

Every job in memory will have this header, if the job is larger than 50K then the rest of memory occupied by the job is filled with the PID of the job. Job PIDs are implemented in a similar way to a basic UNIX system where as each job is created the PID is increased by one.

Duration remaining is the time that is left to execute for the job, each time the CPU is allowed to execute on the job this number is decreased by the time it was executed. This allows the CPU to know how much time is left on the job and when to remove the job from memory because it has been completed.

Memory Manager

The memory manager is given a job and based on what algorithm the simulation is using, first, best, or worst fit, it will attempt to place the job in memory. If there a hole that is able to fit the job it adds the job PID to the ready queue. If it is unable to find a hole large enough in memory it places the job header into the pending list. It will not ever reject job since there will always be a large enough hole at some point in the simulation to fit the job. When removed jobs from memory the memory manager now just replaces that block of memory with zeros to allow for immediate coalescence.

Pending List

The pending list is an array of job headers which were unable to fit into memory when they first arrived in the simulation. This pending list is checked top to bottom every time a job is completed to see if it can fit any jobs in memory. This is truly a list of jobs and thus the entire list is checked and it will attempt to fit all jobs in memory every time a job completes. As per requirement this list has priority over any new jobs arriving in the system, meaning that all the list is attempted to be placed in memory and then the new arriving job is attempted to be placed in memory.

This method of FIFO pending list has a problem where the simulation is giving priority to shorter jobs and as the simulation rolls on there is mostly likely starvation for large jobs in the pending list. However, this will allow the simulation to complete more jobs since the simulation is not being stalled by waiting on memory for a large hole to open up. With the addition of compaction and coalescence this stalling for large jobs is less of a problem than it would be in phase 1, since all the small holes will create one large hole and allowing for no memory to be lost to fragmentation.

Round Robin Scheduler

The ready queue is a circular queue that contains all the jobs currently in memory. As a job is executed on is moved to the bottom of the list. When a job is added to the ready queue it is placed on the bottom of the list or behind the current pointer.

CPU

The CPU has a time quantum of 5 VTU, which means is only operates for 5 VTUs per job per loop through the simulation, each time a job is operated on the durations remaining header value is reduced by 5.

Dispatcher

The dispatcher is responsible for removing the job from memory when the remaining duration is 0.

Coalescence

Immediate coalescence is added as a job leaves memory. Immediate coalescence is achieved my clearing memory and setting all holes to zeros. This default value for memory automatically combines creating bigger holes if there is any small holes (zeros) directly around the job that just got removed from memory.

Compaction

Compaction was added to combine all the holes into one giant hole in memory. There are three different situations where compaction can be called depending on the options selected for the simulation.

1) Every 250 VTUs
2) Every 500 VTUs
3) Memory Request is denied (Job added to the pending list)

Overview

The simulations have 9 different options that are being compared.

1) First Fit, Compaction Every 250 VTU
2) First Fit, Compaction Every 500 VTU
3) First Fit, Compaction Every Memory Request Denial
4) Best Fit, Compaction Every 250 VTU
5) Best Fit, Compaction Every 500 VTU
6) Best Fit, Compaction Every Memory Request Denial
7) Worst Fit, Compaction Every 250 VTU
8) Worst Fit, Compaction Every 500 VTU
9) Worst Fit, Compaction Every Memory Request Denial

Memory Placement Strategies

First Fit

First fit is a memory placement strategy that looks for the first hole that is big enough to place the job in and the places the job there.

Best Fit

Best fit is a memory placement strategy that looks for the best possible current hole, meaning the hole that would create the smallest hole and places the job there.

Worst Fit

Worst fit is very similar to best fit except that it finds the spot that would create the worst possible hole, or largest left over hole and places the job there.

Implementation First Fit vs Best Fit vs Worst Fit

The implementation of First Fit was very easy as you are just required to look through memory until there is place large enough for the job and place the job there. Very simple to implement and requires less overhead than the other two placement strategies to implement as there is no variables needed to store the best location in memory. Overall, this was by far the easy memory placement strategy to implement.

Next, Best Fit and Worst Fit these two memory placement strategies are very similar in one is looking for the best hole and one is looking for the complete opposite. These were almost identical to implement, just a change of a sign. They were not significantly harder to implement than first fit but they were more difficult, in that you had to search the entire memory looking at every memory block to find the best/worst place to put the job.

Implementation Compaction Scenarios

Compaction is very easily preformed when the performance of the algorithm is very important. The algorithm in the simulation is an $O(n^2)$ algorithm that looks for the first zero value and then searches for the next non zero value and swaps them and continues to do this until all the zero memory has been combined at the bottom of them memory array.

The three scenario were very easy to implement, the 250 and 500 VTUs just compare the current VTU time and see if it mods evenly with 250 or 500 respectively and then run the compaction method. The scenario that runs when a memory request is denied was similar in difficulty, each time a job was added to the pending list, the compaction method was called too.

Performance

In comparing performance many things have to be considered such as jobs completed, pending list size, round robin scheduler size, average hole, fragmentation, storage utilization and average turn-around time, as well as the comparison between the different compaction scenarios for each.

First Fit – Comparing Compaction Scenarios

Completed Jobs –Chart A

Comparing the slopes and final values of the completed jobs gives the perception that compaction at memory denial is the better of the scenarios for completed the most jobs, but only by a very small margin. The difference here is very small and just above the standard deviation of the set making it still a valid difference but only barely.

Pending List Size – Chart B

This graph is showing the number of jobs in the pending list, and almost had no difference at when the pending list is full and the jobs stop entering the system. There are no

major improvements between the three scenarios here.

Round Robin Scheduler Size – Chart C

This graph shows the number of jobs in the ready queue at one time, the higher the number here the more multiprogramming that the system could have. This graph shows that the compaction at 250 VTUs gives you a higher number of jobs on average over the simulation window. However, there is only 1-3 more jobs, which is not a significate difference in the three scenarios.

Average Hole Size – Chart D

The average hole size starts to show a difference between the three scenarios, the compaction at memory denial has more times when it completely fills up memory and there are no holes. However, it also has the highest points at when memory is the most empty. The other two scenarios are very consistent throughout memory at the level of blank holes in memory around 30-50 KB at any one time.

Fragmentation – Chart E

Fragmentation has a very similar trend as hole size, in that compaction at memory denial has a much better fragmentation average than any of the other scenarios. Compaction at memory denial has almost no points when there is a significant amount of fragmentation.

Storage Utilization – Chart F

Storage Utilization has a much better picture at how efficiently memory is being assigned in compaction at memory denial, there are many points when the memory is perfectly assigned, where the utilization is 100%. There are almost no times when the utilization of the other compaction scenarios are 100%.

Average Turn-Around Time – Chart G

The average turn-around time of the three scenarios is very similar. They all have a

slightly increasing turn-around time.

Best Fit – Comparing Compaction Scenarios

Completed Jobs –Chart H

Very similar to the first fit case, memory compaction at denial completes the most jobs but there is not many more jobs completed, only around 10 more jobs than the other method of compactions.

Pending List Size – Chart I

The pending list size for all of the scenarios is very similar in that the list is filled to its capacity very quickly and there is no noticeable performance difference between them.

Round Robin Scheduler Size – Chart J

The round robin queue size trend is different than that of first fit, where memory at compaction was the best. For best fit, compaction at every 250 VTUs consistently has the most jobs in memory at once, allows for the most multiprogramming by having the most active jobs at a time.

Average Hole Size – Chart K

The average hole size of the jobs in memory for best fit is pretty even for all of the compaction scenarios with low average hole size of around 3-5 KB. Compaction at memory denial has the most time spend where the average hole size is 0KB, this is reflection even more clearly that compaction at memory denial has the least amount of fragmentation.

Fragmentation – Chart L

Compaction at memory denial has almost a perfect 0 KB of fragmentation from 1000-4000 VTUs. If the system needs to have very low memory fragmentation, then this is the scenario to put it in. The other compaction scenarios have fragmentation that are much higher than compaction at memory denial.

Storage Utilization – Chart M

The storage utilization between the three scenarios is very similar, there is no major differences between them.

Average Turn-Around Time – Chart N

The average turn-around time for each of the scenarios is very similar, they are slightly different values but the overall trend of them is very similar to that of first fit. The compaction scenarios do not provide any specific benefit over the other.

Worst Fit – Comparing Compaction Scenarios

Completed Jobs –Chart O

The number of jobs completed between the three scenarios is very similar to each other and no significant difference is noticed between them. A similar trend as in best fit and worst fit, compaction at memory denial does finish more jobs but the number is not a significant difference.

Pending List Size – Chart P

The three scenarios of memory compaction shows nearly no difference here.

Round Robin Scheduler Size – Chart Q

The round robin scheduler size is larger than that of first fit and best fit, but only by a very small margin. There are no noticeably differences between the compaction strategies.

Average Hole Size – Chart R

The average hole size between the compaction scenarios does not show any trend or difference between the three. Each of them has benefits in this category and neither is that much better than the other.

Fragmentation – Chart S

Comparing the differences between them, compaction at memory denial has very little

fragmentation that is above 0KB, making it the clear winning in this metric. The other compaction scenarios have much higher and sporadic fragmentation throughout the simulation.

Storage Utilization – Chart T

The storage utilization between the three compaction scenarios is very similar to each other and no clear difference is observable.

Average Turn-Around Time – Chart U

The average turn-around time for each job in the three compaction scenarios is very similar to the other compaction scenarios and no significant difference is noted.

Comparing Placement Strategies

Completed Jobs –Chart V

This graph maps all compaction scenarios and all placement strategies, for the 9 different simulations. The trend of every single one of the simulations is almost identical to the last. The simulation that completed the most jobs was Best Fit compaction at memory denial, but this simulation only completed 10 more than the one that completed the least, worst fit compaction every 250 VTUs. This difference is only slightly more than the standard deviation, 8.68 jobs, which means that is only about 2 more jobs than the expected error of the data set, not nearly enough to call it a significant difference.

Storage Utilization – Chart W

The storage utilization of the 9 simulations shows that the simulations are very similar now, most of the time the memory in the simulations was between 85-100% utilized, which is incredible for this type of simulation. Referencing phase 1, the simulation would quickly go to 0% and this has improved the simulation where it no longer even gets below 80% but only a couple times.

Average Turn-Around Time – Chart X

The average turn-around time of all the simulations shows that there are no major waiting conditions that appear from the different scenarios that the memory was tested with. The average-turnaround time is very similar over the entire graph, no simulation is better in this category.

Selection First Fit vs Best Fit vs Worst Fit and Compaction at every 250 VTUs vs every 500 VTUs vs every Memory Denial

Several features have been added in the memory simulation to make the cases very even. There is now memory compaction and memory coalescence. These features alone help the memory to recover from any bad choices that the memory manager made, this normalizes the effects of the worst fit and first fit not having the best choice on which location to place memory. Combine this, with round robin schedule and the CPU is almost always working on some job making the simulation not penalized by long jobs.

The best memory setting therefore cannot be determined by just looking to the ending metrics but rather to complexity and run time of the algorithm. The best memory scenario therefore would be first fit with compaction at every 500 VTUs. First, this setting has the best run time because the complicated algorithm of compaction is run only 10 times in a 5000 VTU simulation which reduces the large run time of that $O(n^2)$ algorithm. Second, first fit is a linear algorithm compared the n^2 algorithms that are best and worst fit, reducing the run time even more.

Improvements Best Best Fit and Worst Worst Fit

After much thought about how to improve the memory placement in the simulation, I decided to look at what would happen if the memory manager would only place a job in the best

ever going to be hole and the worst possible hole for worst. This is what I coined to be best best fit and worst worst fit. This algorithm is trying to place a job in only a location that would be perfect location for it. This algorithm limits the wrong choices that would be made by a non-perfect best fit or worst fit algorithm.

Completed Jobs – Chart Y

This completed job graph describes every simulation option that my program could handle and nothing can break from this normalized slope of completing the average 160 jobs per 5000 VTUs, even when never having to settle in memory for a bad place, the other features normalized this and produced the same output in terms of jobs completed.

Storage Utilization – Chart Z

The storage Utilization graph shows the same result that even when the algorithm is designed to not pick a bad location you get a very similar result because there is not unlimited jobs to pick from and the memory manager eventually has to make a bad choice and place a job to not idle the CPU and even though most of the time there is 95-100% utilization there are points where this will drop because of a forced bad decision, but it will right away be evened out by the jobs coming in behind it.

Selection First Fit vs Best Fit vs Worst Fit vs Best Best Fit vs Worst Worst Fit

After all the different simulation that were made the best solution is still the simplest solution. The fancy best best fit no longer gain the memory manager anything like it did in phase one, these placement strategies just cause longer run times for similar number of jobs in memory. This is just wasting other resources to get insignificantly more memory utilization. The choice is still clear to choose first fit with compaction at every 500 VTU for the simplicity and the shorter run times.

Appendix

Chart A – First Fit

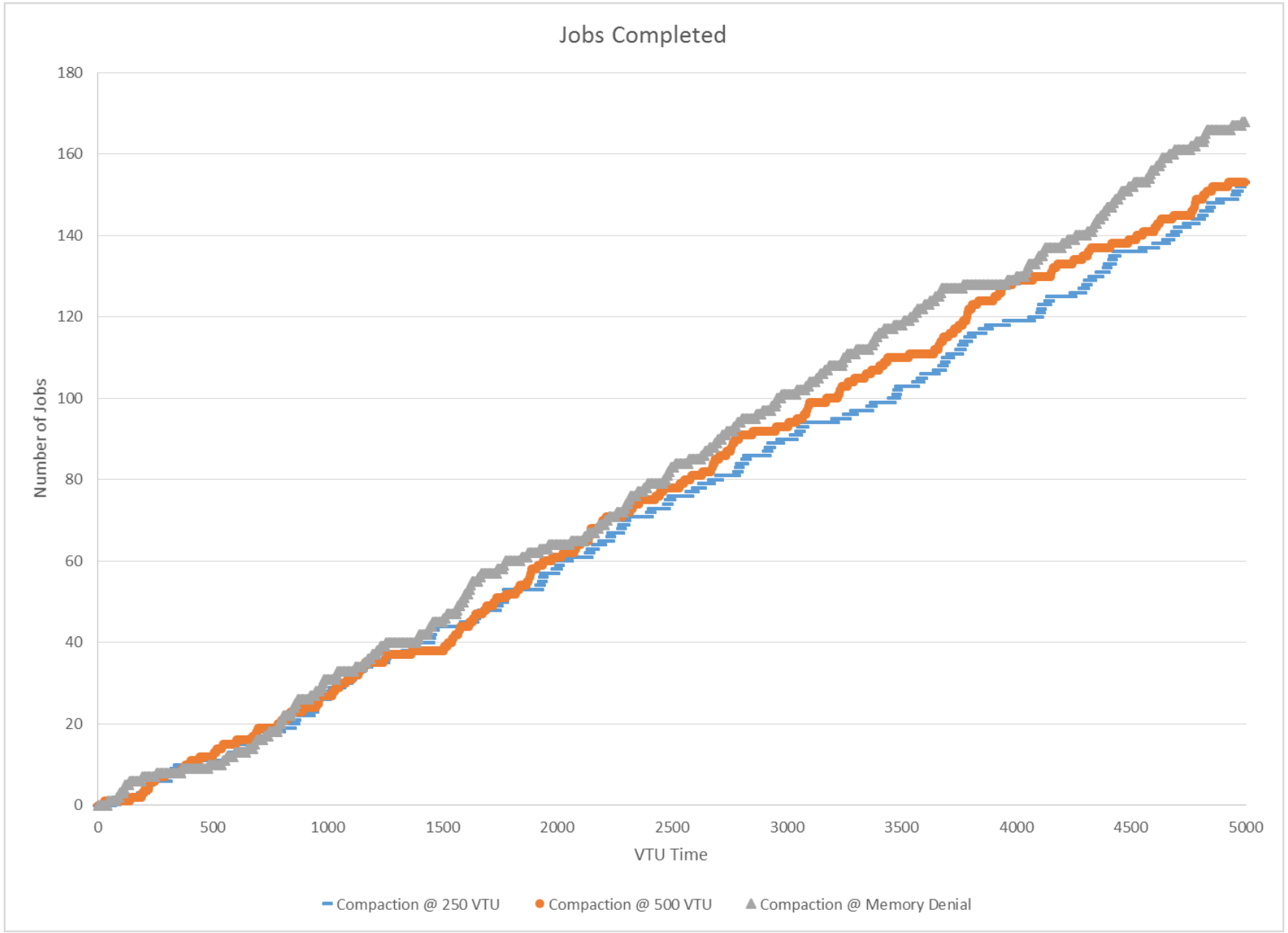


Chart B – First Fit

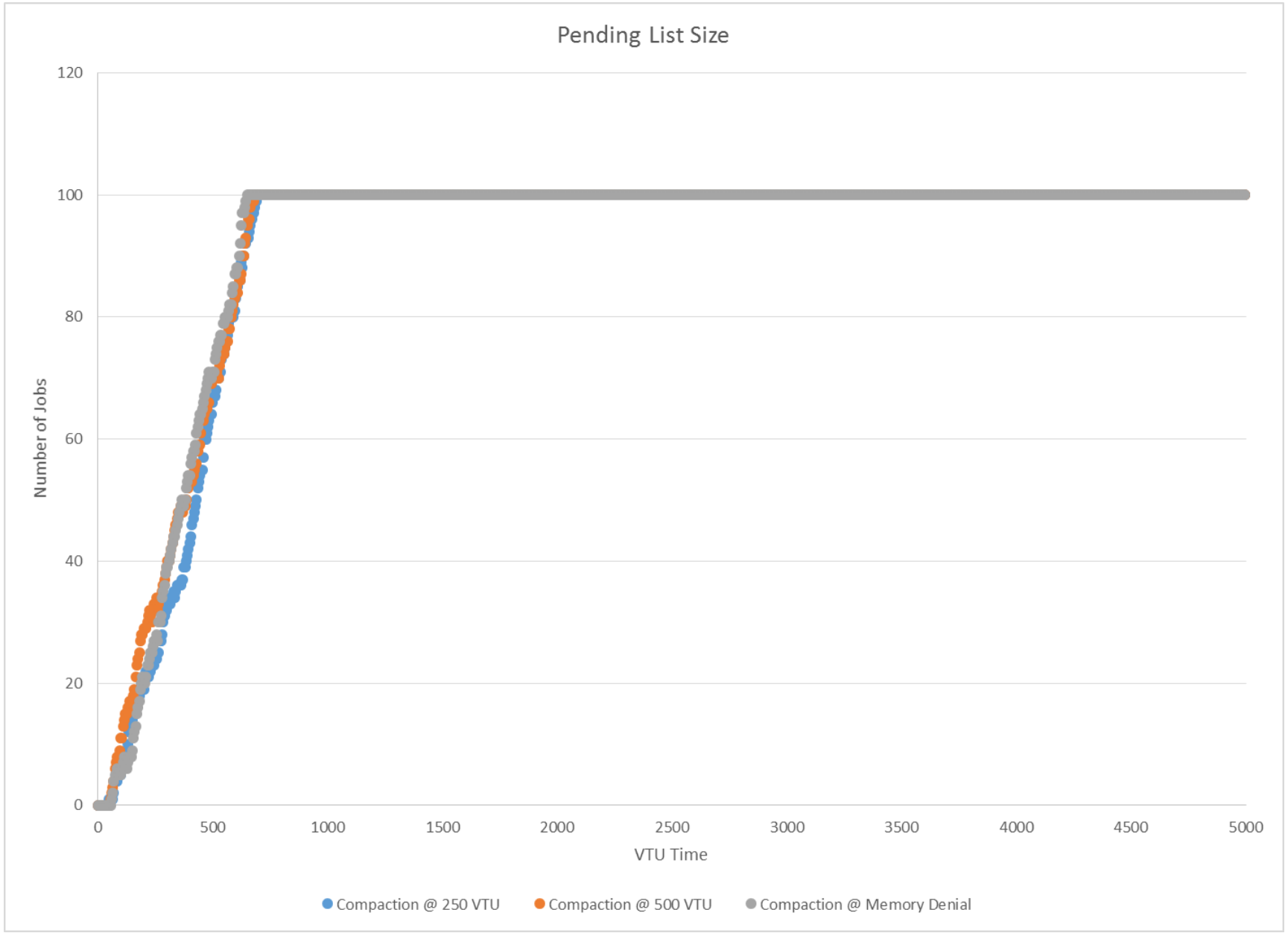


Chart C – First Fit

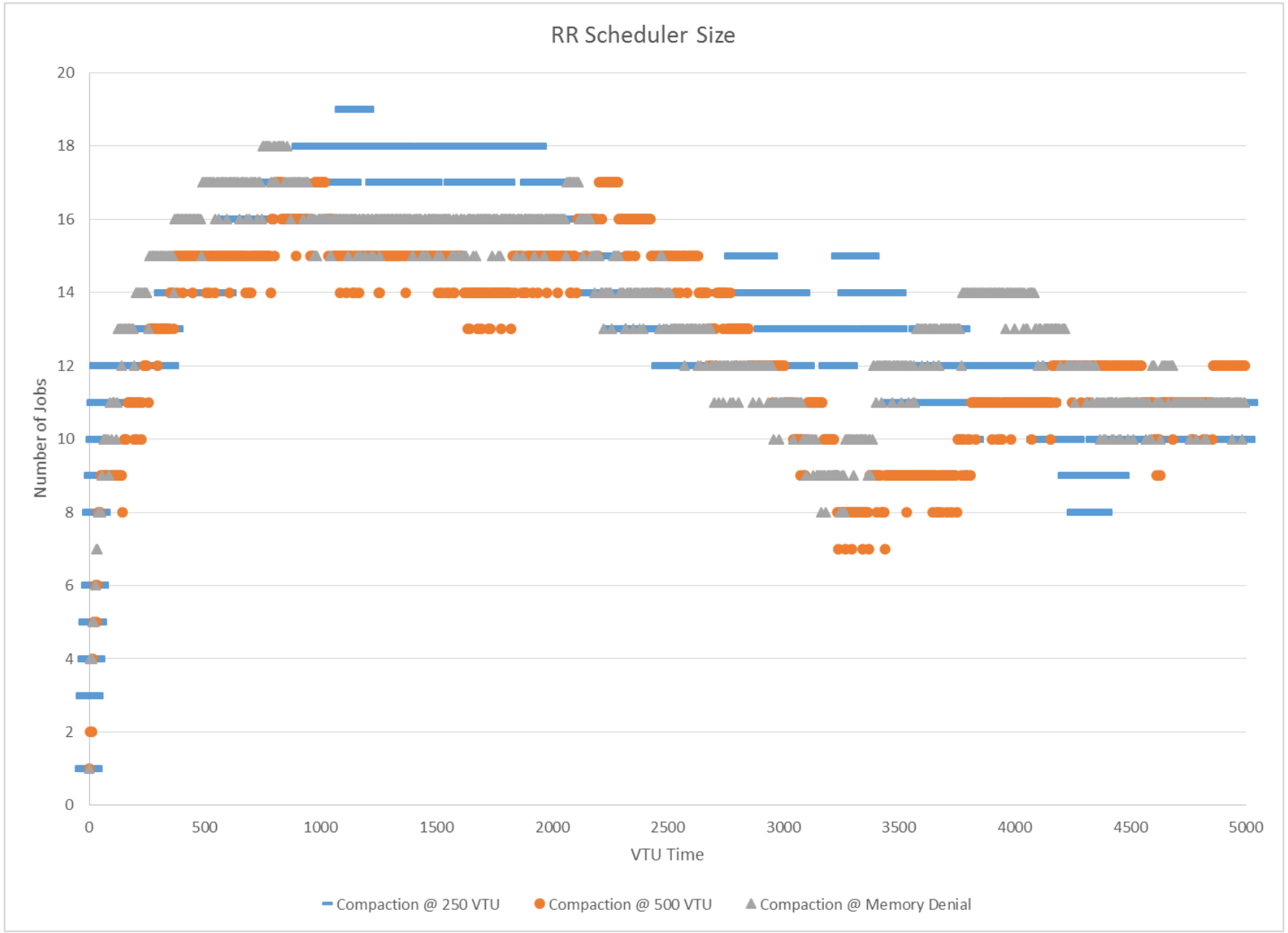


Chart D – First Fit

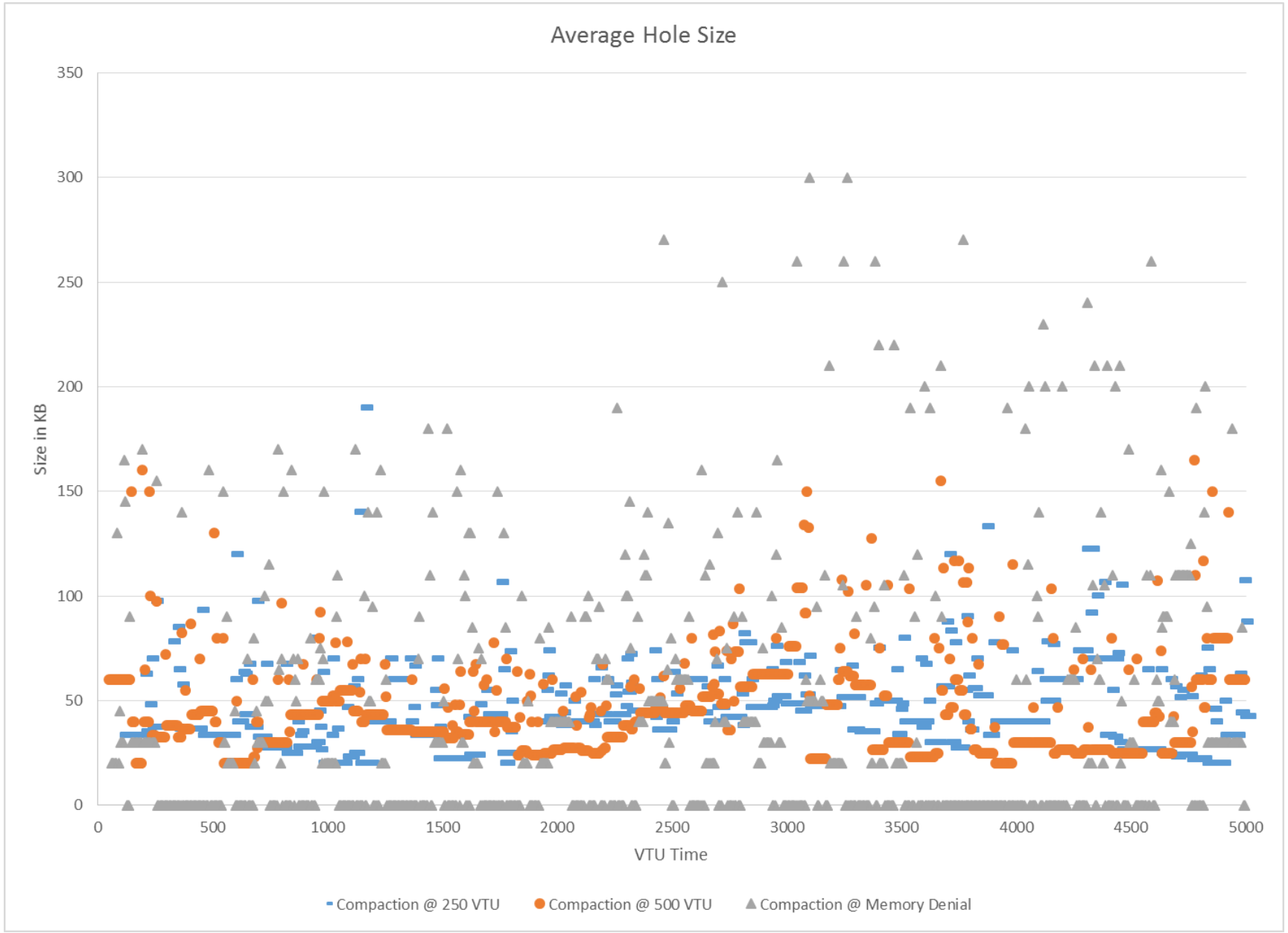


Chart E – First Fit

Chart F – First Fit

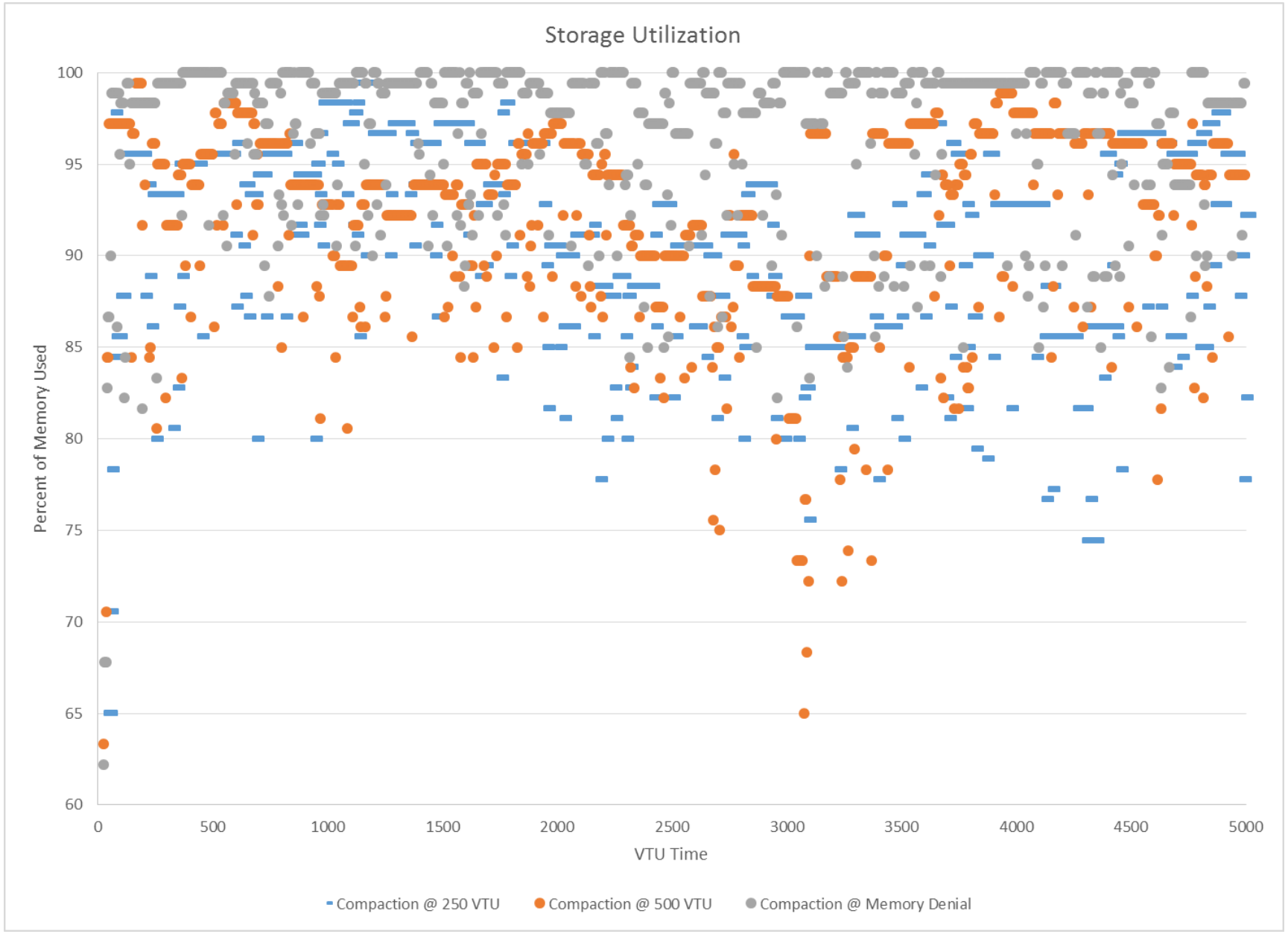


Chart G – First Fit

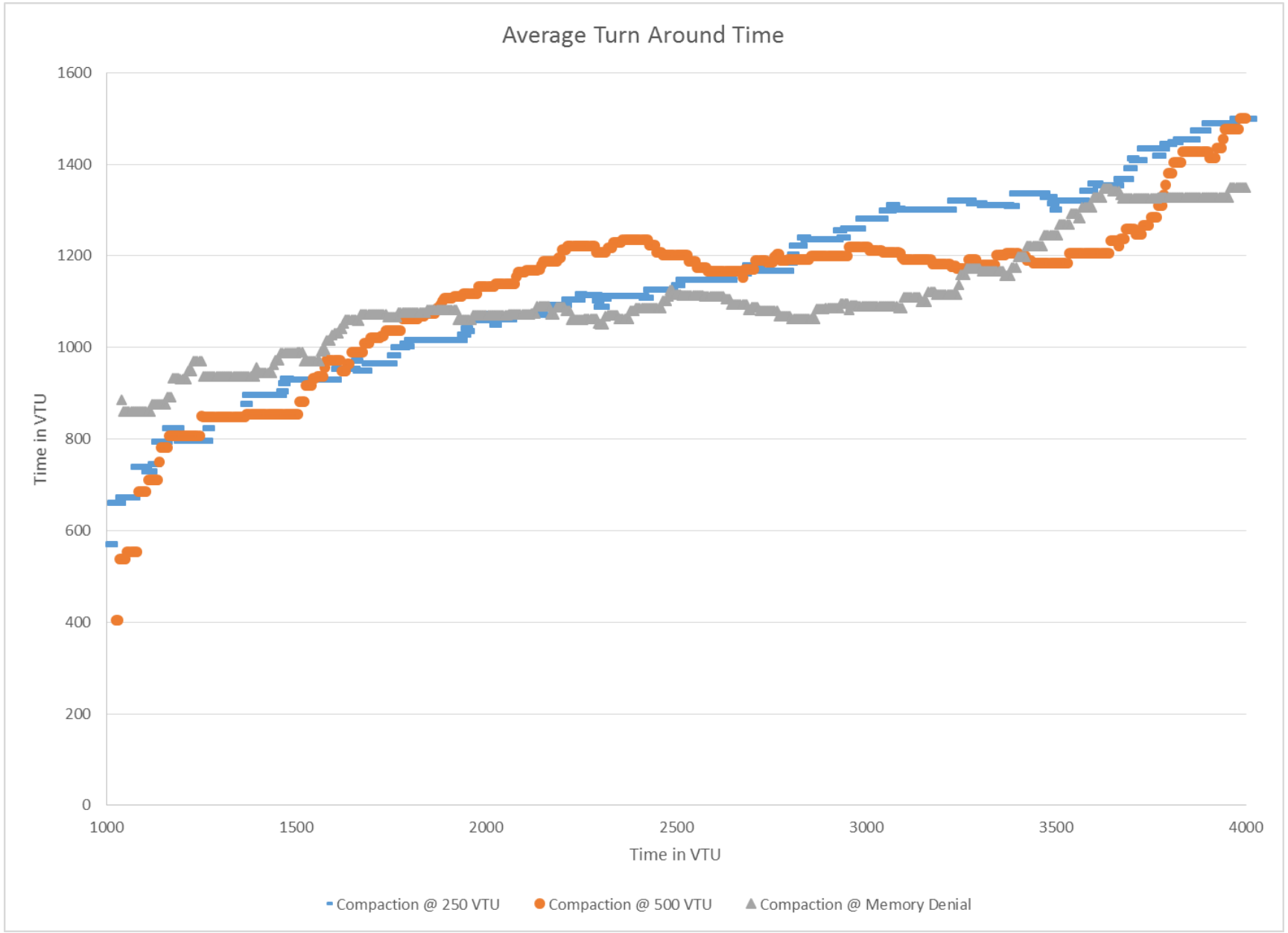


Chart H – Best Fit

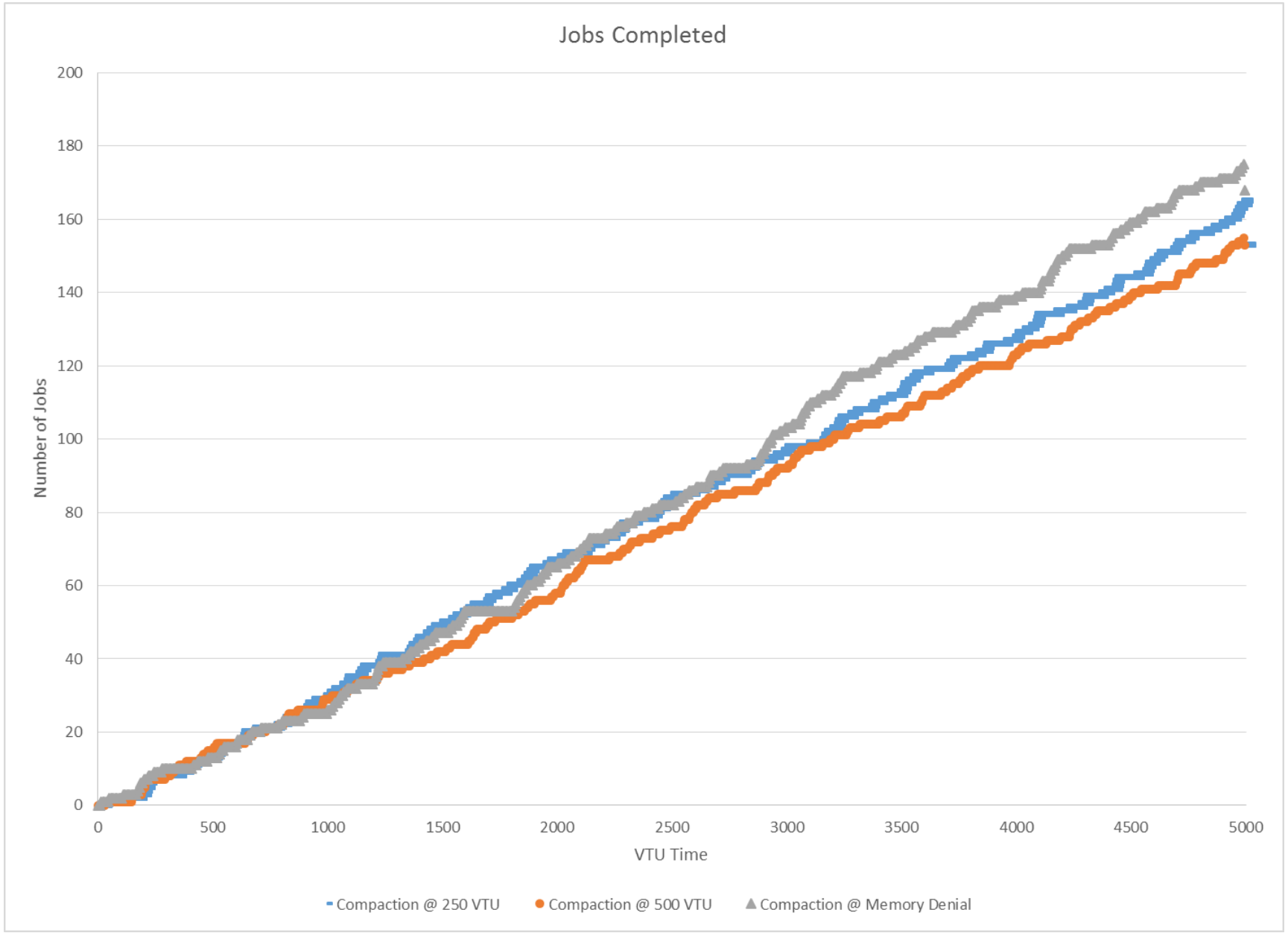


Chart I – Best Fit

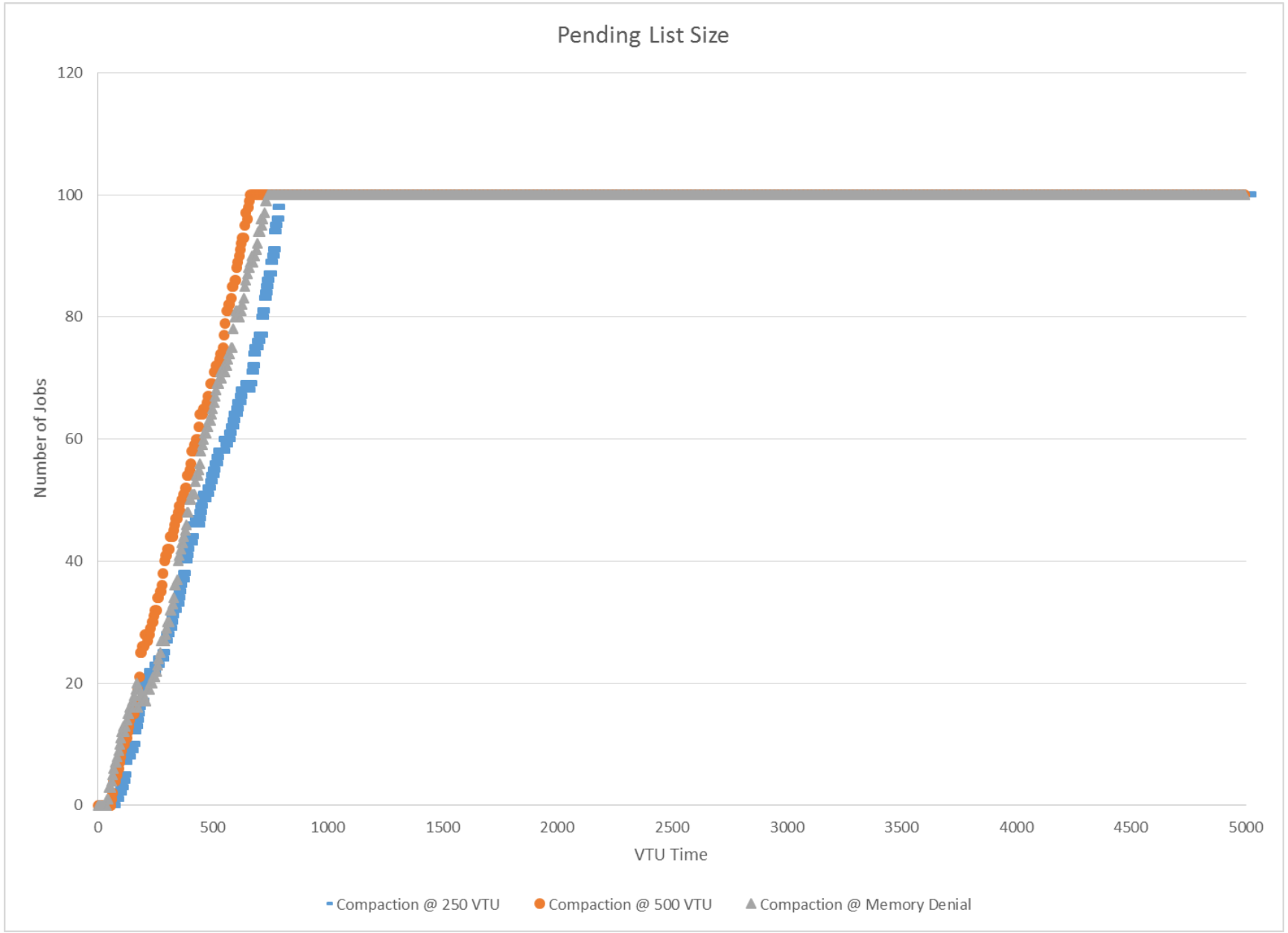


Chart J – Best Fit

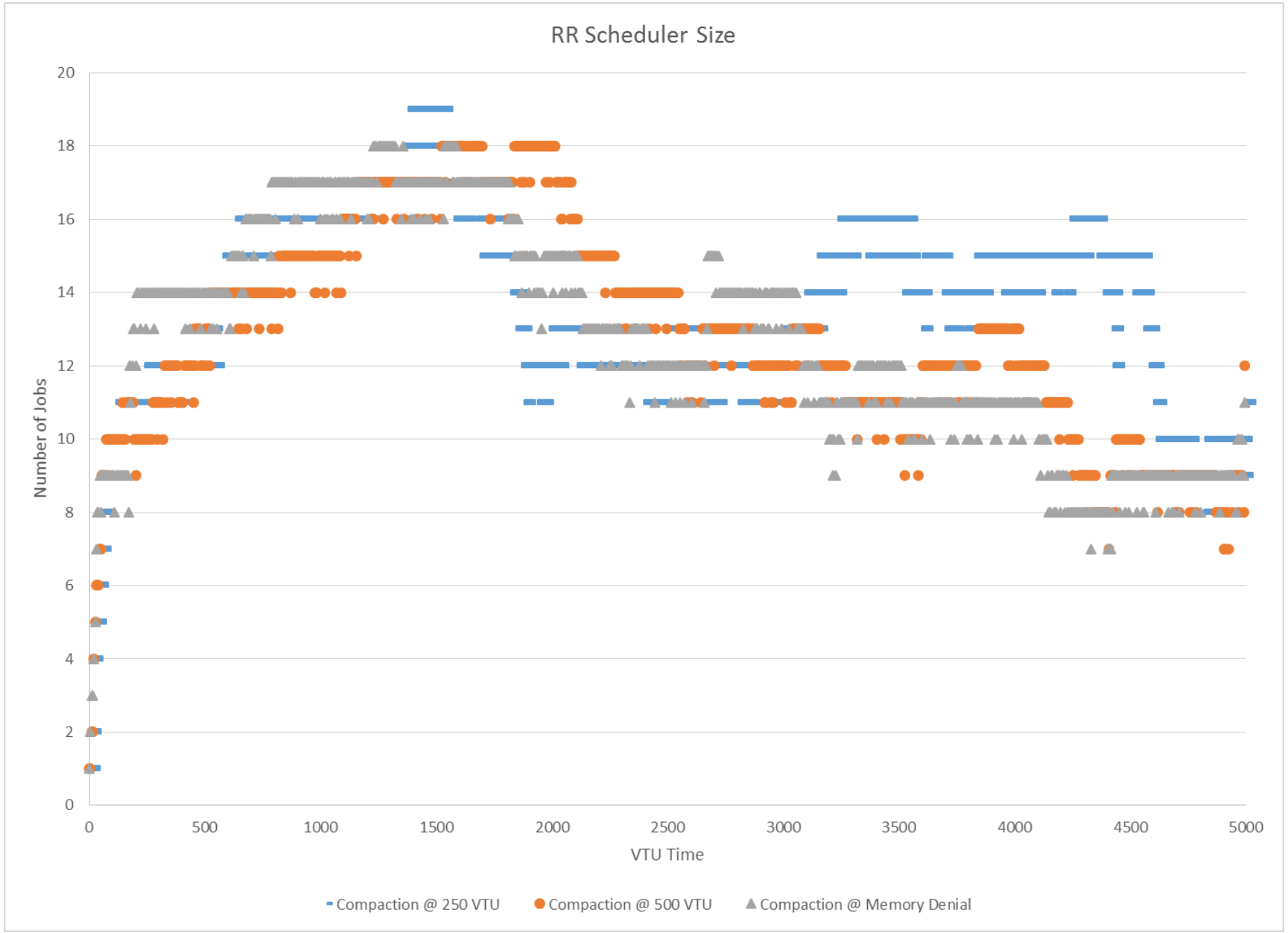


Chart K – Best Fit

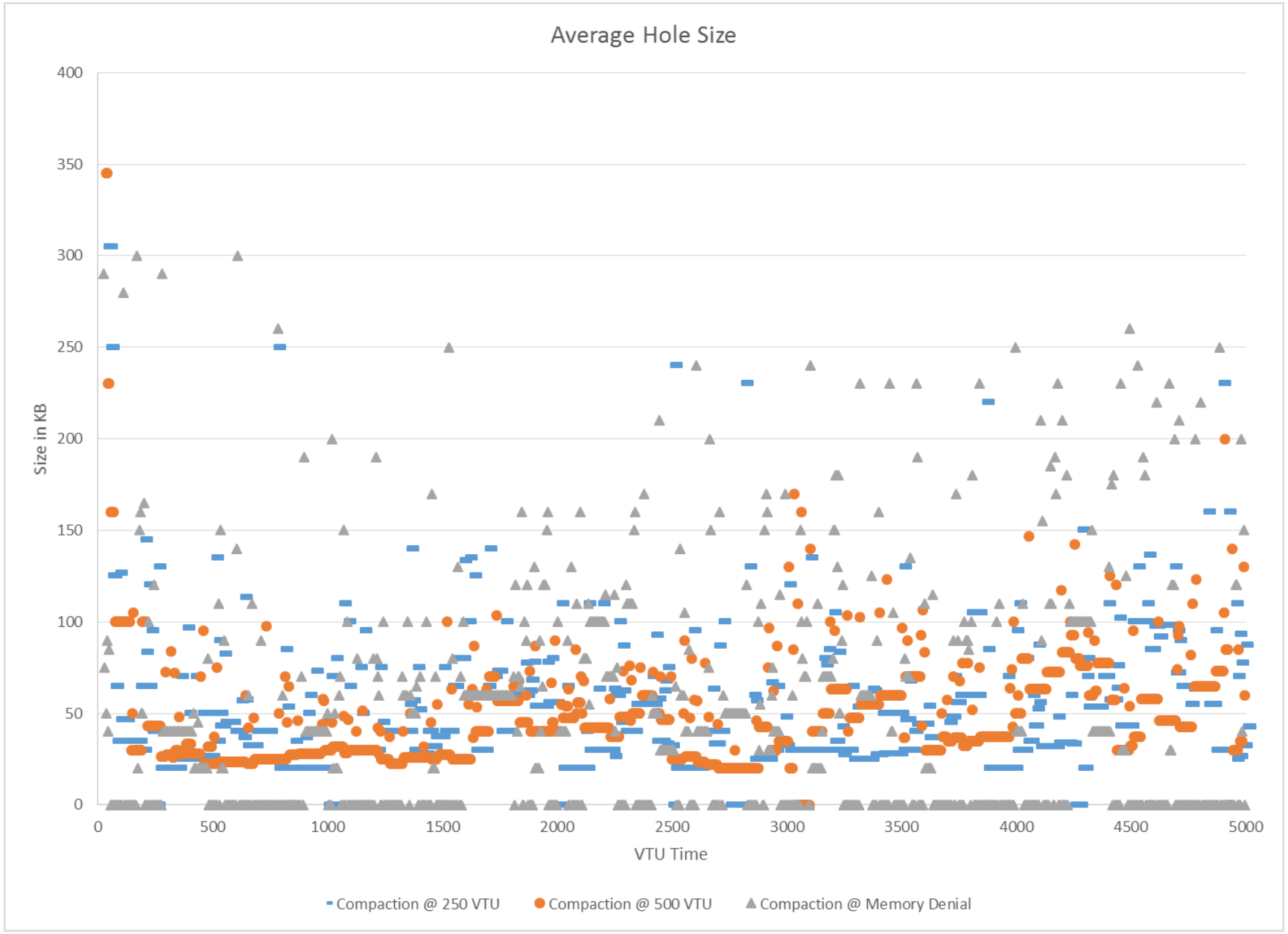


Chart L – Best Fit

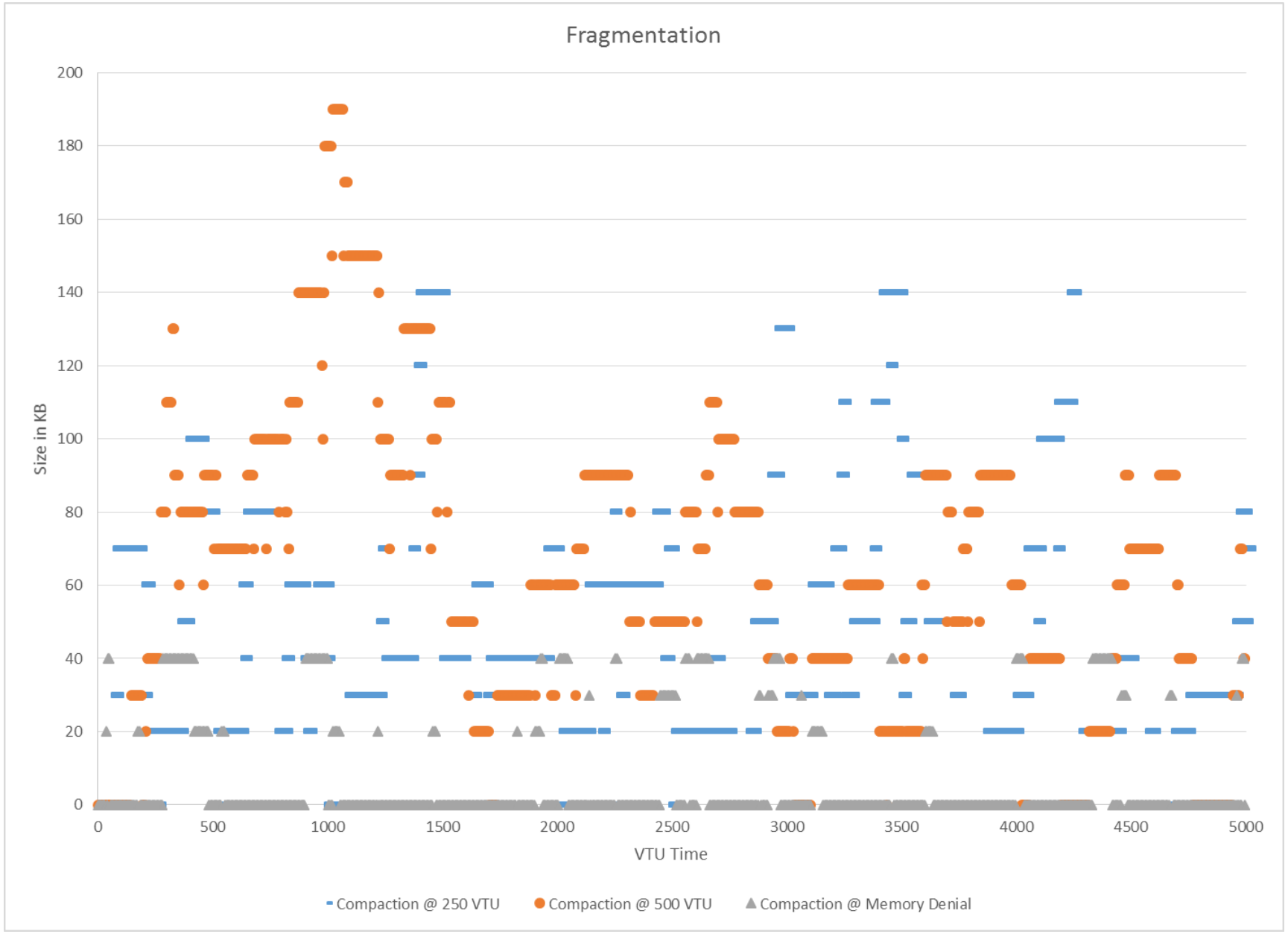


Chart M – Best Fit

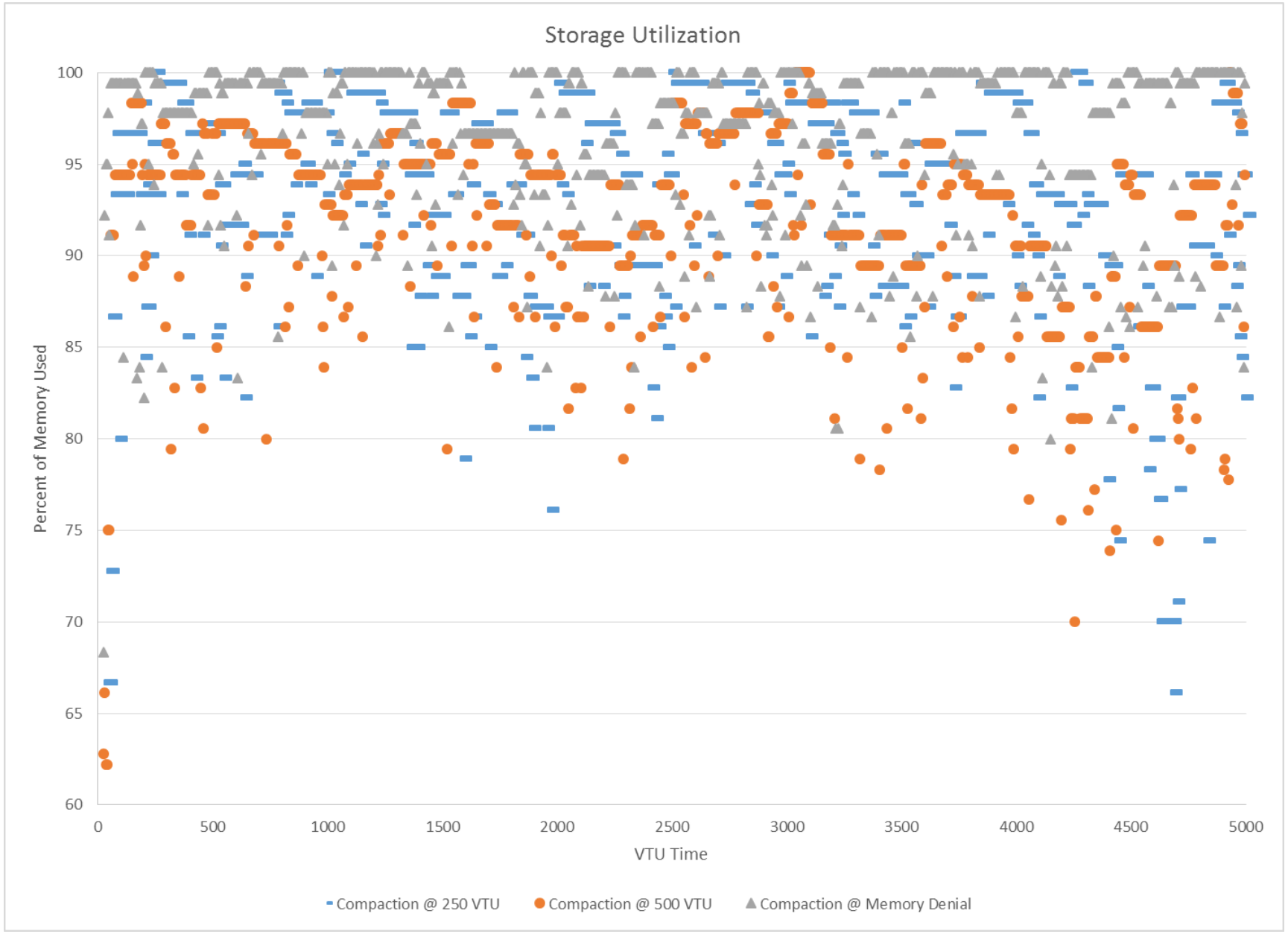


Chart N – Best Fit

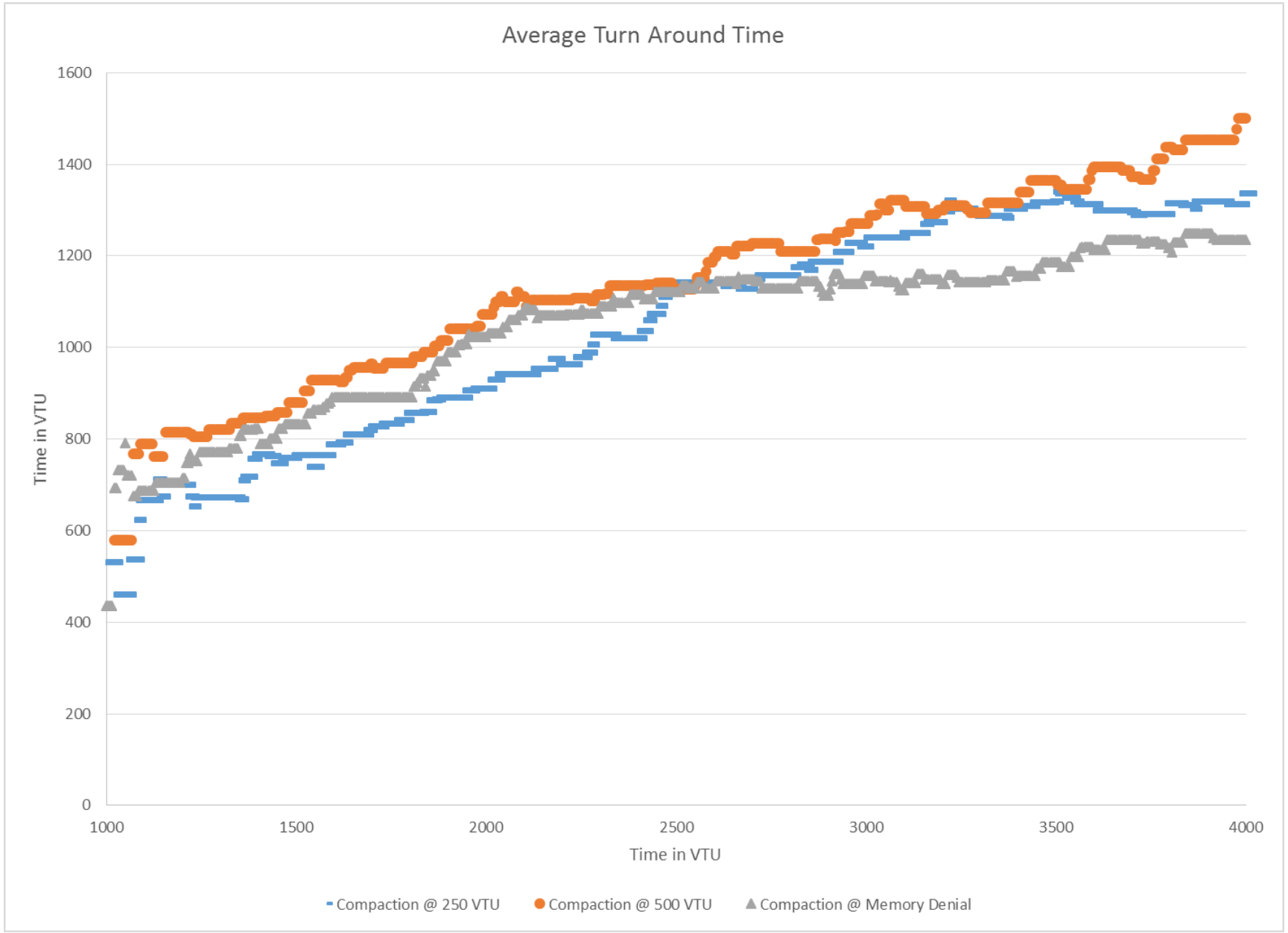


Chart O – Worst Fit

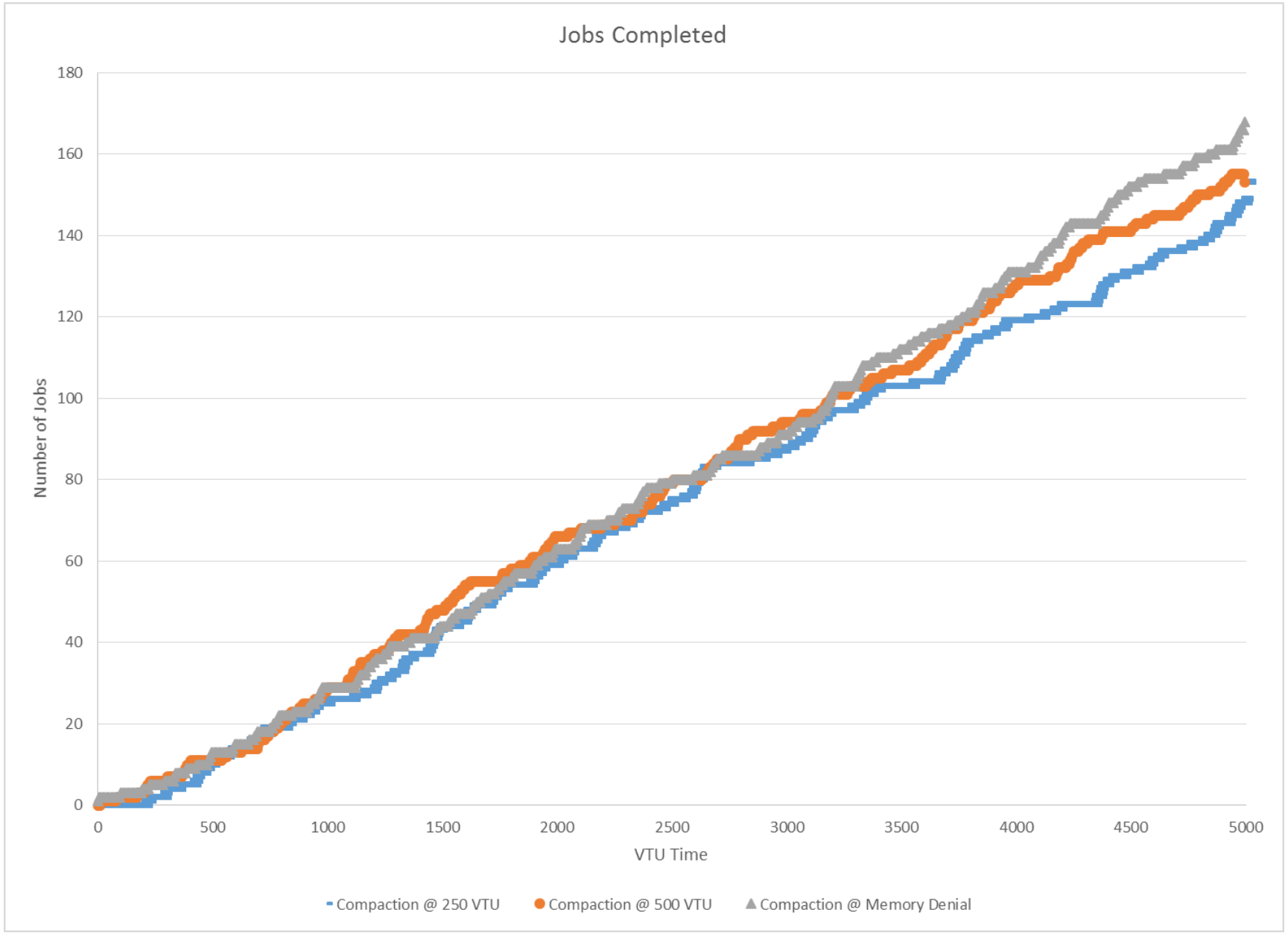


Chart P – Worst Fit

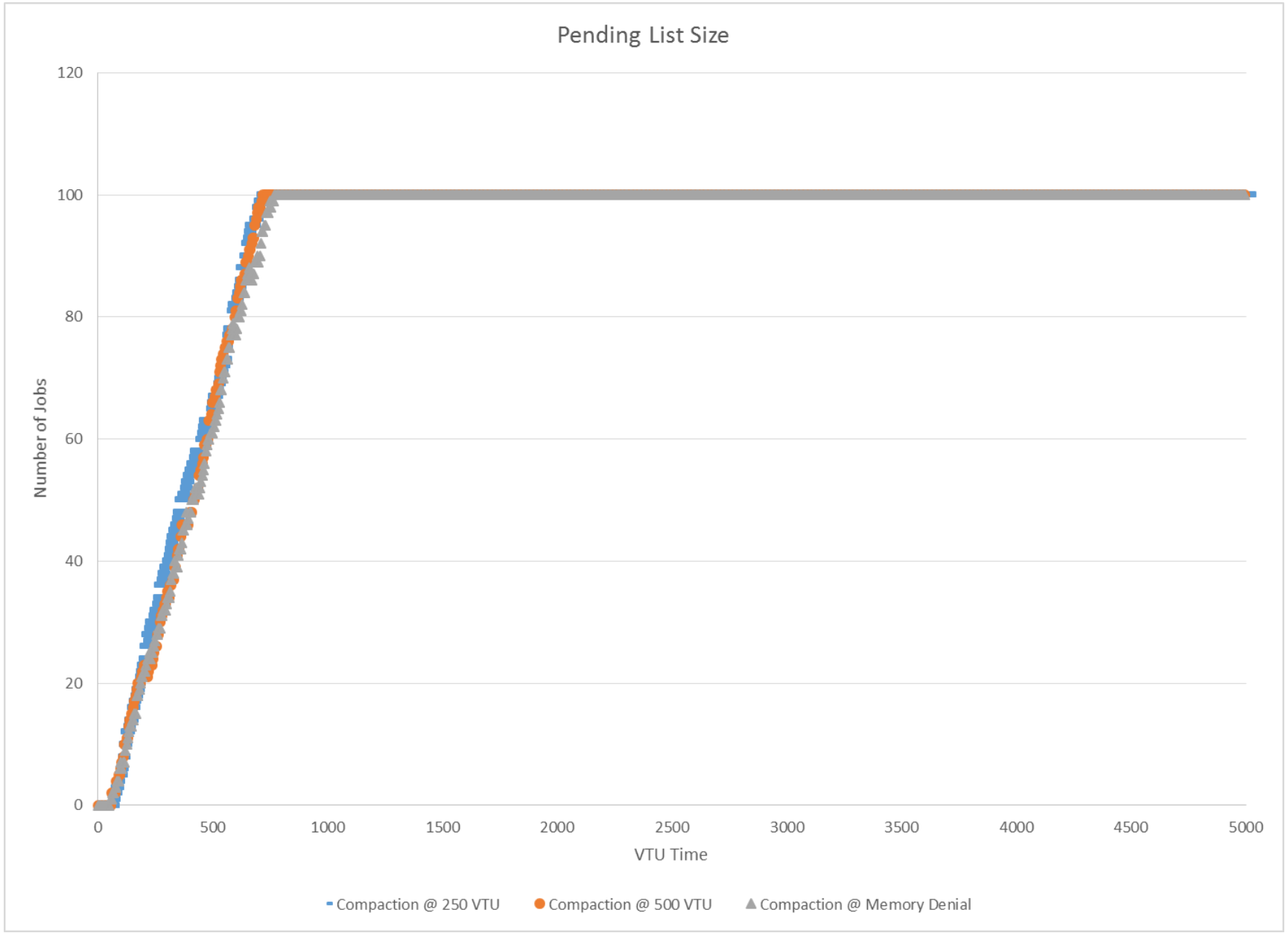


Chart Q – Worst Fit

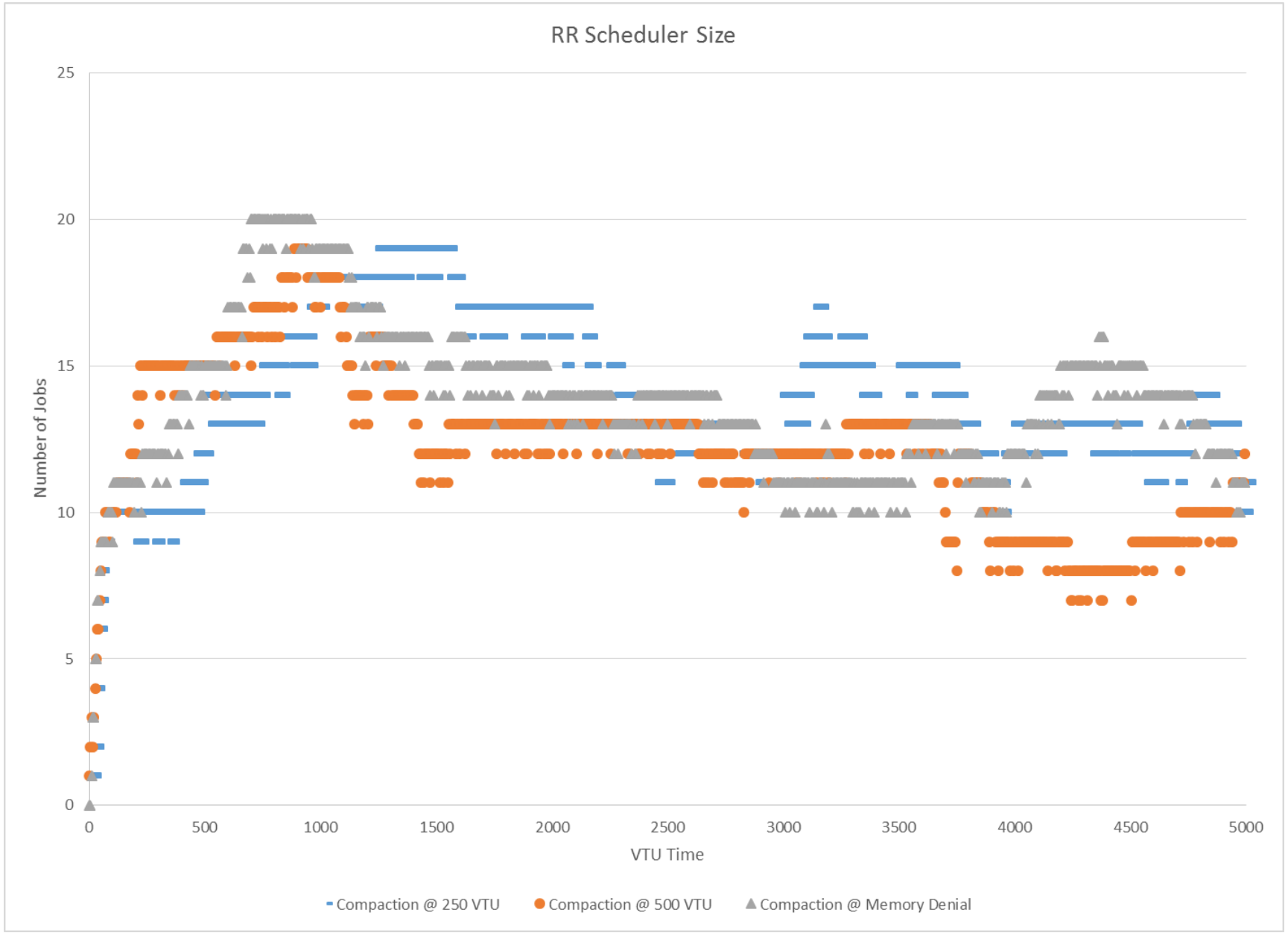


Chart R – Worst Fit

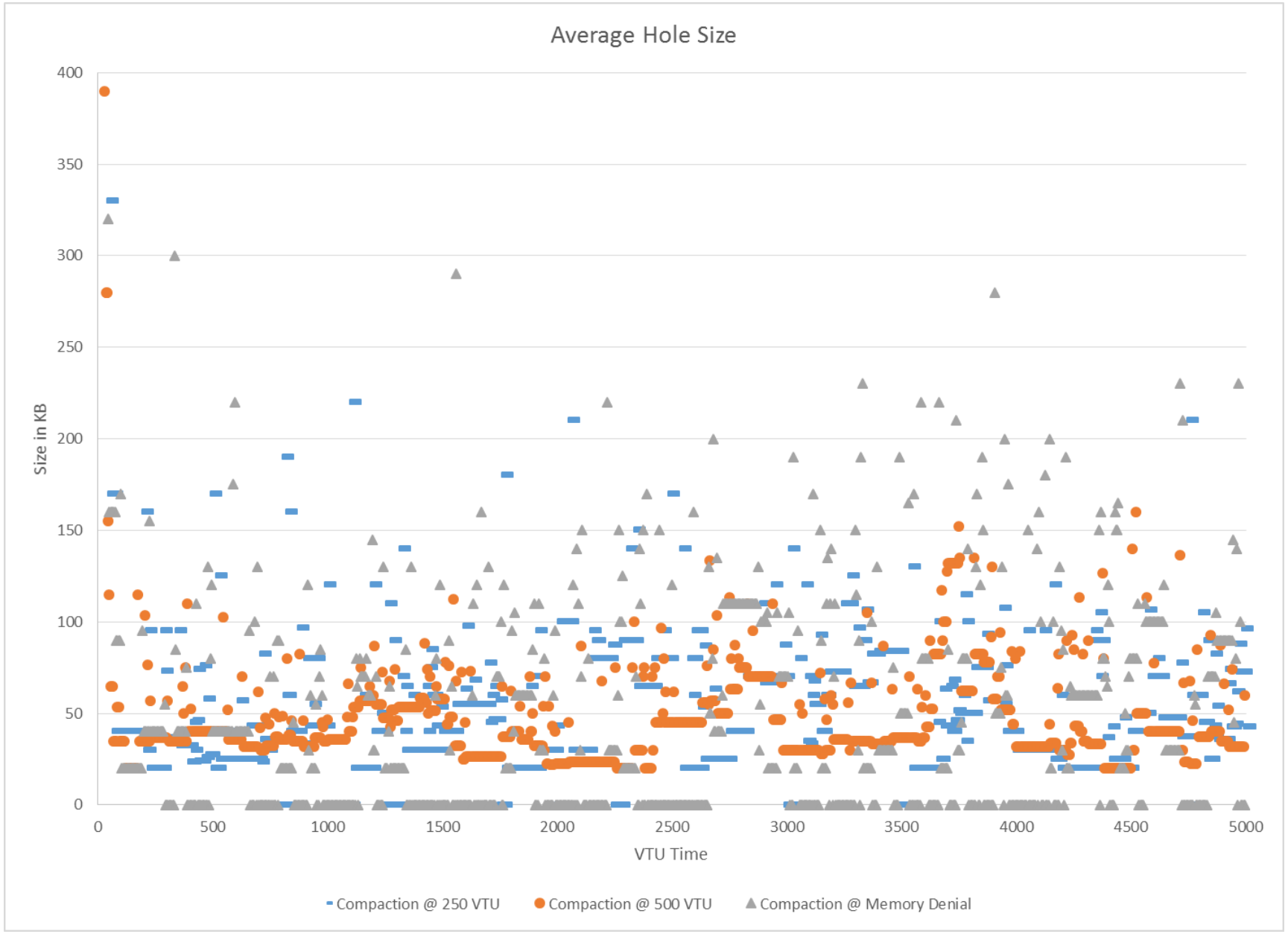


Chart S – Worst Fit

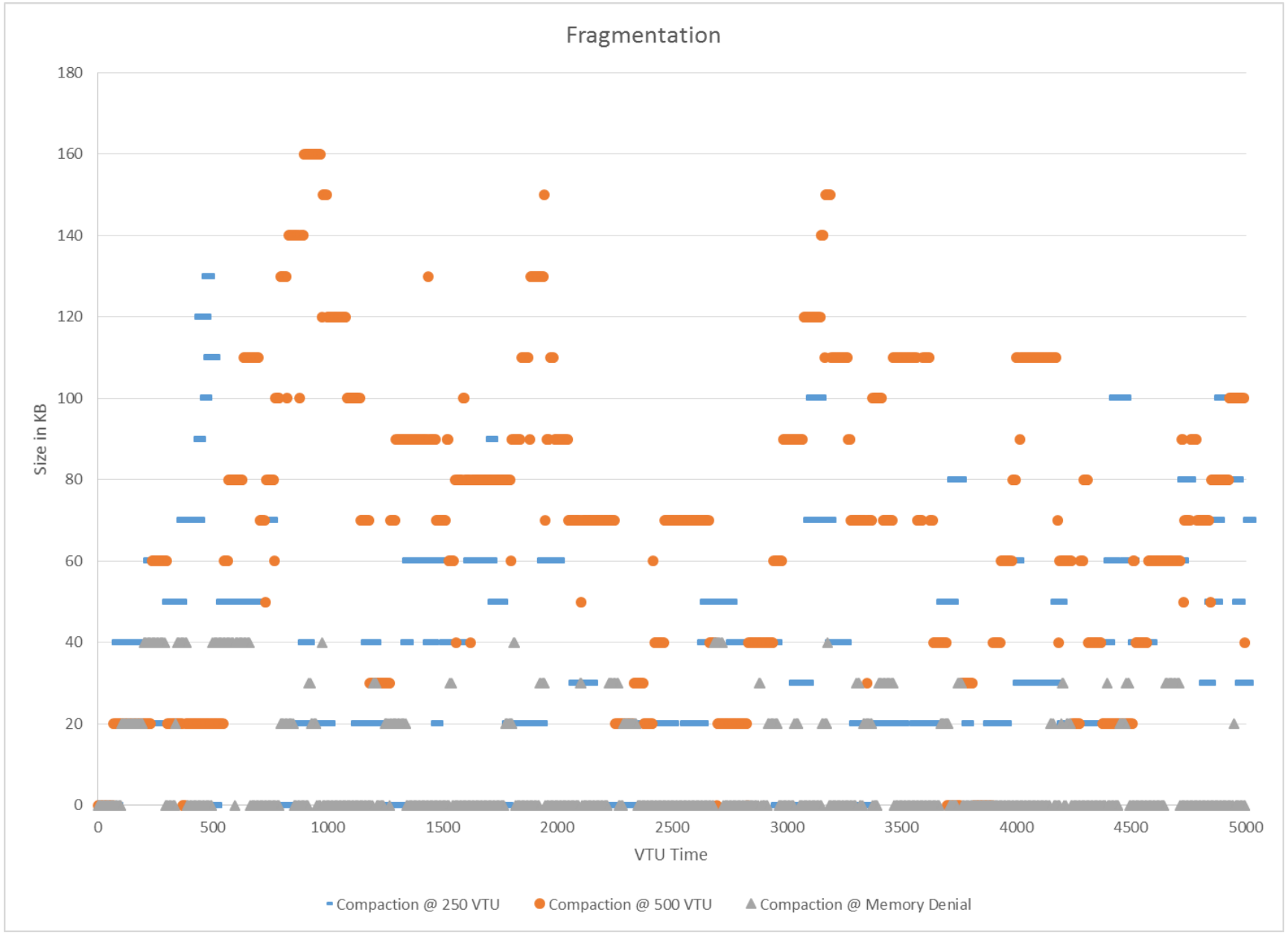


Chart T – Worst Fit

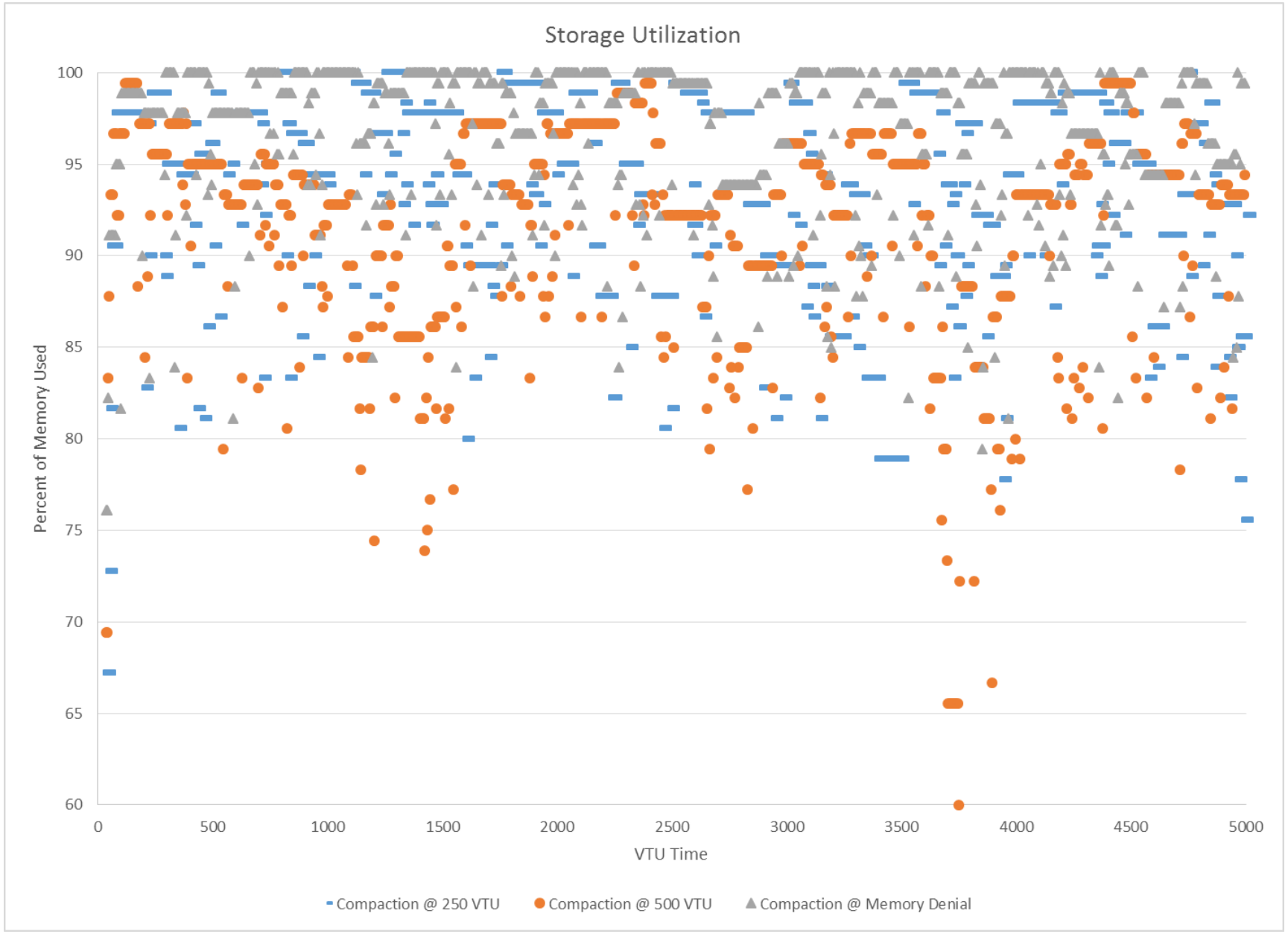


Chart U – Worst Fit

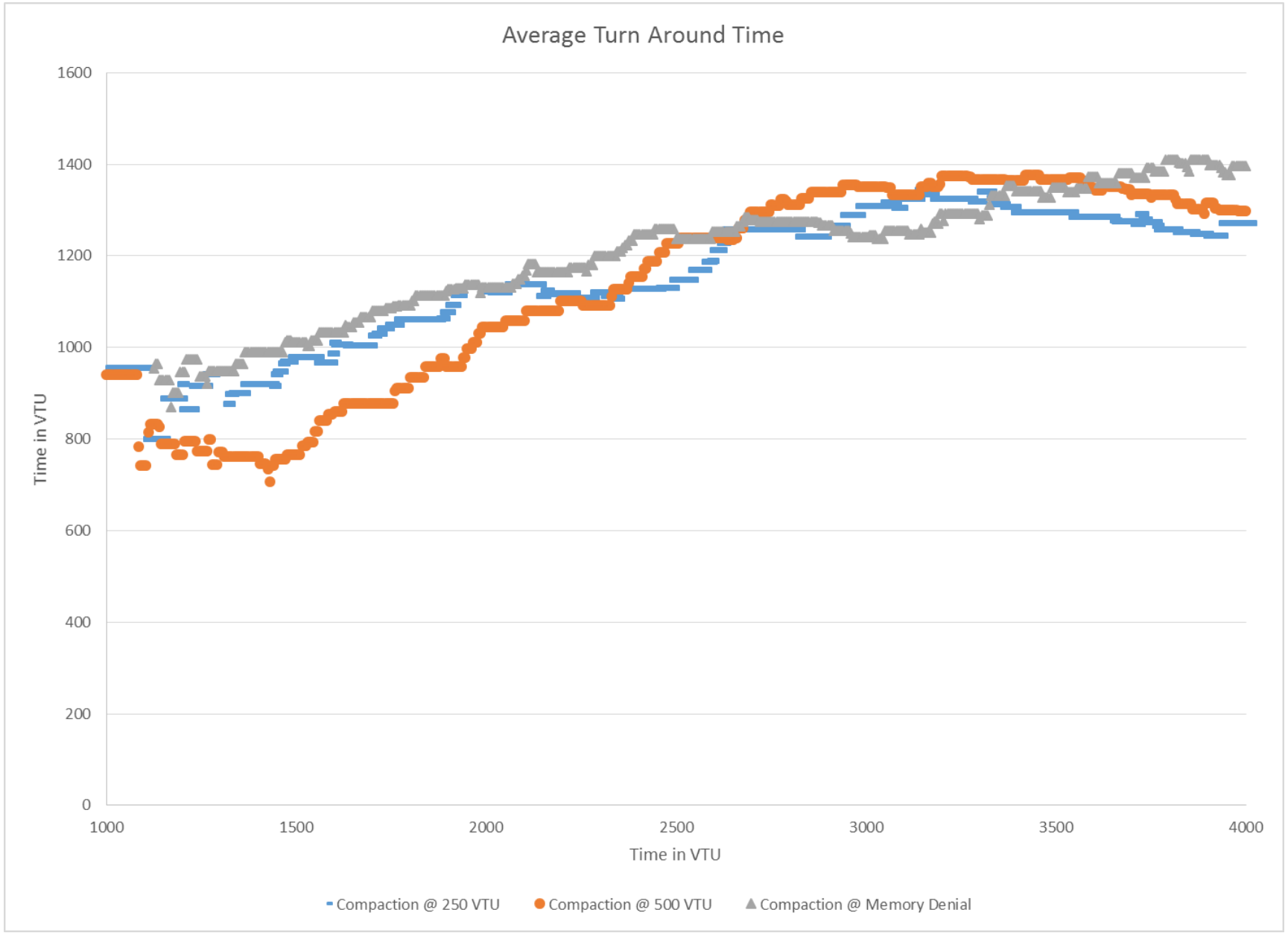


Chart V – Cumulative

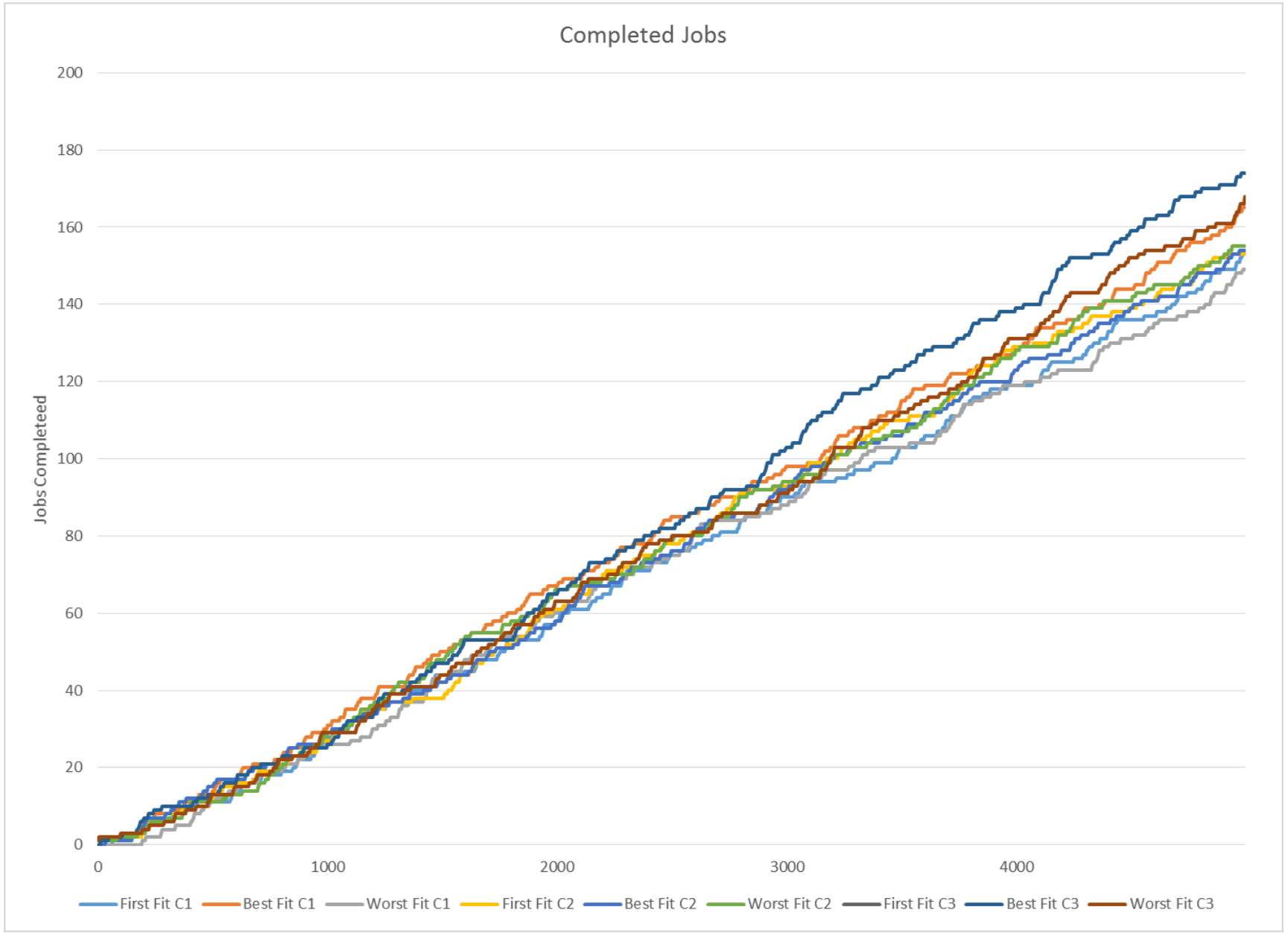


Chart W – Cumulative

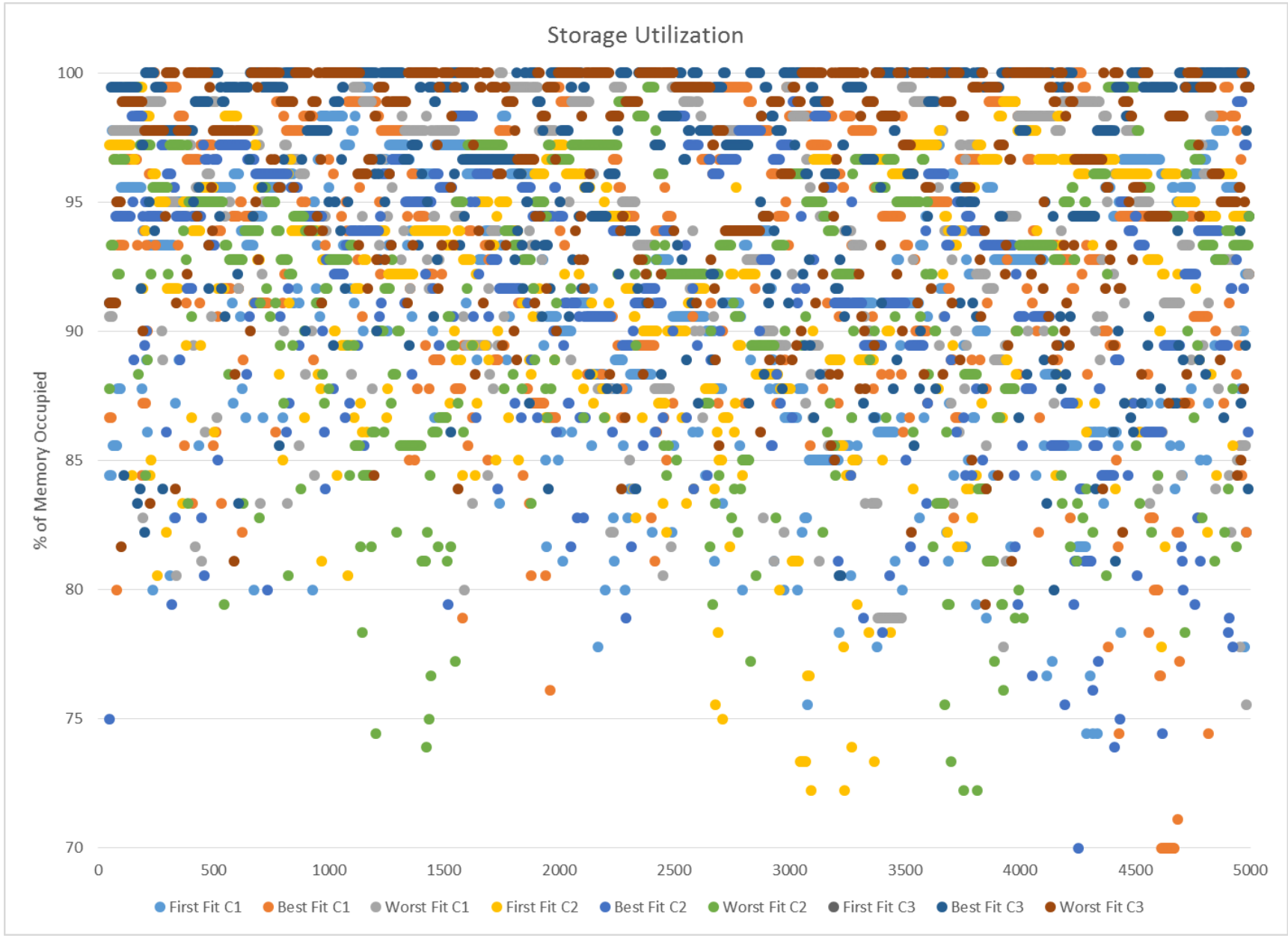


Chart X – Cumulative

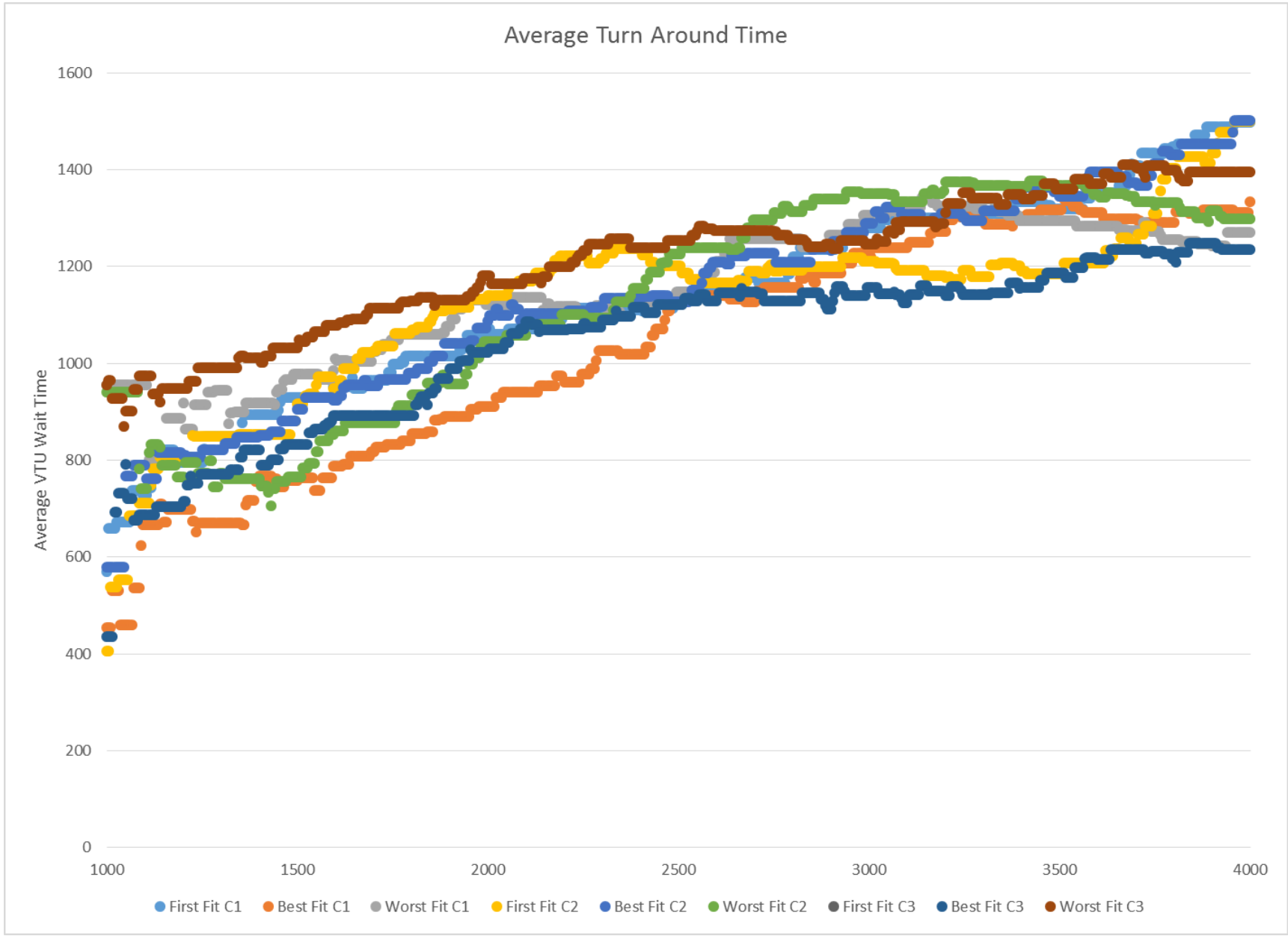


Chart Y – Cumulative – Best Best and Worst Worst

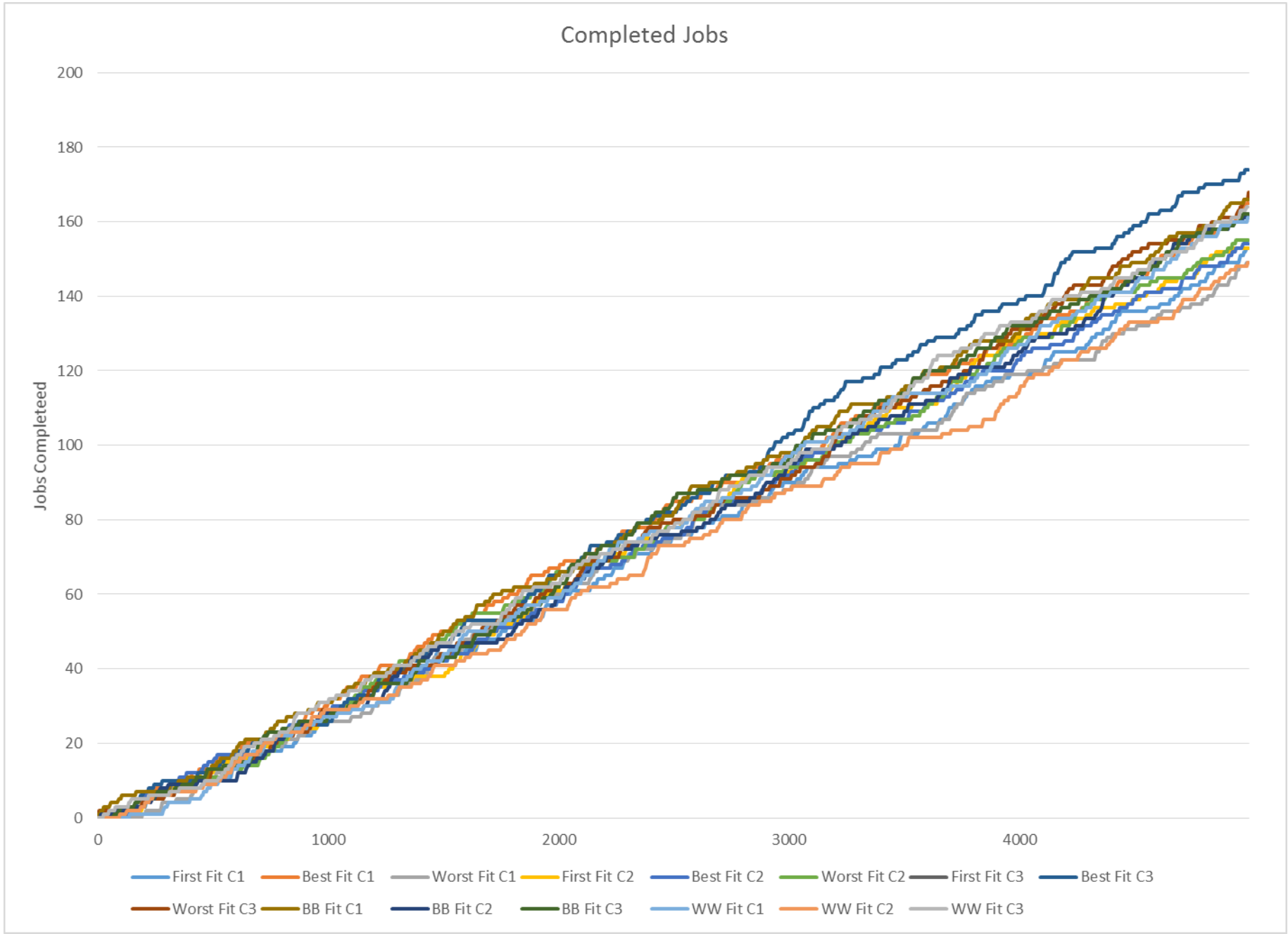


Chart Z – Cumulative – Best Best and Worst Worst