Drake Tetreault
35571095

CS 143B – Project 2

Project Report

Introduction

The goal of this project was to evaluate the performance of two different memory allocation strategies in an execution environment where the maximum amount of memory available is expected to be in use. The criteria which were used to evaluate the strategies are the average memory utilization produced by each strategy as well as the average search time per memory reservation request. Average memory utilization is the percent of the total available memory that is actually put to useful purpose over the lifetime of the testing simulation. Average search time measures the number of noncontiguous free memory segments that an allocation strategy examines for each memory request. The ideal allocation strategy would exhibit high memory utilization at all times along with low constant search times. As the data I collected demonstrate, neither strategy I implemented achieved this ideal at all times.

The two allocation strategies I chose to implement are Next Fit and Worst Fit.

Data Gathering

To collect meaningful data, I varied three primary inputs to the simulation driver beyond the allocation strategy itself.

The first input I varied is the total amount of memory available to the system. The amount of total memory available is the coarsest of the three varying inputs to the driver, as I chose to measure performance at 1000, 10 000, and 100 000 memory units. I had originally planned to gather data at 1 000 000 and above, but even after several optimizations running the simulation still took too much real world time to be feasible.

The remaining inputs I varied are the size of memory requests per simulation step. Because memory requests are generated according to a Gaussian distribution, the second and third varied inputs are the mean and standard deviation of the memory request distribution. To decide on the ranges for both these values, I ran the simulation over a very wide but coarse range of values. After examining the resultant data, I decided on a final fine-grained ranges of 1% to 20% for the mean and 1% to 10% for the standard deviation. These values are fractions of the total memory size.

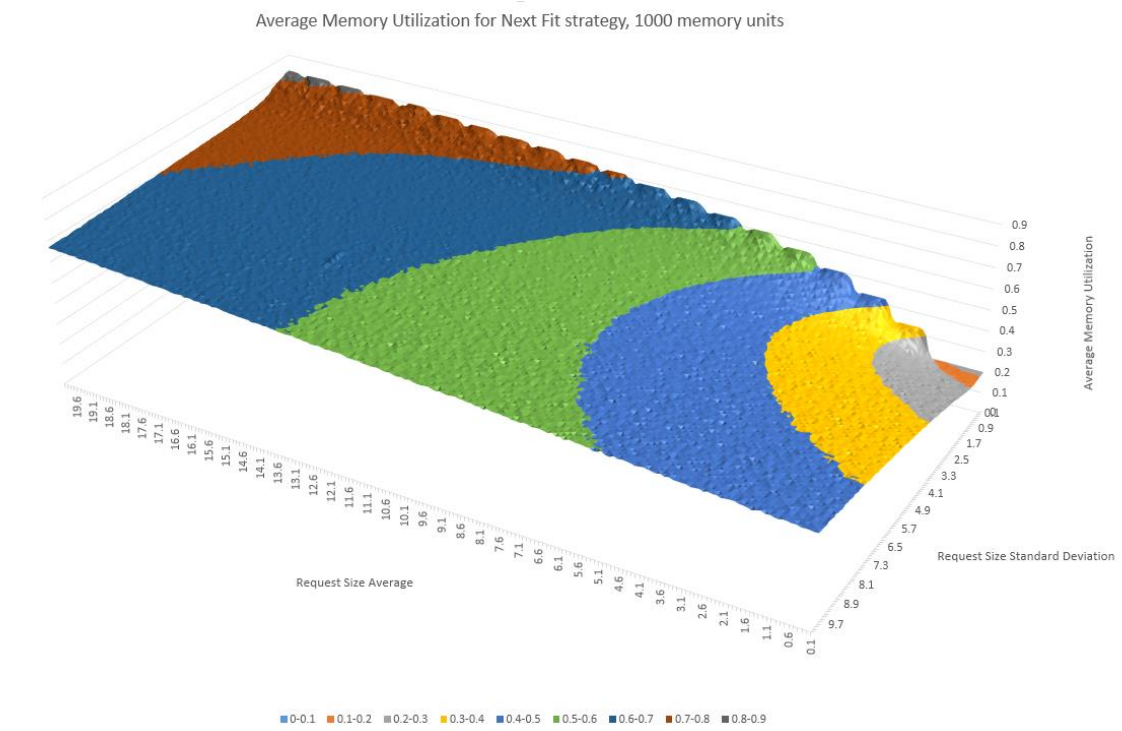
Each final data point represents the average value over 10 000 simulation steps.

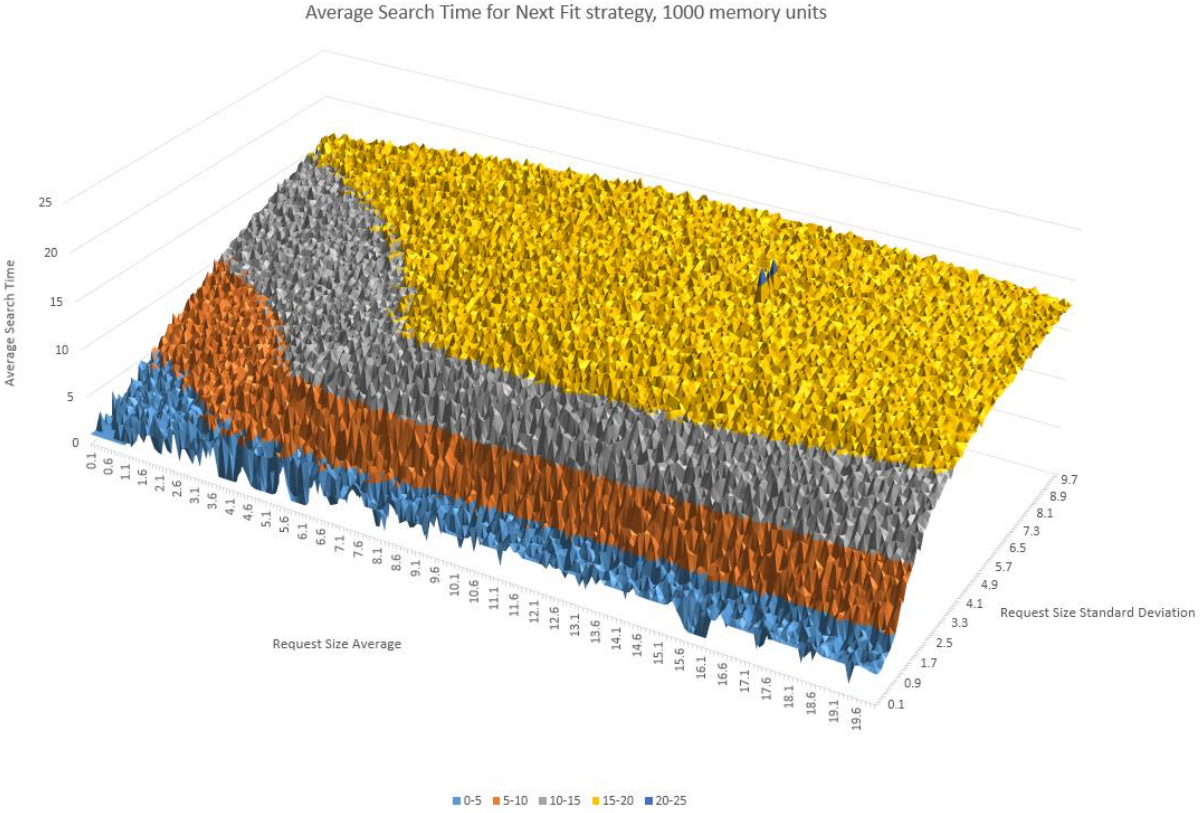
Results

Below are the results I obtained for the Next Fit and Worst Fit memory allocation strategies.

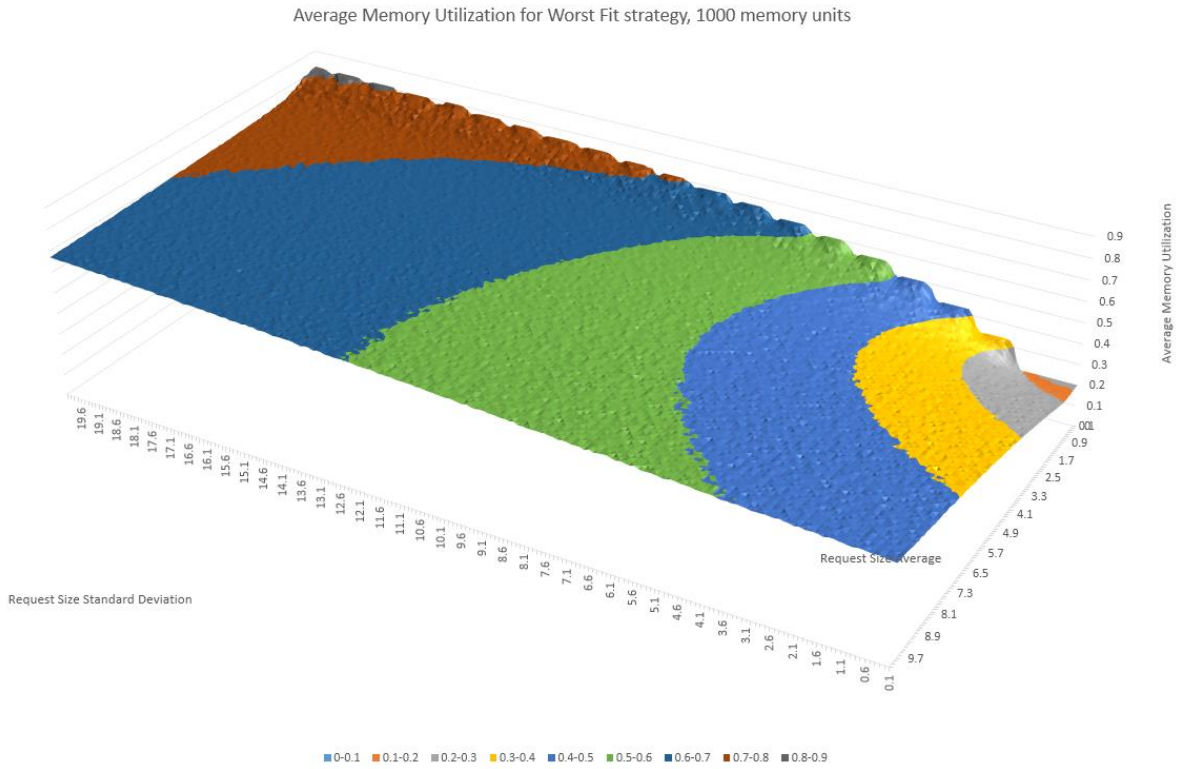
Note that the decreased resolution of data points for the 100 000 memory units cases is due to the prohibitively long simulation time at that memory size for simulations with data density equal to the 1000 and 10 000 cases.

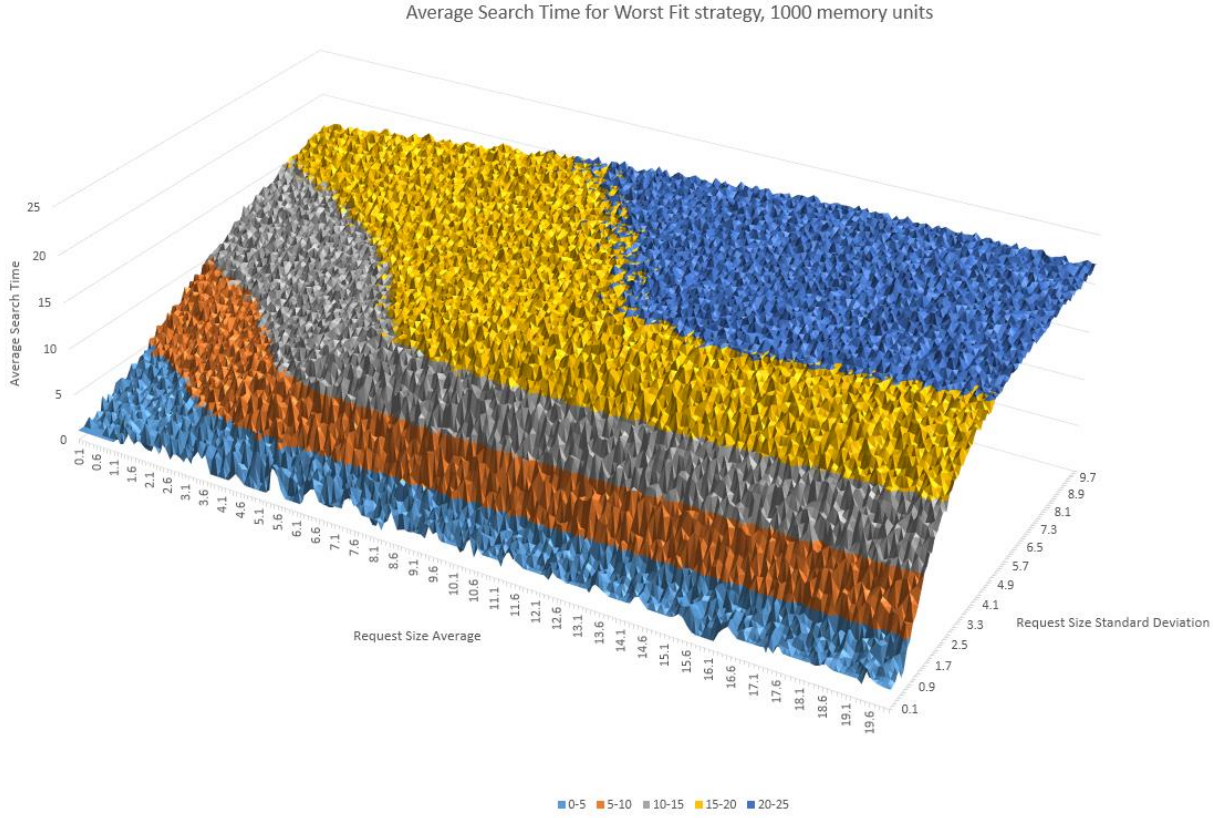
Drake Tetreault
35571095



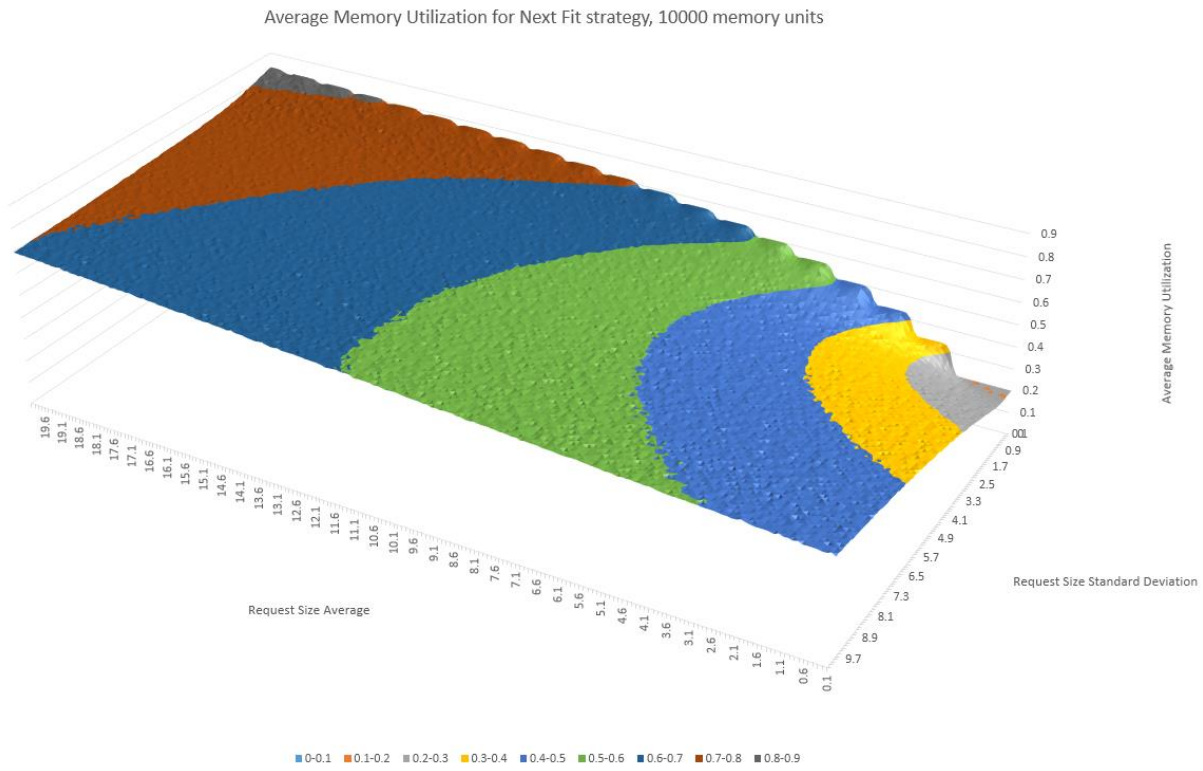


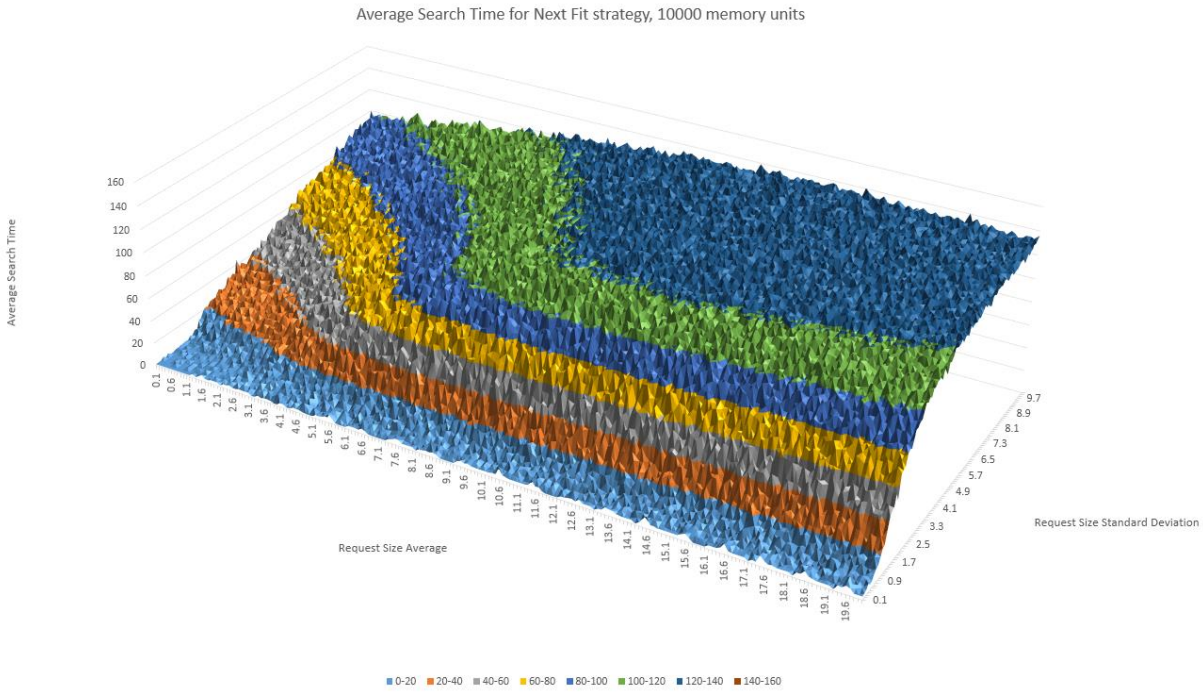
Drake Tetreault
35571095



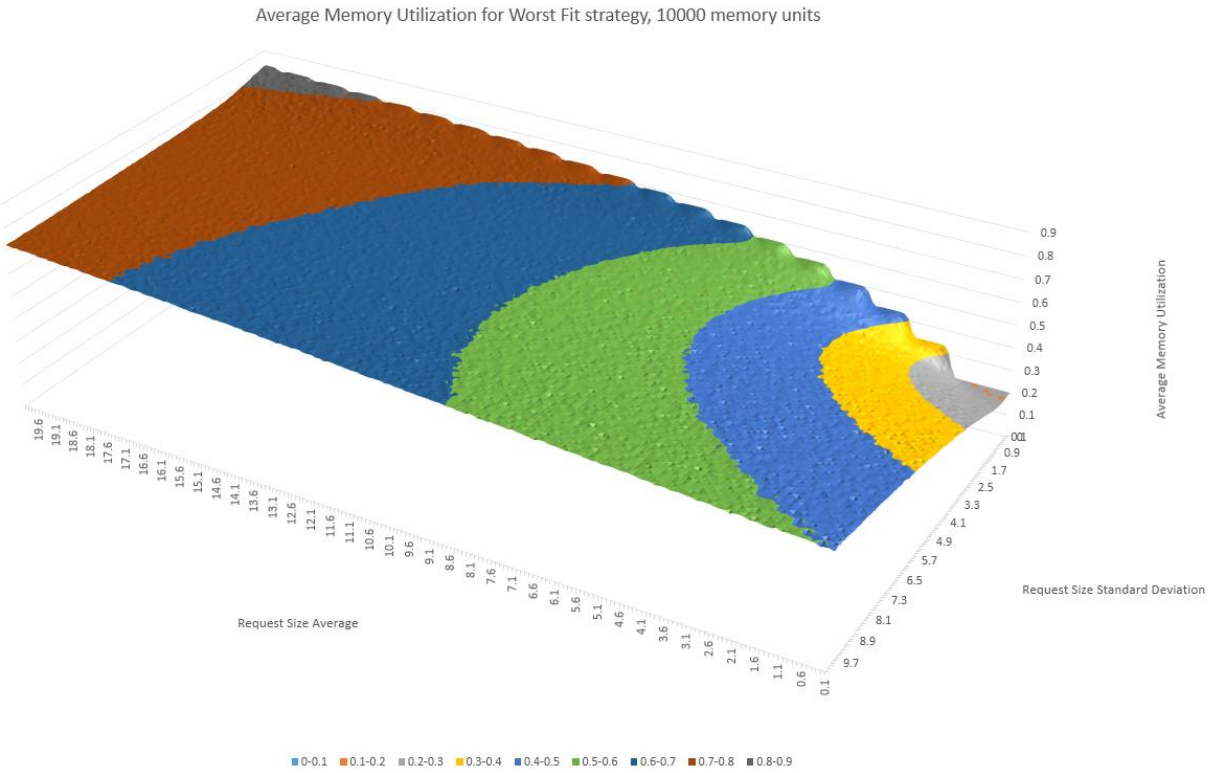


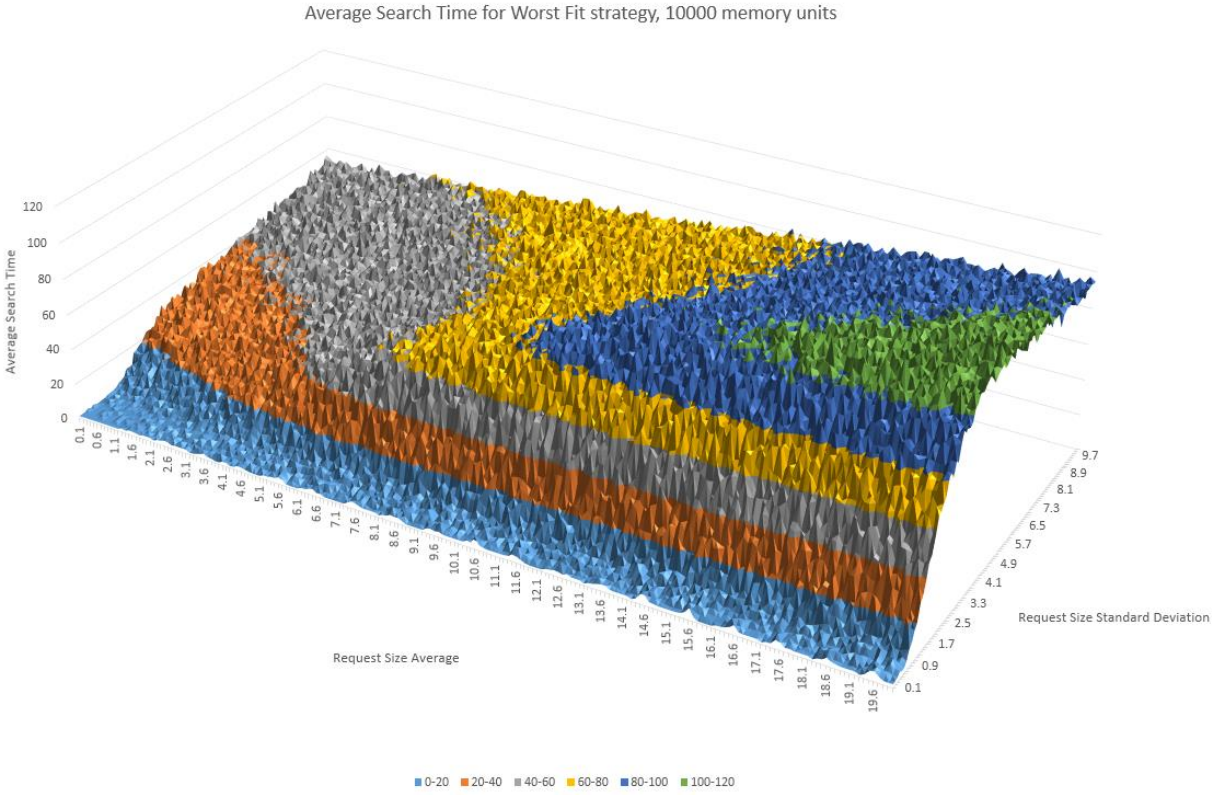
Drake Tetreault
35571095



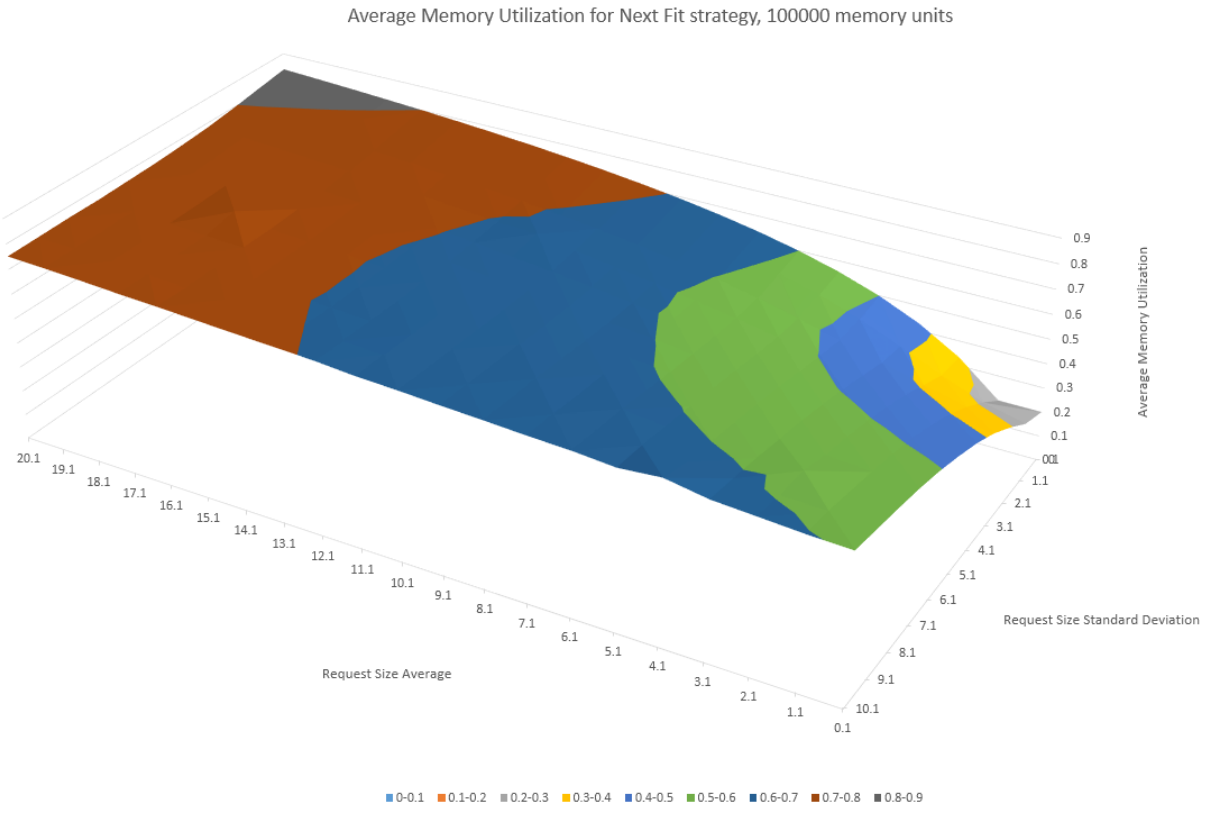


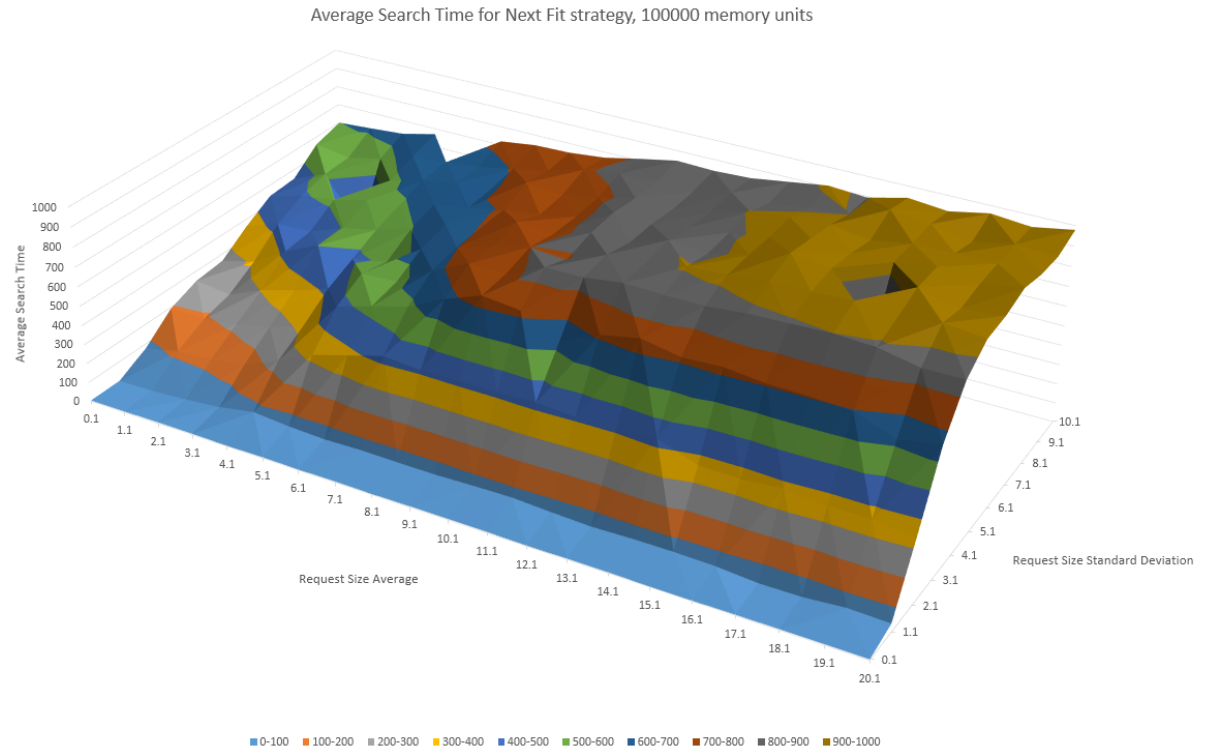
Drake Tetreault
35571095



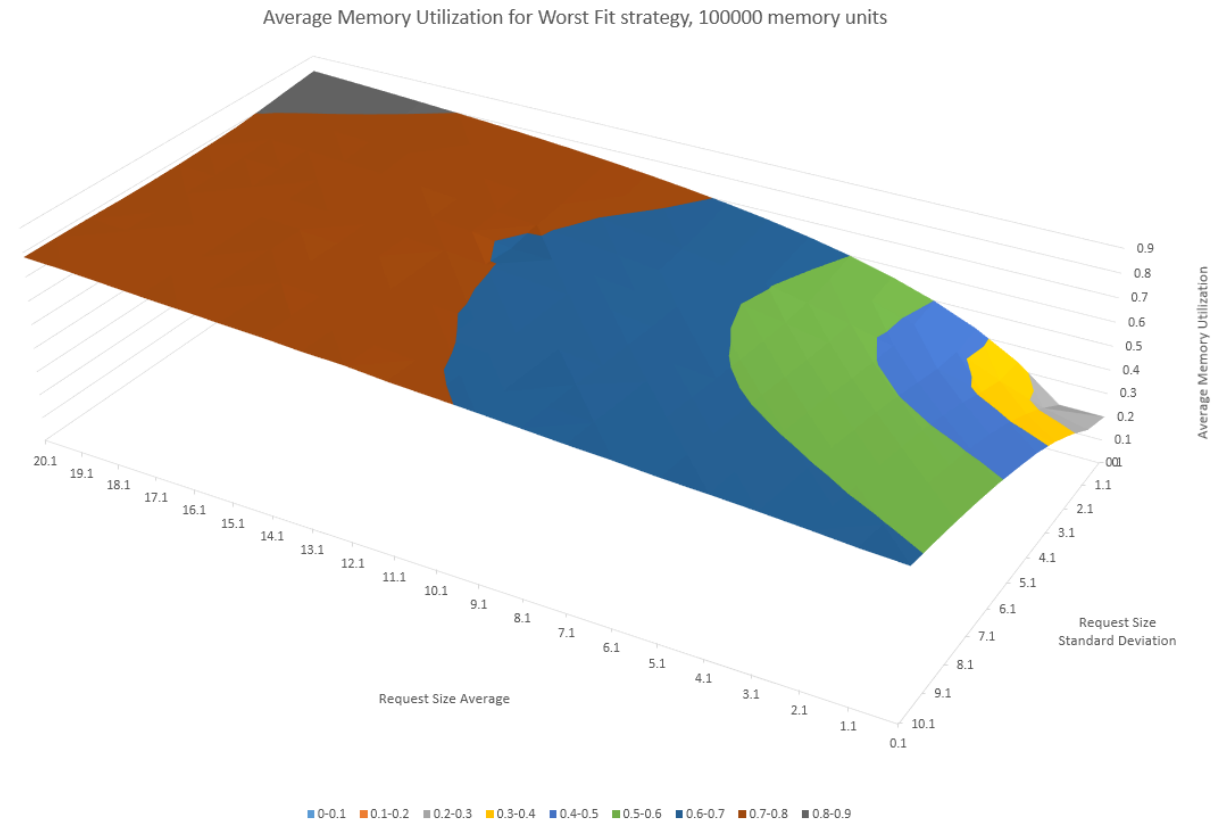


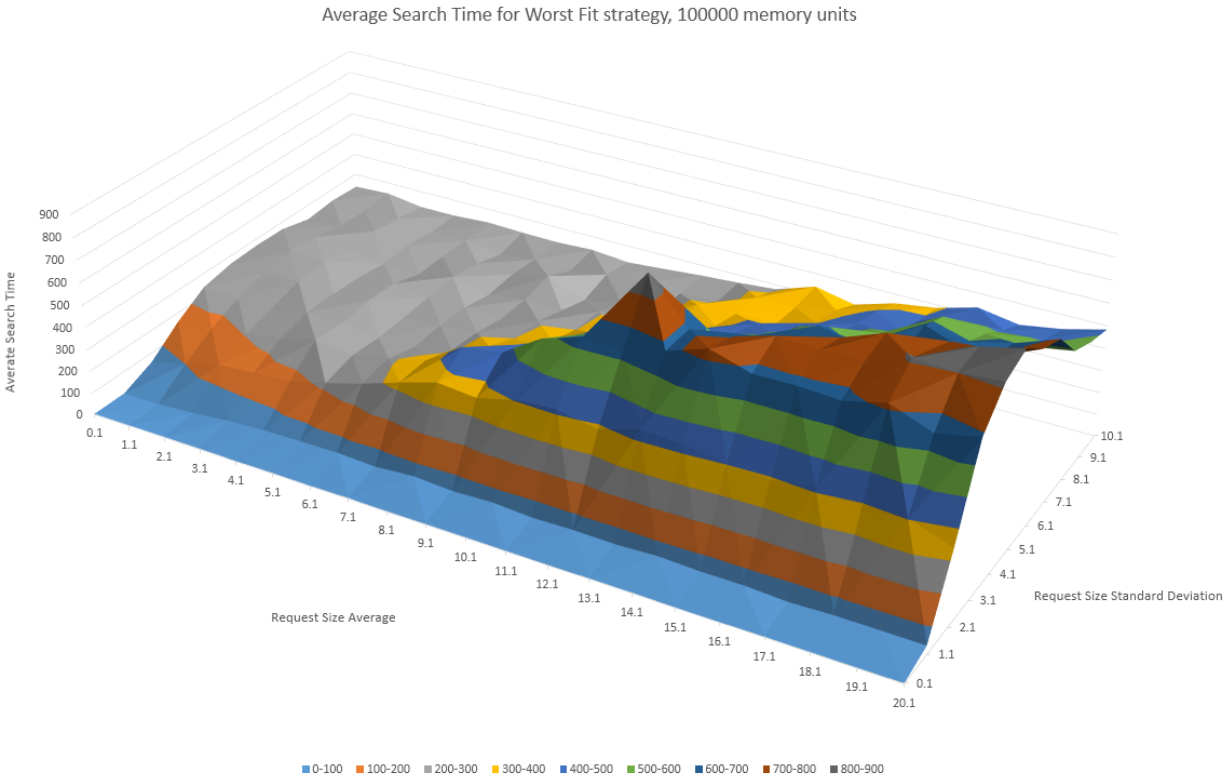
Drake Tetreault
35571095





Drake Tetreault
35571095





Drake Tetreault
35571095

Analysis

My first observation once I had plotted the results was that the shapes for both average memory utilization and average search time are broadly similar across the two different allocation strategies and the three tested memory sizes.

For both strategies, average memory utilization is minimized when the request size mean and standard deviation are both small. The values increase steadily when both mean and standard deviation are increased, but increasing the mean has the greater effect. The general maximum for average memory utilization occurs when the mean request size is maximized and the request size standard deviation is minimized, suggesting that large allocations of nearly constant size best utilize memory space under both algorithms.

Both strategies also exhibit similarly shaped behavior for average search time. In most memory size cases, increasing the request size standard deviation leads to a large jump in the average search time, and thus to decreased performance. Increasing the mean request size also leads to increased average search times, but not to such a dramatic extent as increased standard deviation.

Although the general shape of the result surface graphs are similar between the two allocation strategies, there are also differences in the results. The average search time for the Worst Fit algorithm increases sharply with increased standard deviation, but then plateaus at around a 5% standard deviation. The Next Fit algorithm, however, increases less rapidly with standard deviation but never stops increasing up to 10% standard deviation.

Another surprising results is that as the total memory size increases, the Worst Fit strategy begins to significantly outperform the Next Fit strategy in terms of average search times. This is surprising because the Worst Fit strategy must scan every node in the linked list of free memory segments before deciding on the one to return, whereas the Next Fit strategy returns the first matching segment. This results suggests that the Worst Fit algorithm more optimally selects segments to split to a degree that outweighs its theoretical disadvantage in search times. Another consideration is that the simulation constantly kept the memory state at its maximum possible utilization, which benefits the Worst Fit algorithm's search time disadvantage by ensuring the size of the free list remains small. Further research is called for that analyzes the relative performance of these two algorithms at e.g. 50% maximum utilization to see if Worst Fit's performance remains optimal.