Results 3:

Greedy Hillclimber

To test our benchmarks, we extracted the Greedy Hillclimber from exercise 2 and ran it against our set of modified benchmarks.

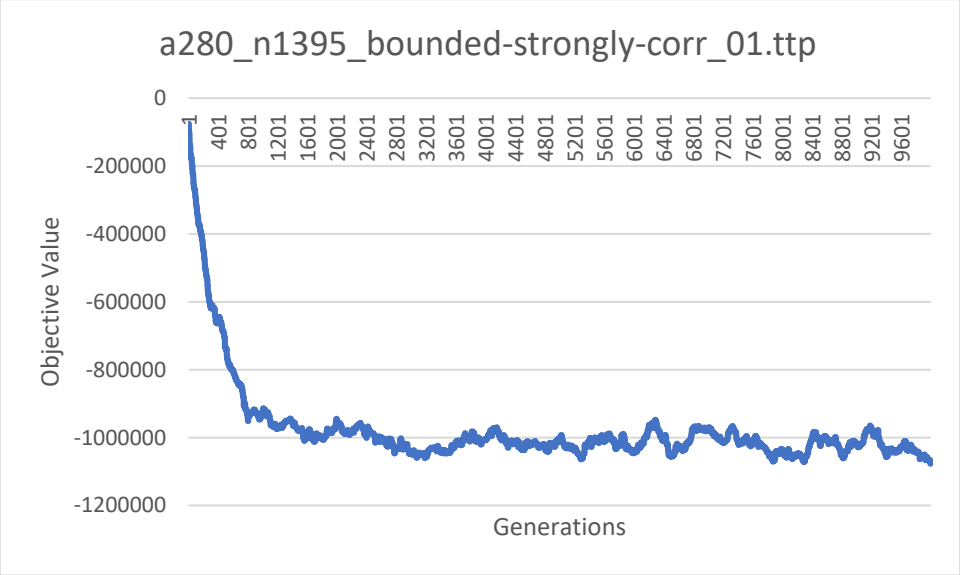

Figure 1 - Greedy Hillclimber vs Exchange every 50 generations

As we can see the tour modifications significantly affects the success of the traveling thief. Initially we thought that there was an error in the comparator causing us to minimise our objective value rather than maximising it. However, on closer inspection we can see that the damage done by de-optimising the tour vastly outweighs the progress that can be made during 50 and 500 generations, respectively.

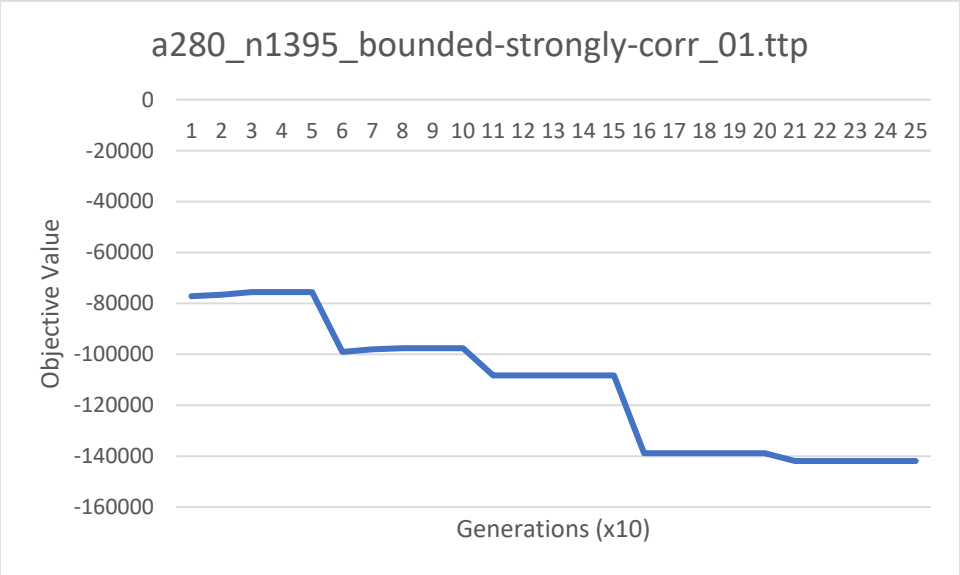

Figure 2 - Greedy Hillclimber vs Exchange every 50 generations, closer

Richest Ant

The idea behind the Richest Ant is to borrow some of the concepts from ant colony optimisation to create a highly diverse, probabilistic packing plan. The algorithm works as follows:

For each item in the packing plan (i_k) there exists a p value (p_k) where p_k is the probability that the item will be picked up during the next iteration. p_k begins at 0.5 and is modified by adding or subtracting 1/n where n is the number of items in the plan. The idea is that the most successful algorithm would modify the probability such that commonly successful items are re-enforced while less commonly successful items are discouraged. The probabilistic nature of generation encourages high novelty, while converging as the probabilities are re-enforced. Fortunately, the algorithm proved very successful in the latter goal. However, while we weren't able to improve the initial result significantly, the fact that we managed to recover fairly quickly from most changes was encouraging.

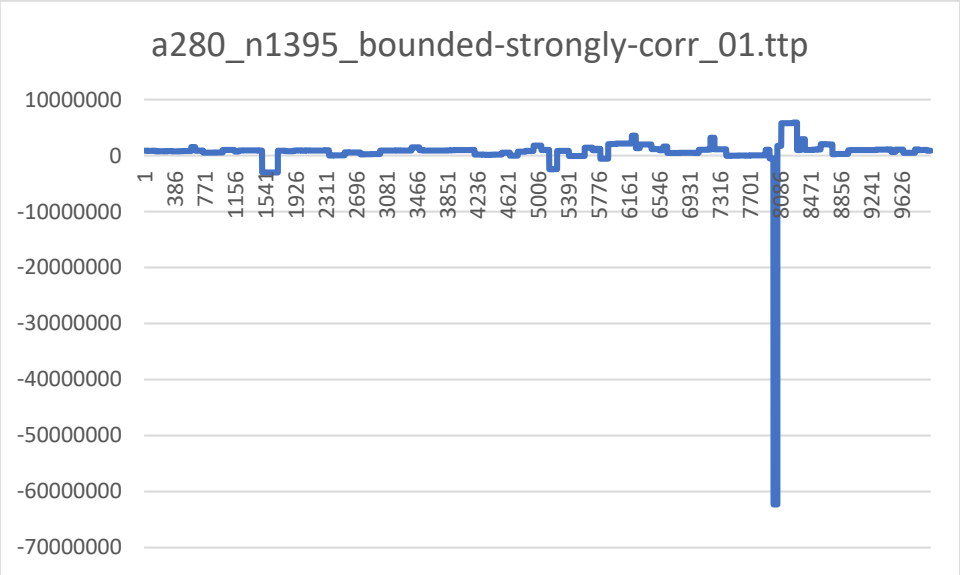

Figure 3- Richest Ant vs. 2-opt every 500 generations.

Tracking Plan

Since the tour mutation is clearly highly correlated with the success of the packing algorithm, rather than aiming for pure novelty we decided to mirror the mutation operations for the tour with a matching mutation of the packing plan. The number of items per city was calculated based on the item count / node count ratio, the items were then moved with the cities. Additionally, the initial packing plan was generated using a probability of 1/ the rental rate to reduce the initial items for instances which are more expensive to travel through. Additionally, weighting was given to the probabilistic mutation step favouring not picking up an item early in the tour, but picking up items more readily later on. The basic algorithm was using an elitist ES-(1 + 10) algorithm, with probabilistic mutation and no cross over.

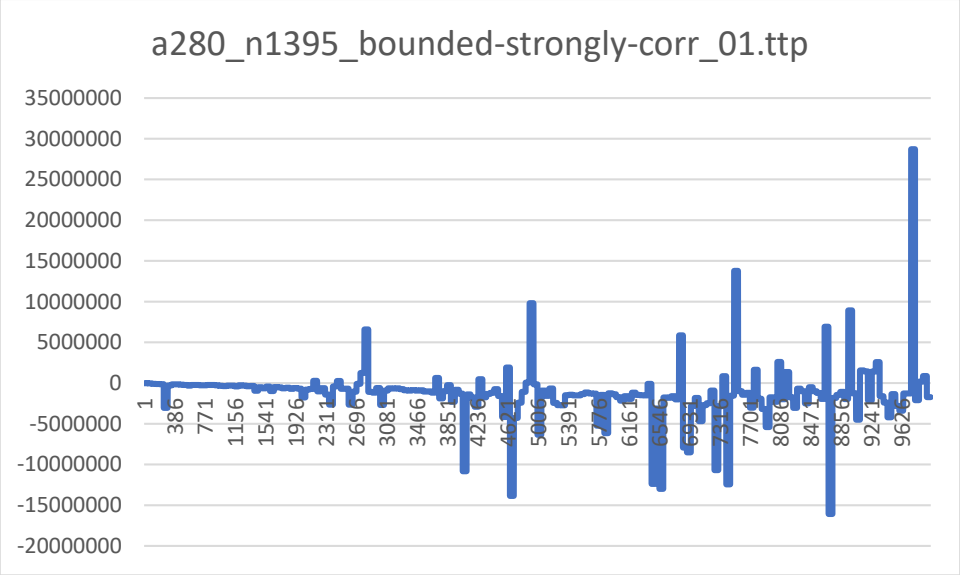

Figure 4 – Tracking Plan vs. 2-opt every 500 generations

We can see that there is some progress being made for the best tour, and we're able to find better solutions as the generations go on. However, in order to get the algorithm to recover from the tour modifications, we needed have extremely aggressive mutation. The solutions are still heavily coupled to the tour which means that most progress is lost with each generation, making for an erratic algorithm which may or may not produce a good result at any given stopping time.

We also imagine that the linken tours given are the optimum solution (although this is not clear). If this is the case the tour will be moving further from the optimum with each mutation which is the opposite of what we'd expect for a real scenario of concurrent optimisation.

Results

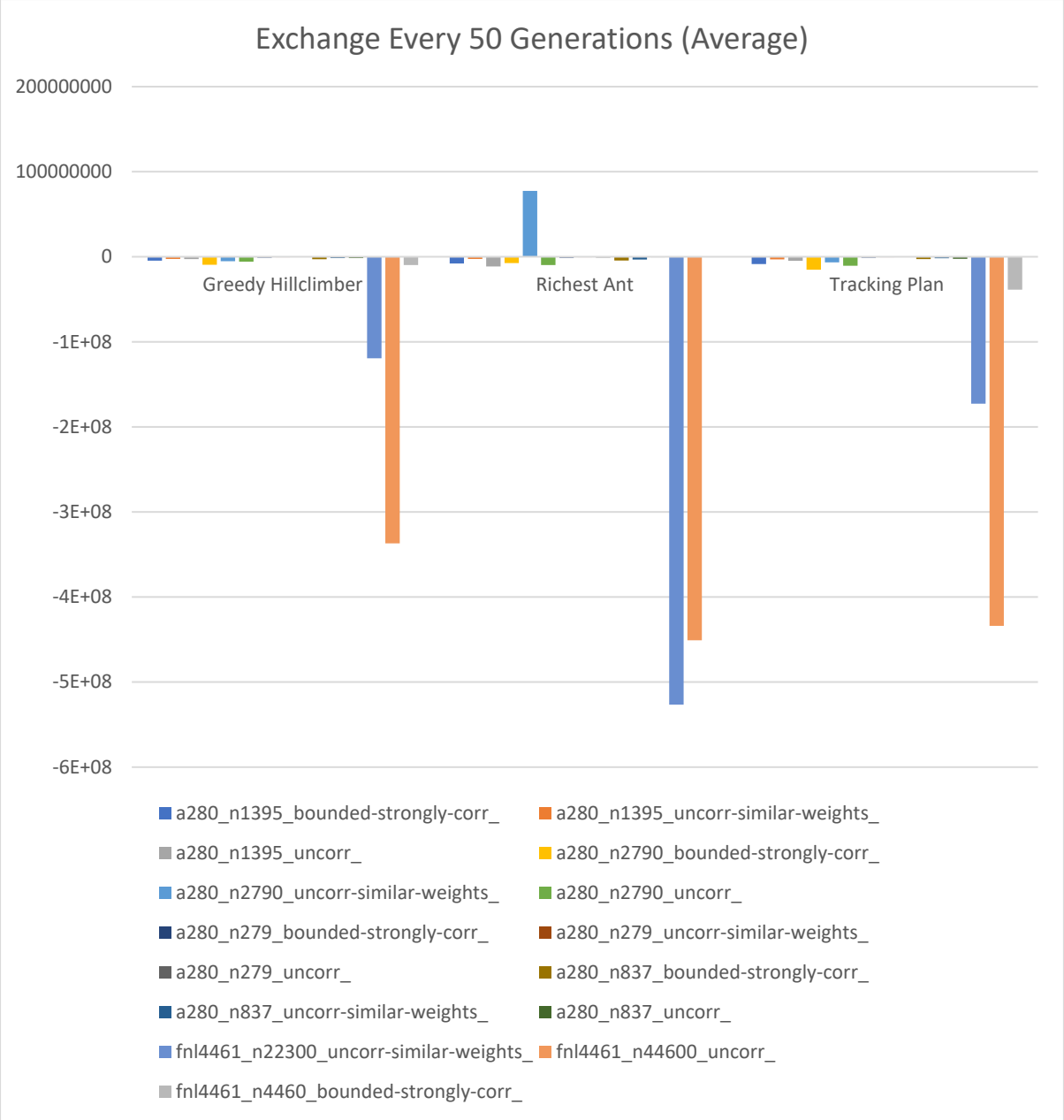

Figure 5- Summary of results for mode 1 (average)

This doesn't look very good…

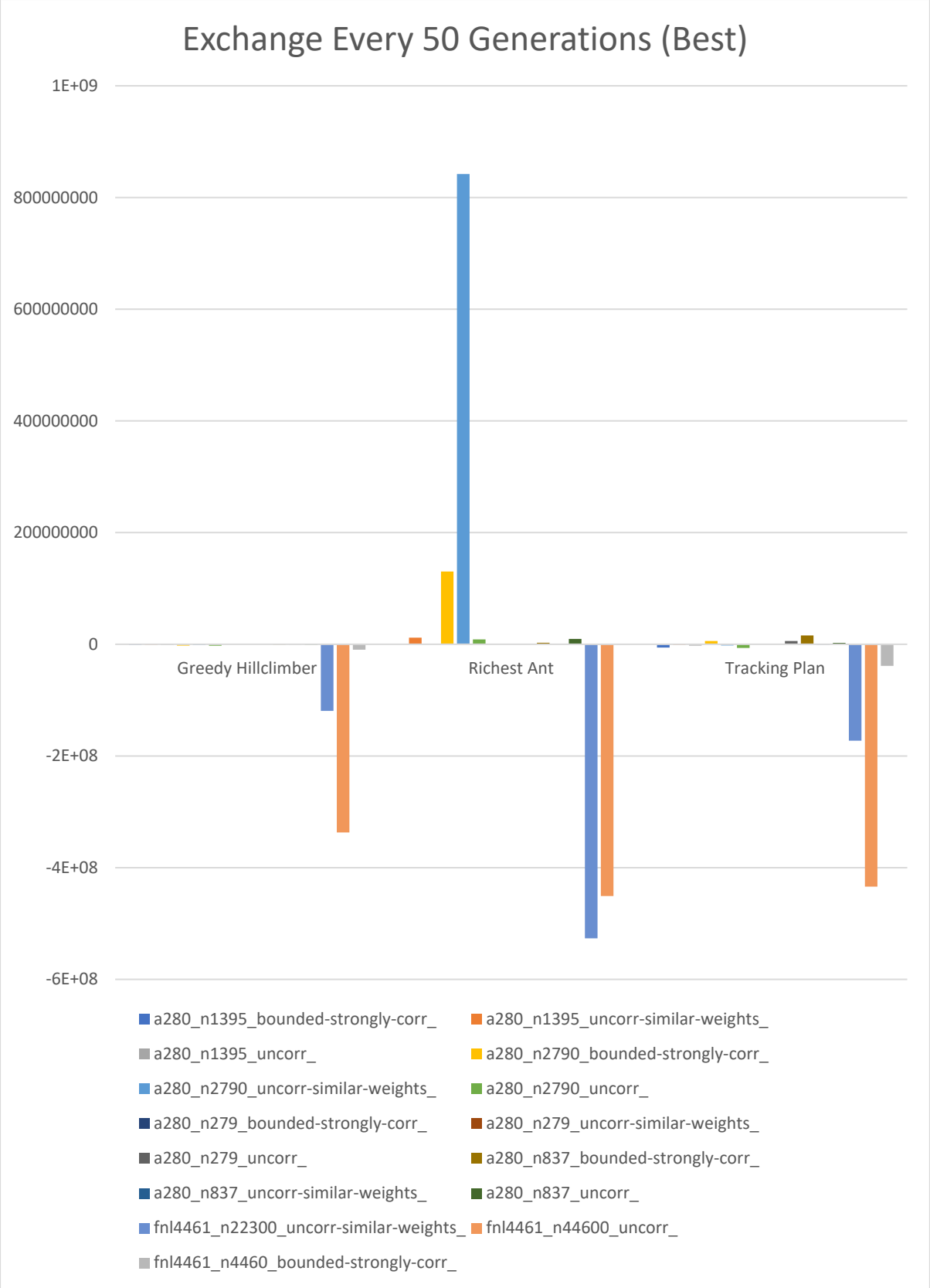

Figure 6- Summary of results for mode 1 (best)

However our more erratic algorithms pay off when it comes to their best scores.

And with a little more time to evolve between upsets…

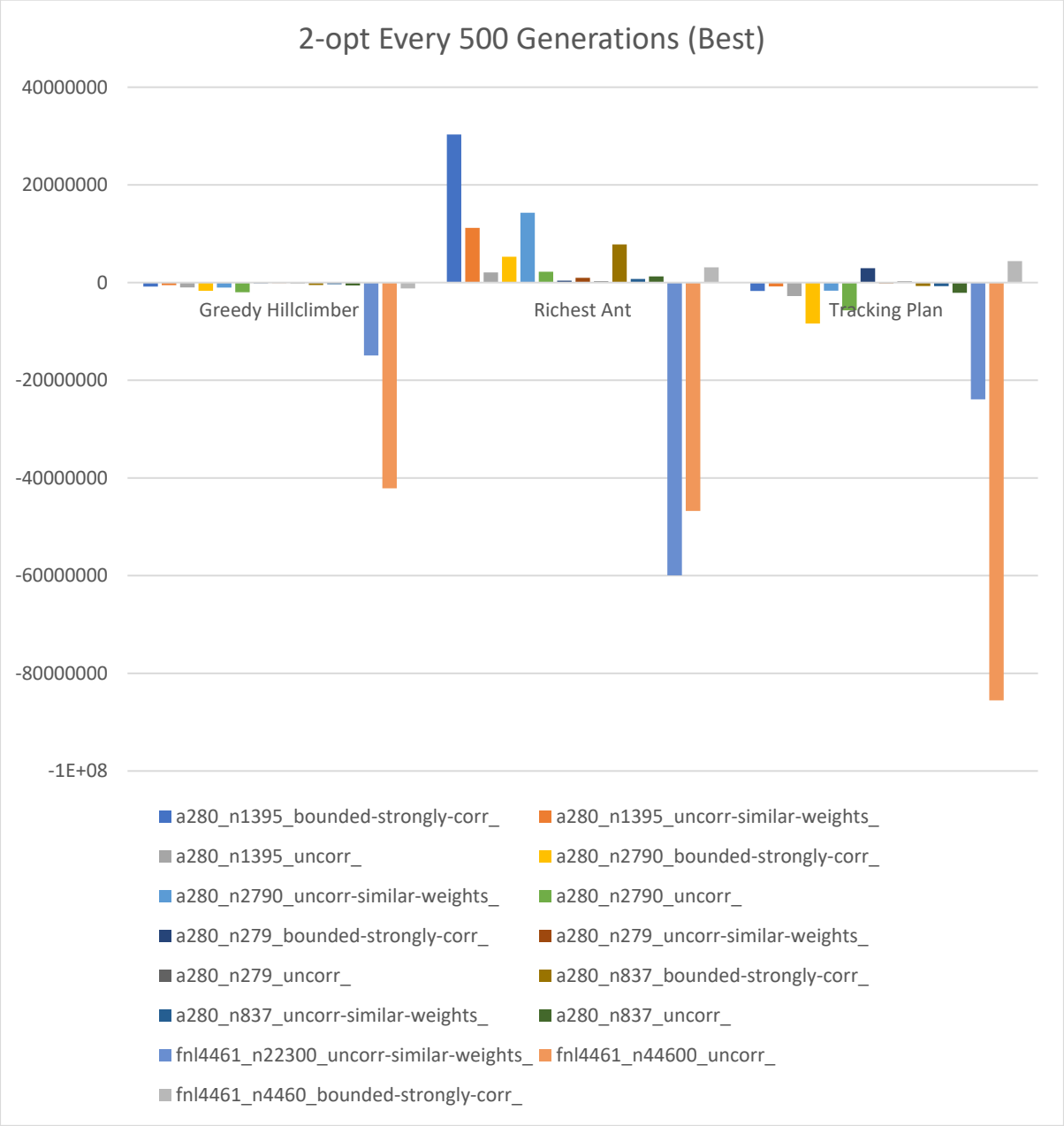

Figure 7 - Richest Ant can bring results

Although the data is promising we've certainly learnt that getting the tour right is an extremely valuable part of solving the TTP.

Data

Raw output data can be found in the output/ex3 folder.
The data can also be reproduced by running the main method in ec2017.ass2.ex3.Ex3Runner with the appropriate arguments where the algorithms are numbered as: 1 - Richest Ant, 2 - Tracking Plan, 3 - Greedy Hillclimber. Mode 1 uses exchange every 50 generations while mode 2 uses 2-opt every 500 generations.