Exercise 1: GA/Ants

Haubenburger Gabriel (11840531)   Seka David (11902064)

2024-11-17

Introduction

We chose to apply Ant Colony Optimization (ACO) and Genetic Algorithms (GA) to the Knapsack problem (with multiple knapsacks, and item dimensions), to TSP and to minimizing the Rastrigin function.

Findings

Runtime

For run times relative to instance size, we tested 5 iterations/generations each for ACO and GA on three different instance sizes.

For Rastrigin, we chose $n = 2, 3, 5$ and saw ACO increase from around 1 second to nearly 10 seconds. GA stayed roughly the same, increasing from around .021 seconds to .24 seconds.

For Knapsack, we chose $ks = 1, 5, 10$, $i = 10, 50, 100$ and $n = 1, 2, 3$ (these being the number of knapsacks, the items to choose from and the dimensions of the items respectively). We saw ACO increase from around 0.95 seconds to nearly 33 seconds. GA increased less, from around .1 seconds to 3.3 seconds.

For TSP, we chose 10, 20 and 50 cities, and saw ACO increase from around 0.09 seconds to around 14.3 seconds. GA once again was well below that, only increasing from around .3 seconds to .75 seconds.

Convergence

Below is a plot showing a comparison of GA and ACO on our three problems. Both were set to run for 200 iterations/generations, with an early stop for ants in case there was no improvement for 50 generations.

We can see that GA works decently for all three problems. ACO is generally worse, aside from quickly finding a good solution for the knapsack problem that is better than the one found by GA. For Rastrigin, GA outperforms ACO across the board. For TSP, ACO initially finds a decent solution, but gets outperformed by GA eventually.

We can further see that for GA, the global best is only rarely and then narrowly better than the current best, while the same isn't true for ACO.

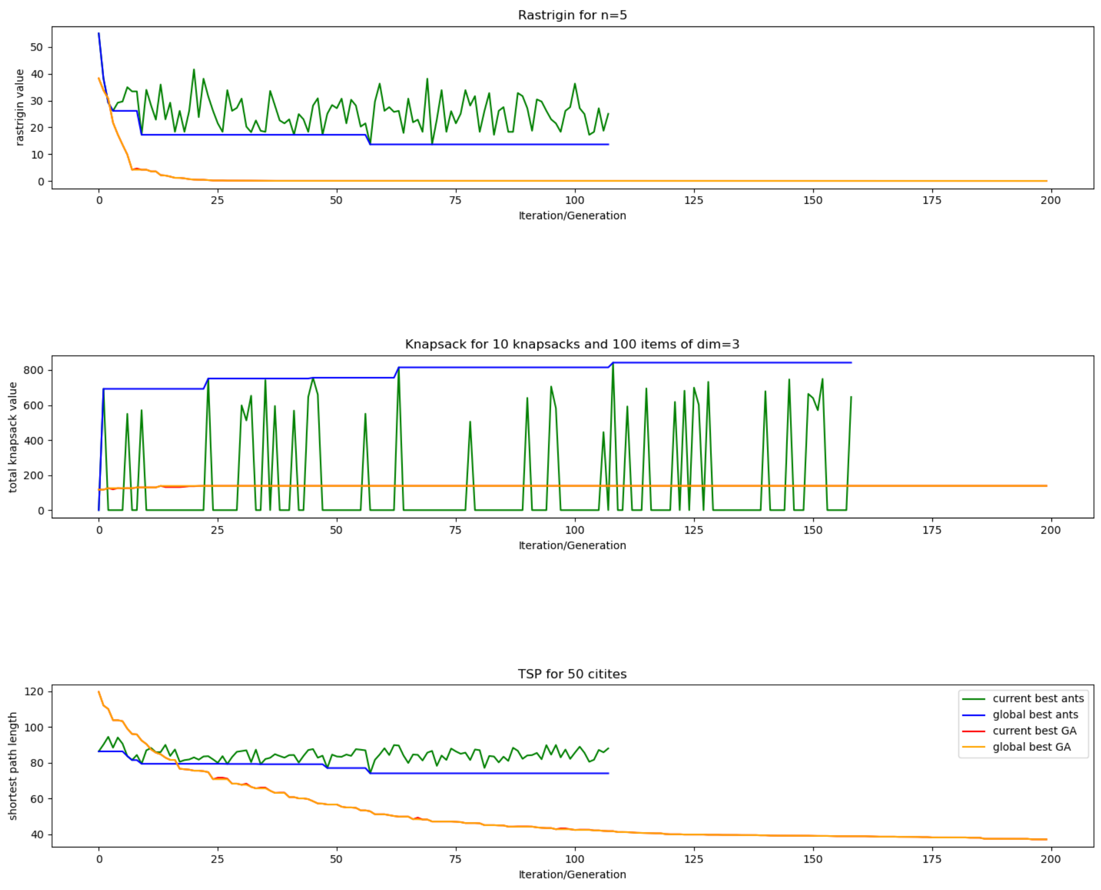

Abbildung 1: Convergence on the long run. Notice that for knapsack, GA and ACO had different start values

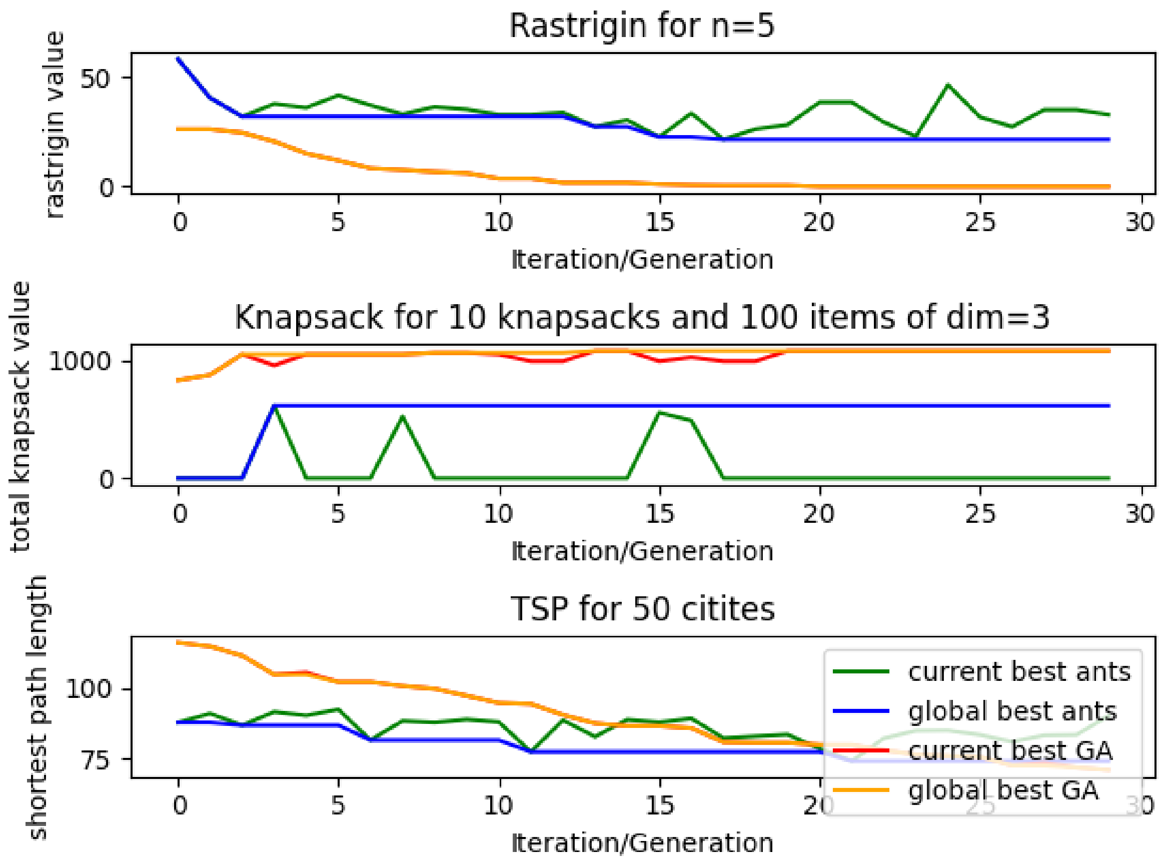

Abbildung 2: Shorter re-run of the suite due to time reasons. Now, Knapsack values are from the same instance.

Parameter tuning for Genetic algorithm

The genetic algorithm techinque turned out to be quite sensitive to hyperparameter tuning. Our tests showed, that if the mutation mutates too little or too much, the convergence is affected adversely. We conjecture, that this is due to the fact that too little change will not escape a local maximum, while too much change makes it hard to optimize the global optimum.

Furthermore, modifying too many values has a negative impact. Therefore, we scaled the probability of a dimension being modified for rastrigin and for a position being swapped for the tsp to the size of the instance. Tests showed, that too much variance does not tend to yield a good outcome.

Conclusion

- Rastrigin: GA is the obviously better choice.
- Knapsack: GA finds a better solution. As the solution of ACO is worse and finding it takes considerable time, GA would be advisable here.
- TSP: GA will likely be the better choice, since it scales better to larger instances and also outperforms ACO in the long run. However, ACO has an edge for the first few iterations. Therefore it might be used to generate cometitive solutions which might be refined by a GA eg.