ECMM409 – Nature Inspired Computation
Individual Report – Group B
Nils Wapki - 730080733

<u>My part in the project</u>

After researching potential algorithms in the literature and choosing a suitable approach my initial coding task was writing functions that deal with the multi-objective nature of the problem. For details on the algorithms surveyed I refer to the group project.

The functions I initially wrote are the following:
Dominates: This function takes two vectors as input and determines if the first vector dominates the second vector.
Non-dominated: I then used that method to implement *non_dominated()*, which checks for a given vector if it is not dominated by any point in a given set. This function is crucial for the outcome of the multi-objective problem since the final Pareto front will only consist of non-dominated points.
Non-dominated Sorting: I further wrote the function *non_dominated_sorting()*, which calculates the first Pareto front for a given set of points, removes these points from the set and then calculates the next best front. This process repeats until all points got categorised into a front. This process is similar to the one used by NSGA-II [1] (Non-dominated Sorting Genetic Algorithm II). By doing this, an order is imposed onto a given set of points. However, among all the points of one particular front, no order is established. The function *non_dominated_sorting()* can either return just the first Pareto front, or the best n points (where points are drawn randomly from the last front).
Tournament: The previous functions lay the groundwork for the tournament function I wrote. This function draws n individuals from the population randomly. These n individuals are then categorised into fronts by *non_dominated_sorting()*, and a random individual from the first Pareto front is returned.
Pick Best Pareto Points: Since the final submission will only allow 100, 50 or 20 Pareto optimal points, depending on the size of the data set, a subset of points has to be picked. A good Pareto front should be spread out as much as possible, because having Pareto optimal points clustered together does not contribute to the hypervolume as much as if they were spread out. Therefore, I calculated the Euclidean distance from each point to its neighbours. This is similar to the Crowding Distance used in NSGA-II [1]. Then, points with the smallest distance to its neighbours are being removed iteratively, until the needed number of Pareto optimal points is reached. The function makes sure that edge points (first of last point in Pareto front) are not being removed, as they contribute significantly to the hypervolume.

This concludes the initial part of my work, after every team member submitted their code fragments, Richard tried to put all pieces together and to get it work with a main function he wrote. Since code from other team members did not work properly, I helped him to rewrite parts of it. Richard and I rewrote the fitness function because it was not working properly. Further, I rewrote the *calculate_speed()* function because it featured logical errors and was unnecessarily complex.

In the main function, I added code to calculate the hypervolume of a given Pareto front. For that I also needed to choose a suitable reference point. Usually, it should be the nadir of all Pareto points under review (for all runs) multiplied by a factor slightly larger than 1 (e.g. 1.01), to make sure that the edge points can contribute to the hypervolume. However, for our test cases it was sufficient to choose a reference point that is definitely larger than any nadir for any run of a given dataset. After a few test-runs I chose [1e-3, 1e5] for dataset 1-3, [1e-3, 1e7] for dataset 4-6, and [1e-3, 5e10] for dataset 7-9.

Before starting the real runs with the final parameters, I made sure the objective values and the tour and packing plans were properly saved in the right format and that the necessary plots were being generated. I then ran all final experiments and saved the data that we had to submit.

Finally, Richard and I wrote the group report, as we conducted most of the research and made most improvements to the algorithms.

Improvement of the Algorithm and Experiments Conducted

Even though a discussion of the algorithm survey was listed as a requirement for this individual report, I would like to refer to the group report for details on this to avoid any potential overlap of information. The reasons for the algorithm selection and the related research are discussed in the group report, which was written by Richard and me.

After managing to run the algorithm for the first time, we noted two main problems: There was a lack of diversity of solutions on the Pareto front and the performance decreased significantly for even the medium-sized datasets.
Even though we all agreed in one of our first meetings to use numpy arrays for efficiency, all team members, except for Richard and myself, used normal Python lists. Furthermore, a lot of unnecessary for loops were used that slowed down the computation time. Thus, Richard and I rewrote functions to feature numpy arrays and then, most importantly, replaced for loops with much faster numpy operations. For example, calculating the profit in the fitness function can be achieved with a combination of np.sum and np.where, instead of a for loop, which increased efficiency by orders of magnitude. I further rewrote the calculate speeds function again to remove a for loop with numpy searches, which greatly increased performance. Richard even managed to increase performance even further by pre-calculating arrays outside of the loop.
Furthermore, I added Ordered Crossover (OX) and the Cyclic Crossover (CX) to our set of crossovers for the TSP tour, both employing numpy arrays.

In the beginning we agreed on using local searches for both the TSP and the KP, and to omit simple mutations instead. However, the TSP local search did often not result in better objective values, and the KP local search used up a lot of computation time. Therefore, we discarded the local searches and only employed the TSP local search for the initialization of the population. Instead, we resorted to simple mutations, which worked well in promoting exploration. By improving on our local searches, however, we might have achieved better exploitation of the results.

In order to increase the quality of the Pareto front and to explore in more directions, I thought about interactive preference incorporation by weighing the objective functions. I further researched into using decomposition based MOEAs, such as MOEA/D [2], where a set of reference vectors, which MOEA/D needs as input, is being adjusted to incorporate a preference towards a specific reference point. I discarded this approach eventually, because, as I will explain further down, the quality of the Pareto front improved greatly by adjusting the way we replace individuals in a population each iteration.

After fixing most of the efficiency issues, further test runs revealed that the convergence of the hypervolume was not steady. Sometimes, the hypervolume decreased, which means that an individual that is featured on the Pareto front gets replaced by a new offspring which is not Pareto optimal. To fix this, I altered the replacement function to check if the individual that is supposed to be replaced is present in the Pareto set. If yes, the replacement function should look for another point that is not in the Pareto set. This fixed the convergence issue and the hypervolume then increased steadily. However, replacing points semi-randomly was still not an optimal solution. Richard improved this further by taking the crowding distance into account and replacing the point in the last front that has the smallest crowding distance, i.e. the point that is closest to its neighbours. This step greatly improved the quality of our final Pareto front.

After the parameter search (for more information on that I would like to refer to the group report) was done, I drew the conclusions out of the data and chose the parameters, taking the resulting hypervolume, the size of the Pareto front and the runtime into account.

<u>**Reflection on the Teamworking Process**</u>

All team members attended all online meetings. I personally think that meeting in person is more productive and better for the outcome as a whole because you get to know your team members on a personal level as well. However, only Richard wanted to meet up in person. Sadly, I have never seen any of the other team members.
All team members were always kind and polite. Everybody readily accepted their assigned task and worked on them.
The main issue during the meetings was the language barrier. Yinghui, Binnan, and Zhen possessed very limited English proficiency, which made a live conversation barely manageable. Therefore, mostly Richard and I were talking, the rest was mainly listening. Ashutosh spoke English well but also rarely contributed to conversations. We tried our best to value everybody's opinions and ideas, and I hope this was achieved despite the existing language barrier.
Texts in the chat were translated to English, which sometimes featured ambiguous translations. The fact that some team members were unable to properly communicate in English made working in a group very hard. This is, however, partly to blame on the University for accepting students that do not possess the necessary language skills to properly understand academic content and express themselves, just because those students bring in a lot of money for the University.

An even bigger problem, however, was the lack of initiative of the other group members, except for Richard. Richard gave the most input in all our conversations, did the most research on how to further improve our algorithm, and was overall the driving force for the whole project. From the start of the project, it became clear that Richard and I have to distribute and assign tasks and, most importantly, set deadlines. Tasks we assigned to other team members were happily accepted and worked on. But, here a lack of initiative to write test cases, check the own code (most importantly parameters) against the code of other team members and to understand the algorithm and problem case as a whole rendered the submitted code fragments often unready to deploy.
We also set up a One-Drive with a folder dedicated to the current version of the project to work on. Still, some members of the group continued to work on their local code files that were outdated for days.

After managing to run the algorithm for the first time, our next strategic goals were to improve the runtime of the algorithm and the quality of the Pareto front. These next steps required a deeper understanding of the numpy library to improve efficiency, and a good understanding of how to handle multi-objective problems to achieve a better Pareto front. Both these skills, or rather interests therein, were hardly present in the other team members except for Richard and me, which made it hard to further assign tasks. We distributed the conversion to numpy arrays and more importantly the improvement of the efficiency of the code, but there were basically no usable solutions from other team members. Tasks were only done on a very basic level, without any deeper understanding. Most tasks would have needed too much explanation for someone with little knowledge of the topic and no initiative to deep-dive into the topic oneself. Therefore, considering the cost-benefit ratio, Richard and I basically did most of the work on our own from thereon.
I am aware of the fact that distributing tasks in a smart way and assigning tasks to fellow team-members based on their interests and competencies is a skill in itself that I did not express by doing most of the work on my own. However, the lack of initiative shown by other team members and the very clear directions they needed left me little room for action.
I liked the in-person sessions with Richard the most about this groupwork. It is really fun to try to solve a problem with someone that is equally enthusiastic and interested in the topic and has equal proficiencies. By discussing and working in a team, I have definitely broadened my horizons and managed to think outside the box.

Conclusion
I would consider the algorithm both in terms of efficiency and of objective values achieved a success. I have learned a lot during this project, in terms of knowledge about multi-objective optimization and nature-inspired computation, but also in terms of working as a team and communicating effectively.
Regarding further improvements of the code, I would like to try out dynamic programming as a packing heuristic, since initial heuristics have proven to be very valuable for our results.
I would like to conclude this report by saying that this course, and especially the hands-on projects, reinforced my opinion to pursue this field of research in the future.

Literature

[1] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan: "A fast and elitist multiobjective genetic algorithm: NSGA-II," in IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182-197, April 2002, doi: 10.1109/4235.996017.

[2] Q. Zhang and H. Li, "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition," in IEEE Transactions on Evolutionary Computation, vol. 11, no. 6, pp. 712-731, Dec. 2007, doi: 10.1109/TEVC.2007.892759.