Universität Bielefeld

Fakultät Wirtschaftswissenschaften
Lehrstuhl für Decision and Operation Technologies

Bachelorarbeit

im Studiengang Wirtschaftswissenschaften

Solving the SpOC GECCO Trappist Tour challenge

von
Paulina Heine
3977204
Im Flachskamp 5, 32107 Bad Salzuflen
paulina.heine@uni-bielefled.de

vorgelegt bei
Prof. Dr. Kevin Tierney
J.-Prof. Dr. Michael Römer

July 8, 2022

Abstract

Inspired by the recent SpOC GECCO system exploration competition, this thesis aims to find the best possible approximation of a pareto frontier for a multi-objective optimization problem by decomposing it into partial Problems. The decomposition leads to two partial problems: identifying optimal sequential orders of visiting the exoplanets and, based on these results, determining the optimal values for the continuous parameters for these permutations to minimize both objectives. Multi-objective optimization is frequently discussed in quantity of papers, notwithstanding scrutinizing if a decomposition of this complex problem and applying the elaborated approach for the partial problems leads to valuable solutions. Which sequential orders are optimized highly affects the quality of the solution, wherefore identifying and optimizing good sequential orders is crucial to heighten the chance of attaining a good solution. To mitigate randomness in the set of sequential orders, valuable sequential co-occurrences of planets are analyzed and used for new sequences to find high-quality sequential orders, which can then be optimized. The concept of finding those valuable solutions is constructing and improving an approach for the permutation problem and testing if a genetic algorithm is capable of finding a valuable score. This thesis aims to present an approach to solve this problem by decomposing it and reducing the dimensionality. The final approach will be tested by experiments and then be applied in order to demonstrate how this astrophysical problem can be solved without technical knowledge about astrophysics but with well-drafted approaches.

Keywords: multi object optimization, decomposition, complex TSP, permutations

Contents

1 Introducing the system exploration challenge

Real life is defined by complexity. Since decisions often have far-reaching consequences and therefore affect many things, which in turn affect other things, controlling and steering these consequences in as many possible directions is desired in almost every decision in real life. Consequently, no matter in which field problems arise, in most cases, they have more than one objective. These objectives are not independent of each other, hence the level of satisfaction of one influences the level of satisfaction of the others. In most cases, the objectives conflict with each other, which is why a trade-off decision comes up. Because this applies to most problems in real-life, developing new approaches and developing already worked out approaches further to solve those problems by optimizing the values that determine the level of satisfaction is a primary goal in the optimization field.

Although it is a hypothetical futuristic challenge, being multi-objective adds to the realism of the current system exploration space optimization competition challenge provided by the European Space Agency. The goal consists of determining trajectories for exploring the seven currently known Trappist One exoplanets while using as little time and propellant as possible to take samples from these planets. Minimizing both objectives which are the needed time and the used propellant is essential to finish the mission, but because they conflict with each other, simultaneously minimizing both is not possible since it costs more propellant to reduce the required time and more time needs to be afforded to reduce the required propellant.

Not only does the multi-objectivity makes this problem very interesting, finding the order of planetary visiting, which is determined by one part of the decision vector, shows similarities to the NP-complete Traveling Salesmen Problem, which is a highly relevant problem in combinatoric optimization.

In many cases, the setting of the original Travelling Salesman Problem is not complex enough to depict reality which is why more complex settings are developed like the deliveryman problem, the repairman problem, or the minimum latency problem (Bigras et al., 2008). These are different from the origin TSP because they are time-based and hence more complex. Nevertheless, the system exploration problem is more complex because it adds the astrophysical component with moving planets instead of fixed points on a graph to which individual orbits and gravitations belong. Here again, it is evident that real life is complex

Considering the problem abstractly shows another aspect that adds to the importance and realism of this problem. The urge to explore space may become a requirement in the future. It is unsure how long the earth will be capable of hosting human life. Even if exploring planets to mine resources or even host human life is not required soon, it probably will be in the future.

The setting of the challenge makes the astrophysical component vast, not only because all of the values in the decision vector determine astrophysical components of the trajectory but because the problem contains high complex astrophysical problems like the Lambert problem, which constrains the values for the decision vector. To be capable of solving this problem without prior knowledge about those astrophysical components, it ought to be considered a black box. Optimizing the problem without any or only with brief expertise is a considerable ability, but the lack of knowledge needs to be balanced by a promising approach.

Decomposing into partial problems is a reasonable approach used in this thesis. It allows focusing on smaller and manageable partial problems. This decomposition resolves into two partial problems. The first is a combinatorial optimization problem about locating the best possible sequence of visiting the seven planets through analyzing their sequential co-occurrence. The second is to optimize the continuous variables for each sequence of the set of sequences that resolves from the first partial problem by using genetic algorithms.

2 Literature regarding components of the problem

Since the challenge concerning this thesis took place while the thesis was being worked on, there are no publications for this exact problem. Nevertheless, publications on parts of the problem, like multi-objective optimization and the Travelling Salesman Problem, are available. Since these topics are an essential field of research, approaches for solving problems that show similarities are developed.

2.1 Decomposition

Wu et al. (2018) introduces how decomposing optimization problems is a common approach when solving complex problems. According to Ma et al. (2015), it is interesting to decompose a complex high-dimensional problem into a set of more straightforward and low-dimensional sub-problems that are easier to solve.

Through the decomposition of the system exploration task, two partial problems must be resolved: seven categorical integers determining the sequence of visiting the exoplanets and a 27-dimensional vector containing continuous variables that need to be optimized.

2.2 References to the Travelling Salesmen Problem

Arranging the seven integers shows similarities to the Traveling Salesman Problem, an NP-complete combinatorial optimization problem, which aims to find a path in a fixed graph from one beginning node to an ending node, visiting each node exactly once with the goal of minimizing the costs of traveling through the pointsBaumgardner et al. (2009). The given problem appears to be much more complex than this. Besides the classic Travelling Salesman Problem problem only has one objective, in the system exploration task, every exoplanet shows an individual orbit, gravity, and other astrophysical parameters. That makes a huge difference since the planets are no fixed points on a graph.

Since in many real-life cases, there are no fixed points in a graph, there is a lot of research regarding more complex Travelling Salesman Problem problems like a time-based Travelling Salesman Problem for trucks delivered by unmanned drones Boccia et al. (2021). Even if extensions increase the complexity of the combinatorial optimization problem, the system exploration problem is much more complex

because an orbital trajectory is requested with no fixed trajectories between the planets. From this can conclude that applying approaches for the origin Travelling Salesman Problem or Time-based Travelling Salesman Problems is not enough to solve such a task. An approach specific to space is required.

Zhang et al. (2022) introduced complex problems like these in the setting of debris removal missions and named three possible ways of locating the best sequences, which are explicit enumeration where every permutation is going to be optimized, implicit enumeration where only a set of promising permutations is going to be optimized and stochastic programming methods. This thesis will focus on implicit enumeration because of the savior of computation time and the significant number of possible permutations, which would be 5040.

López-Ibáñez et al. (2022) worked on a problem quite similar to the partial problems of the system exploration problem. The author refers to them as outer and inner problems. While the first includes finding optimal permutations, the latter aims to find the optimal values for the remaining parameters in the setting of the outer problem. The author tested approaches like a combinatorial Efficient Global Optimization and an Unbalanced Mallows Model. He concludes that more research is required to find optimal permutations.

What stands out is that the approaches for finding optimal sequences in space are mostly very young. For those problems, it is hard to find an approach that can be applied to every other complex Travelling Salesman Problem, which is why analyzing every problem individually and looking for approaches is reasonable.

2.3 References to multi-objective optimization

Assuming that valuable permutations are found, the 27 continuous variables for each permutation needs to be optimized. This is a multi-objective optimization problem with reduced dimensionality. Like Deb (2011a) describes, the problem of conflicting objectives is that the optimal solutions of the objective functions are different, which prevents simultaneous optimization of each objective. Two main approaches are named in publications.

2.3.1 Traditional approach

Konak et al. (2006), as well as Deb (2011b), explains one popular approach for optimizing conflicting objectives: Simplification of multi objectives into a single objective, which is the traditional way of solving those problems. The authors introduce two ways to achieve this.

The first one is to evade the conflicting objectives and the resulting trade-off by scalarizing multiple objectives into one objective. This simplification can be archived

by combining the conflicting objective functions into a single function. A typical application would be the weighted sum method, where a set of objectives is scalarized into a single objective by adding each objective a user-supplied weight and summing them up in one function. This reduction might be problematic when selecting the weights to show preferences. Further, in the case of a non-convex objective space, some Pareto-optimal solutions are not obtainable if this specific approach is applied.

The second way of simplification is to move all but one objective to the constraint set. The problem is that when objectives are moved to the constraint set, a constraining value must be established for each of these former objectives.

In both simplification cases, an optimization method would return a single solution rather than a set of solutions that can be examined for trade-offs. Often a set of valuable solutions is preferred over a single solution, which is why another method gained popularity.

2.3.2 Usage of genetic algorithms

"A reasonable solution to a multi-objective problem is to investigate a set of solutions, each of which satisfies the objectives at an acceptable level without being dominated by any other solution." What Konak et al. (2006) describes in this citation is the usage of genetic algorithms, another popular approach to solve multi-objective optimization problems. These are meta-heuristic algorithms that are based on the evolutionary principles of Charles Darwin. Because of the advantages, many genetic algorithms are developed and are a popular way of solving multi-objective optimization problems.

The fact that a challenge was published that deals with the topic of multi-objective optimization and a complex version of the Travelling Salesman Problem shows the relevance of these topics and explains why investigating such a challenge is essential.

3 Problem description

3.1 The system exploration challenge

The main goal of the system exploration challenge is finding the best possible approximation of a pareto frontier whose solutions complete the space mission. This contains visiting successively every of the seven currently known exoplanets of the Trappist one system exactly once, starting from a fixed point 10 AU around the Trappist one system. While doing that, the required time and $\Delta(V)$(the cumulative change in spacecraft velocity)for this mission should be minimized.

The spaceship's trajectory and the used time and $\Delta(V)$ are determined by the values of the 34-dimensional decision vector. Figure 3.1 shows how the decision vector is composed. The blue part represents continuous variables, while the green part represents categorical integers. The categorical integers determine the order of planetary visiting while the continuous determine the details about the trajectory. The exact meaning of the continuous values in the continuous part will not be explained further since they are astrophysical variables and are not included in the scope of this thesis.

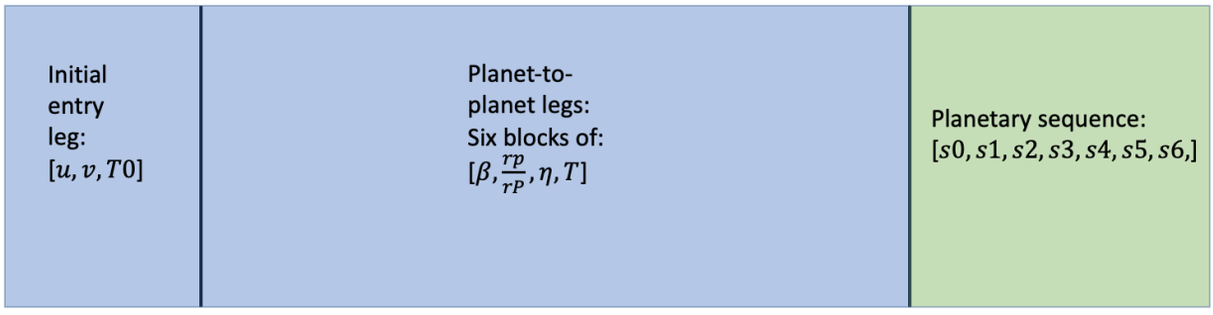


Figure 3.1: Visualisation of the variables in the decision vector

The difficulties of this task consist in satisfying both objectives at the same time. Optimizing both time and velocity leads to a conflict, as the optimal solution of the objective time function is different from that of the objective ΔV function, which is why the minimization of one of the objectives leads to the increase of the other

objective and vice versa, for which reason a trade-off needs to be computed. The included astrophysical components of the problem constrain every variable in the decision vector and lead to a restricted feasible decision space.

Figure 3.2 shows a visualization of how the values of the decision vector determine the objective vector. It needs to be taken into consideration that the decision vector has more than two variables, so the visualization shows a simplified decision space. The difficulty consists in not knowing how the values in the decision vector influence the objectives. What is represented by the black arrows in the figure is impossible to calculate without prior knowledge through which leading back, which value influences which objective in which way is not possible and not the aim of this thesis.

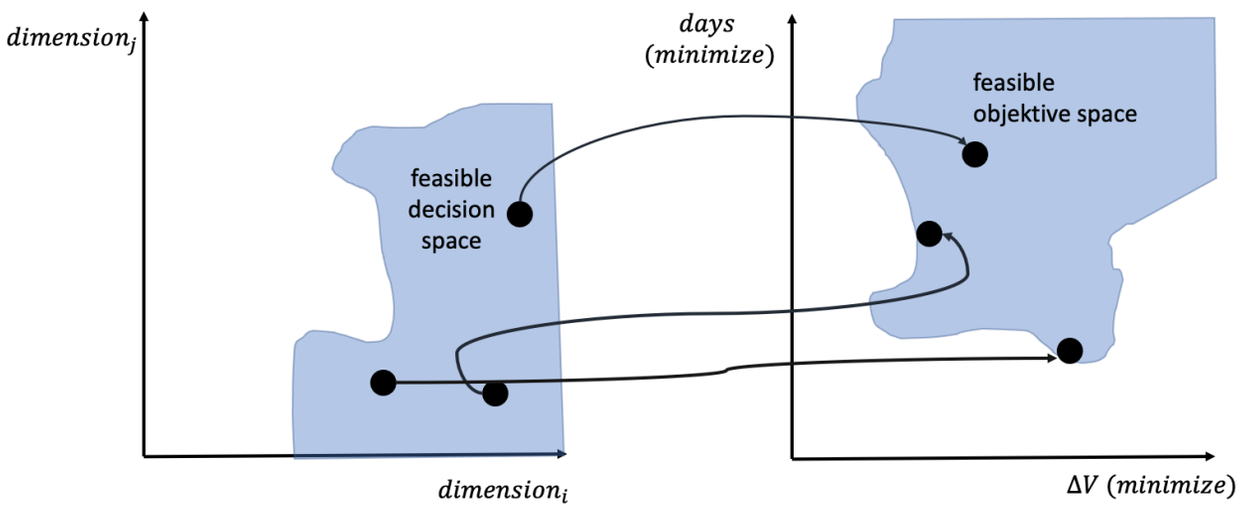


Figure 3.2: Visualisation of connection between decision space and objective space

3.2 Matematical formulation

A general mathematical formulation cited from Tušar and Filipič (2014) for those problems is:

$$\boldsymbol{f} : X \to F$$

$$\boldsymbol{f} : (x_1, \ldots, x_n) \mapsto (f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n))$$

Where X is an n-dimensional decision space and F is an m-dimensional objective space ($m \geq 2$). Each solution $x \in X$ is called a decision vector, while the corresponding element $f(x) \in F$ is an objective vector. Without losing generality it can be assumed that $F \subseteq \mathbb{R}^m$ and all objectives $f_i : X \to \mathbb{R}$ are to be minimized for all $i \in \{1, \ldots, m\}$.

In a mathematical form, the given problem is:

$$Objective\,1 :\ minimize \quad f_{time}(x)$$

$$Objective\,2 :\ minimize \quad f_{\Delta(V)}(x)$$

$$Inequality\,Contrain :\ g(x) \leq 0$$

$$Equality\,Contrain :\ h(x) = 0$$

$$Boundaries :\ x_i^L \leq x_i \leq x_i^U \quad i = 1, ..., N$$

The inequality constrain guarantees that the closest distance of the spacecraft to the star TRAPPIST-1 must at no point in time be smaller than 834840 km while the equality constrains ensures each planet is visited exactly once.

3.3 Solutions

Due to the conflicting objectives, there is not only one optimal decision vector that satisfies both objectives completely but a set of mutually incomparable decision vectors that satisfies the objectives at an acceptable but varying level. Since all the decision vectors focus on a certain objective at varying degrees, some of these vectors satisfy one objective more than the other. At the point where solution X satisfies objective one more than another solution Y and the second objective is at least equally satisfied, Y is dominated by X

$$f_i^X \leq f_i^Y \, for \, \forall \, i \in \{1, ..., m\} \, and \, f^X \neq f^Y$$

(Tušar and Filipič, 2014)

The set of objective vectors, which are not dominated by another objective vector constitutes the pareto front. The goal is to locate the decision vectors, that provide

objective vectors which dominates as many as possible other objective vectors.

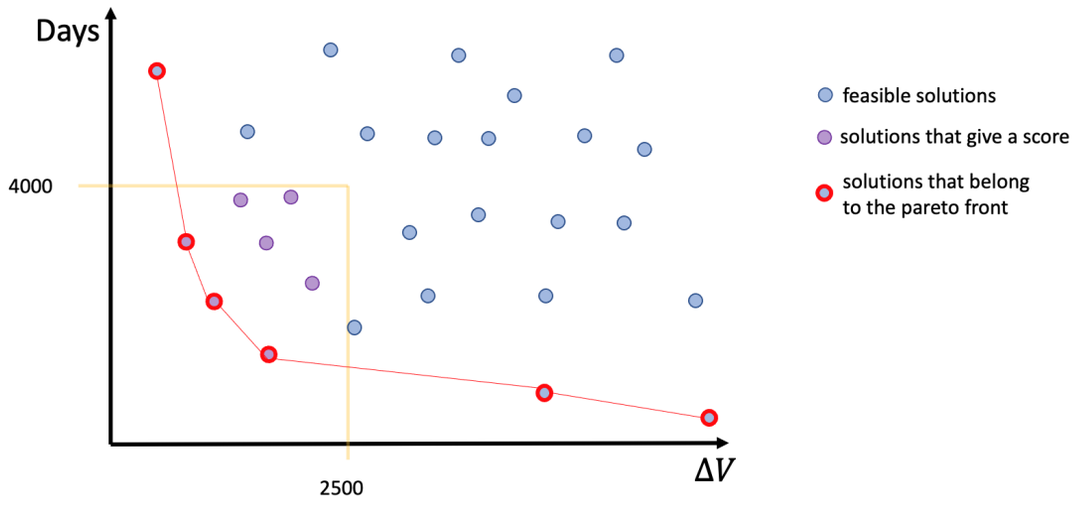


Figure 3.3: Depiction of possible pareto front

As introduced by Pardalos et al. (2017) the feasible objective space of the problem can have a convex or a non-convex form, drastically affecting the possibilities of solving such a problem. Due to the complexity of the given problem, the objective space is in a non-convex, so adjusting the parameters' values would not necessarily lead to a satisfying solution. It would lead to a pareto frontier, but perhaps the best possible one.

In consequence of the mutually incomparable solutions, evaluating the quality of single objective vectors or a set of them is required. Several indicators can measure this quality. The hypervolume indicator(also known as the Lebesgue measure or S-metric) is used to evaluate solutions in this challenge. (Nowak et al., 2014).

4 Approach for achieving a valuable score

4.1 Decompose problem

The astrophysical problem and the size of the presented problem add to its complexity. Based on this complexity, it can be asked if a decomposition would be rational. Ma et al. (2015) explains that prior knowledge about the objective functions is required to decompose a problem. Even if there is no prior expertise about the astrophysics behind these functions, like in figure 3.1 shown, the decision vector contains different kinds of variables, which enables decomposing the problem. Instead of considering the problem with all 34 values, decomposing it into partial problems leads to the following two partial problems:

Permutation problem
: Find the optimal sequential order of visiting the exoplanets, determined by the seven categorical integer variables.

Solving continuous parameters
: Determine the optimal values for the continuous astronomical parameters in order to minimize both objectives

One reason for doing this is the difficulty of optimizing complex problems. The more complex the problem is, the more demanding valuable solutions are to achieve. Decomposing the problem enables looking at two reduced problems more precisely and independently. Hence understanding these problems and finding options to solve those problems as well as locating the origin of problems is more manageable, which is helpful regarding the complexity and the astrophysical-related field of the problem (Ma et al., 2015).

The decomposition reduces the dimensionality of the multi-objective optimization problem from a 34-dimensional vector to a 27-dimensional vector which allows focusing on those 27 dimensions while the order of planetary visiting is in the first partial problem.

A disadvantage of this decomposition that has to be considered is that it would result in 7! possible multi-optimization problems instead of only one. Solving all of these would increase the computational time of the problem drastically. Because of this, an approach needs to be developed that classifies the permutations and determines which ones will be optimized.

Considering the advantages of decomposing the problem, this will be done in order to solve these problems independently and result in better solutions.

4.2 Permutation problem

This partial problem aims to find promising permutations out of the 5040 possible permutations. A reasonable question is if the time needed searching for those permutations would be better invested in optimizing random permutations. This would lead to a trade-off between exploitation and exploration. The number of possible permutations needs to be examined to assume. Considering the opportunities named by Zhang et al. (2022) for solving those complex combinatorial optimization problems, optimizing all 5040 permutations, known as explicit enumeration, would be an option. According to him, explicit enumeration is not possible if more than ten objects need to be visited. Although only seven objects need to be visited in this problem, the question should be asked if explicit enumeration would be reasonable since some permutations seem more promising than others. Optimizing permutations that cannot deliver valuable solutions, even if they are optimized with highly complex algorithms, would mean losing time since high complex algorithms require much time. Suppose an algorithm's complexity is increased to find better solutions, the computation time of the algorithm increases as well. Only promising permutations should be optimized to maximize the chance of locating reasonable solutions. To optimize those permutations, a small subset of permutations needs to be located. In the following, three opportunities to select a set of permutations are presented.

4.2.1 Randomness

A quite naive opportunity is to select a random number of permutations out of the 5040 and then optimize these. The reason for this being naive is that, besides the optimizing algorithm used, the solution quality depends unmitigated on which permutations are selected and because this happens randomly, the solution quality is also random. This is a significant problem because optimizing unnecessary permutations expends computational time and effort that could be used to optimize valuable permutations. The more permutations are tested, the higher the chance of optimizing a valuable permutation. However, testing an increased number of permutations, the computational time for each permutation needs to be decreased to execute the approach on a regular computer.

Despite this approach having significant issues, a suitable decision vector can be located because optimizing a valuable permutation is possible with good luck.

What results is the need to control the quality of the permutations that are optimized by classifying the permutations, which can be done by reducing the randomness and controlling which permutations are tested.

4.2.2 Rank permutations based on fixed decision vector

Evaluating the potential quality of permutations based on a fixed solution by evaluating how all the permutations perform in that solution to select the best-performing permutations to be optimized leads to significant issues. The selected permutations depend on the fixed solution. Since the values of the continuous variables determine if permutations seem promising here, optimizing the best permutations seems not promising. Difficulties may arise in ranking whole sequential orders of 7 planets because it is too complex to rank them in one whole system. Because of these disadvantages, this approach is discarded.

4.2.3 Analyse sequential co-occurrence

Neither the randomness nor the ranking method seems promising. Observing the weakness of the randomness approach, a mechanism to control the quality of the optimized permutations is required to use computational time on the right permutations. The weakness of the second approach is that ranking complete permutations could fail because of the sensitivity according to the continuous variables. An approach that the weaknesses of both approaches equalize is required.

Since whole permutations are hard to evaluate, analyzing sequential co-occurrences of planets seems to be an auspicious method. If optimizing the permutations "0-1-2-3-4-5-6" and "6-0-4-5-3-2-1" leads to a high-quality solution, it can be assumed that the sequence "4-5" is promising. Evaluating which co-occurrence seems valuable can be done with a score. After several randomly selected permutations are optimized and divided into sequences of two, depending on their score, a counter is added or subtracted to a belonging sequence. Resuming the most promising sequences of two gaining a high score.

Algorithm 1 Classifying sequences of two

```
Find x random permutations

for each permutation do
    optimize permutation
    if Score of permutation < X then
        add value on score for each sequence of two
    if Score of permutation = 0.0 then
        subtract value on score for each sequence of two
for every sequence of two do
    if Score of sequence of two > Threshold then
        append to list of best sequences
```

The number of first optimized permutations and the settings of the scores have a major impact on which permutations are optimized in the end and, by that, the quality of the solution.

These promising sequences of two attached might form beneficial new permutations that could be optimized further.

An issue of attaching every sequence as long as every planet is only once in a new permutation is that it does not pay attention to the transitions between those sequences. Even if two sequences of planets that work very well are attached, the transition between them might perform so poorly that it is impossible to generate a valuable solution.

To compensate for this issue, the approach might be developed further. The sequences should not just be attached. The transition between two sequences needs to be promising to heighten the quality of new permutations. This is done by searching for sequences in which the first planet is equal to the last planet of the current sequence. In the case of no proper sequence, there is just the smallest possible planet appended. This approach's risk of optimizing poor permutations should be minimal. The following pseudocode shows how new permutations out of sequences are built.

Algorithm 2 Bulding new permutations

```
while length of new permutations <7 do
    for all good sequences of two do
        initialize an empty list for all matches
        initialize search planet as the last value of the sequence
        for all good sequences of two do
            if Fist value of other sequence != search planet then
                Transition unsafe → continue
                if Second value of new potential match is not visited then
                    append sequence to matches list
                    continue
        if matches found then
            append every match to the old sequence
        else
            for all possible planets do
                if smallest possible is not visited then
                    append the smallest possible
                    break
return new permutations
```

This approach's issue is that it is not entirely controllable how many permutations will be built out of the first optimized permutations because these first ones are

selected randomly, and there could be more or less good ones. This can be somehow controlled by the number of optimized permutations in the first place.

Only selecting a fixed number from the new set would be inconsistent since this approach aims to reduce the randomness of selecting permutations to be optimized.

What is controllable are the thresholds that determine if a sequence is classified as good or bad. Setting the thresholds low will allow more permutations to be built, and setting the thresholds high will allow more sequences of two to be classified as optimal to build new permutations.

4.3 Continuous multi-optimization problem

Assuming that the set of valuable permutations is located, a multi-objective optimization problem with 27 dimensional remains, consisting of continuous variables.

From this point on, either the problem can be simplified into one objective, or it remains with both objectives. While the decomposition to reduce the dimensionality appears to be helpful, the simplification to one objective can lead to poor results because if the objective functions were scalarized into a single objective, there would need to be weights set, but because of the complexity of the objective functions and the lack of expertise, this can not be done correctly. Instead of reducing the number of objectives, these remain in their original form. An effective way of optimizing both objectives needs to be applied.

There are several methods to do this. This thesis focuses on a search through genetic Algorithms because the ability of genetic algorithms to simultaneously search different regions of a solution space makes it possible to find solutions for complex problems with non-convex, discontinuous, and multi-modal solutions spaces (Konak et al., 2006). Genetic algorithms yield a set of solutions rather than one single solution, as in traditional simplification methods, which offer a pareto front.

Many libraries offer genetic algorithms that solve multi-objective optimization problems. This thesis focuses on the frameworks of pymoo because it not only supports optimizing multiple conflicting objectives at one time but also offers tools for complete multi-objective optimization tasks (Blank and Deb, 2020). It offers a diverse set of genetic algorithms optimizing the given permutations.

Since the approach presented to locate valuable permutations can take much computational time, it might be helpful to look at the most promising algorithms of pymoo for this task instead of testing it for all algorithms. The scope of this thesis is too small to analyze the algorithms in-depth. Nevertheless, they are going to be presented briefly.

NSGA2
: The non-dominated sorting genetic algorithm works based on crowding distance.

NSGA3
: An improvement of NSGA-II for multi-objective optimization problems with more than two objectives and works with reference directions.

U-NSGA3
: A generalization of NSGA-III to be more efficient for single and bi-objective optimization problems because of applying tournament pressure.

R-NSGA2
: An extension of NSGA-II where the user can provide reference points and the surviving solutions are selected based on a rank instead of a distance measurement.

AGE MOEA
: Similar to NSGA-II but uses a modified crowding distance.

(Blank and Deb, 2020)

Based on the description of the algorithms, NSGA3 might perform worse that the other algorithms due to its focus on problems with many objectives. The performance of U-NSGA3 might be improved compared to the others because it sets the focus on single and bi-objective problems but builds up on the NSGA3 which is an improved version of NSGA2, so U-NSGA3 might perform best. To analyze which algorithm performs best based on distance measurements would require an accurate analysis of these measurements and how they work, which would not be in the scope of this thesis and is therefore excluded.

The belonging settings determine the complexity of the described algorithms and, therefore, the chance of achieving a valuable score. Because of the saved time when optimizing only promising permutations, the complexity of those algorithms can be increased.
The algorithm for optimizing the given set of permutations looks like this:

Algorithm 3 Optimizing permutations

for All given permutations **do**
    Optimize and save pareto front of this permutation
Remove all dominated solutions of the saved solutions

Calculate score of pareto frontier

5 Computational results and experiments

This chapter presents experimental results on which of the presented algorithms works best to test the elaborated approach for the challenge with this algorithm.

A preliminary remark is that the experiments' results are influenced by the complexity of the algorithms that are used. These algorithms have different features to control this complexity. The main ones are:

- offspring = the number of offspring through a generation
- termination: stop if the mean of indicator of last x generations is below y with a maximum number of generations of z
- pop_size= The population size that is used for the algorithm

(Blank and Deb, 2020)

The settings are going to be stated in every experiment.

5.1 Experiments

5.1.1 Does pymoos algorithm U-NSGA3 outperform other algorithms on optimizing the continuous variables?

Applying the elaborated approach to all of pymoos algorithms and comparing them might be interesting, but it is not the focus of this thesis. Since the focus is on achieving a valuable score, only the best-performing algorithm will be applied to the approach.

The algorithms' performance will be compared by optimizing the same 100 not classified permutations that every presented algorithm will optimize. This procedure is repeated five times to classify which algorithm performs best on this problem.

Since the tested algorithms all work differently and are determined by different settings, their performance is hard to compare. Therefore settings that occurred in several algorithms are set to equal values, and the main influences for complexity are in all algorithms: offspring = 250, termination: stop if the mean of indicator of last eight generations is below 0.002 with a maximum number of 200 generations and pop_size= 50 which results in a low to medium complex algorithm.

	NSGA2	NSGA3	U-NSGA3	AGE MOEA	R-NSGA2
Run1	0.0	0.0	0.0	0.0	-1 235 462.3406
Run2	0.0	0.0	0.0	0.0	-433 599.7465
Run3	0.0	0.0	0.0	0.0	0.0
Run4	0.0	0.0	0.0	0.0	-47 156.6268
Run5	0.0	0.0	0.0	0.0	0.0
Mean	0.0	0.0	0.0	0.0	-343 243.743

Table 5.1: Comparison of performance for a number of pymoos algorithms

From this experiment can be concluded that U-NSG3 does, against prior assumptions, not outperform the other algorithms. R-NSGA2 is the only algorithm that achieves a score. What can be taken away from this experiment is that testing the whole elaborated approach with all algorithms seems unreasonable because of not achieving any score from all runs, which is why the whole approach will be tested with R-NSGA2. Giving reasons for the outperforming of R-NSGA2 can not be done in its entirety without analyzing the distance measurements, which would go beyond the scope of the thesis. Even if the permutations are selected randomly, and the outcome could be different if the experiment is repeated, the other algorithms achieving no score lead to the conclusion that they cannot solve this problem. Nevertheless, adjusting the other algorithms or increasing the complexity might enable the algorithms to achieve a score.

5.1.2 How does the elaborated approach for finding promising permutations performs compared to optimizing random permutations?

To test if and how the developed approach of finding promising permutations performs, it will be tested against optimizing the same number of randomly found permutations. It needs to be considered that this approach is only tested with R-NSGA2 because it performed best in the previous test. Also, it is essential to mention that there are three main influences on the results. The first is the first selected permutations. The second is the algorithm's complexity, and the third is the level of strictness when classifying the sequences.

Testing with different configurations of these influences would take too much time. This is why this experiment focuses on selecting 100 permutations randomly first and optimizing them with R-NSGA2 with the following settings: offspring=100, pop_size=70, termination : stop if the mean of indicator of the last eight generations is below 0.002 with a maximum number of 200 generations. The score of the sequences of two increases by one if the belonging permutations score is greater than 1000, increases by three if the belonging permutations score is greater than 20 000, increases by five if the belonging permutations score is greater than 30 000,

and increases by ten if the belonging permutations score is greater than 300 000, it decreases by four if the score is 0.0. To calculate the score, the reference point of the hypervolume indicator has changed from [2 500, 4 000] to [2 600, 4 100] to find more scoring permutations with the low complex algorithm. The sequences with scores greater than 800 are then selected to form new permutations. These permutations are optimized with the settings: offspring=300, pop_size=200, termination: stop if the mean of indicator of last eight generations is below 0.002 with a maximum number of 200 generations. The exact number of newly built permutations are then optimized with the same settings but with the difference that these are selected randomly. The scores of this experiment are sown in table 5.2.

	Number of generated permutations	Score of generated permutations	Score of same number of randomly selected permutations	Improvement of score, when permutations are generated
Run1	34	-784 370.9850	0.0	-784 370.9850
Run2	13	-195 065.9599	0.0	-195 065.9599
Run3	26	0.0	-233 471.9950	233 471.9950
Run4	21	0.0	0.0	0.0
Run5	41	-558 605.4606	-557 041.5671	- 1 563.8935
Run6	48	-392 214.4009	- 20 719.9554	-371 494.4455
Run7	22	- 92 947.1512	-402 067.5178	309 120.3666
Run8	44	-343 636.73949	- 91 253.7700	-252 382.9694
Run9	62	-492 950.3222	- 86 370.7174	-406 579.6048
Run10	26	-471 813.0312	-815 616.1894	343 803.1582
Mean	33.7	-333 160.4050	-220 654.17121	-112 506.2338

Table 5.2: Comparison of scores from generated permutations and randomly selected permutations

Still, the results of the scores from the completely random selected permutations can differ a lot which is why there are ten runs so the mean of those scores can be compared fairly. What can be concluded from this experiment is that the elaborated approach works slightly better than optimizing the same number of randomly selected permutations with a mean improvement of -112 506.2338.

5.1.3 Does the final approach achieve higher scores if the number of randomly selected permutations is increased?

Based on the results of the previous experiment, it may be asked if the time of searching for promising permutations would be better invested in optimizing more randomly selected permutations.

To check this, the time needed to evaluate the sequences and build new permutations may be measured, and an algorithm needs to be implemented that uses this time to optimize other randomly selected permutations. Not only that, but implementing this would require much time. Since all implementations are executed with consoles in pycharm, the required time to build new permutations depends on how many consoles are currently running. Nevertheless, optimizing an increased number of permutations should be tested in a simplified version to tell if building promising permutations is worth the effort and time. For this simplified experiment, the runs of the previous experiment are used, but the number of randomly tested permutations is increased by 50% and optimized with the same settings of the previous experiment. The results will be compared to the score of the original number of permutations and the score of the original number of generated permutations.

	Increased number of permutations	Score of increased number of randomly selected permutations
Run1	51	-247 585.8107
Run2	20	0.0
Run3	39	-996 612.2483
Run4	32	-27 455.0533
Run5	52	-270 222.9295
Run6	62	-839 040.7961
Run7	33	0.0
Run8	66	-878 891.4864
Run9	91	-387 831.6426
Run10	39	-400 182.6533
Mean	47.2	-404 782.2620

Table 5.3: Comparison of performance for some of pymoos algorithms

The result of this experiment leads to the conclusion that testing an increased number of randomly selected permutations leads to a higher average score than optimizing constructed permutations. Even if the results of this experiment are that the score improves by -71 621.857, it has to be taken into consideration that randomness significantly influences these scores of randomly selected permutations.

5.2 Which score is achieved when applying the elaborated approach several times?

Even if the previous experiment gives evidence that testing an increased number of permutations leads to a good score, the performance of the elaborated approach is tested. Since the chance of only optimizing non-valuable permutations remains if the approach is carried out only one time, the approach will be applied repeatedly, and the solutions building the pareto frontiers of each run are combined and again checked for dominance. The aim of this is to find solutions that dominate solutions of other runs and, through this improve the score of the final pareto front. The best approximation of the pareto front was achieved by building new permutations and optimizing them with high complex R-NSGA2 algorithms several times over the last weeks. These algorithms' settings varied to find most of the possible solutions.

The algorithms used to achieve the final pareto front had settings in the range of:
offspring = 400 -1 500
termination : stop if the mean of indicator of last 8 - 10 generations is below 0.002 with a maximum number of generations of 200 - 400
pop_size= 200 - 400

Figure 5.1 presents the final pareto front, containing the solutions that resulted in a score and hence, sufficiently satisfy both objectives. With a score of -3 440 253.65, it can be concluded that with the elaborated approach of building new promising permutations and optimizing their continuous variables with the genetic algorithm R-NSGA2, the SpOC Trappist Tour system exploration challenge is completed.

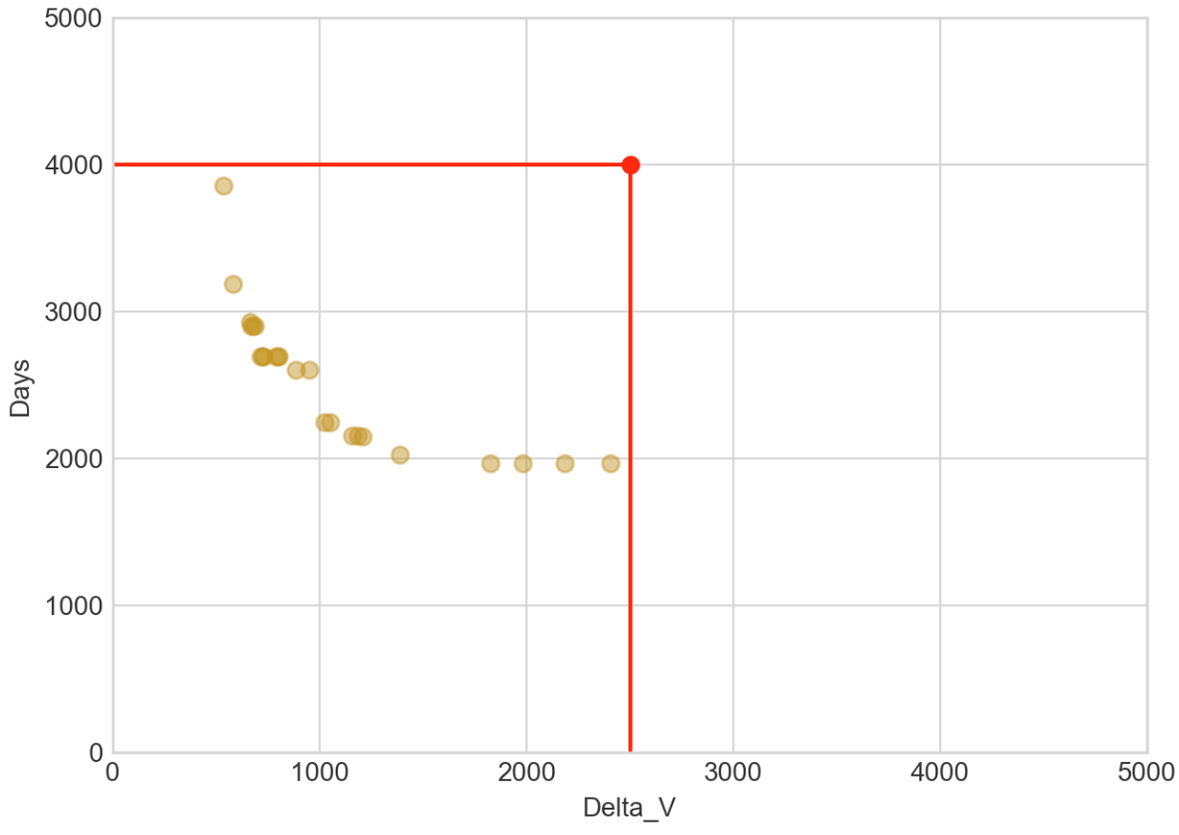


Figure 5.1: Visualisation of the final pareto front

6 Conclusion and outlook

The achieved score indicates that the challenge is solved successfully. Nevertheless, the score achieved is not the maximum score that can be achieved. To be able to attain a higher score, one possibility to develop the elaborated approach further could be to find a way to start optimizing the best solutions of a previous run and not optimizing them from the beginning, which would allow optimizing the values of the continuous variables further. Another opportunity would be to find a way of selecting the permutations at the beginning of the elaborated approach that is not based on randomness.

Even if the challenge was solved without prior knowledge of astrophysics, what can be learned is what the trajectory of a good solution against a poor solution looks like.

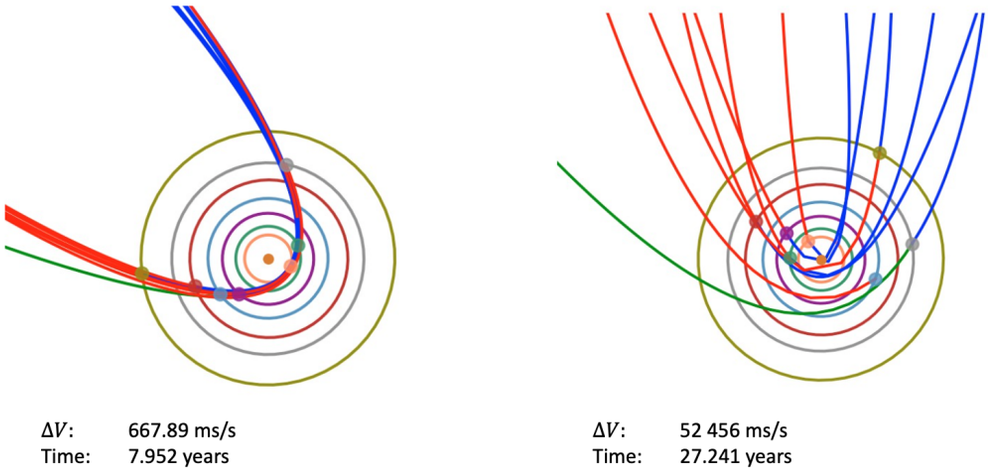


Figure 6.1: Comparison of trajectories from a good solution vs. a poor solution

As Figure 6.1 shows, the trajectory consists of quite similar orbits. This is possible because the planets stand in a certain way. To improve the score further, expert astrophysics knowledge would also help find those solutions.

Although further research is always reasonable, the elaborated approach allows achieving a valuable score, which proofs the ability of solving complex problems without any expert knowledge.

Bibliography

Jordan Baumgardner, Karen Acker, Oyinade Adefuye, Samuel Thomas Crowley, Will DeLoache, James O Dickson, Lane Heard, Andrew T Martens, Nickolaus Morton, Michelle Ritter, et al. Solving a hamiltonian path problem with a bacterial computer. *Journal of biological engineering*, 3(1):1–11, 2009.

Louis-Philippe Bigras, Michel Gamache, and Gilles Savard. The time-dependent traveling salesman problem and single machine scheduling problems with sequence dependent setup times. *Discrete Optimization*, 5(4):685–699, 2008.

J. Blank and K. Deb. pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509, 2020.

Maurizio Boccia, Adriano Masone, Antonio Sforza, and Claudio Sterle. An exact approach for a variant of the fs-tsp. *Transportation Research Procedia*, 52:51–58, 2021.

Kalyanmoy Deb. Multi-objective optimisation using evolutionary algorithms: an introduction. In *Multi-objective evolutionary optimisation for product design and manufacturing*, pages 3–34. Springer, 2011a.

Kalyanmoy Deb. Multi-objective optimisation using evolutionary algorithms: an introduction. In *Multi-objective evolutionary optimisation for product design and manufacturing*, pages 3–34. Springer, 2011b.

Abdullah Konak, David W Coit, and Alice E Smith. Multi-objective optimization using genetic algorithms: A tutorial. *Reliability engineering & system safety*, 91 (9):992–1007, 2006.

Manuel López-Ibáñez, Francisco Chicano, and Rodrigo Gil-Merino. The asteroid routing problem: A benchmark for expensive black-box permutation optimization. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*, pages 124–140. Springer, 2022.

Xiaoliang Ma, Fang Liu, Yutao Qi, Xiaodong Wang, Lingling Li, Licheng Jiao, Minglei Yin, and Maoguo Gong. A multiobjective evolutionary algorithm based on decision variable analyses for multiobjective optimization problems with large-scale variables. *IEEE Transactions on Evolutionary Computation*, 20(2):275–298, 2015.

Krzysztof Nowak, Marcus Märtens, and Dario Izzo. Empirical performance of the approximation of the least hypervolume contributor. In *International Conference on Parallel Problem Solving From Nature*, pages 662–671. Springer, 2014.

Panos M Pardalos, Antanas Žilinskas, Julius Žilinskas, et al. *Non-convex multi-objective optimization*. Springer, 2017.

Tea Tušar and Bogdan Filipič. Visualization of pareto front approximations in evolutionary multiobjective optimization: A critical review and the prosection method. *IEEE Transactions on Evolutionary Computation*, 19(2):225–245, 2014.

Mengyuan Wu, Ke Li, Sam Kwong, Qingfu Zhang, and Jun Zhang. Learning to decompose: A paradigm for decomposition-based multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 23(3):376–390, 2018.

Nan Zhang, Zhong Zhang, and Hexi Baoyin. Timeline club: An optimization algorithm for solving multiple debris removal missions of the time-dependent traveling salesman problem model. *Astrodynamics*, 6(2):219–234, 2022.

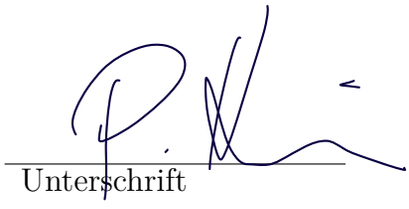
Versicherung

Name: Heine

Vorname: Paulina

Ich versichere, dass ich diese Bachelorarbeit selbständig verfasst und keine anderen, als die angegebenen, Quellen benutzt habe. Die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen habe ich als solche kenntlich gemacht. Diese Versicherung gilt auch für alle gelieferten Datensätze, Zeichnungen, Skizzen oder grafischen Darstellungen. Des Weiteren versichere ich, dass ich das Merkblatt zum „Umgang mit Plagiaten“[1] gelesen habe.

Bielefeld, den July 8, 2022



Unterschrift

[1] www.wiwi.uni-bielefeld.de/organisation/pamt/~/organisation/pamt/uploads/PlagiatInfo-BlattStudenten.pdf