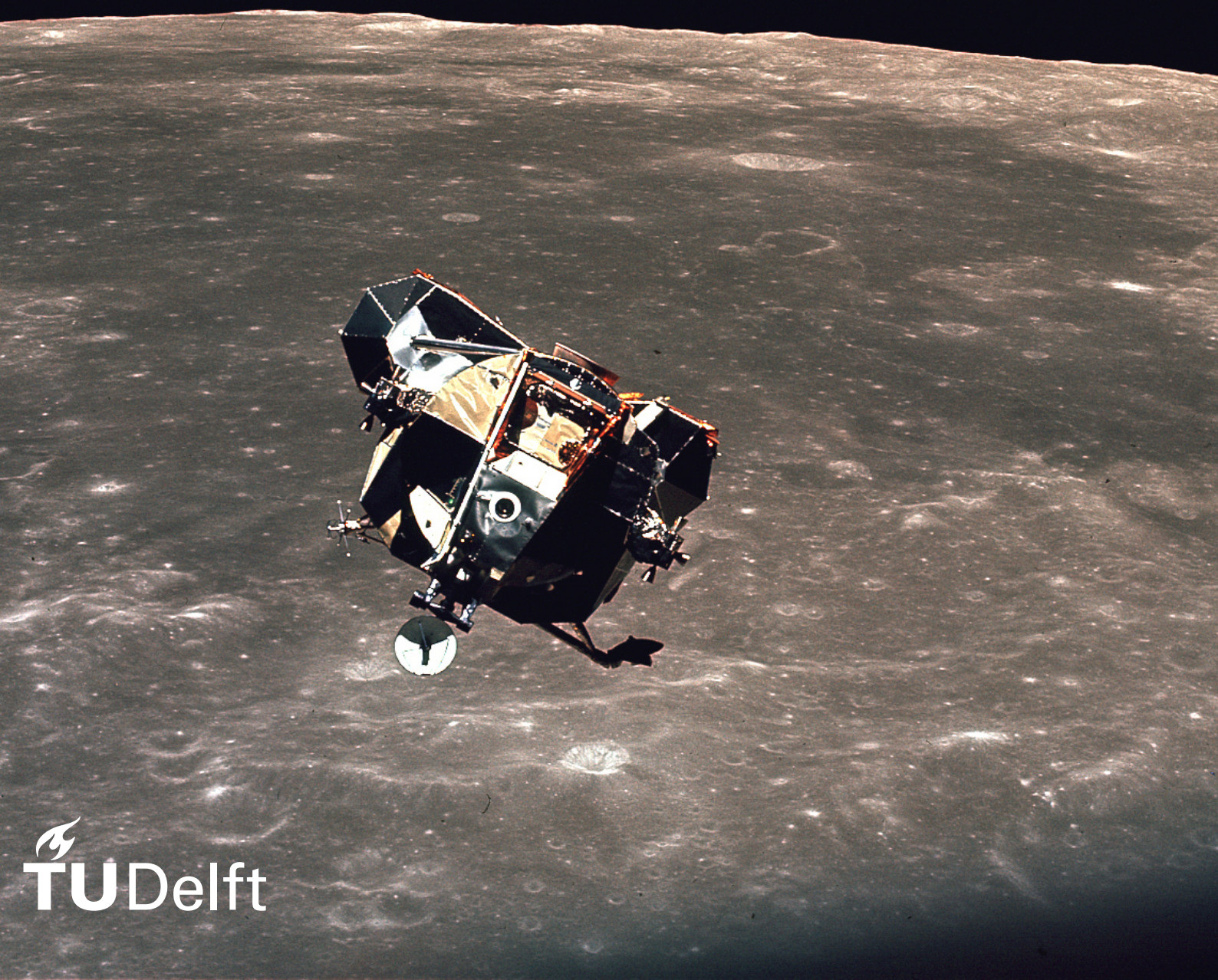
Propagation and Optimisation

Assignment IV

Daniel Calliess, Mitchell van Doorn, Lorenz Veithen, Jesse Voskuilen



TU Delft

DELFT UNIVERSITY OF TECHNOLOGY

FACULTY OF AEROSPACE ENGINEERING

AE4866-1

PROPAGATION AND OPTIMISATION IN ASTRODYNAMICS

Lunar Ascent Trajectory Propagation and Optimisation

Global Optimisation of Trajectory

Lecturers: Dr. D. Dirkx

Github repository link: https://github.com/LorenzVeithen/Propagation_Optimisation_Group_2D_2023

July 2, 2023

Name	Student Number	Hours Spent
Daniel Calliess	4869710	≈ 50
Mitchell van Doorn	4954297	≈ 35
Lorenz Veithen	5075211	≈ 45
Jesse Voskuilen	4795768	≈ 42

Cooperating Groups: 2A

TU Delft Delft University of Technology

[1]Cover Image URL: https://nssdc.gsfc.nasa.gov/imgcat/html/object_page/a11_h_44_6642.html

1 Problem Statement

In this section, first the problem statement which was decided upon in the previous assignment is reiterated. For the underlying reasons of this chosen problem statement the reader is referred to [1].

1.1 Problem Definition

In Table 1, the numerical and physical model definition is shown.

Table 1: Numerical and physical model definition as decided upon in [1]

Integrator	Propagator	Accelerations	Environments
RKF4(5)	Cowell	Thrust, Earth PM, Moon SH (100, 100)	constant ephemeris, Spice Moon rotation, Oblate spheroid

Furthermore the selected decision variables and constraint and objective functions are reiterated in Table 2. Note that a few constraints have been rephrased to 'problem boundaries'. These constraints did not depend on the output of simulations and could therefore not be violated. Rather they are boundary conditions. Some other constraints have been rephrased to 'termination conditions' as they cause a simulations to terminate once their maximum or minimum value is reached. Also the maximum mechanical load constraint has been changed from $< 6g$ to $< 3g$ since this is considered to be a more realistic acceleration that a human being could sustain for multiple minutes.

Table 2: Selected Problem boundaries, termination conditions, decision variables, constraints and objectives from [1].

Problem Boundaries	**Definition**
Initial state	$r_0 = r_M + h_0, \quad \lambda_0 = 23.4333°, \quad \phi_0 = 0.6875°$
Initial state	$\gamma_0 = 89°, \quad v_0 = 10m/s, \quad \psi_0 = 90°$
Specific impulse	$I_{sp} = 452s$
Termination conditions	**Definition**
Maximum altitude	$h_{max} < 100km$
Minimum mass	$M(t_e) > M_{dry} = 2250kg$
Decision variables	**Definition**
Thrust magnitude	$7.61kN \leq T \leq 20kN$
Node spacing	$10s \leq \Delta t \leq 100s$
Thrust angle at node 1	$0.0rad \leq \theta_1 \leq 0.8rad$
Thrust angle at node 2	$0.0rad \leq \theta_2 \leq 1.0rad$
Thrust angle at node 3	$0.0rad \leq \theta_3 \leq 1.1rad$
Thrust angle at node 4	$0.0rad \leq \theta_4 \leq 1.3rad$
Thrust angle at node 5	$0.0rad \leq \theta_5 \leq 1.6rad$
Constraints	**Definition**
Maximum time (redundant)	$t_{max} \leq 3,600s$
Minimum initial thrust load (redundant)	$\psi_0 = \frac{T}{M_0 g_m} > 1$
Maximum acceleration / mechanical load (redundant)	$a_{max} < 3g$
Minimum altitude	$h_{min} > 0km$
Thrust angle rate	$\dot{\theta}_{max} < 20deg/s$
Check if target orbit is reachable with remaining fuel mass	$\Delta V_{buffer} = I_{sp} g_0 ln(\frac{M(t_0)}{M(t_e)}) - \Delta V_{tar} \geq 0m/s$
Objectives	**Definition**
Minimize propellant mass	$min(M_f), \qquad M_f = M(t_0) - M(t_e)$
Minimize corrective	$min(\Delta V_{tar}), \qquad \Delta V_{tar} = \Delta V_1 + \Delta V_2$

1.2 Constraints Implementation

The design space exploration has shown, with the latest decision variable ranges for the thrust magnitude, that the maximum time, minimum initial thrust load and maximum acceleration cannot be violated anymore (section 3.3 of [1]). Therefore, these constraints have become redundant and could be left out. They are however still shown in Table 16 for completeness. This is only remaining constraints that can be violated are minimum altitude, thrust angle rate and ΔV_{buffer}. Of these, the Delta V buffer is the constraint that was by far violated the most.

The incorporation of constraints can be done in multiple ways. The easiest way would be to 'kill the individual', once a solution has shown to violate a constraint. Here a 'solution' means a specific combination of decision variables. For

solutions violating the minimum altitude constraint, this will be the strategy, since those solutions are far from an optimum because they crash into the Moon's surface. This will also be the strategy for solutions violating the other constraints by a significant amount. For this a maximum allowable constraint value is established in Table 3. For those solutions it is considered not to be worth trying to change them to steer them towards an optimum through the use of an optimisation algorithm. However, for solutions slightly violating a constraint, it could be that such a solution is still close to an optimal solution, meaning that if some decision variable(s) change slightly the solution might end up just on the right side of the constraint near the optimum. Therefore, it would be a waste to discard that solution immediately and so the strategy to incorporate those constraints will be to give the objective value of those solutions a weighted, normalised 'penalty' per violated constraint according to Equation 1.

$$\text{Fitness } = f_{\text{obj}} + \sum_{i=1}^{m_{\text{constraints}}} w_i \hat{g}_i \tag{1}$$

where w_i and g_i are the weight and normalised constraint value, respectively, of constraint i. The constraints will be normalised with respect to the maximum allowable constraint value. If no constraints are violated all $\hat{g}_i s$ will be zero and a fitness value equal to the objective value will be obtained. In the multi-objective optimisation algorithms, this penalty will be given to each of the two objectives. Note that the outcome of the optimisation may be sensitive to the chosen weights and normalisation of the constraints. Therefore, for each optimisation algorithm, additional tuning is required. In this section, some initial values for the weights and maximum allowable constraint values are given. These can be found in Table 3.

Table 3: Assigned weights and normalisation's to constraints

Constraint	**Constraint value**	**Weight**	**Maximum allowable constraint value**	**Normalisation**
Thrust angle rate	v_i	10	30 [deg/s]	$\frac{1}{30[deg/s]} \cdot 100 \cdot v_i$
ΔV_{buffer}	v_i	20	-1500 [m/s]	$\frac{1}{-1500[m/s]} \cdot 100 \cdot v_i$

As an example: let's say the thrust angle rate constraint is violated by 10 deg/s and the ΔV_{buffer} constraint by 500m/s (so the constraint value is then -500), both $\approx 30\%$ of the maximum allowable constraint values, then the assigned penalty to that solution would be $\sum_{i=1}^{m_{constraints}} w_i \hat{g}_i = 10 \cdot 10 \cdot \frac{1}{30} \cdot 100 + 20 \cdot 500 \cdot \frac{1}{1500} \cdot 100 = 1000$. This 1000 is then added to the objective value to obtain the fitness value of a solution (to be minimised). The design space exploration has shown that objective values are usually in the 0-2000 range for both consumed fuel mass [kg] and corrective Delta V [m/s], showing that the weights have a reasonable magnitude.

NB: it was also considered to let the thrust angle rate penalty depend on the time it is violated according to: $\int_0^{t_f} v_i \mathrm{d}t$. This would make sense since if the thrust angle rate is violated by e.g. 3 deg/s for a whole minute, the penalty should be higher than when the thrust angle rate is violated by that same value for only a couple seconds. However, the design space exploration has shown that the thrust angle rate constraint is only violated very few times, and when it is violated, also for a very short amount of time. Therefore, it has been decided that it was not necessary to implement it (to save coding time) as it would not influence the assigned penalty's much and therefore also not the optimisation results.

The ΔV_{buffer} is considered to be the most important constraint as it determines if the trajectory could enter the desired orbit. While violation of constraints on the maximum angular rate would mean that the trajectory is infeasible, slight corrections during the trajectory could render the trajectory feasible. However, a misfit of the orbit insertion by a small delta-V could result in a later crash, if no propellant is left for corrections. For this reason the ΔV_{buffer} constraint is given the largest weight. The maximum allowable constraint values are based on the results of the full Monte Carlo analysis of in the Design Space Exploration (section 2.3 of [1]) and aim to give a realistic estimate of when a solution would be too far from an optimum so that it is best to discard it during the optimisation process (for efficiency purposes).

1.3 Termination and Convergence Criteria

It is important to establish a suitable termination and convergence criteria in order to make sure an optimisation algorithm stops after it has reached an optimum (or is very close to it). Then once it has stopped it needs to be checked whether the apparent optimum is the actual optimum. In other words, it needs to be checked whether the algorithm has truly converged and no better solution is possible. For the termination criteria it has been decided that if the mean fitness of the population does not improve by 0.01% after 5 iterations, the algorithm stops. Waiting for five iterations is a result of the random nature of the global optimisers used, which can have very sudden improvements, even if the values do not improve after a number of generations. After this, local refinements will be performed to check whether all individuals in the Pareto front are indeed all optima. A (limited) Monte Carlo analysis around a chosen point in the Pareto front will be performed, with changes of 10% in the decision variables. This Pareto point can be chosen according to the preferred combination of the two objective function values. Here the convergence criteria comes into play. The convergence criteria will be that if no better (more optimal) solution is found in this local refinement, the optimisation has converged. If a better solution is

found a local refinement for every Pareto individual in the Pareto front will be done with the same Monte Carlo analysis. Then a new Pareto front can be established, and local refinement will be done again. The termination and convergence criteria are summarized in Table 4.

Table 4: Termination and convergence criteria summarized

Termination criteria	Terminate optimisation if $\lvert\frac{mean(Fitness_{(i)})-mean(Fitness_{(i-5)})}{mean(Fitness_{(i-5)})}\rvert < 1E-4$ where i is the current iteration
Convergence criteria	Converged if after local refinement MC no better solution is found w.r.t. Pareto front

2 Multi-Objective Global Optimisation

In this section the global optimisation is performed using the problem definition defined in Section 1. The scientific Python library Pygmo is used to perform the optimisation [2].

2.1 Multi-Objective Algorithm Selection

There are six multi-objective optimisation algorithms available in the Pygmo toolbox [2]. These are: Improved Harmony Search (IHS), Non-dominated Sorting Genetic Algorithm (NSGA2), Multi-objective Evolutionary Algorithm (EA) with Decomposition (MOEA/D), Multi-objective EA with Decomposition Generational (GMOEA/D), Multi-objective Hypervolume-based ACO (MHACO) and Non-dominated Sorting PSO (NSPSO). The global optimisation is run once with each of these algorithms without tuning the settings. The standard settings chosen are identical for each optimiser and are 1 generation with 48 individuals and 24 evolution steps. To evaluate convergence and robustness, the mean fitness of each optimiser for different seeds is shown in Fig. 1 to Fig. 6. 24 evolutions are performed per optimiser to evaluate the general behaviour over time and allow a fair comparison, while convergence according to the definition in Section 1 is verified in parallel.

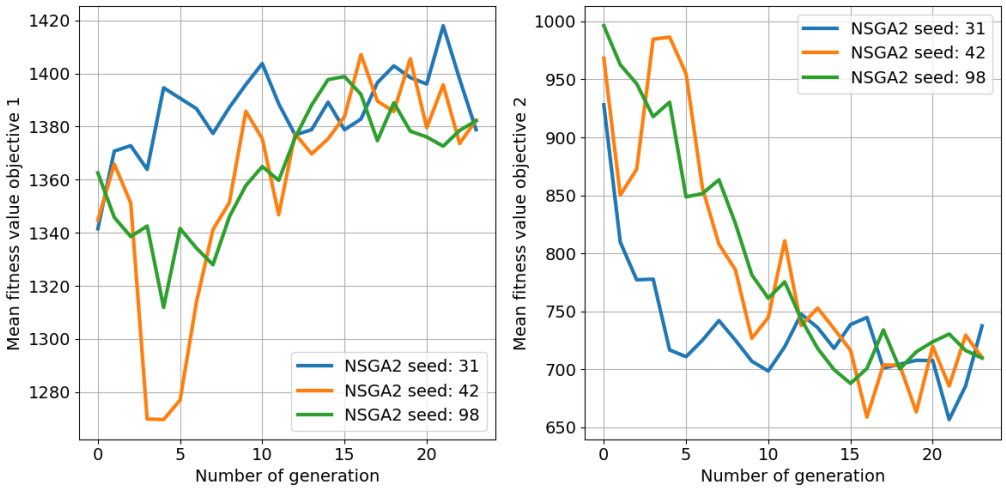


Figure 1: Evolution of mean fitness value for objective 1 (consumed fuel mass) and objective 2 (Corrective ΔV) using NSGA2 algorithm in Pygmo. Population size = 48, Number of generations = 24, seeds: 31, 42, 98.

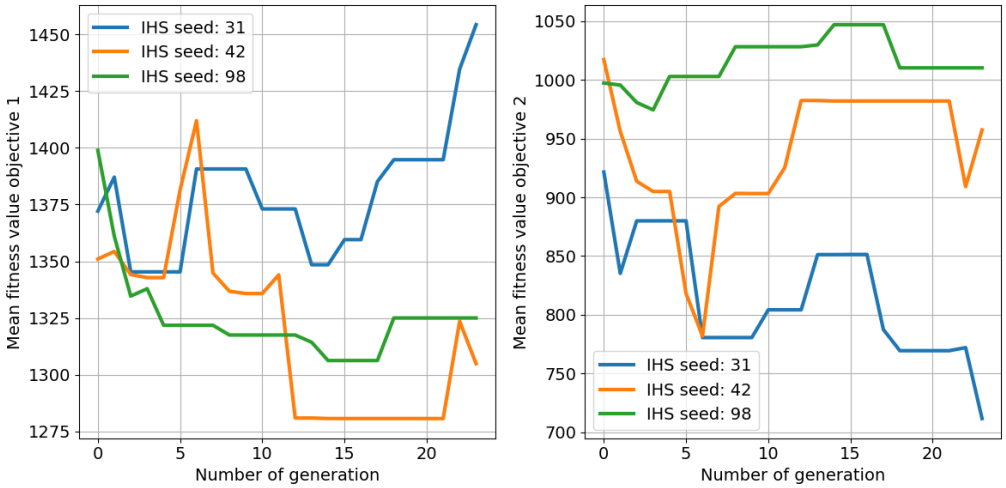


Figure 2: Evolution of mean fitness value for objective 1 (consumed fuel mass) and objective 2 (Corrective ΔV) using IHS algorithm in Pygmo. Population size = 48, Number of generations = 24, seeds: 31, 42, 98.

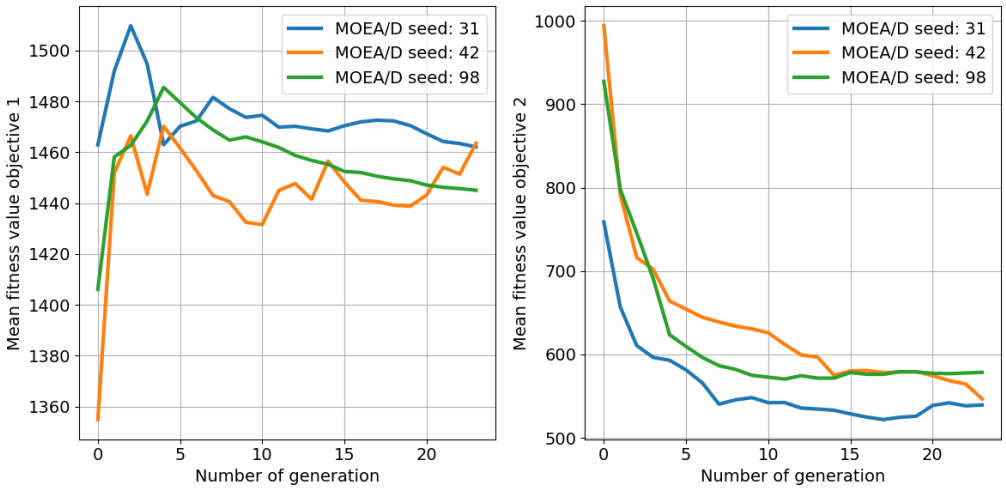


Figure 3: Evolution of mean fitness value for objective 1 (consumed fuel mass) and objective 2 (Corrective ΔV) using MOEA/D algorithm in Pygmo. Population size = 48, Number of generations = 24, seeds: 31, 42, 98.

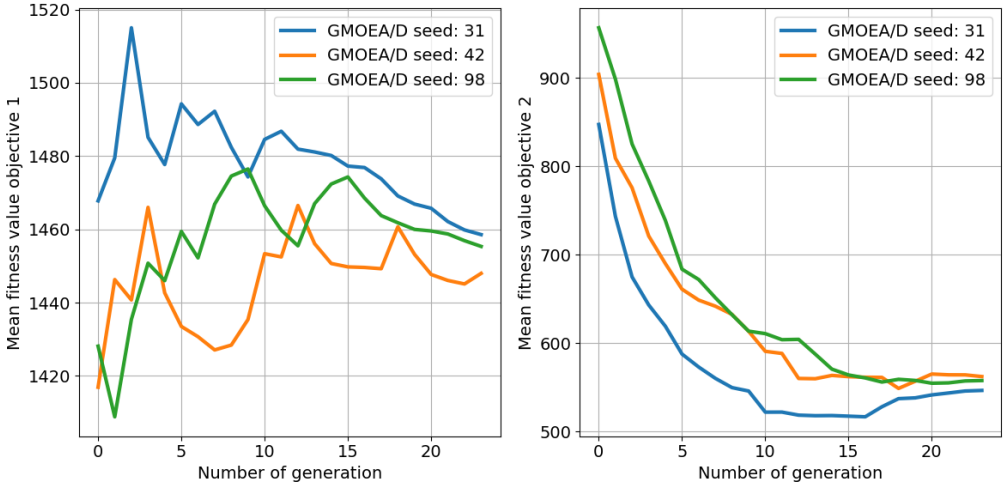


Figure 4: Evolution of mean fitness value for objective 1 (consumed fuel mass) and objective 2 (Corrective ΔV) using GMOEA/D algorithm in Pygmo. Population size = 48, Number of generations = 24, seeds: 31, 42, 98.

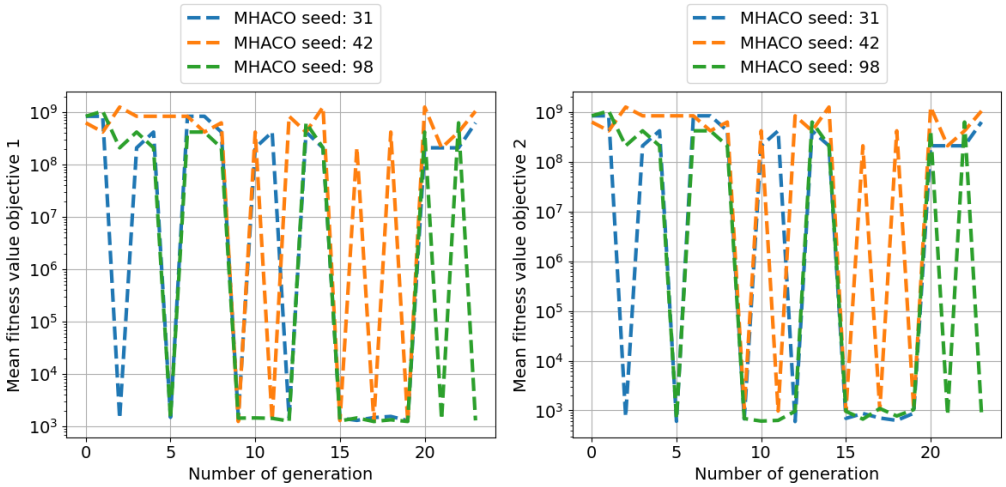


Figure 5: Evolution of mean fitness value for objective 1 (consumed fuel mass) and objective 2 (Corrective ΔV) using MHACO algorithm in Pygmo. Population size = 48, Number of generations = 24, seeds: 31, 42, 98.

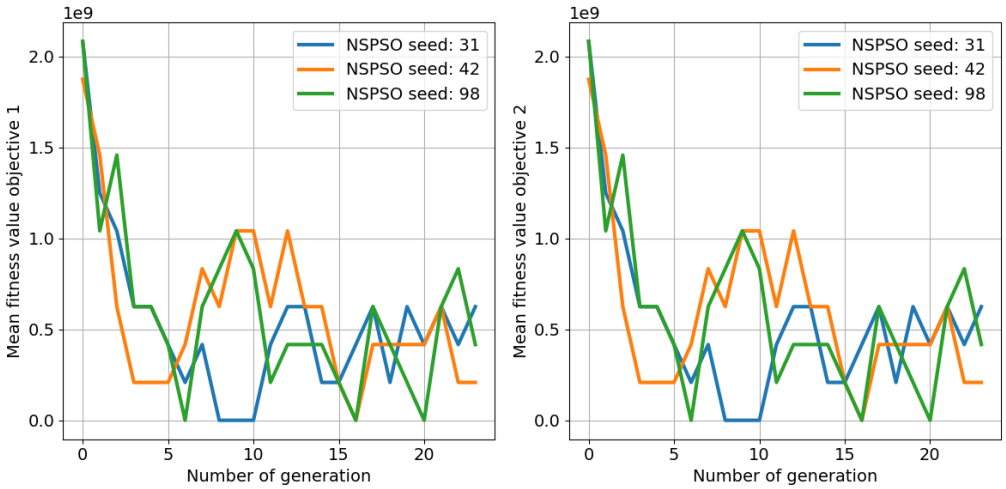


Figure 6: Evolution of mean fitness value for objective 1 (consumed fuel mass) and objective 2 (Corrective ΔV) using NSPSO algorithm in Pygmo. Population size = 48, Number of generations = 24, seeds: 31, 42, 98.

Both algorithms MOEA/D and GMOEA/D show the smoothest and most robust behaviour over various seeds and converge to low fitness values quickly. Their change in mean fitness values per subsequent generation decreases for all three random seed numbers. NSGA2 shows medium robustness behaviour reaching slightly lower values for objective 1 and higher values for objective 2, hence it is converging to different optima. The IHS algorithm is the fastest to run but does not converge smoothly for all seeds, showing considerable difference when changing seed number. Both NSPSO and MHACO produce very high fitness values and do not converge. The mean fitness fluctuates around 1×10^9 due to some population members breaking a hard constraint (eg. minimum altitude) and their penalty is increased by this value. Unlike the other algorithms and expectations they do not seem to filter out these individuals. NSPSO and MHACO are not suitable for optimisation since they are not robust for changing seed number and do not converge to low fitness values like the rest. The other four algorithms are chosen for further consideration and their Pareto front is plotted in Fig. 7 to Fig. 10.

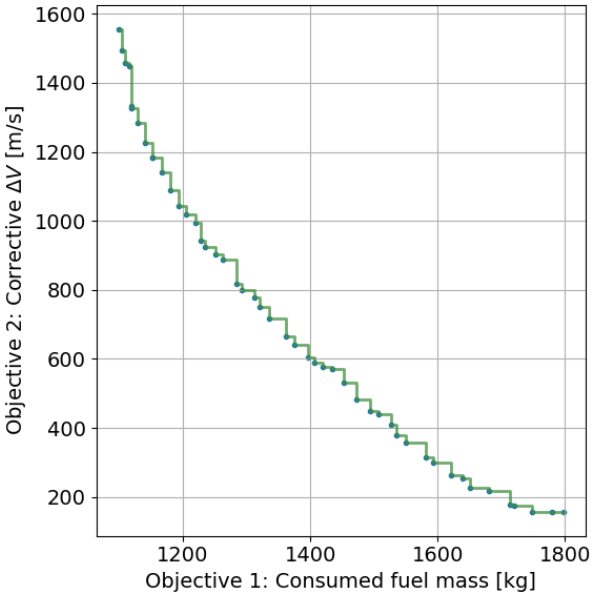


Figure 7: Pareto front of final population using NSGA2 algorithm in Pygmo. Population size = 48, Number of generations = 24, seed: 31.

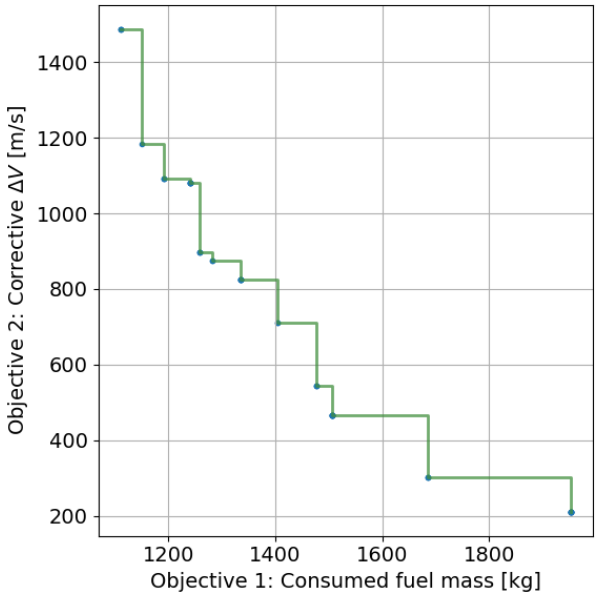


Figure 8: Pareto front of final population using IHS algorithm in Pygmo. Population size = 48, Number of generations = 24, seed: 31.

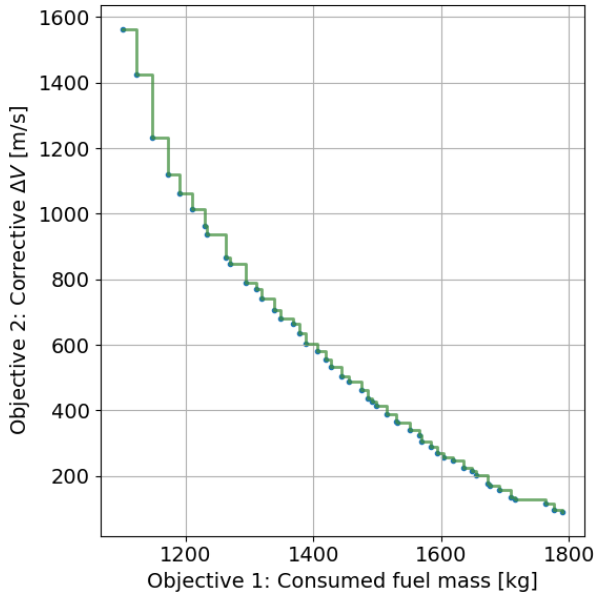


Figure 9: Pareto front of final population using MOEA/D algorithm in Pygmo. Population size = 48, Number of generations = 24, seed: 31.

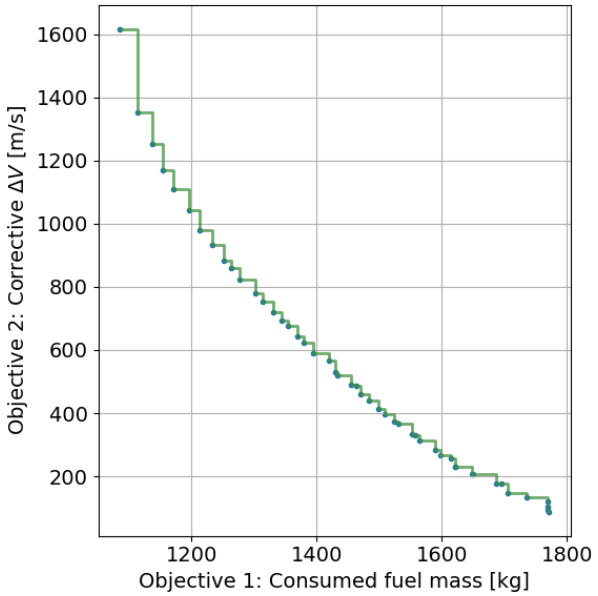


Figure 10: Pareto front of final population using GMOEA/D algorithm in Pygmo. Population size = 48, Number of generations = 24, seed: 31.

Figure 8 shows that the Pareto front found by IHS contains a very limited amount of Pareto individuals. Since this algorithm was not found to be robust it will not be considered. Both MOAE/D and GMOEA/D show very similar Pareto fronts with a high number of Pareto individuals spread out evenly over the front. It is observed that MOEA/D and GMOEA/D show a slight preference to minimise objective 2 instead of objective 1 as more Pareto individuals are located towards minimising fuel mass. The NSGA2 algorithm finds a slightly more uniform Pareto front, finding more Pareto individuals that minimise objective 1. Although Figure 7 proves that NSGA2 results in a slightly better Pareto front, its robustness could not be verified for varying seed numbers and its mean fitness fluctuates over time. Its competitors MOEA/D and GMOEA/D show smooth and reliable convergence for multiple seeds while producing a similar Pareto front. Considering the trade-off between convergence, robustness and diversity/density of Pareto front solutions, the NSGA2 algorithm is not selected due to its unreliability and non-convergence. This leaves both MOEA/D and GMOEA/D as most suitable global optimisation algorithms.

(Li and Zhang, 2007) developed MOEA/D as modification to existing evolutionary algorithms that turns a multi-objective problem into a number of smaller optimization sub-problems [3]. Each sub-problem is optimised simultaneously by using information from its neighbours to find the true Pareto front. This new method utilising problem decomposition is an alternative to the non dominated sorting operation used in NSGA2 and usually outperforms NSGA2 [4]. The authors suggest to tune the algorithm settings to obtain a more diverse and uniform Pareto front, similar to NSGA2 [4]. (Izzo et al., 2015) found that problem decomposition is preferred to non-dominating sorting in the context of the design of missions with multiple objectives [5]. For trajectory optimisation problems, the performance of recently developed genetic algorithms like MOEA/D is generally considered to be very robust over a broad range of multi-objective optimisation problems and preferred [6]. GMOEA/D is an adapted version of MOEA/D updating the population on a generational basis. This sacrifices sensitivity to population changes to allow parallel fitness evaluations [2]. As this is not required or used for this problem the choice is made to select the variant MOEA/D.

2.2 Tuning the MOEA/D Algorithm

Table 5: Parameter settings varied for tuning MOEA/D algorithm for performance. Default parameter settings are taken from Pygmo documentation [2] and verified with recommended settings in (Li and Zhang, 2009) [4].

Parameter	**Default value**	**Range of values tested in tuning**
Population size	48	24,48,96,144,192
Weight generation method	grid	grid, low discrepancy, random
Decomposition method	tchebycheff	weighted, tchebycheff, bi
Neighbourhood size	20	5,10,20,30,40
Crossover parameter (CR)	1	0.5,0.8,1.0
Differential evolution parameter (F)	0.5	0.5,0.8,1.0
Mutation distribution index (η_m)	20	10,20,50
preserving diversity mechanism [3]	True	True,False

In Subsection 2.1 the Multi-objective Evolutionary Algorithm with Decomposition (MOEA/D) was chosen. The default settings were used but the optimal settings for high performance are highly problem dependent. The most relevant settings of MOEA/D as explained by (Li and Zhang, 2009) will be varied to see if the number, diversity and location of Pareto solutions can be improved while still ensuring robustness/convergence [4]. The parameters that are varied and their default values are shown in Table 5. Each setting is varied independently to isolate and analyse its effect on the performance and a seed of 31 is used. Generation size as input affects the number of local runs performed in each population and it was found that this setting is identical to number of evolutions; 2 generations and 12 evolutions are identical to 1 generation and 24 evolutions. The preserve_diversity setting is an additional modification that showed no effect hence its sub-settings (limit,realb) were not modified, see Figure 18.

The overall trend observed is that all settings produced robust results, which was tested for varying seed numbers similar to Subsection 2.1. The convergence to $\Delta\bar{f} < 0.01\%$ was achieved within 50 evolutions for the majority of settings, with some settings causing convergence problems and not behaving as robust, see Figure 19. The non-converged parameter settings are shown in color and these are a population size of 96, random weight allocation and F=0.8. Since the rest of the settings converge and behave robustly, the MOEA/D parameter settings are tuned to improve the performance of Pareto solutions found and ideally approach the true optimum. This is further justified as more than 200 evolutions are common in literature for similar problems (less than 50 evolutions needed here across all converging settings) and there is no strong need for shorter run-times, given that all settings are already robust/convergent [6] [7] [8].

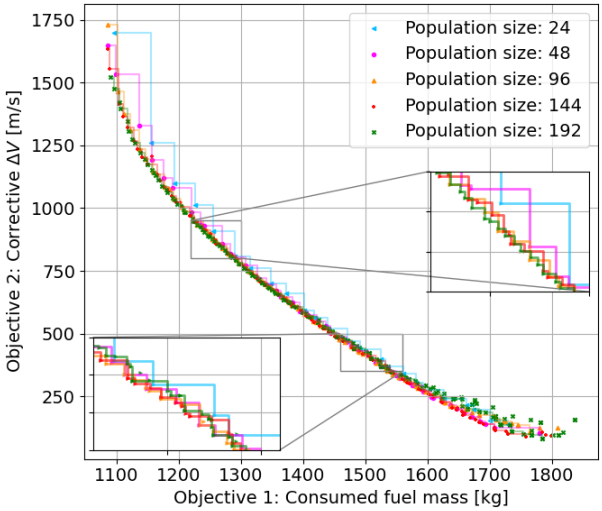


Figure 11: Pareto fronts of final populations found by MOEA/D optimiser using default settings from Table 5, seed: 31 and varying population size.

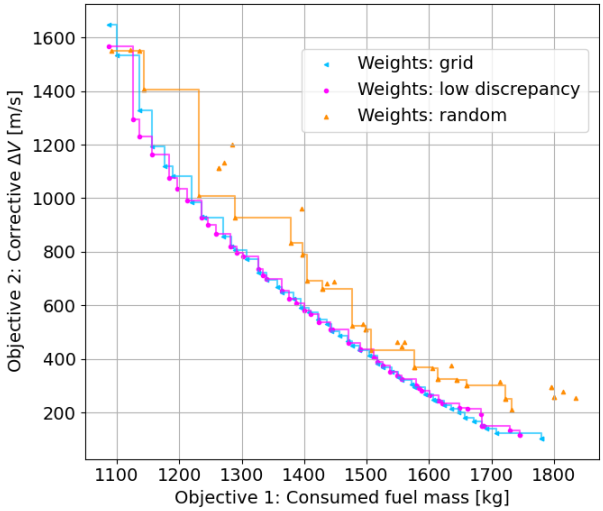


Figure 12: Pareto fronts of final populations found by MOEA/D optimiser using default settings from Table 5, seed: 31 and varying weight generation method.

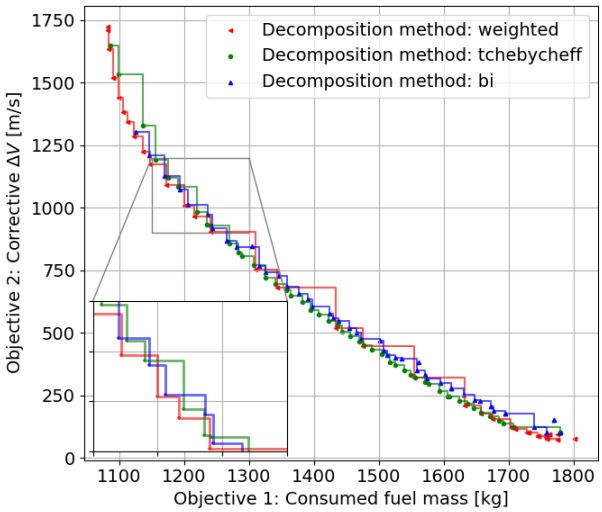


Figure 13: Pareto fronts of final populations found by MOEA/D optimiser using default settings from Table 5, seed: 31 and varying decomposition method.

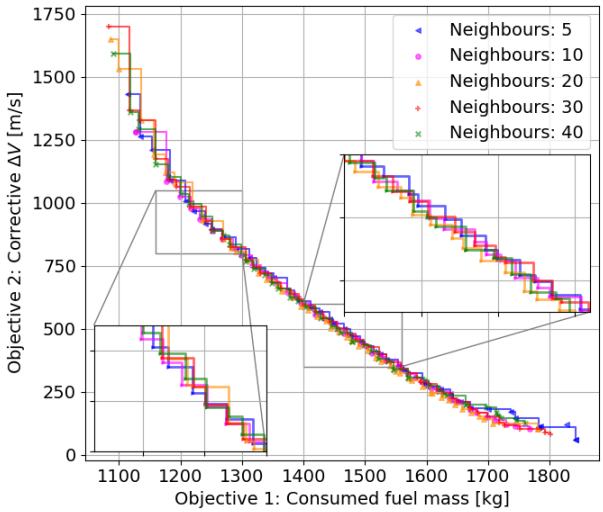


Figure 14: Pareto fronts of final populations found by MOEA/D optimiser using default settings from Table 5, seed: 31 and varying neighbour size.

The Pareto fronts of the final population found by MOEA/D varying each setting are shown in Fig. 11 to Fig. 18. It is clear

that MOEA/D is most sensitive to changes in population size, weight generation, decomposition method and neighbour size. CR, F, η_m and the diversity setting have far less influence on results. Many parameter changes were shown to have a tendency to improve the Pareto front towards one objective only. For this reason two classes of optimum settings were chosen to optimise the Pareto front towards both objective directions and a combined final Pareto front is shown in Figure 21.

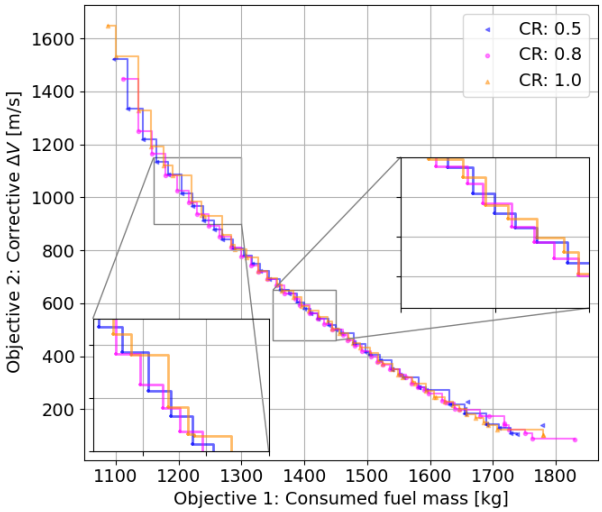


Figure 15: Pareto fronts of final populations found by MOEA/D optimiser using default settings from Table 5, seed: 31 and varying CR parameter.

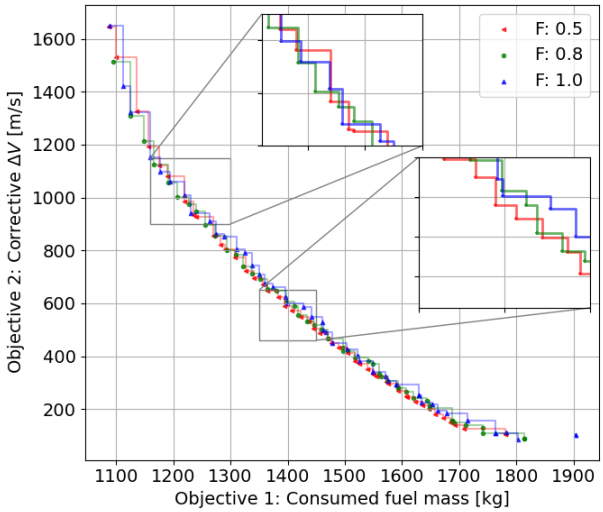


Figure 16: Pareto fronts of final populations found by MOEA/D optimiser using default settings from Table 5, seed: 31 and varying F parameter.

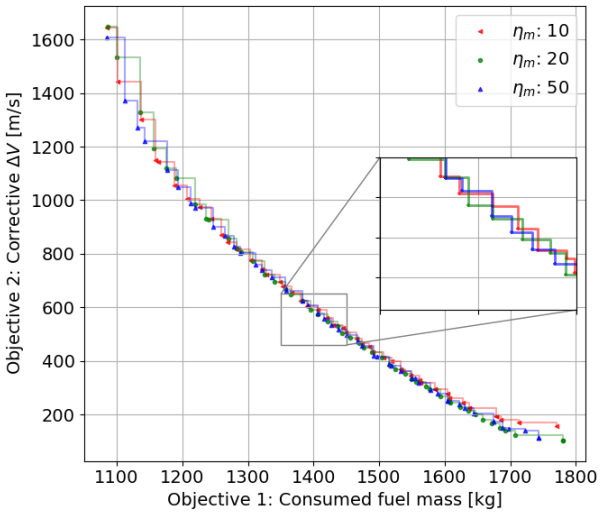


Figure 17: Pareto fronts of final populations found by MOEA/D optimiser using default settings from Table 5, seed: 31 and varying η_m parameter.

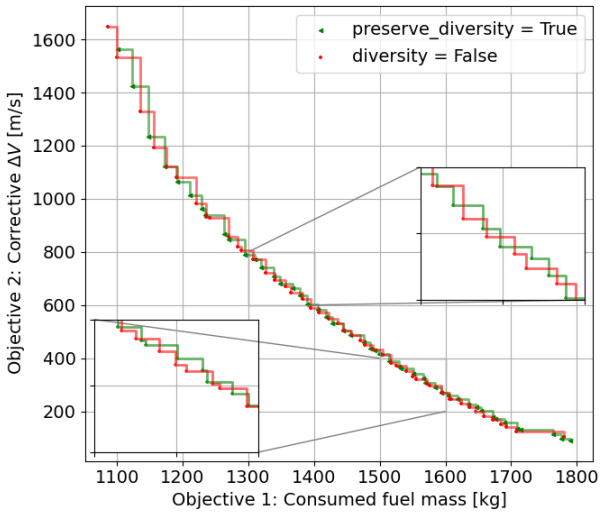


Figure 18: Pareto front of final population found by MOEA/D optimiser using default settings from Table 5, seed: 31 and varying diversity setting.

Population sizes of 24 to 192 are shown in Figure 11. Low population sizes result in a sparse Pareto front and perform worse. A population size of 144 produced better results for objective 2 and 192 individuals better results for objective 1. The default population size of 48 seems too low and finds less and inferior Pareto individuals towards objective 1. A random weight allocation deteriorates the Pareto front results to sparse and worse-performing solutions; this setting also does not converge as seen in Figure 19. The default grid weight allocation works better towards objective 2 while the low discrepancy option improves the density and location of the Pareto front primarily towards the left, see Figure 12. The decomposition method is tchebycheff by default, resulting in a Pareto front that is sparse and inferior at the boundaries but clearly dominating in the center of the front. The weighted setting has the opposite effect, producing a high diversity of well-performing Pareto individuals at the edge cases but failing to produce good results in the center meeting both objectives. The bi decomposition approach results in inferior results compared to the other two settings. The effect of neighbor size is displayed in Figure 14, where the default setting of 20 results in the most optimum Pareto front. Other neighborhood sizes cause inferior results especially at the boundaries where the number of solutions is sometimes decreased. (Li and Zhang, 2007) recommend a "neighborhood size much smaller than the population size N", which is in

agreement with the results obtained here for a default N=48 [3].

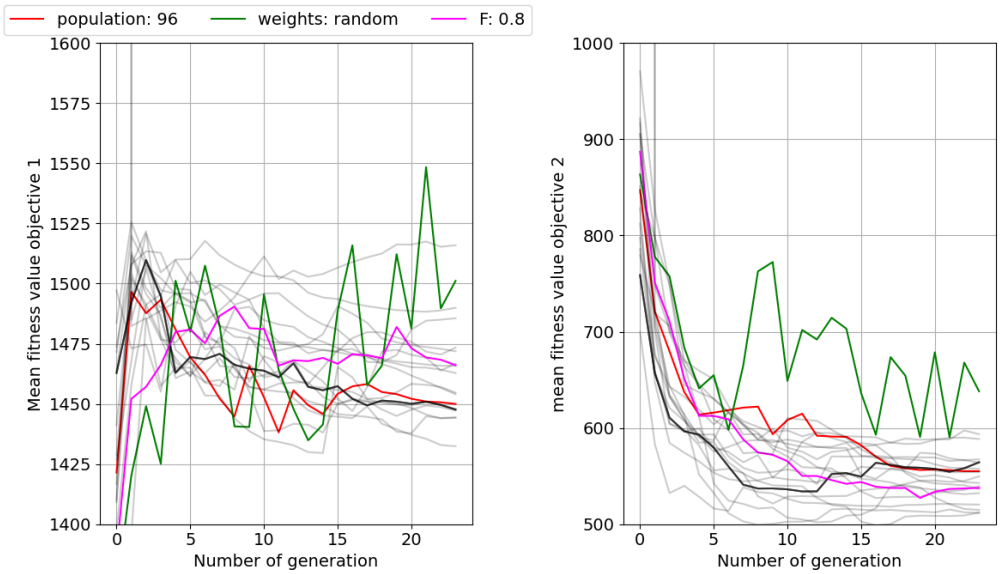


Figure 19: Evolution of mean fitness value for both objectives found by MOEA/D optimiser and all paramter settings as defined in Table 5, seed=31. All parameter settings that do not converge are shown in color, the rest in grey.

Figure 15 shows that the default setting CR=1 is of worse quality than CR=0.8. In both zoomed in regions it is clear that CR=0.8 improves the overall Pareto front slightly and is therefore the preferred setting. The default setting of F=0.5 produces the best results as seen in Figure 16, modifying this parameter leads to inferior results in terms of performance and number of Pareto individuals. Setting F=0.8 leads to no convergence as stated in Figure 19. Increasing the mutation distribution η_m from 20 to 50 results in a larger diversity of solutions especially towards objective 1, see Figure 17. A value of $\eta_m = 50$ is therefore chosen. preserve_diversity is set to True by default using two mechanisms described in (Li and Zhang, 2007) to eg. include parent individuals in the next evolution. This setting proved to have negligible effect on results and will be left to True [3]. The mean fitness for all settings over 25 runs is displayed in Figure 19, proving that all converge in under 50 runs except the previously mentioned failed settings. The converged majority settings was furthermore verified to be robust over changing seed numbers.

Table 6: Final selected tuning settings

Parameter	**Class 1**	**Class 2**
Population size	192	144
Weight generation method	low discrepancy	grid
Decomposition method	weighted	tchebycheff
Neighbourhood size	20	20
Crossover parameter (CR)	0.8	0.8
Differential evolution parameter (F)	0.5	0.5
Mutation distribution index (η_m)	50	50
preserving diversity mechanism [3]	True	True

The final tuned settings are shown in Table 6. CR, F, η_m, neighborhood size and the diversity option were optimised previously and are equal for both classes of settings, since they showed one clear optimum paramter choice. MOEA/D results were more sensitive for the other four settings, three of which (population, weights, decomposition) had settings tending to increase primarily performance of one objective. 192 individuals , low discrepancy weights and weighted decomposition are selected to improve the Pareto front towards objective 1. 144 individuals, grid weighting and tchebycheff decomposition were best to minimise objective 2. Two optimisation runs with settings of class 1 and 2 are produced and shown in Figure 20. Here it is very clear that class 1 settings produce superior results at minimising fuel mass and class 2 works much better in the middle and lower right part of the Pareto front. Both results are combined into the final optimum Pareto front displayed in Figure 21.

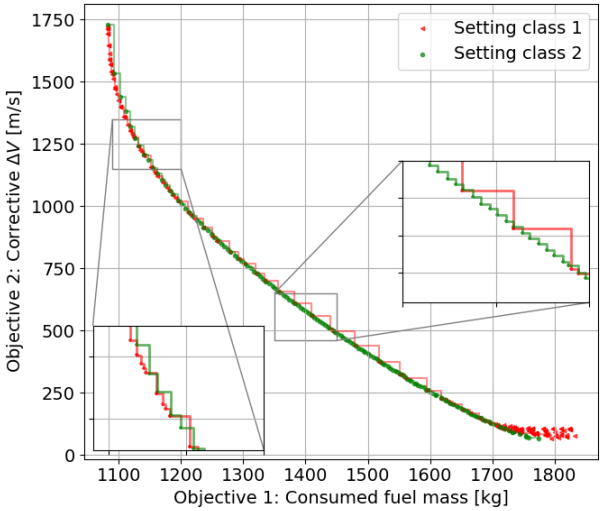


Figure 20: The 2 Pareto fronts of the final populations found by MOEA/D optimiser using selected settings from class 1 and 2 from Table 6, seed: 31.

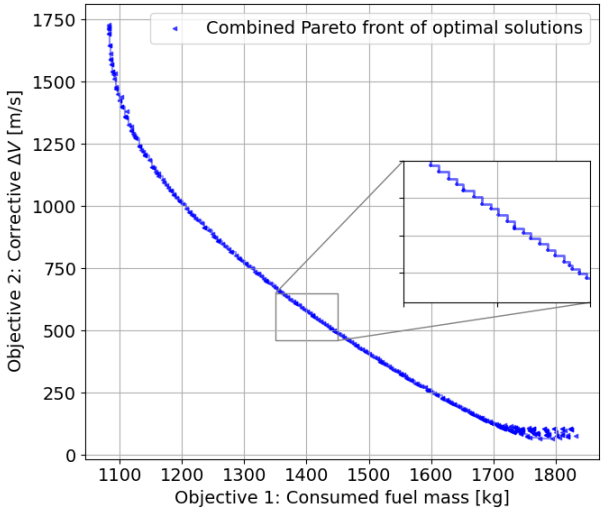


Figure 21: Final combined Pareto front with all 336 solutions of final population found by MOEA/D optimiser using both settings from Table 6, seed: 31.

2.3 Behaviour and Feasibility of the Pareto-optimal Solutions

In this subsection the final Pareto front found in Subsection 2.2 will be analysed. Using the decision and objective space, as well as the behavior of a number of parameters, the realism of the solutions and likelihood of them being close to the true optimum will be determined.

The final population found by MOEA/D combining two optimum setting classes are shown in Figure 21, showing the position of all 336 individual on the objective space. The Pareto front has been plotted in Figure 22, showing only 225 Pareto optimum solutions. Seven solutions (labelled 1 to 7) are highlighted and will be investigated in detail throughout the entire analysis. This is to emphasize the behaviour of solutions on different edges of the Pareto front. Point 1 is the minimum for objective 1 and Point 7 results in minimum ΔV required, the others are equally spaced along the number of points. The lowest fuel mass consumed is 1081 kg at 1720 m/s ΔV, the minimum corrective ΔV is 67 m/s at 1794 kg of fuel consumed, the other solutions are a compromise of both. The region around Point 5 and 6 is the densest implying that most optimum solutions are found here. The Pareto individuals at the boundaries are not optimal since they result in large penalties on the competing objective; going from Point 1 to 2 required only 100 kg more fuel but decreases objective 2 by more than 30%.

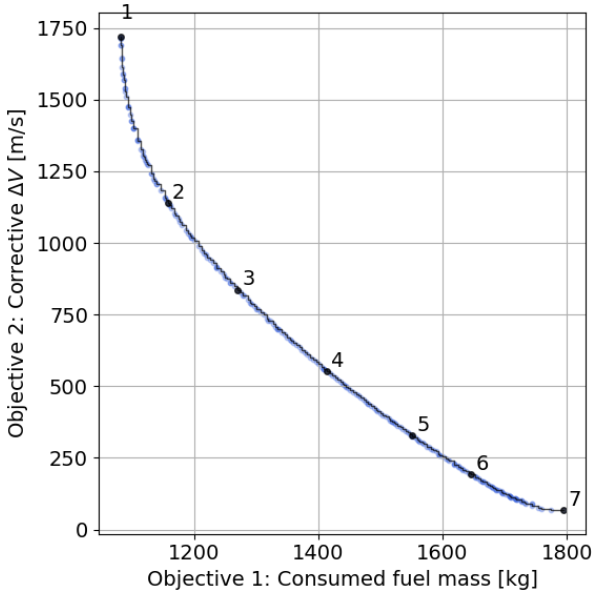


Figure 22: Pareto front with all optimum solutions found by MOEA/D and tuned settings. 7 points are selected for further analysis.

Figure 23 shows the relation between the thrust magnitude and both objectives. The optimum thrust magnitude seems to lie between 13 kN and 15 kN, which is close to the middle of the selected thrust range. There is no direct correlation between thrust and both objectives, since fuel mass consumed initially increases with higher thrust but then lowers because burn time increases for the lunar ascent trajectories optimising objective 2.

The node spacing decision variable in Figure 24 clearly shows all node spacings being between 83-100 s , but no correlation to the objectives. The trajectories minimising fuel mass tend to select 100s node spacing values, which is unexpected since they are very steep (terminating at 350 s) and thus terminate before even reaching the last two thrust angles. The exact behaviour/decisions taken of the evolutionary algorithm MOEA/D are not known but for the long trajectories (up to 650 s) minimising objective 2, the boundary of 100 s is never preferred. The node spacing range is therefore not limiting the optimisation problem.

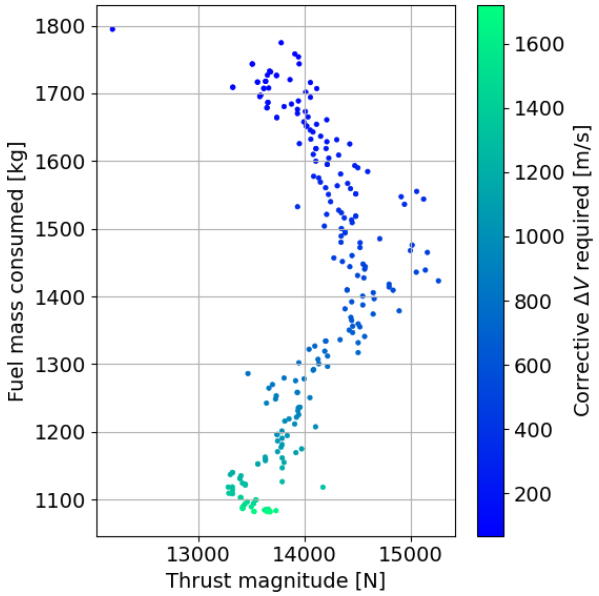


Figure 23: Thrust magnitude [N] of Pareto-optimum solutions against consumed fuel mass [kg]. Corrective ΔV [m/s] shown in colorbar.

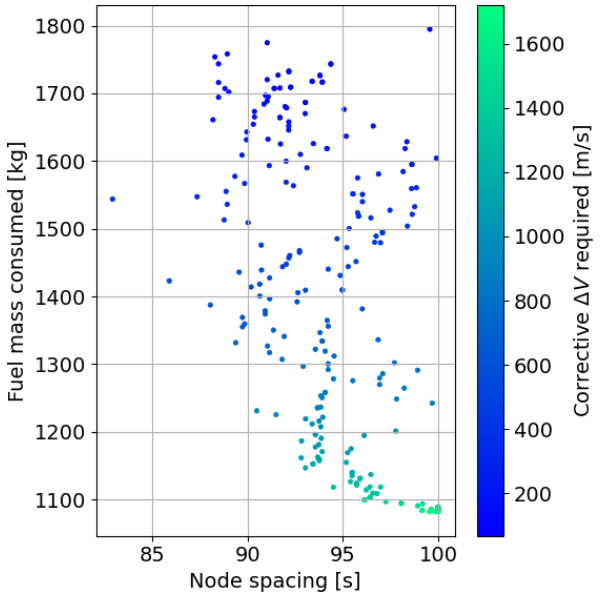


Figure 24: Node spacing [s] of Pareto-optimum solutions against consumed fuel mass [kg]. Corrective ΔV [m/s] shown in colourbar.

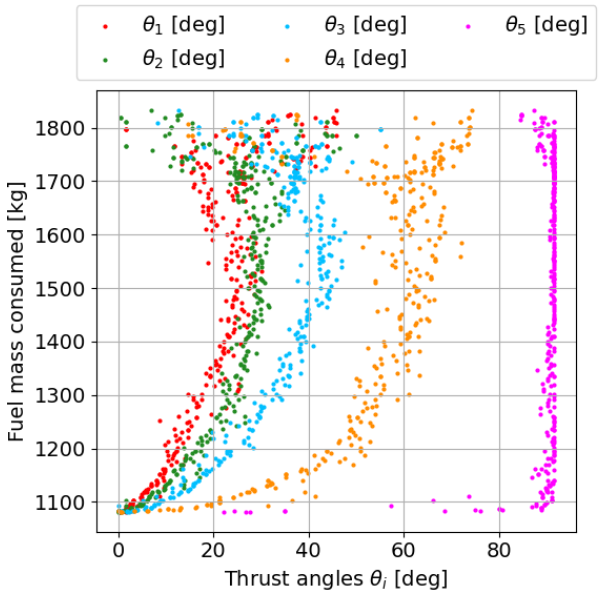


Figure 25: Fuel mass consumed as a function of the node angles values.

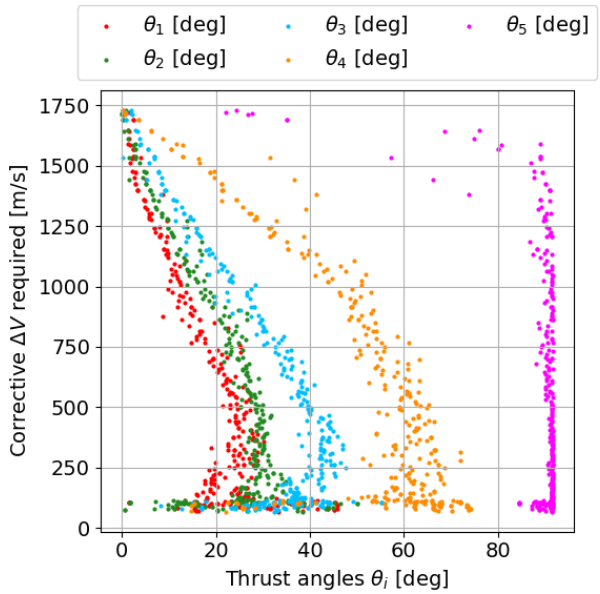


Figure 26: Necessary corrective delta-V as a function of the node angles values.

Figures 25 and 26 show the node angles for all Pareto optima. When optimising for ΔV it can be seen that the node angles get larger, this is so that the trajectory gets more shallow and the orbit more circular, requiring less ΔV. For the minimum fuel mass trajectories the node angles all converge to 0 as the shortest trajectory, which requires the least fuel is essentially vertical. Both plots do however show some limitations for the fifth node, almost all points are at the limit of 1.6 radians, which is actually already slightly over 90 degrees. This suggests that there are possibly optima outside of the chosen decision variable range, which is to be further analysed by the Monte-Carlo analysis performed in Section 4. The optimisation algorithm seems to prefer $\theta_5 > 90°$ likely to reach the circular target orbit. In the previous assignment[1], both objectives were combined to find the optimum thrust angle minimising total fuel and based our decision variable range on this decision. The optimum thrust angles θ_1 to θ_4 found with this method align with the angles found in the densest region shown in Figure 25 and Figure 26, which is where both objectives are equally minimised. This implies that the true optimum lies in this region where also Point 5 and 6 are located and where ΔV buffer is maximised, see Figure 29. Maximising ΔV buffer shows the approximate location of the true optimum since it is calculated from both objectives.

To verify the conclusions drawn in the decision and objective space the trajectory of each of the optimal solutions is plotted in Figure 27. Highlighted are the same seven points that were previously analysed. In correspondence to the density of the nodes in both decision and objective space, again a dense region around the fifth and sixth line can be seen. This is on the more shallow side of the trajectories, which is part of the more realistic set of results. Steep trajectories, like that of 1 and 2, would require high ΔV corrections which in theory are possible using an impulsive shot. However, in reality the ΔV capabilities are limited by the engine performance and large ΔV corrections would take quite some time. In this time the ascent module will have passed the analysed point to higher elevations or even have begun its decent again, which makes large ΔV manoeuvres an unrealistic solution.

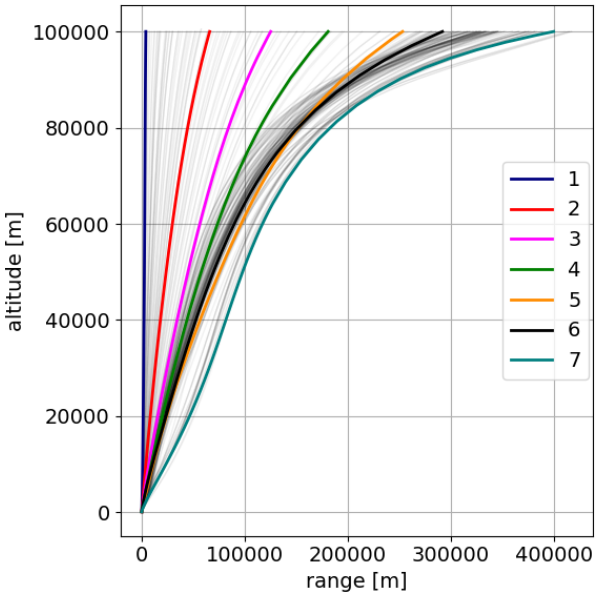


Figure 27: Altitude-range plot for all points on the pareto front, with highlighted trajectories as selected in.

To show the behavior of each of the orbits after propagation has terminated, a selection of Keplerian elements have been plotted in Figure 28. Semi major axis and eccentricity have been chosen to further investigate the final orbit obtained by each solution. The inclination is shown to see any out of plane motions that occur due to perturbations and other third body effects. The first and second plots of Figure 28 confirm the trajectories of Figure 27 by showing that the steep orbits have high eccentricities and low semi major axis, meaning that they are not long term stable and require large corrections. Only a very select number of points actually have an orbit with a semi-major axis larger than the lunar radius, and taking into account the eccentricity a minimum altitude larger than 0 m. These points are so far near the edge of the Pareto front that they likely will not be optima. This means any final orbit will need immediate corrections, not allowing for redundancy in manoeuvring time. In the third plot of Figure 28, all trajectories have a slightly different inclination, which is affected by different perturbing accelerations encountered by different propagation times and ranges for each optimum. Errors like these can be resolved using three dimensional thrust vectoring, of which the capabilities were shown in previous work [1].

The last analyses being done is that of the constraints posed in Section 1. In this case only the expected limiting constraints of acceleration, ΔV buffer, and thrust angle rate are plotted in Figure 29. It is clear that all Pareto solutions are far from violating any constraint. The acceleration is at most 5 m/s^2, whereas the constraint is set at 3g (or 29.43 m/s^2). The ΔV buffer never reaches below 0 m/s, thus all optima have more than enough fuel to perform the required manoeuvres for circularisation and inclination change. This margin would in further analysis allow for an additional study where the excess fuel is removed from the initial mass and a new Pareto front is created, allowing for an ever better optima. This is obviously an iterative process and could be repeated multiple times. Lastly the right plot in Figure 29 shows the maximum thrust angle rate, which never goes near the limit of 20 deg/s. Looking at the first plot it is shown that the acceleration decreases with higher ΔV for steeper trajectories. This makes sense as the steep trajectories have shorter propagation times and thus still have higher fuel mass (since thrust is relatively constant for most optima, as seen in Figure 23), which means that for the same thrust the acceleration is lower. The middle figure is essentially a combination of both objectives, and shows all fuel required for the ascent and ΔV correction. This shows that around the fifth and sixth point the total fuel is minimum and these are likely the true optima. This corresponds to the dense regions found in Figure 27 and the preference of a more circular orbit for lower corrections. The thrust angle rate plot (right plot) shows that the thrust angle rate does not change after 400 s, which is due to the maximum node spacing of 100 s shown in Figure 24. Also, some trajectories have negative thrust angle rates, meaning that a subsequent node has a smaller angle than a previous one, which is not something to be expected to be found in an optimal solution.

Overall the Pareto front is likely to be close to the true optimal front, since the analysis showed a very dense and clear line of best Pareto candidates. Not all individuals of the Pareto front are feasible, likely only the more shallow trajectories around point 5 and 6 in Figure 27 are realistic. These points are fortunately also the best solutions according to the ΔV buffer analysis in the middle plot of Figure 29. The largest uncertainty from the work done in Subsection 2.2 is regarding the limitations on the fifth node angle shown in Figure 25 and Figure 26, which will be studied further using the Monte-Carlo analysis in Section 4

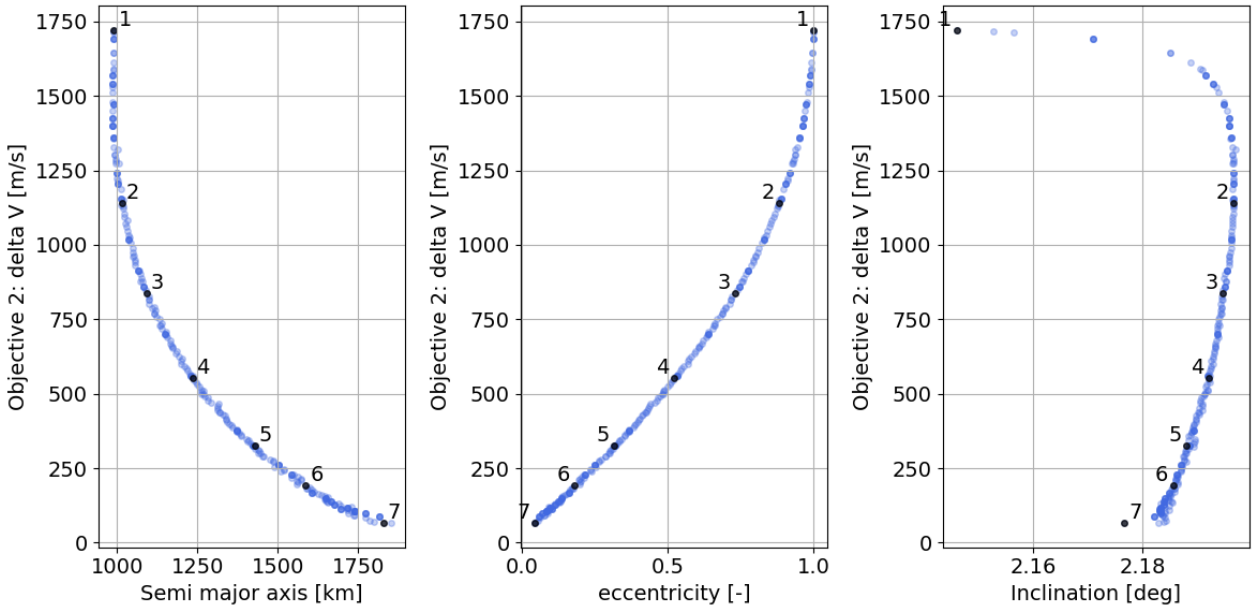


Figure 28: Semi-major axis, eccentricity, and inclination for all final orbits of the ascent trajectory of the pareto front.

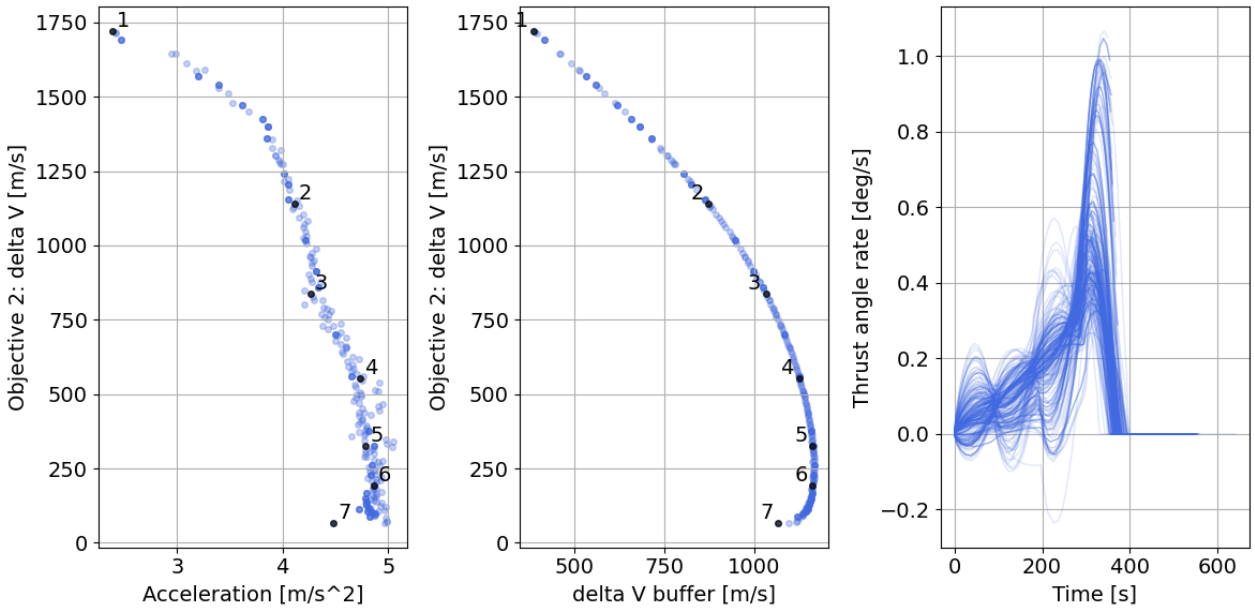


Figure 29: Acceleration, delta V buffer, and thrust angle rate values for the pareto optimal solutions

3 Single-Objective Optimisation

The completeness of the Pareto front obtained in the previous section will be analysed by considering a single-objective optimisation of the Lunar Ascent trajectory optimisation. First, a single objective is selected, followed by a selection of the optimiser and tuning of its parameters. Finally, the obtained optimum is compared to the Pareto front from Subsection 2.3.

3.1 Single-Objective Selection

In this work, two objectives have been considered: (1) minimising the fuel mass required to reach 100 km altitude, which is motivated by the cost associated to carrying unnecessary propellant mass throughout the lunar mission; (2) minimisation of the corrective delta-V necessary to enter a circular orbit at the end of the ascent. While the value of the former is estimated accurately through the dynamical model implemented in TUDAT, the corrective manoeuvre is only modelled through large simplifications (impulsive thrust)[1]. This lack of details of the orbit insertion could result in some trajectories deemed as feasible, while no practical manoeuvre would permit to enter the orbit (for large corrective delta-V). However, this shortcoming can be mitigated by minimising ΔV_{tar}: the smaller the corrective delta-V, the more reliable the impulsive

[1]The corrective delta-V is considered more as an indicator of how far the end state of the module is from the required state to stay in orbit.

thrust approximation becomes. Therefore, the minimisation ΔV_{tar} is considered as the most important objective and is selected as single-objective to investigate the completeness of the Pareto front.

Following, the rest of the optimisation problem formulation is unchanged from Table 2, except from the removal of the propellant mass minimisation, and the same penalty method is used to implement the constraints. Furthermore, objective (1) is not replaced by a constraint, as the ΔV_{buffer} constraint is already present and ensures that the fuel mass necessary to perform both the ascent and the insertion does not exceed the propellant mass on-board. No other physical constraint exists with respect to the amount of propellant mass used during the ascent.

3.2 Optimiser Selection

Having formulated the single-objective problem, the optimisers available in pygmo are considered, and an appropriate one is selected. A total of 14 optimisers are readily available: Extended Ant Colony Optimization (GACO), Differential Evolution (DE), Self-adaptive DE (SADE), Self-adaptive DE (DE1220)[2], Grey wolf optimizer (GWO), Improved Harmony Search (IHS), Particle Swarm Optimization (PSO), Particle Swarm Optimization Generational (GPSO), (N+1)-ES Simple Evolutionary Algorithm (SEA), Simple Genetic Algorithm (SGA), Corana's Simulated Annealing (SA), Artificial Bee Colony (ABC), Covariance Matrix Adaptation Evo. Strategy (CMA-ES), and Exponential Evolution Strategies (xNES). Some of those algorithms being variations or improvements of others, only a subset of the listed options will be considered in the selection, based on the documentation provided by [2], and a brief review of literature. The following are noted:

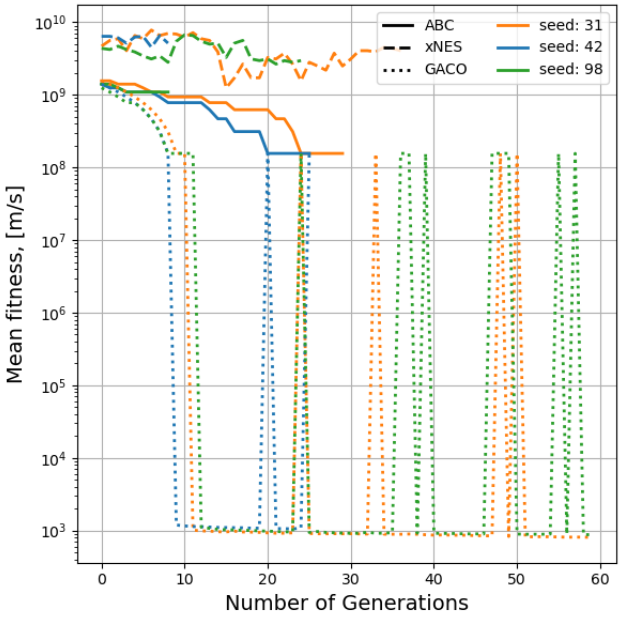
- The DE, SADE, and DE1220 are all subsequent improvements on each other. The DE algorithm was first introduced by Storn and Price [9], but could significantly be improved using the concept of self-adaptation to the C_R and F parameters of the original DE algorithm [2, 10, 11], which was implemented in the iDE and jDE variants of SADE. The DE1220 is an algorithm developed by the Pagmo team to implement the self-adaptation of the mutation variants on top of the existing SADE algorithm [2]. DE1220 is then considered as the most developed version of the DE family available in pygmo, and will be the only one considered further.
- The GWO algorithm was first presented by Mirjalili *et al.* [12]. However, Biscani and Izzo [2] states that the method is merely a small variation on already available methods, and could be interpreted as the result of an evolutionary or a particle swarm metaphor. Furthermore, the algorithm is shows poor performance when the optimal solution is not 0 [13]. For this reason, the GWO method was left out of this selection procedure.
- The IHS was first introduced by Mahdavi *et al.* [14], and makes use of a musician metaphor in the search for a global optimum. Despite its decent performance according to Biscani and Izzo [2], it has been criticised for being only a variation of algorithms of the family of Evolutionary Strategies or Genetic Algorithms, without stating it appropriately, and obscuring its internal functioning. As another Genetic Algorithm will be considered in this work, this method was not considered further.
- The PSO and GPSO algorithms are two versions of the same concept detailed by Poli *et al.* [15]: particles with velocities and positions travel through the design space by being attracted to their own minimum and the minimum of the complete swarm. While in the former approach, the velocities and positions of particles are updated one after another, the latter implementation is generational, meaning that all velocities are updated at the same time. The GPSO alogrithm is then a more general implementation (applicable to a wider range of problems [2]) of PSO. The GPSO will be considered further.
- According to Biscani and Izzo [2], both SEA and SGA rely on the same concept, except that SGA also implements crossovers [16]. As the latter is then an extended version of the former, only the SGA will be further considered in this work.
- While the Simulated-Annealing method was first considered as a good option to be considered, initial numerical results revealed that the method (with default settings) took around 30 minutes per evolution, rendering the method impractical. SA will therefore not be considered further in this work[3].
- Both the CMA-ES and xNES are considered as very successful algorithms [17, 18], with the latter being an improved version of the former [2, 18]. Therefore, only the xNES will be further considered.

This brief selection leaves the following algorithms to be investigated with their default settings: GACO, DE1220, GPSO, SGA, ABC, and xNES. Note that all those methods have been tested against a range of test functions in literature [11, 10, 6, 18], and this process is not repeated here. As a first indicator of the performance of each algorithm, the mean fitness as a function of the generation for three different seeds (31, 42, and 98) is plotted for each. Note that the GACO method requires a minimum population size of 64, which will be used for all optimisers to ensure a fair comparison. Additionally, all optimisations are stopped after 60 generations, such that the overall trend of the convergence can be seen without resulting in unreasonable computational time.

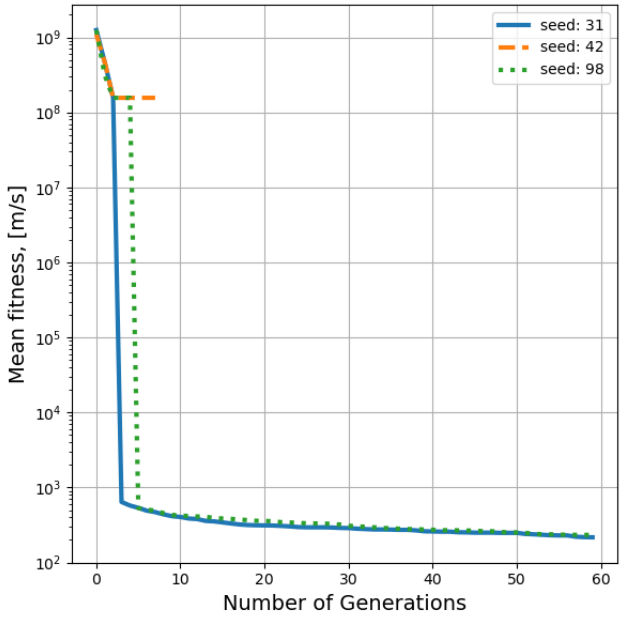
Considering Figure 30, and more particularly Figure 30a, it can be concluded that the ABC, xNES and GACO optimisers are not suited for the Lunar Ascent problem. On the one hand, the former two seem to converge too early due to their inability to remove the individuals violating the maximum constraints (resulting in a penalty of 1E10) fast enough and convergence to local minima. On the other hand, the GACO method shows a bad convergence and oscillates between

[2]Improvement of the jDE or iDE algorithm developed by Pagmo.

[3]It is not excluded that this could be the result of a mistake of the author.

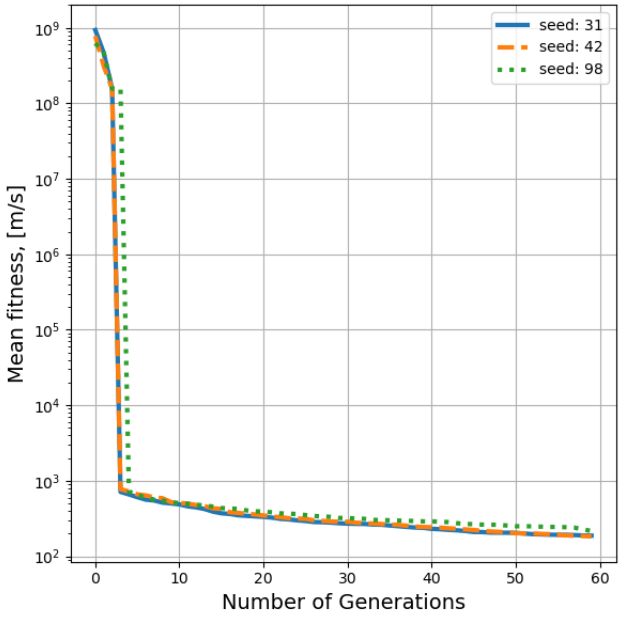


(a) ABC, xNES, and GACO.

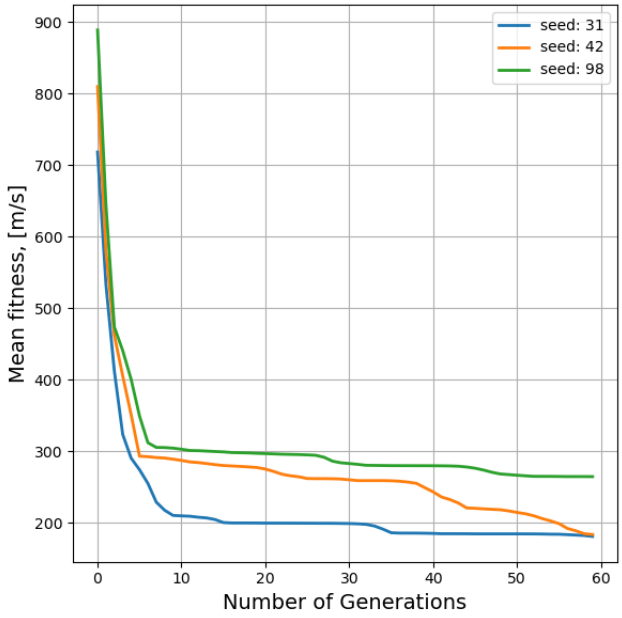


(b) GPSO.

Figure 30: Evolution of the mean fitness as a function of the number of generations of the selected algorithms, for a population size of 64, and seeds: 31, 42 and 98.



(c) DE1220.



(d) SGA.

Figure 30: Evolution of the mean fitness as a function of the number of generations of the selected algorithms, for a population size of 64, and seeds: 31, 42 and 98.

very high and relatively lower (yet still undesirable) values. Those methods are therefore discarded for the rest of the analysis. Following, the GPSO, SGA and DE1220 algorithms seem to be reasonable options, with one main pitfall: the GPSO algorithm converges to a local optimum for seed 42, showing that it is less robust than the other two.

Following, Figure 31 shows the average performance of each algorithm across the three seeds (two seeds for the GPSO algorithm), and providing insights into the performance of the algorithms with respect to each other. It can first be noted that the GPSO method results in the worst performance after 60 evolutions, while having a slightly shallower slope than the DE1220 algorithm. It is therefore unlikely that GPSO would become more advantageous if more generations were considered. A similar conclusion can be drawn for the GSA algorithm, despite being closer to DE1220 performance after 60 generations. Additionally, it is uncertain if GPSO would reach the performance of the SGA if more generations were considered.

Furthermore, Figures 32, 34, and 33 show the population of the final generation through an altitude-range trajectory plots, coloured by the thrust magnitude associated to the trajectory. The thrust magnitude was used due to its large impact on the mission objectives according to Veithen *et al.* [1]. From these figures, it becomes apparent the SGA provides a much

less diverse population despite not being fully converged (according to Figure 30d). This is a result of the structure of the algorithm: the algorithm quickly converges to the best fitness value in the population, and only moves from the local optimum from the addition of newly created individuals which shake the entire population. Less diverse populations indicate that the algorithm is closer to convergence and has less potential for further improvement (despite being less performant than the DE1220 algorithm after 60 generations). Additionally, if the optimisation was to be stopped before reaching the global optimum, a diverse population would be desired such that less optimal individuals could also be considered if they better fit the propellant mass objective or are found more attractive for reasons outside the scope of this work (such as cost). Following, both the GPSO and DE1220 populations show a wider diversity in the solutions, indicating that they both will likely improve significantly with a higher number of generations.

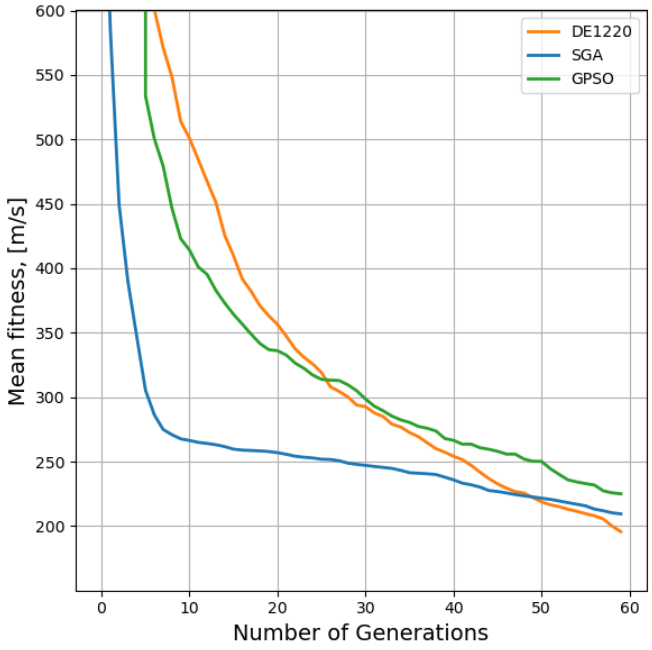


Figure 31: Evolution of the mean fitness as a function of the number of generations of the DE1220-SGA-GPSO, for a population size of 64, and averaged across seeds 31, 42, and 98. Note that the GPSO data was averaged on seeds 31 and 98 only as seed 42 converged to a local optima.

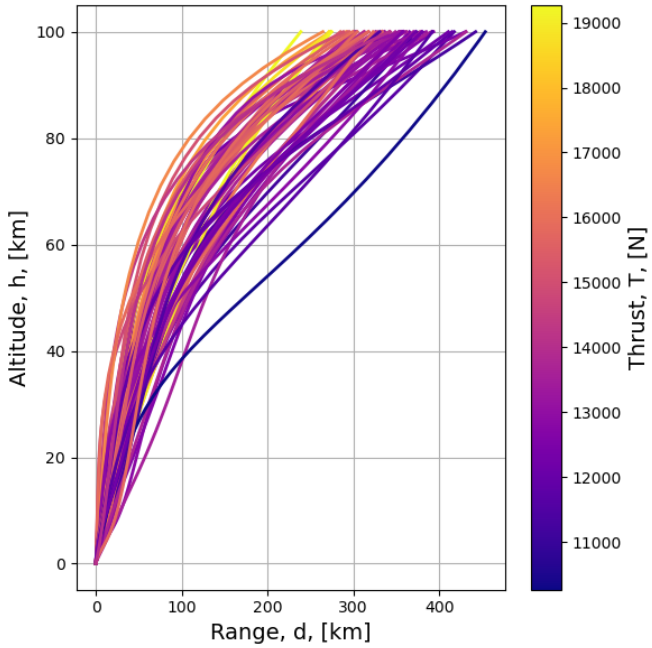


Figure 32: Altitude-range trajectory, coloured by the thrust magnitude, of the final generation of DE1220 with seed 98 and population size of 64.

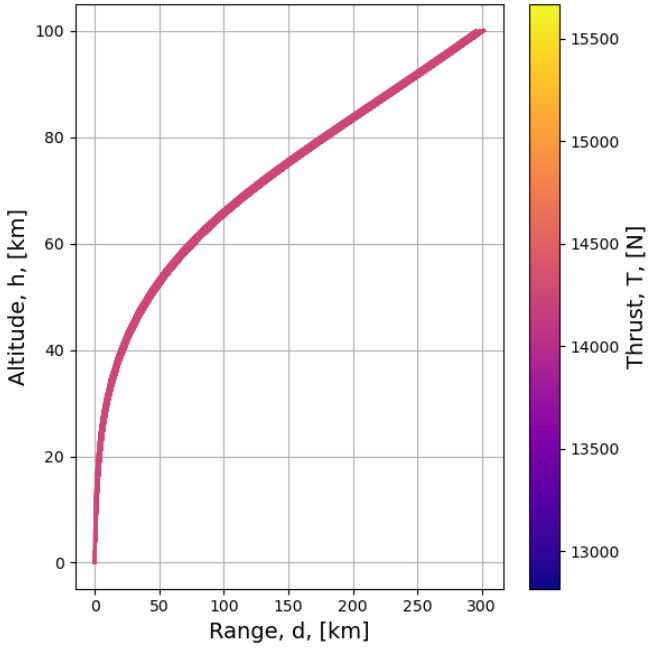


Figure 33: Altitude-range trajectory, coloured by the thrust magnitude, of the final generation of SGA with seed 98 and population size of 64.

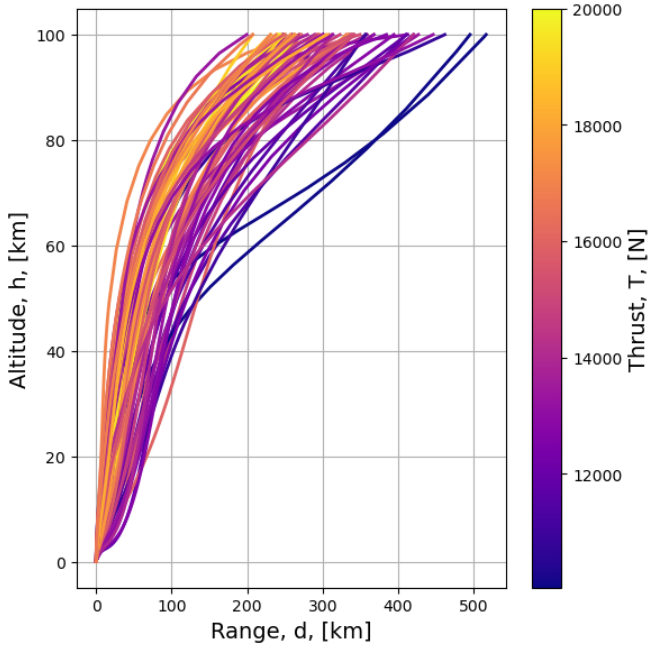


Figure 34: Altitude-range trajectory, coloured by the thrust magnitude, of the final generation of GPSO with seed 98 and population size of 64.

Moreover, the all-time champion of each algorithm for each seed is given in Table 7, showing that DE1220 is more robust and performs generally better than the other two alternatives. Additionally, it can be noted that while the mean fitness of the SGA was shown to be lower than using GPSO, its all-time champion is significantly worse. This is a result of the low diversity in the population of the SGA at all generations.

Table 7: Fitness in [m/s] of the all-time champion of the DE1220, SGA, and GPSO algorithms for seeds 31, 42, and 98. N/C=Not converged.

Seed	**31**	**42**	**98**
DE1220	68.57	62.29	62.79
GPSO	49.17	78.23	N/C
SGA	173.25	127.24	263.59

To conclude, the analysis provided in this section showed that the Simple Genetic Algorithm comes closer to convergence in the number of generations considered, while yielding poor performance in comparison to the DE1220, and has less potential for further improvements than the GPSO method. While the GPSO still showed some room for improvements, its expected rate of improvement is lower than DE1220, while being less robust and demonstrating worse performance after 60 generations. The DE1220 algorithm is robust (with respect to the used seed), provides the best performance in terms of the mean fitness, demonstrates a wide population density after 60 generations (showing room for improvement), and has the best all-time champion for two seeds out of three (with each being within 10% of each other). For those reasons, the DE1220 algorithm is selected for further work.

3.3 Tuning and Robustness of Selected Optimiser

While the DE1220 algorithm already showed a good performance in the previous section, tuning of its parameters can enhance its capabilities to converge faster to a value closer to the global optimum. The following parameters of the DE1220 optimiser[4] will be considered in this section [2]:

- **Seed**: the random seed initialising the pseudo-number sequence used by the algorithm. While the optimiser selected was shown to be robust, some seeds can provide better performance than others. Three seeds will be experimented with: 67, 123, and 989.
- **Variant adptv**: the self-adaptation scheme used to determine the values of C_R and F during the optimisation process: jDE (1) or iDE (2). The latter having been developed at a later date to improve the performance of past methods[11], iDE is expected to yield a better performance.
- **Population size**: the number of individuals considered in the evolutionary algorithm. A higher population generally results in better all-time champions in a given amount of generation. However, a larger population also gives rise to a longer computational time per generation. Therefore, a trade-off between the population size and the amount of generations computed is necessary. Population sizes of 24, 48, 96, and 144 will be considered.
- **Memory**: the F and C_R parameters are reset between successive runs in 10% of the cases only if this option is put to True (compared to 100% of the cases if it is set to False).
- **Allowed variants**: the mutation method which can be used during the optimisation process. The mutation variant is selected randomly from the pool of allowed ones [2]. Note that supplying a single value reduces the algorithm to SADE, and the advantage of such approach is to combine the performance of different methods, or ensure a minimum performance if the different algorithms cannot each be tested beforehand. A total of 18 variants are available in pygmo, and repeated in Table 8. The following allowed combinations will be tested: all variants individually, combination of all the 'best' methods, combination of all the 'rand' methods, and combination of 'rand-to-best', 'rand-to-current' and 'rand-to-best-to-current' together.

Table 8: Mutation variants options [2].

1 - best/1/exp	10 - rand/2/bin
2 - rand/1/exp	11 - rand/3/exp
3 - rand-to-best/1/exp	12 - rand/3/bin
4 - best/2/exp	13 - best/3/exp
5 - rand/2/exp	14 - best/3/bin
6 - best/1/bin	15 - rand-to-current/2/exp
7 - rand/1/bin	16 - rand-to-current/2/bin
8 - rand-to-best/1/bin	17 - rand-to-best-and-current/2/exp
9 - best/2/bin	18 - rand-to-best-and-current/2/bin

[4]Note that the number of generations is not taken into account as the impact of the tuning parameters will be evaluated after 60 generations, and the selected set of inputs will be run until convergence.

Table 9: Summary of the tuned settings.

Parameter	Default	Alternate Options
Seed	randomly selected	67, 123, 989
Self-adaptive scheme	jDE	iDE
Population size	48	24, 96, 144
Memory	False	True
Mutation variants	[2, 3, 7, 10, 13, 14, 15, 16]	[i] for i=1, 2,..., 18; [1, 4, 6, 9, 13, 14]; [2, 5, 7, 10, 11, 12]; [3, 8, 15, 16, 17, 18]

The two parameters which are not considered are the convergence criteria (**xtol**, **ftol**), as they have been replaced by the definition given in Table 4. The tuning of those parameters will be done in three phases: (1) the seed, adptv variant, and population size are selected by comparing the performance of different combinations of their possible values; (2) the effect of the memory on the optimisation performance is investigated; (3) the performance of the allowed mutation variants is considered. Each phase will use the settings determined in the previous, and the default values are used throughout the first phase.

In Figure 35, the performance of different seeds, population sizes, and self-adaptation methods is investigated. First, comparing Figures 35a and 35b, it is clear that the iDE scheme has a tendency to disadvantage larger population sizes compared to jDE, and that mean fitness values are more spread out for the former than the latter. Second, it could have been expected that a larger population would result in a consistently better performance both in terms of mean and best fitness, due to the higher probability of finding better thrust parameters with the larger number of individuals. While this is somewhat the case for the jDE scheme, roles are reversed using the iDE one. In both cases, a population size of 48 individuals outperforms other populations in terms of mean fitness for seed 123 and 989. Considering the best fitness figure of merit, the picture is more expected: larger populations have a higher probability of finding a better champion, and reach lower champion fitness values faster. However, again, optimisation using a population size of 48 individuals performs relatively well for seeds 123 and 989 and competes with the populations of 96 and 144. Third, another important aspect of the performance is the potential of the optimisation process to reach even lower fitnesses if more generations were to be considered. One indicator for such behaviour is the slope of the mean fitness curve at the 60th generation. While all slopes are relatively similar in Figure 35a, the more spread out behaviour in Figure 35b reveals that the best population (seed 123, 48 individuals) has a steeper convergence rate than most of the other curves. Those observations, coupled with the time required for each computation shown in Table 10, reveal the need for a trade-off between the number of individuals considered, and the time needed for each computation[5]. A larger amount of individuals results in better 'best fitnesses' in less generations, at the cost of the increased computation time. However, similar results could be reached with a smaller population in more generations. Given the good performance of the optimisation using seed 123, population size of 48 individuals, and jDE, and its competitiveness with optimisations using larger populations, this setting is selected for further tuning. Its performance is further shown in Figure 36, which evidently reveals the jDE self-adaptation method as the most suited option for this population size and seed[6].

Table 10: Average time per 60 generations as a function of the population size. Effects of other parameters on the computation time were found to be negligible.

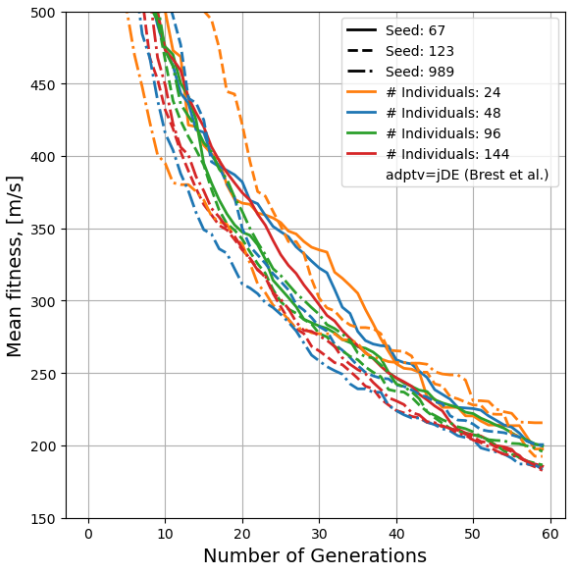
Population Size	24	48	96	144
Computational Time [minutes]	≈ 15	≈ 30	≈ 60	≈ 90

Having selected the population size, self-adaptation scheme, and seed, the effect of memory setting is investigated in Figure 37 (note that the first few generations during which the fitness has values around 1E10 are left-out). It is clear from Figure 37a that the ability of remembering the F and C_R parameters makes the algorithm significantly more performant and allows the mean fitness to reach lower values much faster, increasing the convergence rate. Additionally, Figure 37b reveals that the memory setting permitted the optimiser to find a low champion value much faster, and results in a lower all-time champion after 60 generations. The memory option is therefore selected for subsequent tuning and optimisations.

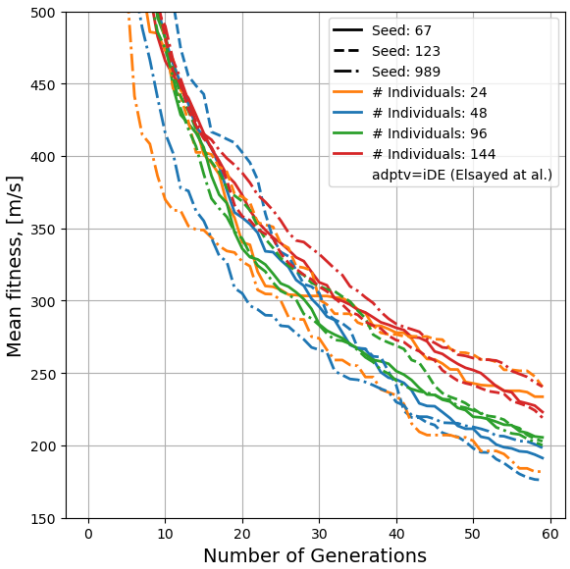
The last phase of the tuning involves the selection of the mutation variants from the set of 18 options shown in Table 8, and some combinations of them. For the DE algorithm, Elsayed *et al.* [11] suggested that the "current-to-best" and "current-to-rand" are not suitable for problems with high-dimensionality such as this one (7 variables), and that "best/1/bin" and "rand/1/bin" are much better than "best/1/exp" and "rand/1/exp". However, the latter statement is challenged by the results shown below. The results are shown in Figure 38, where the focus should be geared toward the lower right of both the mean and best fitness plots. The aim being to find the most promising settings to find a global optimum, one is interested in the combination of a low best fitness and a fast decreasing (and low) mean fitness. However, if a relatively low best

[5]Full convergence could take many more generations

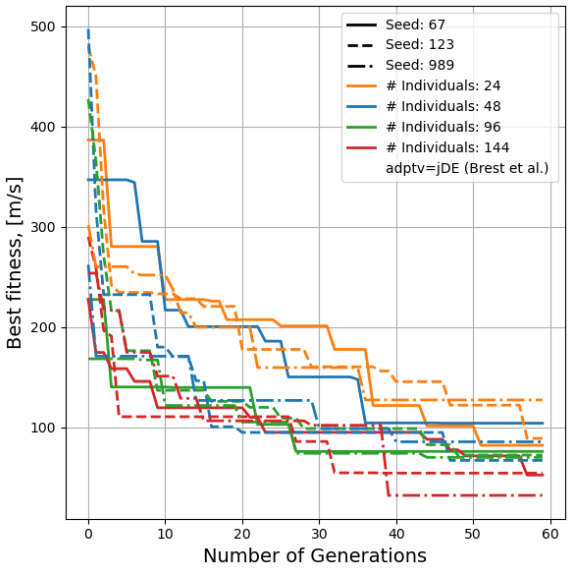
[6]The choice of a particular seed for the optimisation can be perceived as doubtful, however, if that particular seed permits to yield lower fitness values, there is no reason to reject the set of settings because the performance if other seeds is a bit worsened.



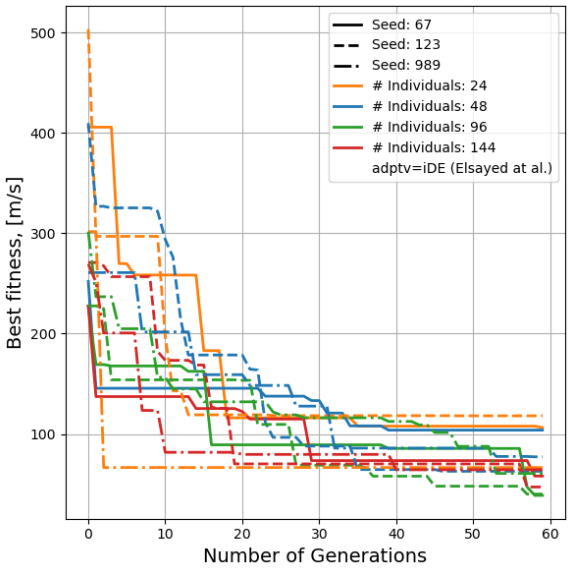
(a) Mean fitness jDE.



(b) Mean fitness iDE.



(c) Best fitness jDE.



(d) Best fitness iDE.

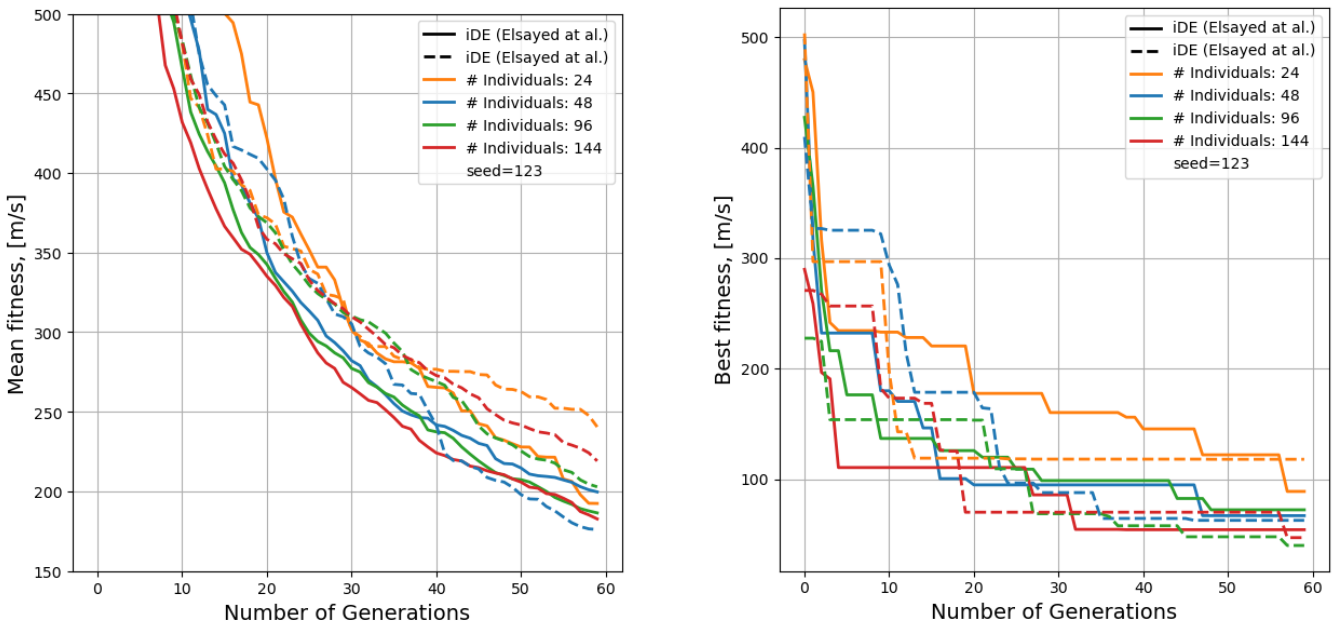
Figure 35: Mean and best fitness as a function of the number of generations for population sizes of 24, 48, 96, and 144 individuals, seeds 67, 123, and 989, and for both the jDE and iDE self-adaptation schemes.

fitness is found in combination to a high mean fitness, this could be an indication that the population still has a lot of room for improvement. The four best mutation variants (in terms of best fitness) are shown in Table 11, and will be investigated further by optimising until convergence for those sets of settings. A particular point of attention is the best/2/bin option showing a high mean fitness with a relatively low best fitness, which shows a lot of potential, as stated previously. Further note that best/1/exp is among the top four, which contradicts directly the advice given in [11].

Table 11: Selected mutation variants settings to be further investigated. Mean and best fitnesses given after 60 generations.

Mutation variant	**Mean fitness**	**Best fitness**
1 - best/1/exp	52.13	23.78
9 - best/2/bin	289.86	33.52
13 - best/3/exp	46.11	21.72
14 - best/3/bin	23.09	17.48

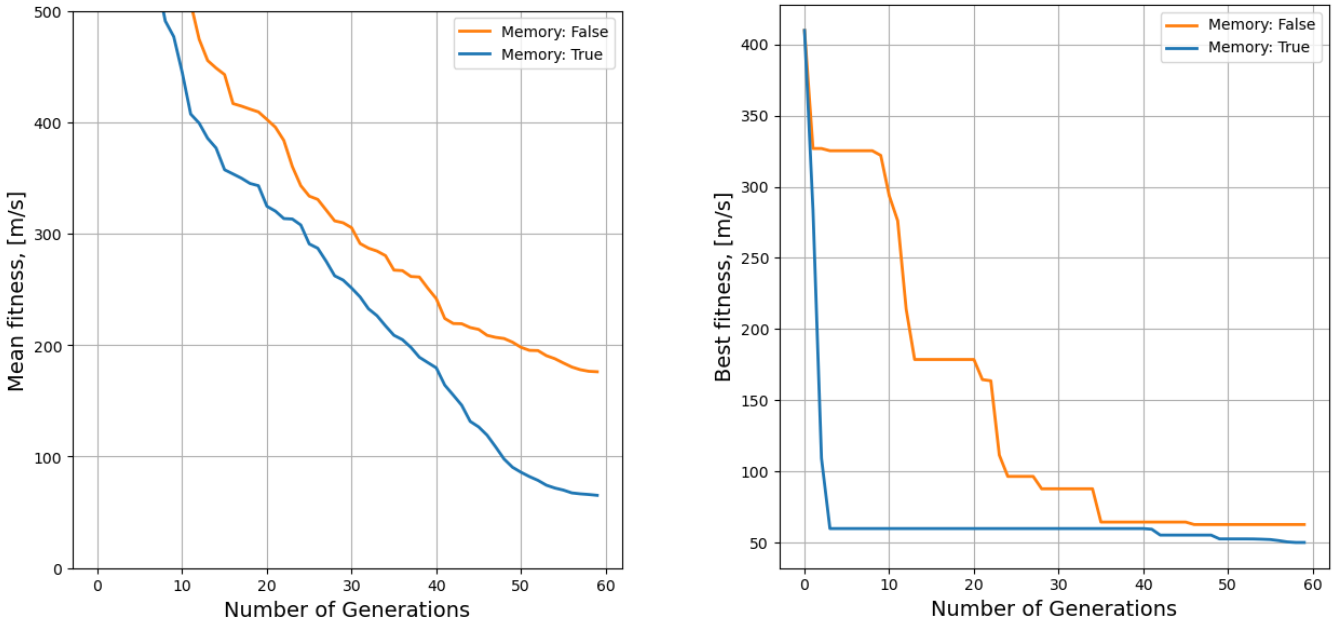
To conclude, the settings shown in Table 12 will be used to perform a total of four full optimisation runs in the next subsection, and the best all-time champion of all four optimisations will be selected as the optimum value for the single-objective problem optimisation.



(a) Mean fitness.

(b) Best fitness.

Figure 36: Mean and best fitness as a function of the number of generations as a direct comparison of the jDE and iDE self-adaptation scheme with seed 123, and population sizes of 24, 48, 96, and 144.



(a) Mean fitness.

(b) Best fitness.

Figure 37: Mean and best fitness as a function of the number of generations for a population size of 48 individuals, the iDE self-adaptation scheme, and a seed of 123. Assessment of the effect of the Memory setting.

Table 12: Selected settings of the DE1220 single objective algorithm for the final optimisation.

Setting	**Selected option**
Seed	123
Population size	48
Self-adaptation scheme	iDE
Memory	True
Mutation variants	1 - best/1/exp
	9 - best/2/bin
	13 - best/3/exp
	14 - best/3/bin

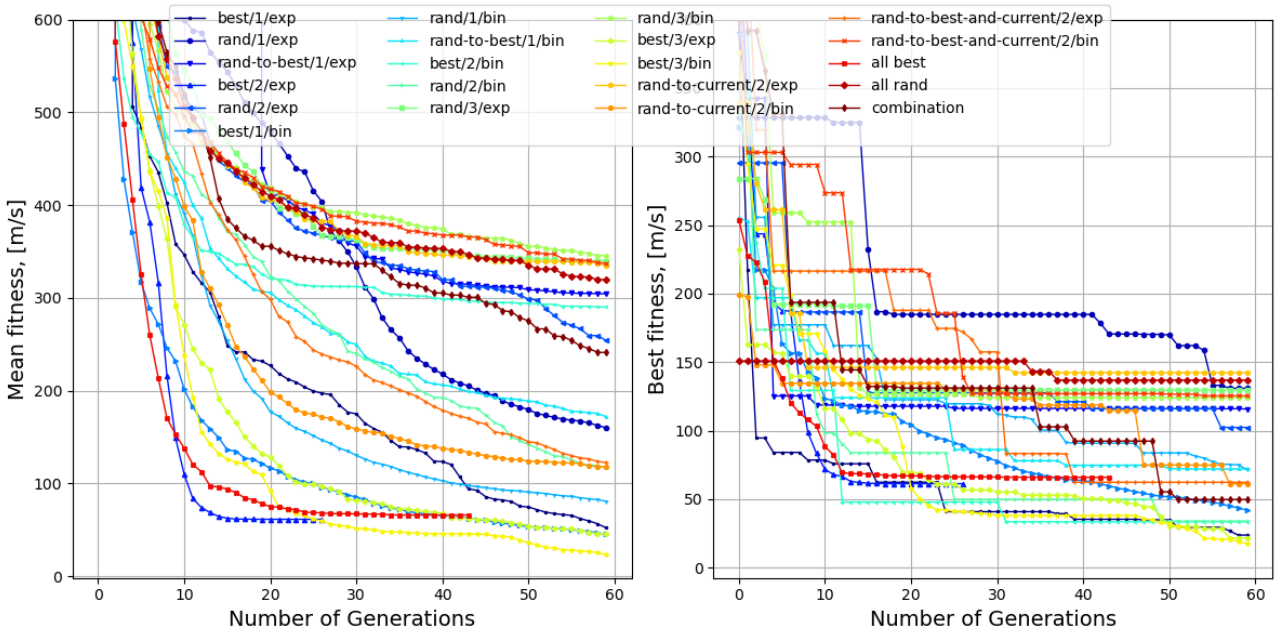


Figure 38: Mean and best fitness as a function of the number of generations using a population size of 48, seed of 123, the iDE self-adaptation algorithm, and the memory setting, for different mutation variants. The 'combination' entry refers to the combinations of 'rand-to-best', 'rand-to-current' and 'rand-to-best-to-current'.

3.4 Completeness of Multi-Objective Pareto Front

The settings of the DE1220 algorithm selected in the previous section are used to find the global optimum of the single-objective problem (or a value close to it). The optimum (and converged) values of each set of settings is given in Table 13, labelled by their mutation variant (as it is the only difference between each set). The best overall fitness was found using the best/3/exp mutation variant. The position of the optimum compared to the Pareto front obtained earlier is seen in Figure 39, showing that it was not previously part of the front. This result could be expected from the nature of the multi-objective optimisation being considerably more difficult to perform than a single-objective: the more objectives are present, the higher the complexity becomes. For a single-objective optimisation, the optimiser is capable to focus on a single direction and becomes more performant overall.

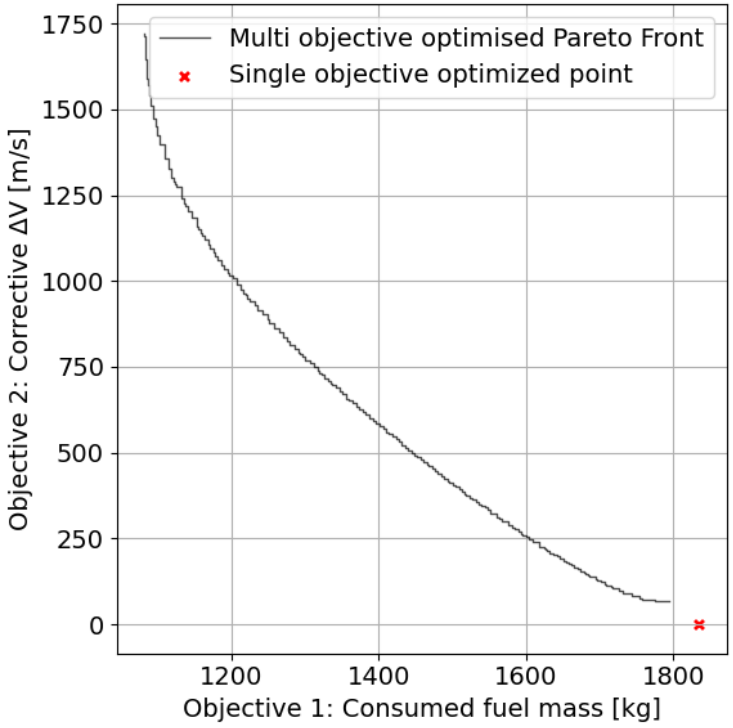


Figure 39: Single-objective optimisation optimum position with respect to the Pareto front determined with the MOEA/D multi-objective optimiser.

Table 13: Converged value of the objective function for the four settings set considered.

Settings	ΔV_{tar} [m/s]
1 - best/1/exp	1.018
9 - best/2/bin	33.520
13 - best/3/exp	0.62
14 - best/3/bin	7.32

This optimum is a very realistic trajectory strategy, as it fits the constraint on the amount of propellant available on board while coming extremely close to directly entering orbit. A delta-V of 0.6 m/s can be achieved in a manoeuvre coming very close to an impulsive shot, as modelled in this work, meaning that the trajectory is very likely to be practically feasible. The trajectory (in an altitude-range plot) and thrust vector orientation as a function of time are shown in Figures 40 and 41, demonstrating that the trajectory comes nearly tangent to the desired orbital altitude. It can be further noted that ΔV_{tar} takes into account the inclination necessary to reach a 2.2deg inclination, but cannot be corrected for during the trajectory due to the two-dimensional thrust vector (restricted to a plane). The only way to reduce that component of ΔV_{tar} is to minimise the impact of perturbations changing the orbital plane or the trajectory, but this cannot reduce the component down to zero. The characteristic of the trajectory and final inserted orbit are given in Table 14. The design parameters of the optimum are given under point 8 of Table 15.

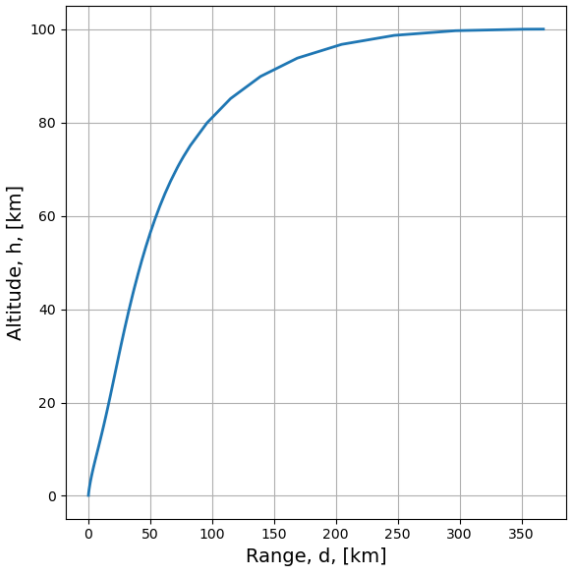


Figure 40: Altitude-range of the optimal trajectory

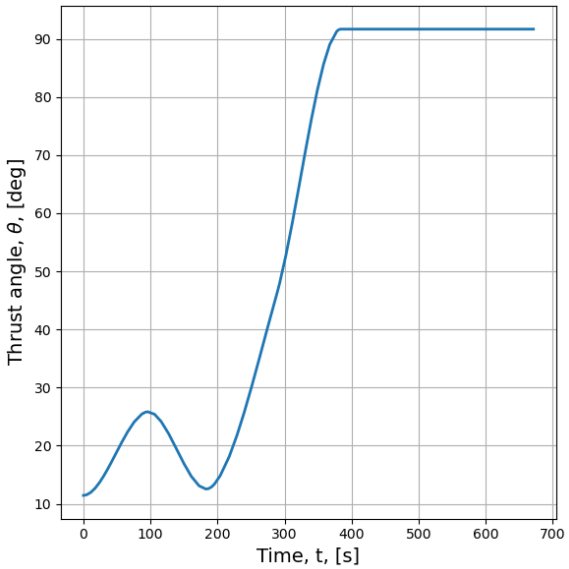


Figure 41: Thrust angle as a function of time for the optimal trajectory.

Table 14: Characteristics of the trajectory and final inserted orbit for the optimal trajectory.

Parameter	Value
Consumed Fuel [kg]	1834.60
Max angular rate [deg/s]	0.64
Max acceleration [m/s/s]	4.51
Semi-major axis [km]	1838.030
Eccentricity [-]	5.348e-4
Inclination [deg]	2.180

4 Robustness and Completeness of the Pareto Front

In this section, a number of points on the Pareto front, as well as the single objective optimized point will be analyzed by means of a refinement study and a sensitivity analysis. First, the points will be selected, to cover a reasonable amount of the Pareto optimum, then a Monte Carlo analysis is done on these points to find potential new optima. In parallel a sensitivity analysis will be done on the same points based on the primary sources of model simplification and uncertainties found in the second assignment.

4.1 Points Selection

For the Monte Carlo and sensitivity analysis, a number of points will be selected. For the analysis done in Subsection 2.3 a number of equally spaced points have already been selected and studied. As these cover the entire Pareto front these points will also be used here. Additionally, the single objective optimization resulted in another point, which will be added to this list. The complete list of initial parameters of these points can be found in Table 15.

Table 15: Data points used for further analysis.

Point	Thrust [N]	Node Spacing [s]	Node 1 [deg]	Node 2 [deg]	Node 3 [deg]	Node 4 [deg]	node 5 [deg]
1	13669.8	99.9	1.6	0.3	0.9	0.6	2.2
2	13629.1	93.8	9.9	11.7	19.2	38.3	89.9
3	13695.9	96.9	14.4	24.3	30.0	51.4	90.7
4	14794.3	90.2	22.9	28.1	37.3	60.6	90.2
5	14222.9	96.0	25.9	30.1	46.9	58.1	91.6
6	14051.3	92.2	19.0	35.0	35.8	61.5	91.6
7	12188.3	99.5	33.2	38.4	32.4	22.1	91.6
8	12117.1	95.6	11.4	25.8	13.0	45.9	91.7

in Subsection 2.3 the trajectories have already been plotted in Figure 27. As discussed in this section not all points are as feasible, mainly those around the dense region are expected to be of value in real life. Despite this the points near the steep side of the trajectory are still added to do a full analysis of the front. On the other side, the most shallow trajectories are also important as the analysis in Subsection 2.3 showed the fifth node angle to be limiting, especially for these trajectories. It is therefore expected that this side of the front has the most room for improvement.

4.2 Local Refinement

In this section, a local refinement will be done around the points selected in Subsection 4.1. This will be done by means of both a Monte Carlo method and a local optimizer from the Pygmo library.

At first, the full Monte-Carlo analysis is done by a variation of 10% around the selected points. For the last three points the variation has been scaled down to 5%, 1%, and 0.1% respectively, as all node angles are quite large, and taking 10% of this would have resulted in an overly large variation, where it would no longer be a local optimization. The analysis is performed using a total of 256 points, which has previously been determined sufficient.[1] The results of this analysis are shown in Figure 42 up to Figure 49, where initially the old front is plotted with all the newly found data points. In the enlarged boxes, some details on the shift of the old Pareto front (in black) and the new Pareto front (in red) have been shown around the initial point (black marker). It can be seen that at every point some slight improvements have been achieved, albeit marginal. This means that the Pareto front found in Subsection 2.2 was not yet the optimal one. It can also be seen that once the original point moves more towards the minimization of objective 2 (Figures 45, 46, 47, and 49) there appear to be some improvements far from the original point, predominantly in the bottom right region. This is because of the previously discussed limitation on the fifth node angle, which is exceeded by the Monte-Carlo analysis. Now even more shallow trajectories can be achieved, which in turn causes better circularization and so less ΔV is required. This results in a relatively larger Pareto front movement around the minimization of objective 2 than elsewhere.

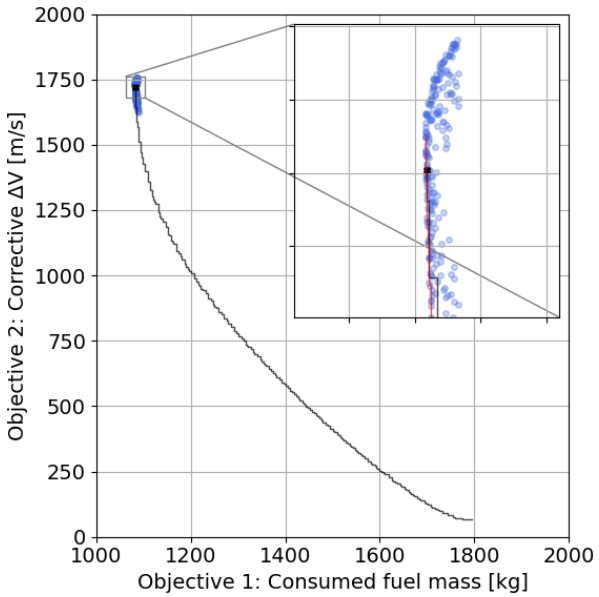


Figure 42: Monte-Carlo analysis around point 1, parameters varied by 10%

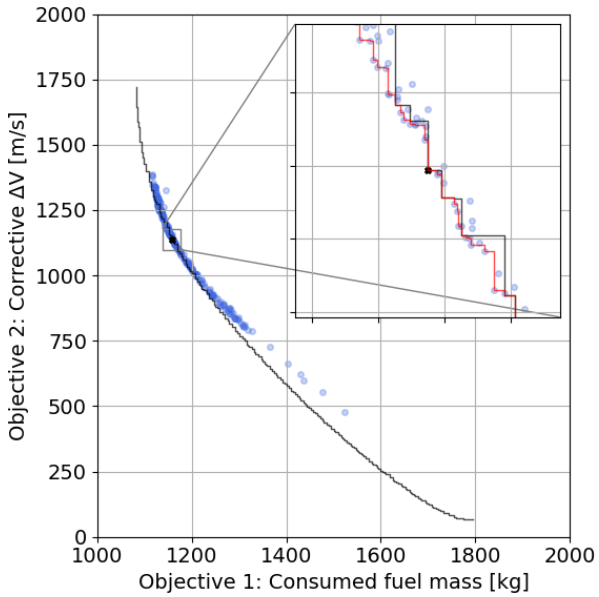


Figure 43: Monte-Carlo analysis around point 2, parameters varied by 10%

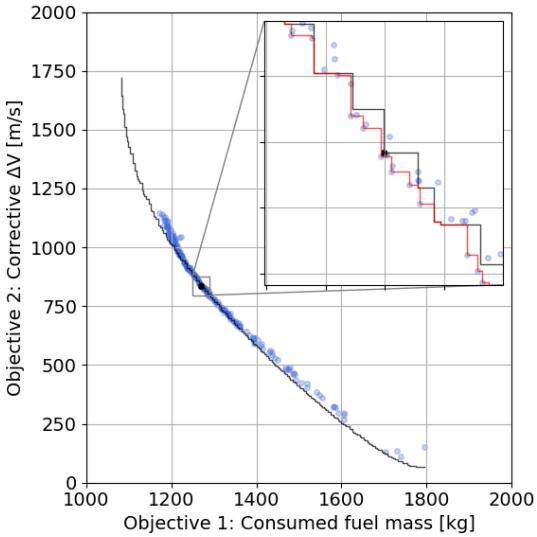


Figure 44: Monte-Carlo analysis around point 3, parameters varied by 10%

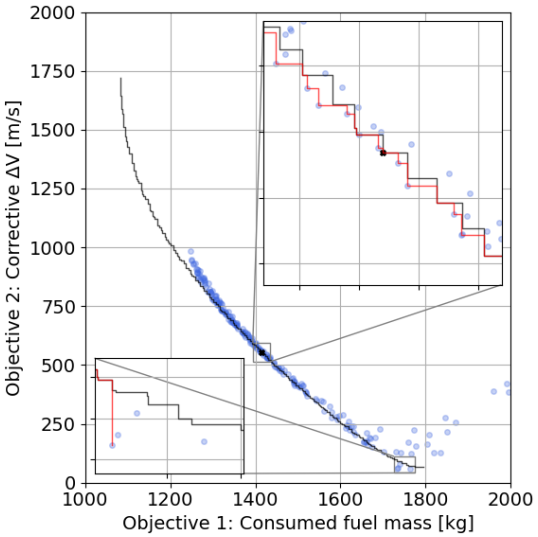


Figure 45: Monte-Carlo analysis around point 4, parameters varied by 10%

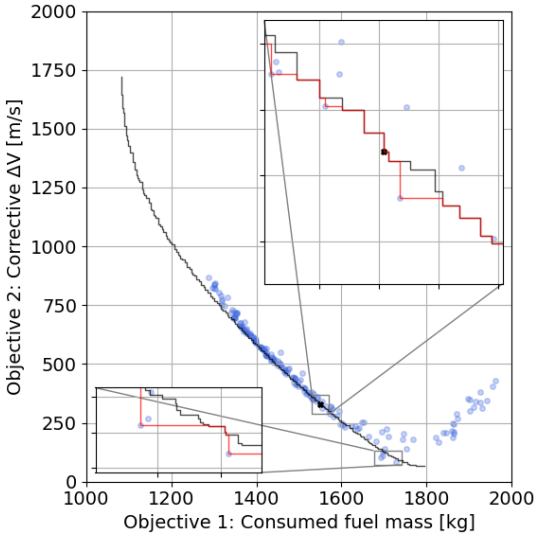


Figure 46: Monte-Carlo analysis around point 5, parameters varied by 10%

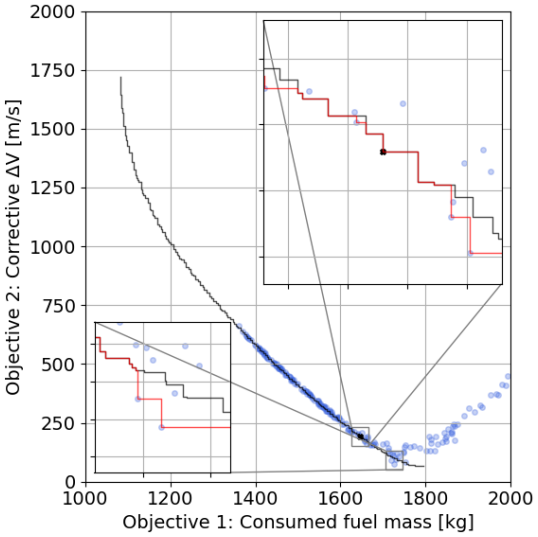


Figure 47: Monte-Carlo analysis around point 6, parameters varied by 5%

Next to the Monte Carlo analysis, a local optimizer will be used to further study the Parteo front. The chosen optimizer is the L-BFGS-B optimizer, as this is commonly used in other practices [19]. All local optimizers implemented in Pygmo have been tested against solving a Pygmo-implemented Rosenbrock function and the L-BFGS-B optimizer showed to reduce the error of 1e-11 m in only a few evaluations. This made it the best optimizer and therefore it will be used in this analysis. It is a limited memory Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm, which makes use of an inverse Hessian matrix to guide the algorithm to a local optimum [20]. Since all local objectives are single objective optimizers the objectives have been combined in this instance. Meaning that the second objective is converted from ΔV to fuel mass and appended to the fuel mass used for ascent according to Equation 2. Which is essentially a conversion of the ΔV to fuel mass by use of the Tsiolkovsky rocket equation [21].

$$M_{f,tot} = M_{f,obj1} + M_{f,obj2} = M_{f,obj1} + \left(M_{final} - \frac{M_{final}}{e^{\frac{\Delta V_{obj2}}{I_{sp} * g}}} \right) \tag{2}$$

The analysis has been performed using a total of 48 individuals for three different seeds (31,42,98), similar to the analysis in Subsection 2.1. The different seeds is to check if convergence occurs towards the same point for all seeds. The results of the analysis are plotted in Figure 50 to Figure 57, where not only the optima but all considered points by the optimizer are shown. The reason for this is that initial analysis has shown that a lot of points allow for improvement of the Pareto front and they therefore are useful in the analysis. Overall all seeds seem to behave quite similarly, with the exception of

seed 31 in Figure 52 where the blue seed completely moves away from the others. The blue seed however does manage the largest Pareto front shift of all and gets Objective 1 very close to 0, meaning the orbit is near circular.

In general, more improvement of the Pareto front is found for the local optimization method than for Monte-Carlo, yet it has proven to be less efficient at improving the region near the minimum of the second Objective. The reason for this is that the local optimizer is a single objective one and the analysis in Subsection 2.3 has shown that this lies more towards points 2 and 3, therefore it does not find as many optima in the lower regime as the (pseudo) random Monte-Carlo analysis.

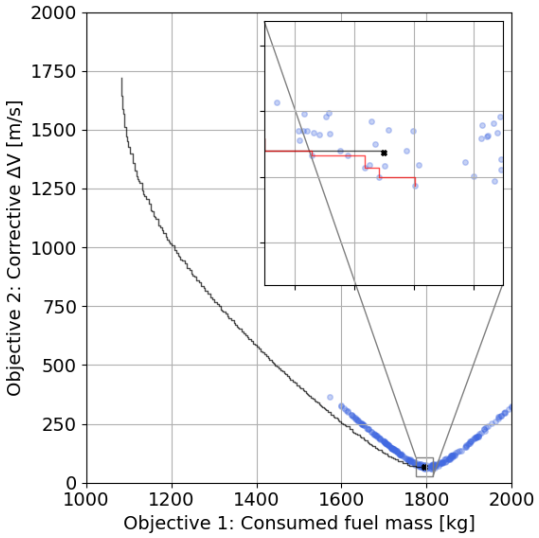


Figure 48: Monte-Carlo analysis around point 7, parameters varied by 1%

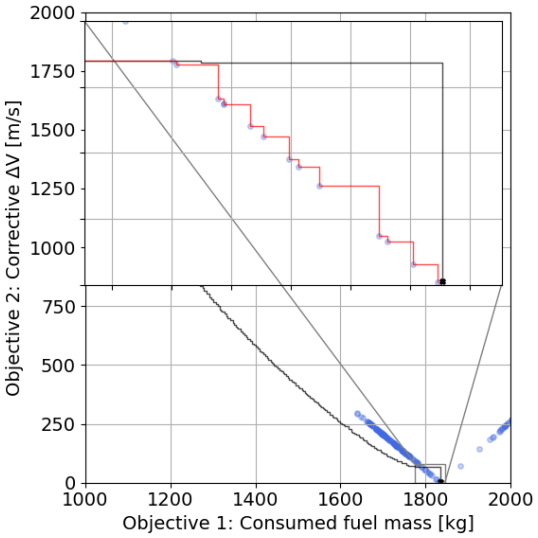


Figure 49: Monte-Carlo analysis around point 8, parameters varied by 10%

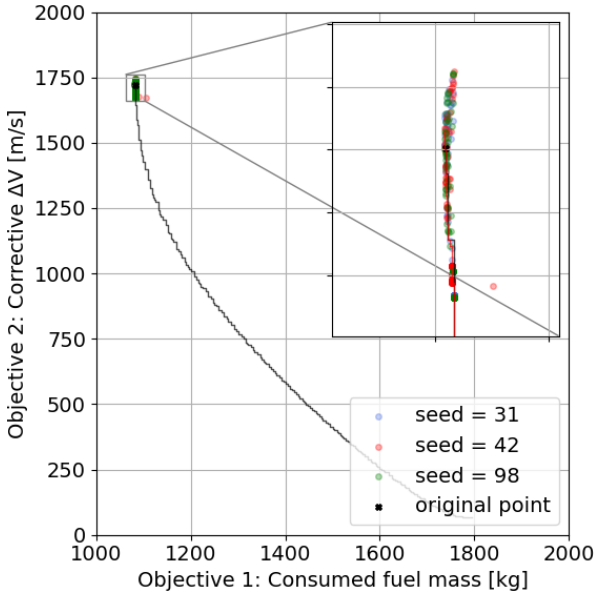


Figure 50: L-BFGS-B local optimization analysis using three different seeds around point 1, parameter space of 5% around the nominal values

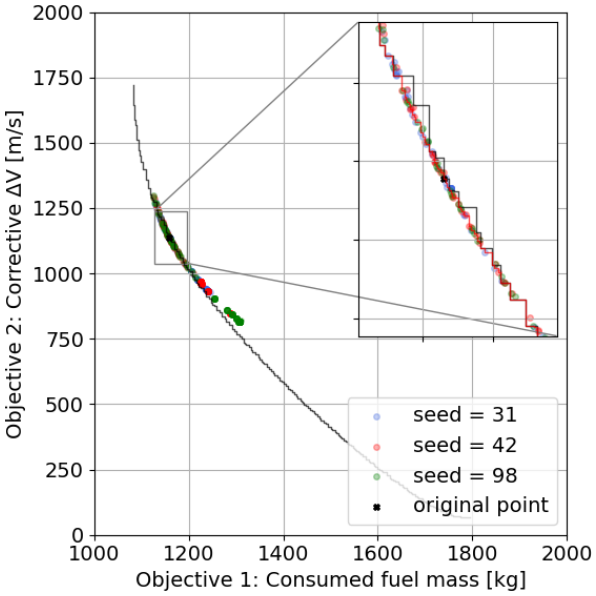


Figure 51: L-BFGS-B local optimization analysis using three different seeds around point 2, parameter space of 5% around the nominal values

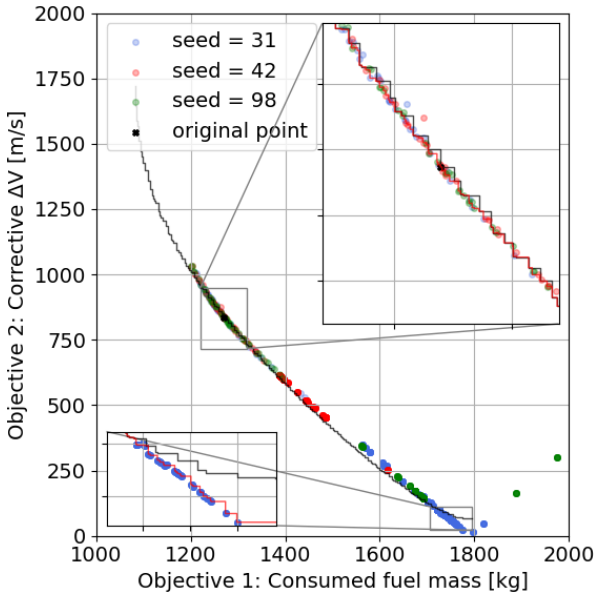


Figure 52: L-BFGS-B local optimization analysis using three different seeds around point 3, parameter space of 5% around the nominal values

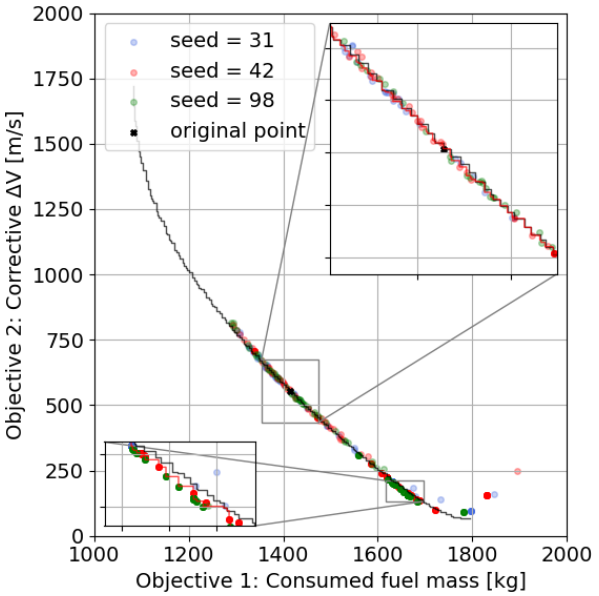


Figure 53: L-BFGS-B local optimization analysis using three different seeds around point 4, parameter space of 5% around the nominal values

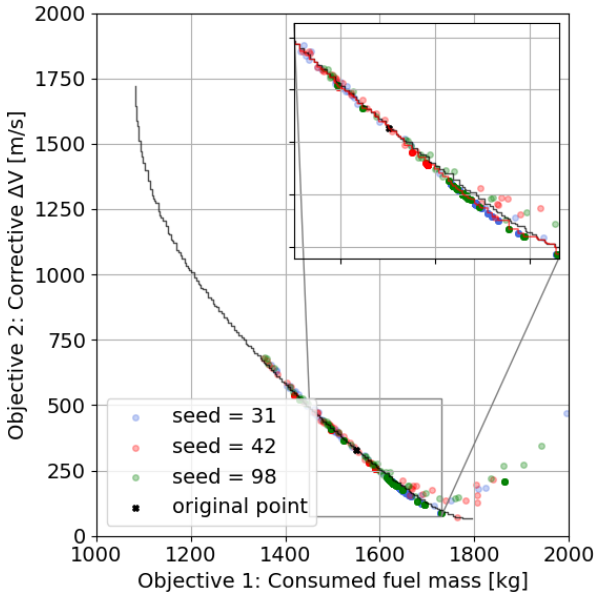


Figure 54: L-BFGS-B local optimization analysis using three different seeds around point 5, parameter space of 5% around the nominal values

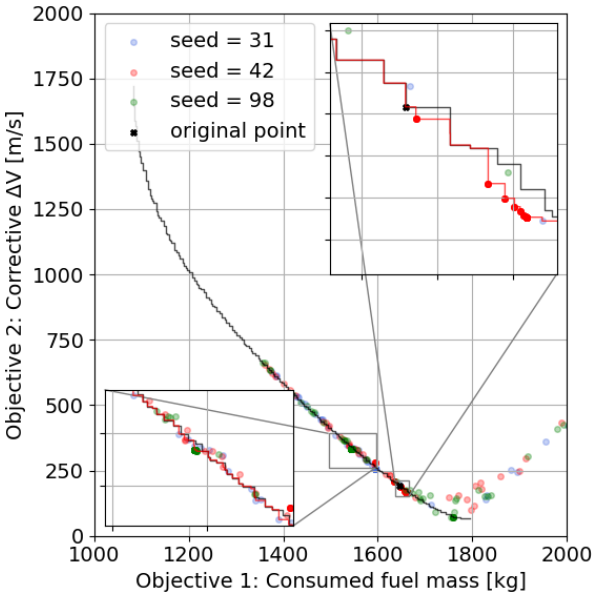


Figure 55: L-BFGS-B local optimization analysis using three different seeds around point 6, parameter space of 5% around the nominal values

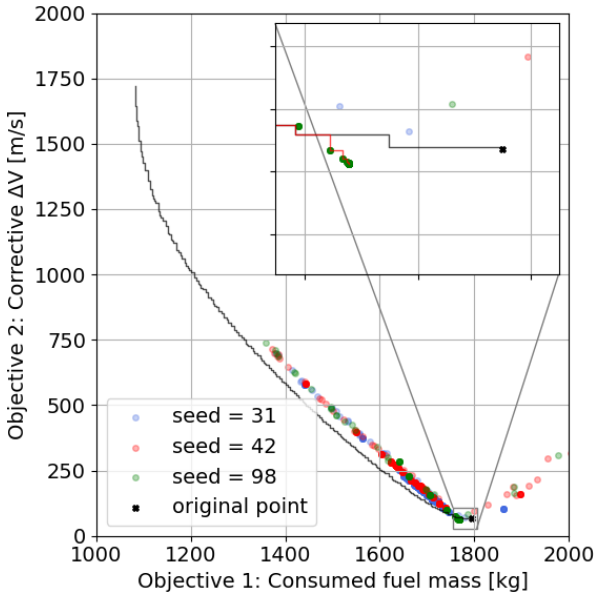


Figure 56: L-BFGS-B local optimization analysis using three different seeds around point 7, parameter space of 5% around the nominal values

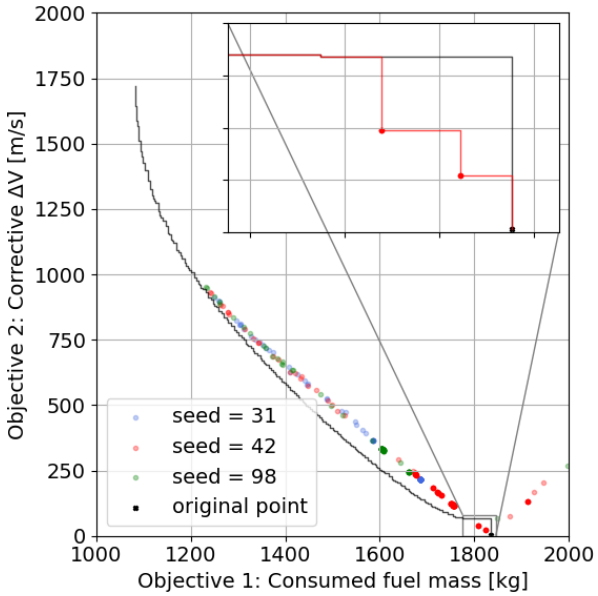


Figure 57: L-BFGS-B local optimization analysis using three different seeds around point 8, parameter space of 5% around the nominal values

Both analyses have been shown to marginally improve the Pareto front that was established in Subsection 2.2. Combining the results of both methods for all the points results in the new Pareto front shown in Figure 58. This plot confirms that the front only improved slightly. At the end where objective 2 is minimized, the improvements are greatest, this is due to the larger fifth node angles than previously were possible.

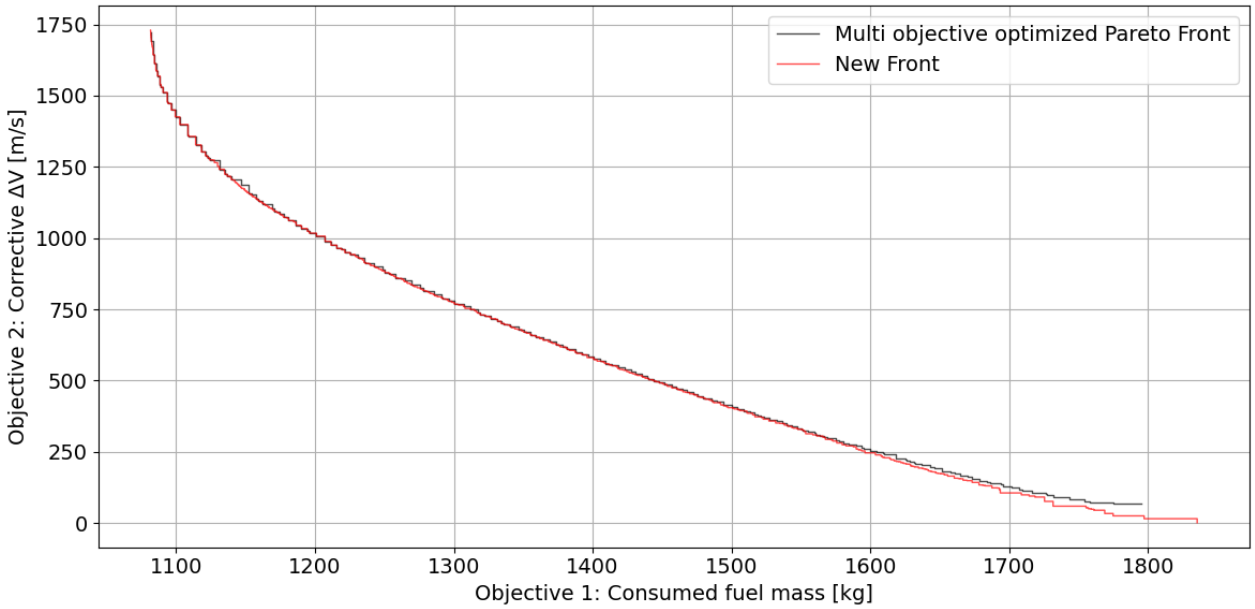


Figure 58: New Pareto front, found by Monte-Carlo and local optimization, compared to the old front.

4.3 Sensitivity Analysis

In this section a sensitivity analysis for each of the 8 points in Table 15 will be done. It will be quantified how much the primary source of model simplification and the main source of uncertainty will influence the constraints and objectives values.

From the second assignments of all the authors ([22],[23],[24],[25]), it can be deducted that the main differences in the models (accelerations and environment) was the amount of spherical harmonics terms used for the moon's gravity field. The decision on how many terms to use was therefore also the most debatable one. Therefore this will be the choice of model simplification on which a sensitivity analysis will be done.

Furthermore in the second assignments of all authors, an uncertainty analysis has been done in which either uncertainties in the initial state or environment and acceleration models have been analysed. Both Doorn [24] and Voskuilen [25] analysed the uncertainty in the initial state and Veithen [23] and Calliess [22] analysed uncertainties in the environment and acceleration models. Comparing all results, it can be deducted that the initial velocity uncertainty had the largest influence on the final position. Therefore this is the uncertainty of choice for which the influence on the objectives and constraints values will be analysed with a sensitivity analysis.

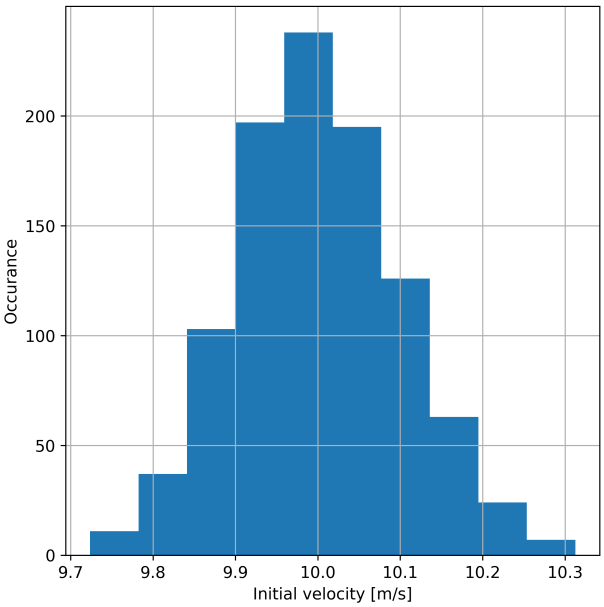


Figure 59: Generated initial velocity perturbations (1000 in total)

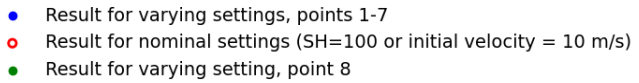


Figure 60: Legend for plots in this section

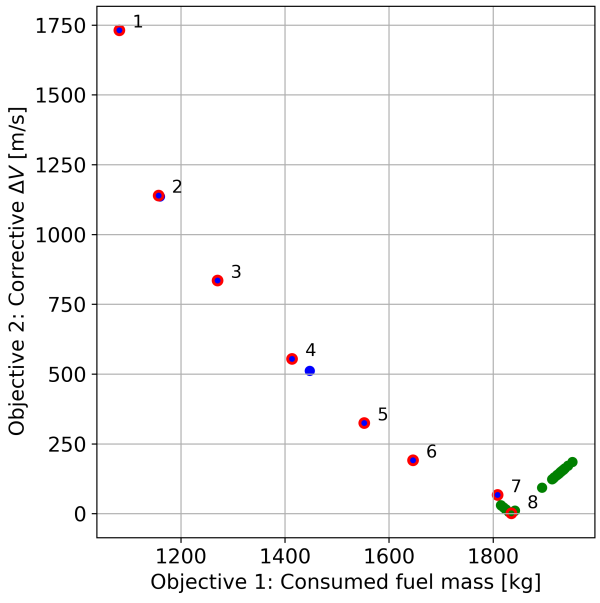


Figure 61: Objective values for varying the amount of SH terms

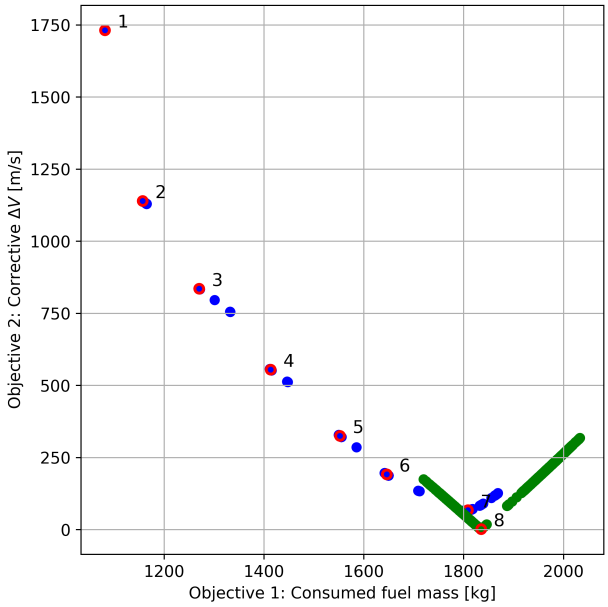


Figure 62: Objective values for initial velocity uncertainties

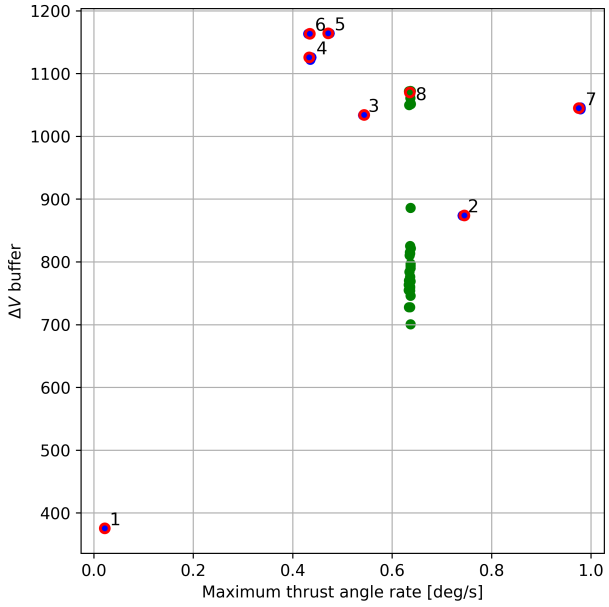


Figure 63: Constraint values for varying the amount of SH terms

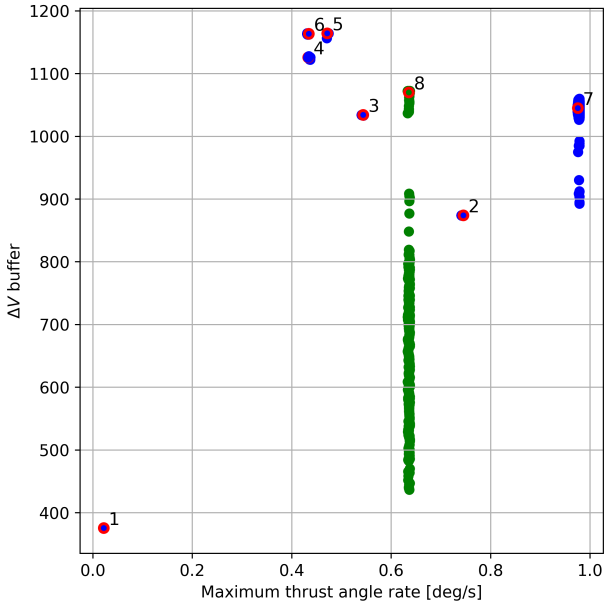


Figure 64: Constraint values for initial velocity uncertainties

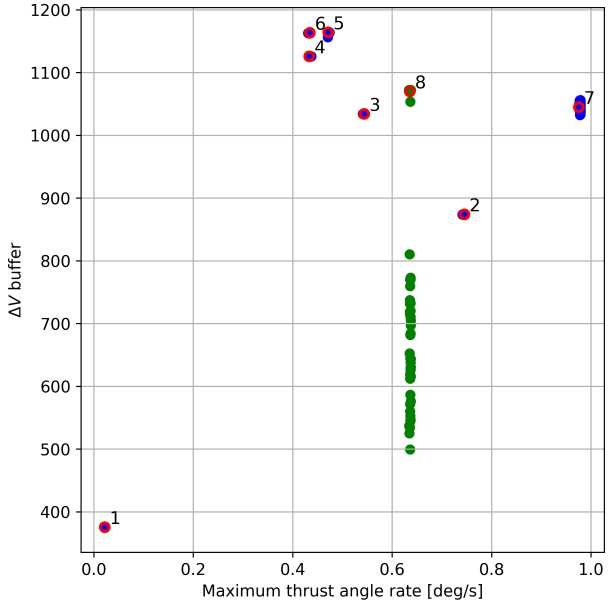


Figure 65: Constraint values for initial velocity uncertainties - 100 samples

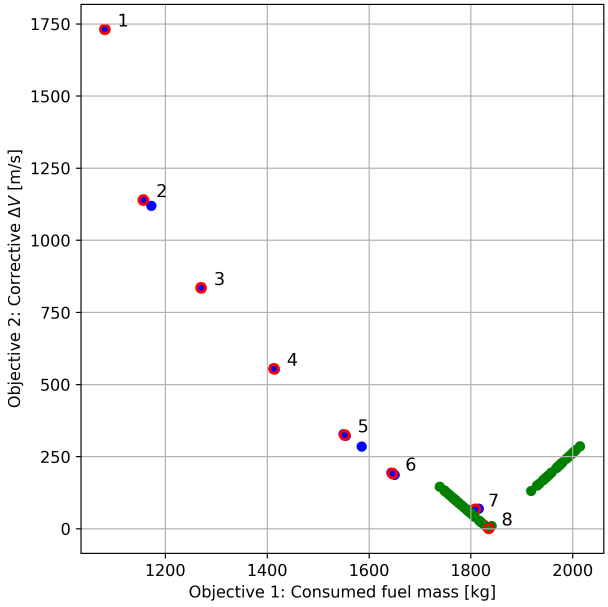


Figure 66: Objective values for initial velocity uncertainties - 100 samples

The chosen amount of SH terms for assignment 3 and this assignment is 100. Therefore, for all 8 points (combinations of decision variables), the amount of spherical harmonic terms for the moon's gravity field have been varied from 60 to 140 in steps of 2 (62, 64, ... 140). In Figure 61 and Figure 63, the spread in objectives and constraints values can be observed. Blue dots correspond to results obtained by varying SH terms (for point 8 those have been made green), red circles correspond to the values for the chosen setting of SH=100. For point 1 to 7, the spread is negligible. For point 8, a significant spread can be observed in the ΔV buffer constraint, as well as in the objective space, indicating this point is sensitive to changes in the amount of SH terms included in the moon's gravity field.

For the generation of initial velocity uncertainties, a Gaussian distribution with a standard deviation of 0.1 has been used. The 1000 generated initial velocity perturbations are shown in Figure 59. A Gaussian distribution has been chosen because it is most likely that the initial velocity is near the nominal 10 m/s. A standard deviation of 0.1 produces initial velocity perturbations that are considered reasonable. In Figure 62 and Figure 64, it can be observed that again for points 1 to 7, the spread is negligible (although a small spread can be observed for the ΔV buffer constraint values of point 7), but for point 8 a significant spread can be observed in the ΔV buffer constraint, as well as in the objective space, indicating that this point is sensitive against initial velocity perturbations. What is also interesting to note is that the spread seems to have

a linear relationship in the objective space, which actually also holds for the model simplifications sensitivity analysis. To prove that 1000 generated initial velocity perturbations are a sufficient amount, results for 100 generated samples have been shown in Figure 65 and Figure 66. These results show that the ranges of the objectives and constraint values are similar. The ranges are just a bit more dense with 1000 samples.

So to conclude: points 1 to 7 of the Pareto-optimal solutions can be considered sufficiently robust, since they have shown to not be sensitive to the primary source of model simplification and uncertainty. Point 8 however, cannot be considered a robust Pareto-optimal solution, since it is very sensitive. Taking different amount of SH terms for the moon's gravity field or perturbing the initial velocity results in very different objectives and constraints values.

5 Conclusions and Recommendations

After a 10 week journey of propagating and optimizing Lunar Ascent trajectories, the time has come to finalize the job and draw conclusions and make recommendations on the optimal trajectory. The group part of the course started of in assignment 3 by combining all individual integrator, propagator, environments, accelerations, decision variables, objectives and constraints choices into one problem definition which would be used in the further analysis. After this the design space was explored with the help of various analysis like Monte Carlo, Response surfaces and full factorial design. This gave the authors a better understanding of the problem. Assignment 3 ended with some tweaking of the decision variable ranges for the thrust magnitude and the node angles, which then provided the starting point for assignment 4. There, first the implementation of the constraints and termination and convergence criteria were established. Then numerous Multi-Objective algorithms available in Pygmo were tested and a final most suitable Multi-Objective optimisation algorithm was chosen in the form of the MOEA/D. This algorithm was then tuned for better performance and with this increased performance, Pareto-optimal solutions were found. Furthermore a Single-Objective optimisation was done in order to analyse the completeness of the Pareto front obtained with the Multi-Objective Global Optimisation. The minimization of the corrective Delta-V needed for orbit circulisation was chosen as the most important objective. After multiple optimisers were analysed, the DE1220 algorithm was selected as the most suitable optimiser. This optimiser was then tuned for even better performance, after which the global optimum (or at least a value close to it) of the single-objective problem was found. Then last but not least the robustness and completeness of the Pareto Front found by the Multi-Objective Global Optimisation, including the newly found point in the single-objective optimisation, was analysed. With local refinement, the Pareto Front was marginally improved and finally a sensitivity analysis of 8 points in the Pareto Front showed that all but the point found in the single-objective optimisation are robust Pareto-optimal solutions.

In Section 2, a Pareto front was established and in Section 4, this Pareto front was marginally improved. The Pareto front provides numerous Pareto optimal solutions. Since there are 2 objectives to be optimised, the solutions can differ a lot. Solutions can provide a very low fuel mass consumption (objective 1 minimized) and still a relatively high corrective ΔV or vice versa or somewhere in between. In theory, all kind of solutions are equally good. However, in this assignment the corrective ΔV manoeuvre calculation involves the simplification of impulsive thrust. For this reason, solutions having a high corrective ΔV could lead to solutions that are in practice not very feasible. For this reason solutions that minimize the corrective ΔV are preferred as they lead to more realistic solutions. Therefore, by looking at Figure 22 in Subsection 2.3, a solution near point 1 would be preferred. What can be observed in Figure 22, is that for the rightmost solutions in the Pareto front near point 1 in the graph, the corrective ΔV is relatively similar. Moving more to the left in the front, at some point near consumed fuel mass values of 1750, the corrective ΔV values start to increase. At this turning point, solutions have the lowest possible corrective ΔV of the front and also the lowest consumed fuel mass. The final recommendation for the optimal trajectory would thus be to pick a solution in this part of the front. For clarity, this preferred Pareto front region has been indicated in Figure 67.

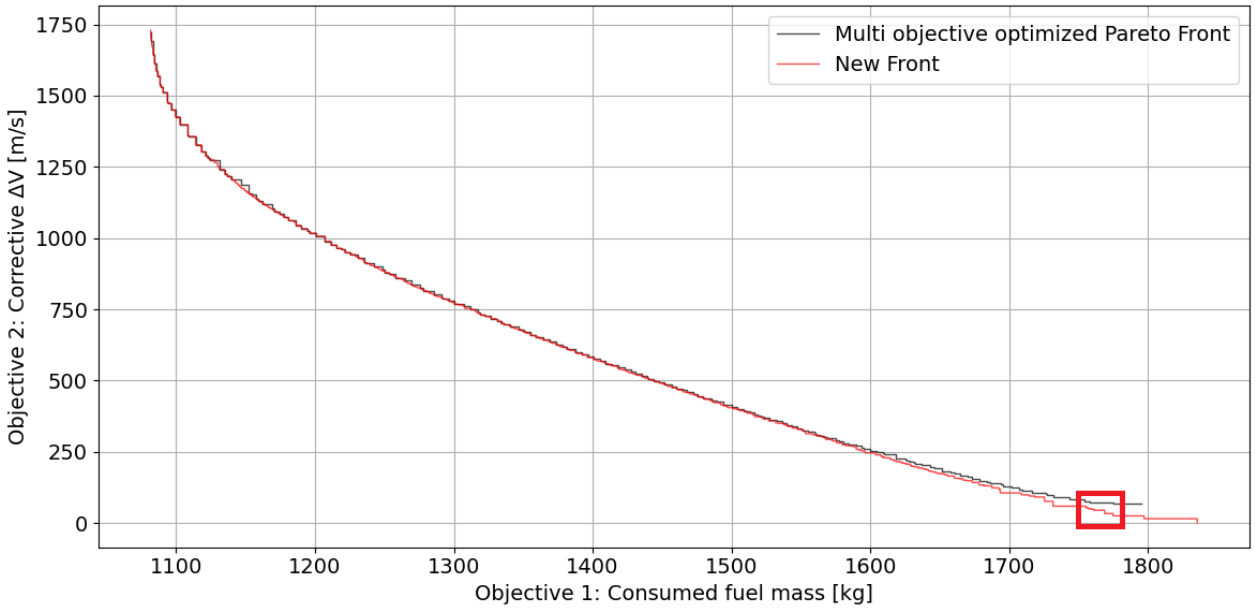


Figure 67: Preferred Pareto front region for optimal trajectory

6 Breakdown of contributions

Table 16: Breakdown of contributions of team members.

Section	Implementing (coding and post-processing)	Analysing (Discussion, thinking, etc.)	Reporting	Hours spent
1.1 Problem Definition	-	Everyone	Mitchell	Mitchell: 1 hour
1.2 Constraints Implementation	-	Draft: Mitchell, Review: Everyone	Mitchell	Mitchell: 8 hours
1.3 Termination and Convergence Criteria	-	Draft: Mitchell, Review: Everyone	Mitchell	Mitchell: 3 hours
2.1 Multi-Objective Algorithm Selection	Daniel	Daniel	Daniel	12 hours
2.2 Tuning the MOEA/D Algorithm	Daniel	Daniel	Daniel	20 hours
2.3 Behaviour and Feasibility of the Pareto-optimal Solutions	Daniel & Jesse	Daniel & Jesse	Daniel & Jesse	Jesse: 18 hours Daniel: 18 hours
3.1 Single-Objective Selection	-	Lorenz	Lorenz	1
3.2 Optimiser Selection	Lorenz	Lorenz	Lorenz	15
3.3 Tuning and Robustness of Selected Optimiser	Lorenz	Lorenz	Lorenz	25
3.4 Completeness of Multi-Objective Pareto Front	Lorenz	Lorenz	Lorenz	4
4.1 Points Selection	-	Jesse	Jesse	2 hours
4.2 Local Refinement	Jesse	Jesse	Jesse	22 hours
4.3 Sensitivity Analysis	Mitchell	Mitchell	Mitchell	Mitchell: 14 hours
5 Conclusions and Recommendations	-	Mitchell	Mitchell	Mitchell: 3 hours

References

[1] L. Veithen, J. Voskuilen, D. Calliess, and M. van Doorn, "Propagation and optimisation - assignment 3," Delft University of Technology, Tech. Rep., 2023.

[2] F. Biscani and D. Izzo, "A parallel global multiobjective framework for optimization: Pagmo," *Journal of Open Source Software*, vol. 5, no. 53, p. 2338, 2020. DOI: 10.21105/joss.02338. [Online]. Available: https://doi.org/10.21105/joss.02338.

[3] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007. DOI: 10.1109/TEVC.2007.892759.

[4] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 284–302, 2009. DOI: 10.1109/TEVC.2008.925798.

[5] D. Izzo, D. Hennes, and A. Riccardi, "Constraint handling and multi-objective methods for the evolution of interplanetary trajectories," *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 4, pp. 792–800, 2015. DOI: 10.2514/1.G000619. eprint: https://doi.org/10.2514/1.G000619. [Online]. Available: https://doi.org/10.2514/1.G000619.

[6] I. Costa-Carrapiço, R. Raslan, and J. N. González, "A systematic review of genetic algorithm-based multi-objective optimisation for building retrofitting strategies towards energy efficiency," *Energy and Buildings*, vol. 210, p. 109 690, 2020, ISSN: 0378-7788. DOI: https://doi.org/10.1016/j.enbuild.2019.109690. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378778819319462.

[7] J. Labroquère, A. Héritier, A. Riccardi, and D. Izzo, "Evolutionary constrained optimization for a jupiter capture," in *Parallel Problem Solving from Nature – PPSN XIII*, T. Bartz-Beielstein, J. Branke, B. Filipič, and J. Smith, Eds., Cham: Springer International Publishing, 2014, pp. 262–271, ISBN: 978-3-319-10762-2.

[8] M. Vasile, "Multi-objective optimal control: A direct approach," *Satellite Dynamics and Space Missions*, pp. 257–289, 2019.

[9] R. Storn and K. Price, "Differential evolution –a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997. DOI: 10.1023/A:1008202821328. [Online]. Available: https://doi.org/10.1023/A:1008202821328.

[10] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006. DOI: 10.1109/TEVC.2006.872133.

[11] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Differential evolution with multiple strategies for solving cec2011 real-world numerical optimization problems," in *2011 IEEE Congress of Evolutionary Computation (CEC)*, 2011, pp. 1041–1048. DOI: 10.1109/CEC.2011.5949732.

[12] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014, ISSN: 0965-9978. DOI: https://doi.org/10.1016/j.advengsoft.2013.12.007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0965997813001853.

[13] P. Niu, S. Niu, N. liu, and L. Chang, "The defect of the grey wolf optimization algorithm and its verification method," *Knowledge-Based Systems*, vol. 171, pp. 37–43, 2019, ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2019.01.018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705119300188.

[14] M. Mahdavi, M. Fesanghary, and E. Damangir, "An improved harmony search algorithm for solving optimization problems," *Applied Mathematics and Computation*, vol. 188, no. 2, pp. 1567–1579, 2007, ISSN: 0096-3003. DOI: https://doi.org/10.1016/j.amc.2006.11.033. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0096300306015098.

[15] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm Intelligence*, vol. 1, no. 1, pp. 33–57, 2007. DOI: 10.1007/s11721-007-0002-0. [Online]. Available: https://doi.org/10.1007/s11721-007-0002-0.

[16] P. S. Oliveto, J. He, and X. Yao, "Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results," *International Journal of Automation and Computing*, vol. 4, no. 3, pp. 281–293, 2007. DOI: 10.1007/s11633-007-0281-3. [Online]. Available: https://doi.org/10.1007/s11633-007-0281-3.

[17] N. Hansen, "The cma evolution strategy: A comparing review," in *Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms*, J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 75–102, ISBN: 978-3-540-32494-2. DOI: 10.1007/3-540-32494-1_4. [Online]. Available: https://doi.org/10.1007/3-540-32494-1_4.

[18] T. Glasmachers, T. Schaul, S. Yi, D. Wierstra, and J. Schmidhuber, "Exponential natural evolution strategies," in *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '10, Portland, Oregon, USA: Association for Computing Machinery, 2010, pp. 393–400, ISBN: 9781450300728. DOI: 10.1145/1830483.1830557. [Online]. Available: https://doi.org/10.1145/1830483.1830557.

[19] R. Malouf, "A comparison of algorithms for maximum entropy parameter estimation," in *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*, 2002.

[20] J. Nocedal, "Updating quasi-newton matrices with limited storage," *Mathematics of computation*, vol. 35, no. 151, pp. 773–782, 1980.

[21] V. Dvornychenko, "The generalized tsiolkovsky equation," in *NASA Conference Publication*, vol. 3102, 1990, p. 449.

[22] D. Calliess, "Propagation and optimisation - assignment 2," Delft University of Technology, Tech. Rep., 2023.

[23] L. Veithen, "Propagation and optimisation - assignment 2," Delft University of Technology, Tech. Rep., 2023.

[24] M. van Doorn, "Propagation and optimisation - assignment 2," Delft University of Technology, Tech. Rep., 2023.

[25] J. Voskuilen, "Propagation and optimisation - assignment 2," Delft University of Technology, Tech. Rep., 2023.