Software Lab 2021
Chair of Modelling and Simulation
TUM School of Engineering and Design

Three-Dimensional Box Topology Optimization with Constrained Space

Authors: Ahmed Dawod, Talhah Ansari, Luis Dario Rueda[1]

Supervisors: Giovanni Filomeno, Moustafa Alsayed-Ahmad

[1] M.Sc. students in Computational Mechanics
Technical University of Munich

ABSTRACT

An efficient algorithm based on population-based optimization (Evolutionary Algorithm) for positioning a transmission assembly comprised of a gearbox and associated electrical components in a constrained assembly space was developed. The algorithm optimized the positioning of the components around the gearbox avoiding any collisions while minimizing the bounded volume and the cable lengths. The results showed that after 20 generations fairly good design solutions were obtained for a population size of 500, and as the algorithm progressed, more feasible solutions were obtained. These results were obtained for a specific gearbox and assembly space with relatively low complexity. The number of generations and suitable population size will vary with the variation of the complexity of the components.

Keywords: population-based Optimization, Evolutionary algorithm, genetic algorithm, assembly space

1 INTRODUCTION

1.1 Motivation

The design of the transmission assembly consists of different components interacting with each other including the gearbox assembly and the electrical components, which work in close proximity. The assembly space (also interchangeably referred to as Bauraum) is constrained and irregular, which usually makes the placement of the gearbox and its associated electrical components inside this space a challenging task. Fitting check is usually done manually for each transmission assembly (Gearbox + Electrical components) using CAD solutions which are inefficient and time consuming. Consequently, it was necessary to develop an algorithm that can efficiently and quickly check the fit of a transmission assembly inside an assembly space and find an optimized arrangement if it exists. Achieving this will result in minimized volume, weight, and electrical losses, which are all favorable improvements for any automotive manufacturer. It will also enable an automotive manufacturer to test the fitness of multiple transmission assemblies for a given assembly space automatically and quickly which allows for an optimized selection between all possible options along with significant time savings.

1.2 Objectives

The objective of this study was to develop an efficient algorithm that optimizes the placement of the gearbox assembly and its associated electrical components inside the assembly space. Optimizing this placement includes minimizing the distance between all components and minimizing the cable lengths connecting them to each other while ensuring physical constraints are not violated if possible. Physical constraints include collision/interference between different components, between components and the assembly space, and interference between the connecting cables or between the connecting cables and any of the components.

1.3 Approach Followed and Software Used

Due to the irregular shape of the constrained assembly space, the placement of components inside is dealt as a packing optimization problem. Being of practical engineering application, it is evident that there isn't a single best solution but rather a set of best feasible solutions that have to be obtained. Some solutions might be better for manufacturing and assembly and hence may be finally chosen. Therefore, an algorithm that obtains a set of best feasible designs was developed. The implemented algorithm was developed from scratch in MATLAB. Some external functions were utilized, and credits are given to their developers and can be found in the code documentation of this study.

The algorithm receives the gearbox geometry along with its associated electrical components' geometries as an input. These geometries are provided as STL files which the code reads and obtains all necessary geometrical information from and prepares all the components for the optimization algorithm by positioning them appropriately in space. The optimization function then works to minimize the distance between the electrical components (IGBT, Control Board, Capacitor, EMI-filter) individually and place them close to the electric motor. The length of the cables connecting the components and the total bounded volume was also to be minimized.

1.4 Literature Review

The implemented optimization algorithm can be classified as an evolutionary algorithm. Evolutionary algorithms are a class of population-based optimization approaches based on biological evolution mechanisms such as selection, crossover and mutation. A generation of population consisting of a set of individuals/ designs undergo the above-mentioned processes to generate a new generation of designs. Figure 1 represents a flowchart that illustrates the steps involved in an evolutionary algorithm.

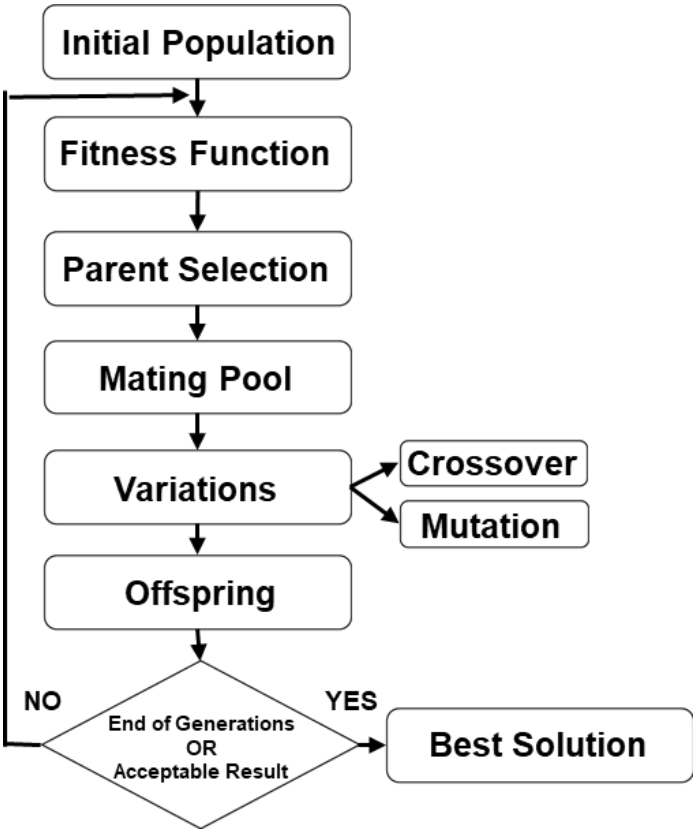


Figure 1: Evolutionary Algorithm flowchart

2 METHODOLOGY

In this section, the process through which the algorithm goes to obtain the optimal solutions is presented. The user is prompted to enter the population size and the number of generations. The algorithm can be divided into three main sections. The first section in comprised of a set of functions that are responsible for reading all the components and positioning them in an appropriate initial position in space. The second section of the code is mainly a population-based optimization algorithm that works to minimize the fitness function by minimizing the distance between all the components and minimizing the length of the cables connecting them to each other while ensuring physical constraints are not violated. The last section of the algorithm is responsible for visualizing the obtained results where it plots the gearbox and its associated electrical components and cable connections between them and presenting them as possible optimal placements inside the constrained space. A simplified flowchart of the implemented algorithm is presented in Figure 2.

The simplified structure of the code shown in Figure 2 can be extended to obtain a detailed one as shown in Figure 3. The detailed flow chart shows all the functions implemented and the objective of each function along with a description of the input parameters and return values of each function. The entire structure of the code will be discussed in detail in the following subsections.

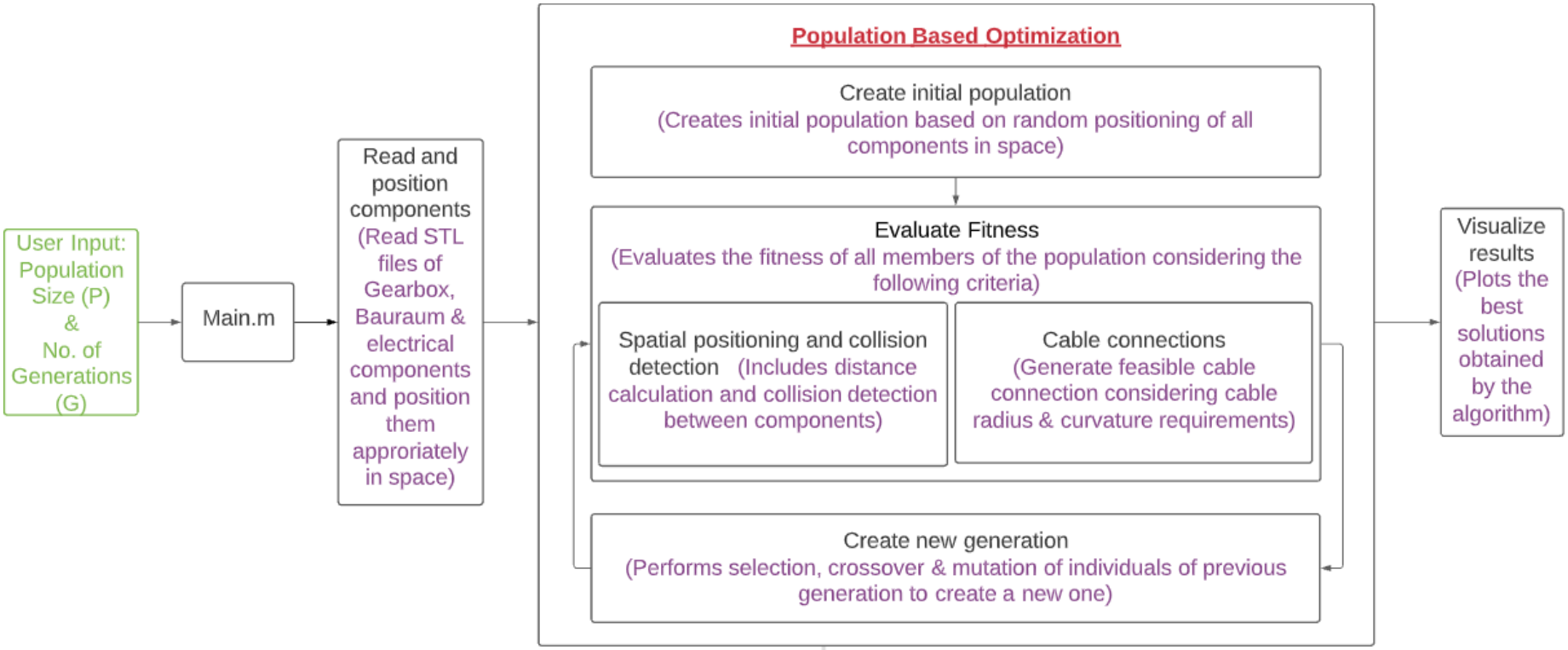


Figure 2: Brief Algorithm Structure

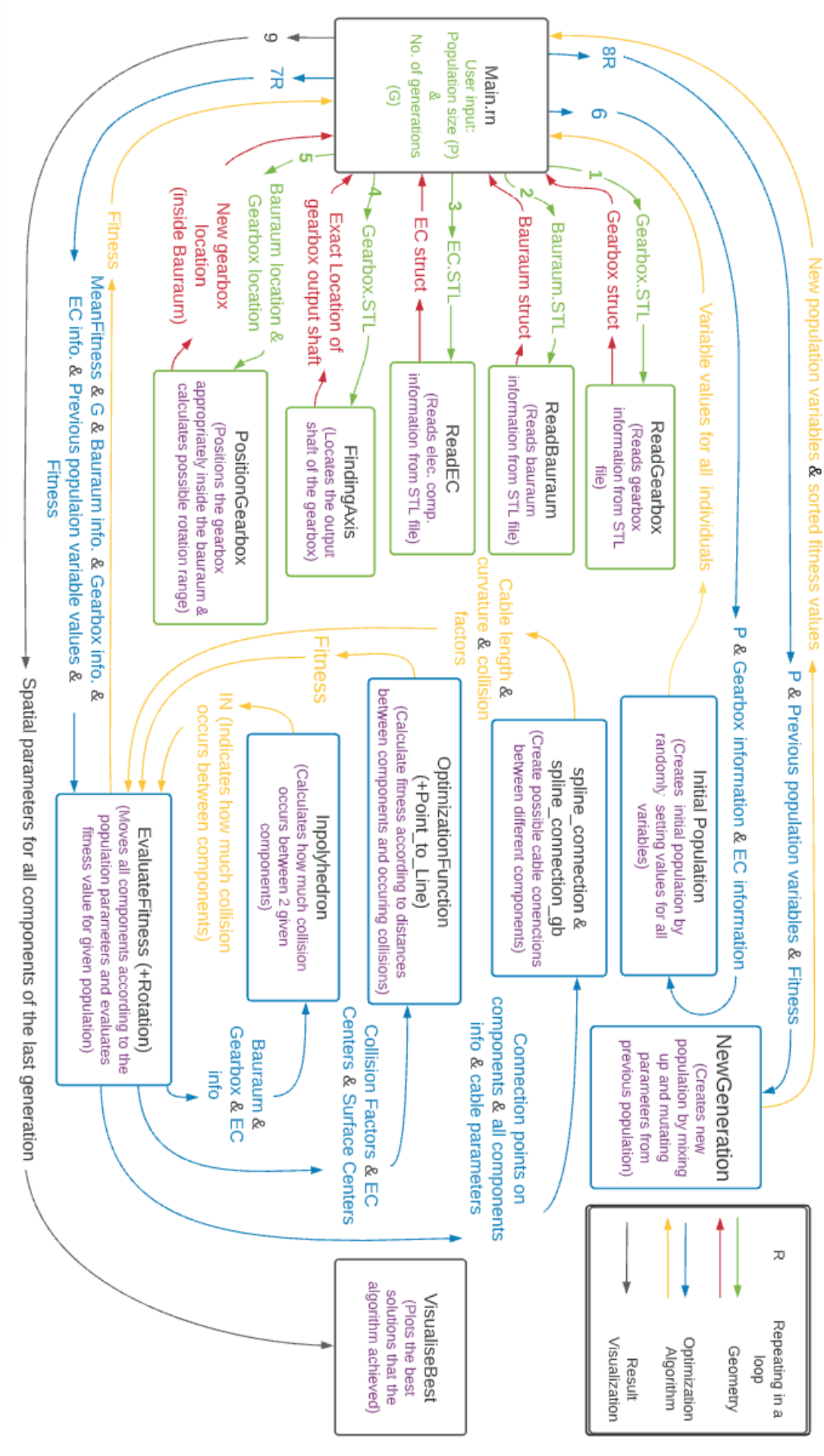


Figure 3: Detailed Algorithm structure

2.1 Reading and Positioning Components

After the user inputs the required number of generations and population size, the algorithm starts by reading and positioning all components. Firstly, the function (**ReadGearbox**) reads the gearbox's STL file. This function reads all the points comprising the gearbox from the STL file and returns a struct containing all the points and faces bounding the volume of the gearbox. The result of this can be visualized in Figure 4.

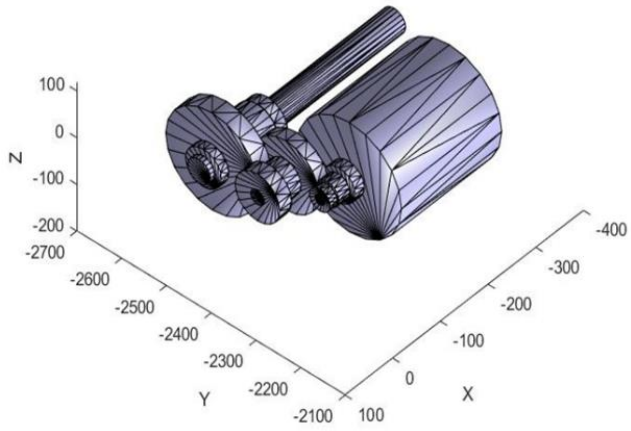


Figure 4: Visualized Gearbox from STL File

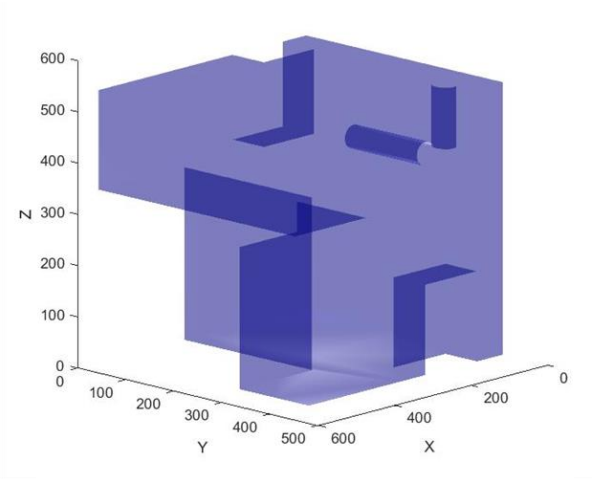


Figure 5: Visualized Bauraum from STL File

Another function (**ReadBauraum**) performs the same function to obtain the geometrical information of the assembly space (Bauraum) to obtain the result shown in Figure 5. The provided gearbox and assembly space models are very unlikely to be placed such that the gearbox is inside the assembly space. An example of this is shown in Figure 6. However, it is an essential early step for the output shaft of the gearbox to be aligned with the drive shaft of the vehicle, which is defined in the assembly space.

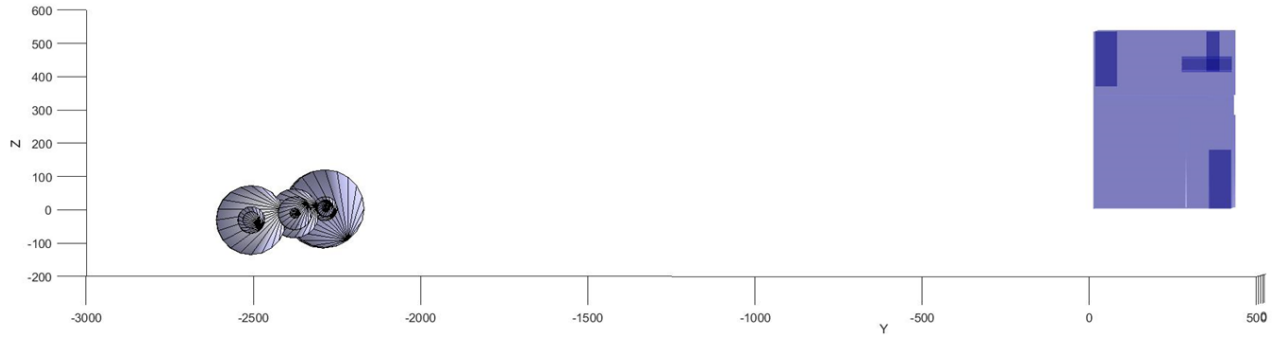


Figure 6: Gearbox and Bauram Geometries Visualized Together

In order to align the gearbox output shaft with the drive shaft, the function (**FindingAxis**) was developed to identify the location of the gearbox's output shaft. This function loops over all surfaces in the gearbox struct to identify the largest cylindrical shape which is in most cases the electric motor. Knowing the cylinder that represents the electric motor, the function then loops over all cylinders starting from the electric motor and identifies the cylinders that have other cylinders tangent to them which represent mating gears. By finding all tangencies, the function identifies the last cylinder in the series of tangencies which would represent the output shaft. The output shaft axis is then defined by a starting point and a direction vector. The obtained output shaft axis is shown in Figure 7. Knowing

the exact location of the gearbox output shaft and the drive shaft in the assembly space, the function (**PositionGearbox**) manipulates the position of the gearbox using rotation and translation processes to align both axes as shown in Figure 8. As an additional but very important step, the gearbox is rotated 360 degrees along the shaft axis and the collision between the gearbox and assembly space is measured. The result of this step is knowing the range of rotation the gearbox can make without going out of the bauraum. This is very useful in the optimization algorithm in order to not waste computation time generating and evaluating solutions that would always be wrong.

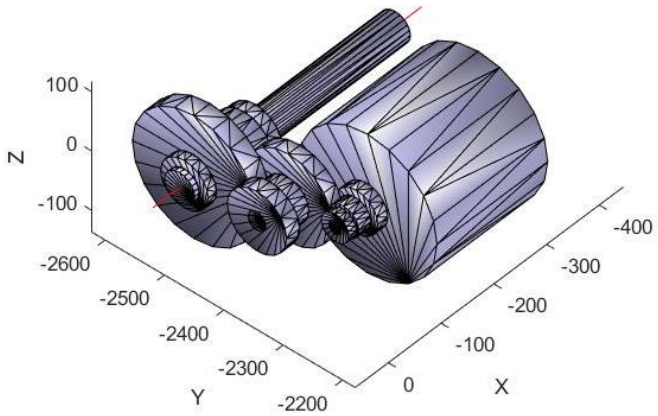


Figure 7: Identified Location of Output Shaft Axis

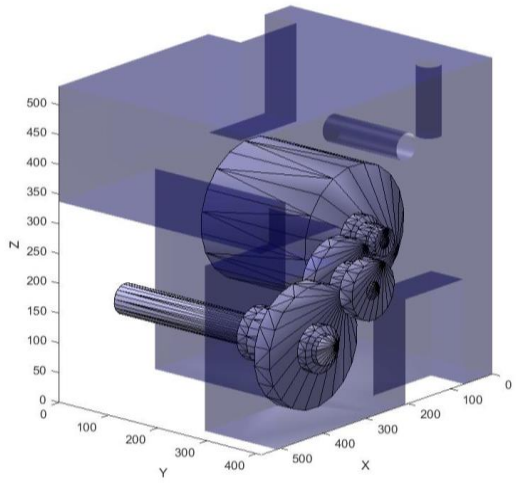


Figure 8: Gearbox Positioned inside Assembly Space

After the gearbox is successfully positioned inside the assembly space, the function (**ReadEC**) reads the STL files of the electrical components associated with the gearbox (IGBT, Control Board, Capacitor, EMI-filter) and places them at an initial position inside the assembly space as shown in Figure 9. At this point, the different dimensions of all components along with their location in space is known. This information is then sent to the optimization algorithm which optimizes their location in space.

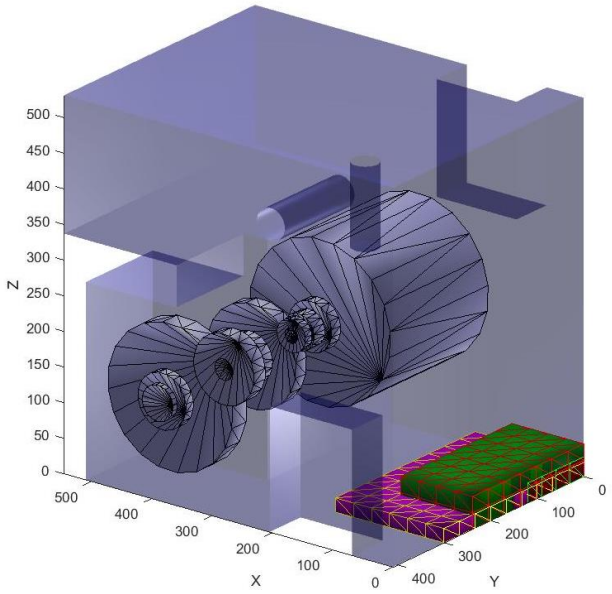


Figure 9: Gearbox and Associated Electrical Components Placed in Initial Position Inside Assembly Space

2.2 Optimization Algorithm

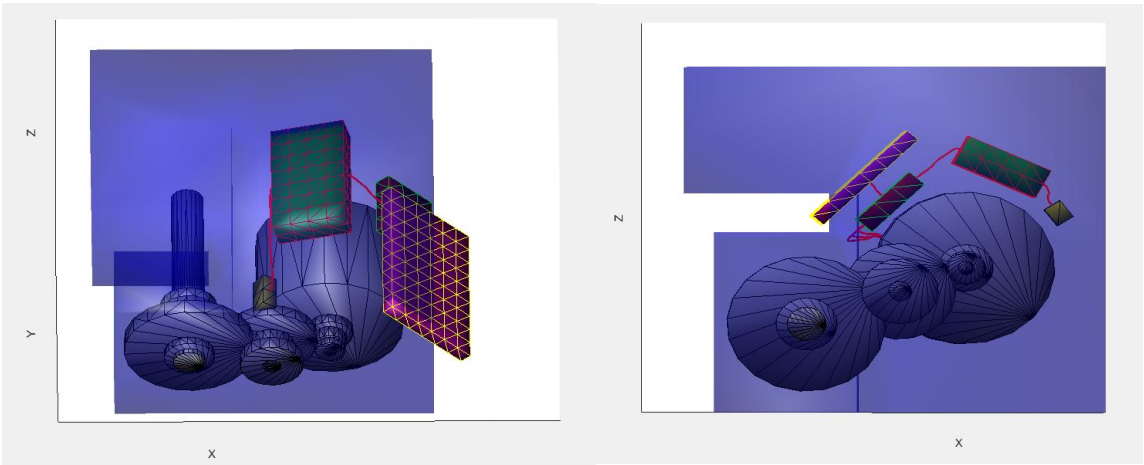
The optimization starts with a function (**InitialPopulation**) that generates an initial set of designs, whose values are randomly generated. The members of this initial population are called the first generation of solutions.

A very important consideration for the generation of this initial population is that, although its individuals are randomly generated, their values are defined by certain limits and relations between the dimensions of the gearbox, bauraum as well as the connectivity between components. For this implementation it was considered as a general assumption that the best solution for the positioning of the bodies was one where the Electrical Components were surrounding or "wrapping" the electric motor of the gearbox. With the axis of the motor as a reference, the best way of defining the position of each body was using cylindrical coordinates.

Each individual in the initial population is made up of a set of variables that define the position and orientation of the geometries that were read in the previous step, except for the bauraum that remains fixed through the entire procedure and the gearbox whose only possible movement is a rotation along its shaft axis. The generated variables are the following:

- **R:** One value for each electric component, represents the perpendicular distance from the electric motor axis to the centre of the respective component.
- **Y:** One value for each electric component, defines the position of a component along the Y axis, which in this implementation is parallel to the electric motor axis.
- **Theta:** One value for each electric component, defines the position of a component around the electric motor.
- **Rotations:** One value for each electric component plus one for the gearbox. This variable determines the orientation of the electric components (rotation along its own centre around Y axis) as well as the gearbox rotation along its shaft.
- **EC2_placement:** One logic value. This variable will define if the capacitor component will be placed to the left or right of the IGBT component.

Once these variables are created for each member of the population, they next step is to evaluate them in the corresponding function (**EvaluateFitness**). In a loop that goes through each individual, the read geometries are moved and rotated to the corresponding positions defined by the variables R, Y, Theta, Rotations and EC2_placement. Figure 10 shows the resulting positioning for some individuals of the first generation.



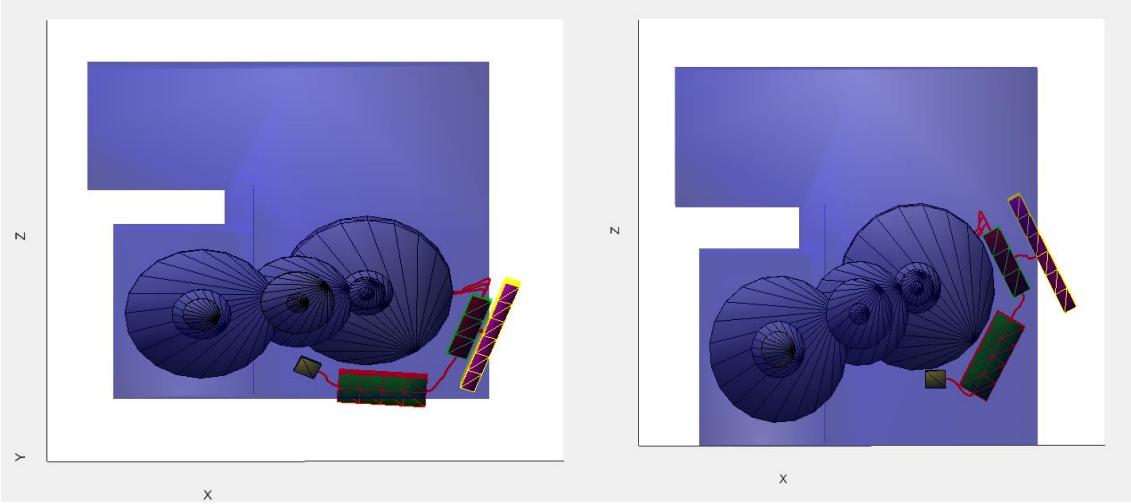


Figure 10: Few different orientation designs from the first generation

As can be seen from Figure 10, even with the limitations in the generation of the variables, there are collisions between components which need to be detected and appropriately dealt with. This is achieved using the collision detection function (**Inpolyhedron** [5]) which measures the collision between two STL files based on the number of nodes of one geometry penetrating the second geometry. Figure 11(a) and 11(b) shows the code detecting collision in case (a) when there is a collision and detecting no collisions in case (b) as clearly seen from the figures.

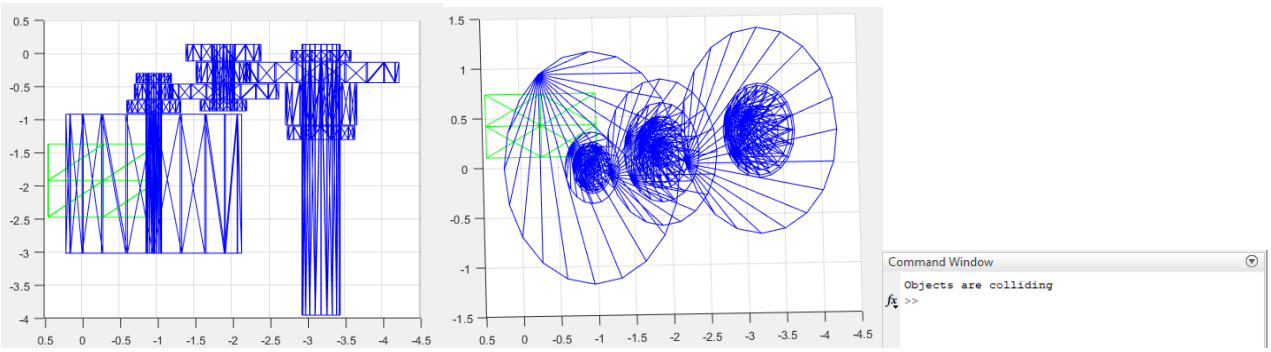


Figure 11(a): Code detecting collision in case of colliding geometries

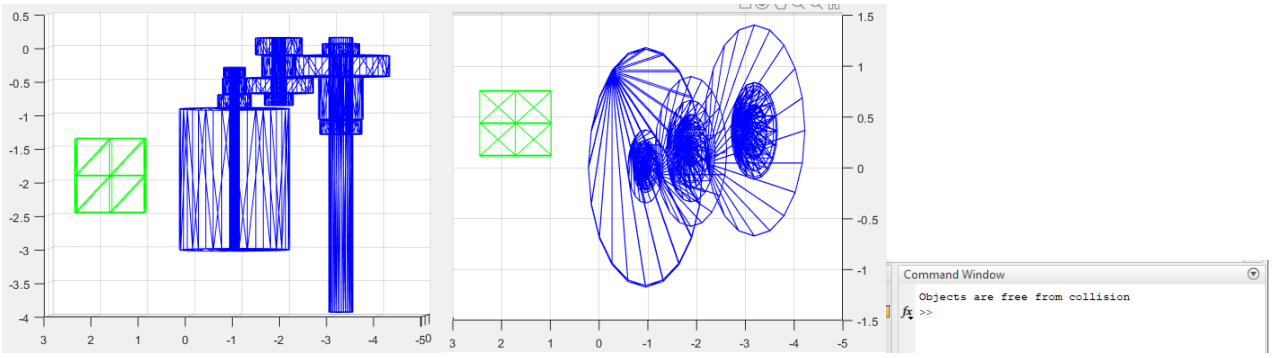


Figure 11(b): Code detecting no collision in case of collision free geometries

Figure 12 shows the types of collisions encountered in the designs. The first type is the collision between components. In this case the desired collision factor is zero indicating that no points (nodes) of either geometry are colliding. The second type is a component (partially) outside the Bauraum. In this case the desired collision factor is one indicating that all points (nodes) of the component are inside the Bauraum.

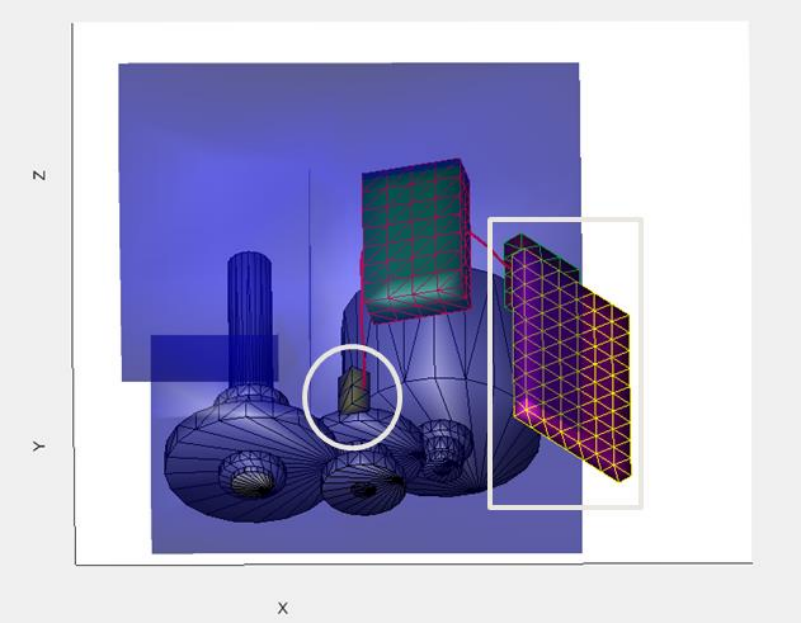


Figure 12: Design showing the types of collisions encountered

Apart from the collisions, the evaluation of an individual also takes into consideration other factors. Figure 13 shows a typical design with the necessary cable connections, the distances to be minimized and the collisions to be avoided.

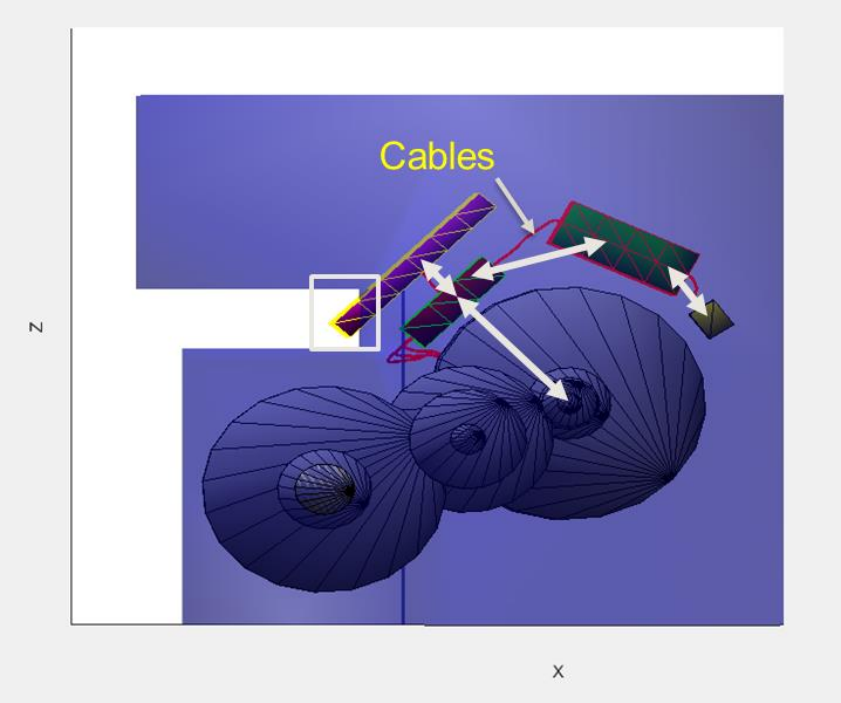


Figure 13: Design showing cables, collisions, and distances to be minimized

The distances and collisions shown in Figure 13 are quantitatively measured in a function (**OptimizationFunction**) to obtain the fitness value which determines the quality of the design. The interference refers to the percentage of collision between geometries. The factor 'X' is a huge number known as the Penalty factor which appropriates the fitness value to account for collisions and constraint violations. Since this is a minimization problem, a lower fitness value is preferred i.e., a better quality of design. In case of collisions or constraint violations, the fitness value increased correspondingly making it a worse solution.

$$Fitness\ function\ (f) = \begin{matrix} Distance\ between\ the \\ motor\ and\ the \\ electrical\ components \\ and\ among\ themselves \end{matrix} + Length\ of\ cables + (interference\ *X)$$

After the fitness values of all the members of a generation are calculated the algorithm moves to the next iteration or generation, where the first step is to create another population, but using the results and the information of the previous generation as an input. This is done in a different function (**NewGeneration**). Figure 14 shows an overview of how this function works.

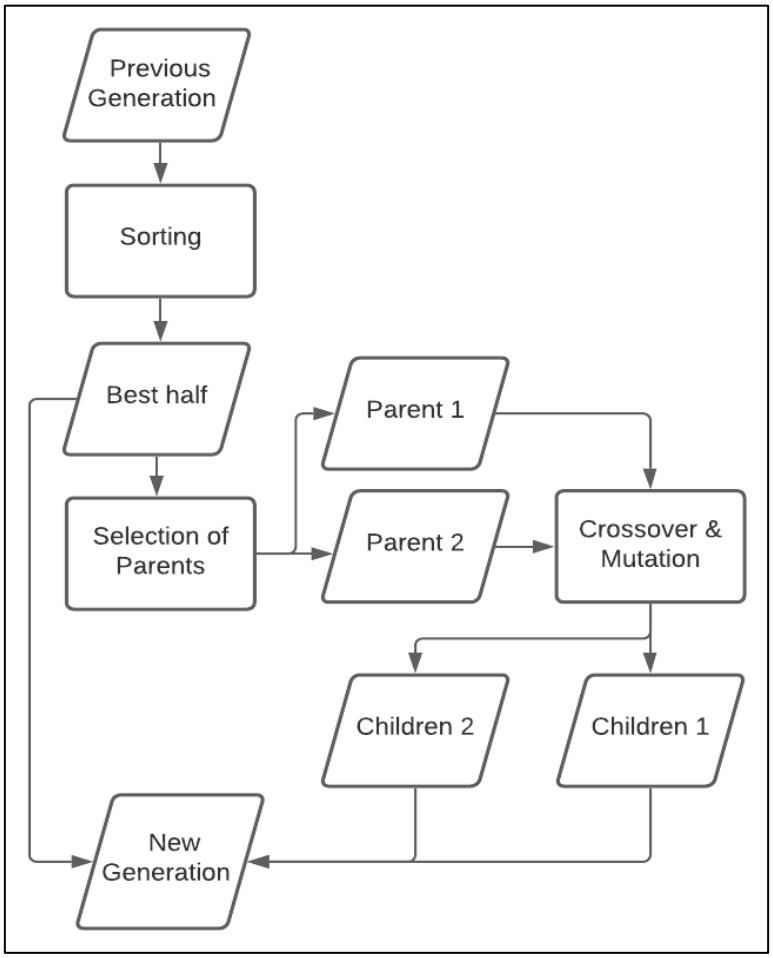


Figure 14: Overview of the process to create the population for the next generation

The population of this next generation will be made with the best performing individuals of the previous one, so the first step is to sort and identify the top half solutions based on the fitness. These top half individuals will constitute the mating pool. From this mating pool two individuals will be selected randomly, with each individual having the same probabilities to be selected as a parent. This process is known as **Selection**. Apart from being used in the mating pool, the top half individuals of the previous generation are also saved and will represent half of the population for the new generation. This is called "**comma strategy**" and ensures that valuable information isn't lost in the case that the new generation performance is worse than the previous one.

The two individuals or "parents" selected from the mating pool will be used to create two new individuals or "children" by mixing the values of the variables from the parents. In order to make the two children different from each other, a weighing factor is used in the mixing process, also called **Crossover.** Finally, in order to add new information to the individuals, each newly generated children can suffer some small and random modification in the value of its variables, defined by a probability parameter in the algorithm. This **Mutation** will add diversity to the new generation in the hopes of finding a better solution. These processes of **Selection, Crossover, and Mutation**, which represent the core of the population-based optimization algorithm, will be repeated until half the population size is complete, then these new individuals, along with the saved best half of the previous generation will be ready to be evaluated.

2.3 Cable Connections

The cable connections are modelled using piece-wise cubic spline polynomials. A four control-point approach is adopted wherein two points near each connection to the components ensure a perpendicular cable connection to the components. The middle part of the spline is then divided into three equal parts to allow further control and flexibility to the cable. The input parameters include the coordinates of the four control points, the radius of the prescribed cable and the minimum radius of curvature of the prescribed cable. The function returns the total length of the cables, maximum curvature and the collision factors.

In case the initial cable encounters interference with other geometries, the algorithm attempts to iteratively modify the spline until a physically feasible spline satisfying all constraints is obtained or the maximum number of iterations is reached. This is shown in Figure 14 where the initial cable (in yellow) encountered interference and hence the final spline (in red) was iteratively obtained which is free from interference and satisfies all the constraints.

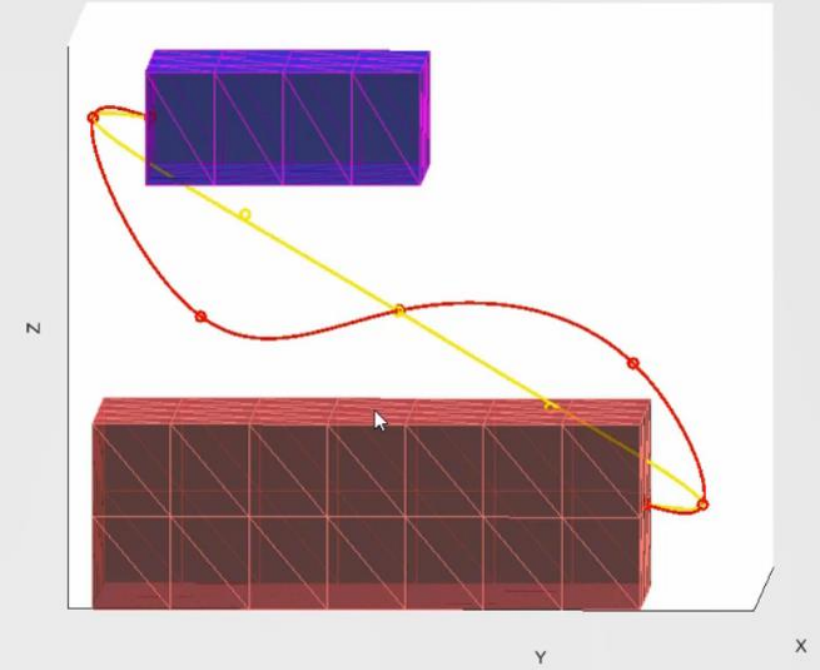


Figure 15: Cable connections between components: initial cable (yellow) and final cable (red)

The case where multiple cables have to be modelled is also dealt with similarly. In this case, the input variables are higher. They include the control points of each cable, the radius of the prescribed cables and the minimum radius of curvature of the prescribed cables. Figure 15 shows a case of three cables connecting the electric motor and the IGBT.

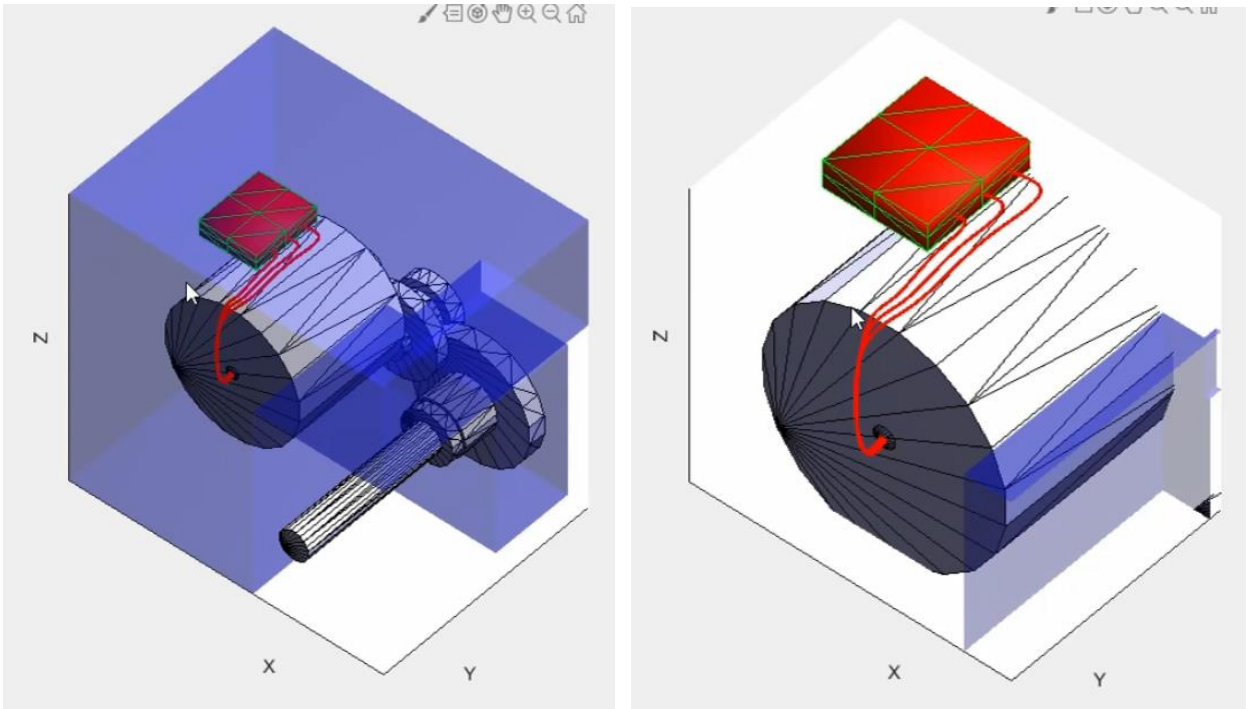


Figure 16: Multiple cable connections between electric motor and IGBT

3 RESULTS

The whole process of creating a new population and evaluating the fitness of the new individuals is repeated until reaching the number of iterations or generations defined by the user. The goal of the optimization algorithm is that by specifying the objective function and the parameters to be minimized, each iteration can create a population whose individuals have better fitness values. After the optimization is completed, it is essential to visualize the obtained results.

Once the algorithm finishes evaluating the fitness of the final iteration, the population is sorted one more time based on its fitness so it can detect which individuals are the ones with the best fitness. The variables (R, Y, theta, Rotations, EC2_placement) for positioning these individuals are sent to a visualization function (**Visualize**) to plot the optimized assembly in a MATLAB figure.

To measure the performance of the optimization algorithm, it is important to check the fitness and how the best individual in the population looks for every iteration. Figure 17 shows the best designs for generation 1, generation 10 and generation 20 respectively on the simple Bauraum configuration with a population of 100. Figures 18(a) and 18(b) show the optimal solution of the positioning objective after 20 generations and the average fitness of the individuals in each generation respectively for the simple Bauraum. The algorithm performs quite good for the Bauraum configuration 1 and give good results after 20 generations.

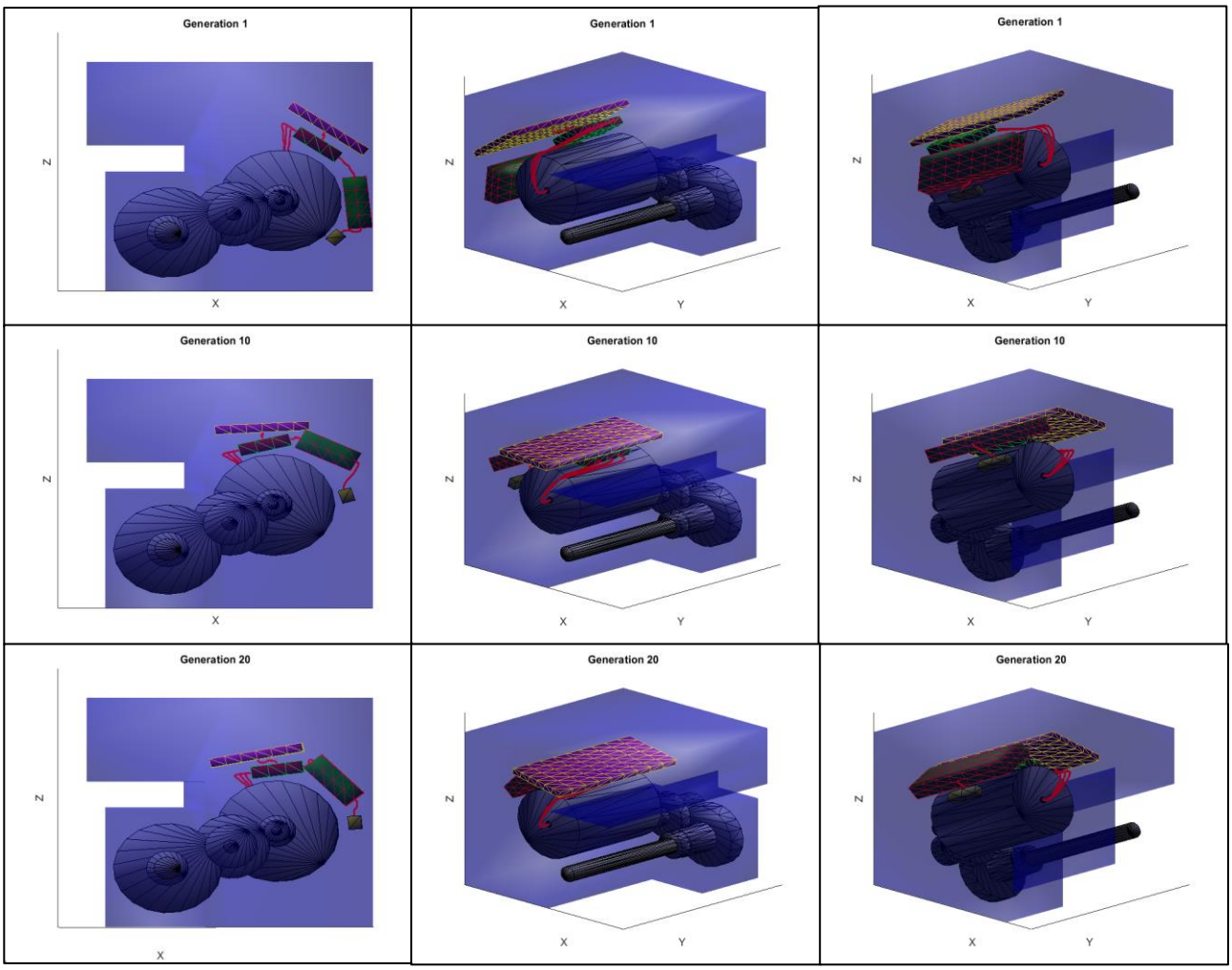
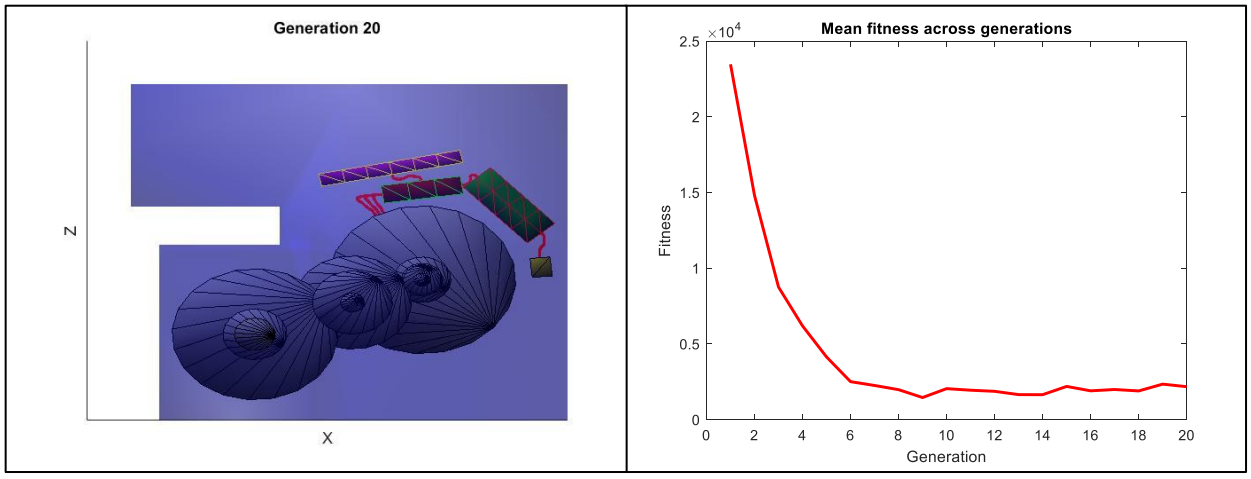


Figure 17: Results showing design improvements over Generation 1, 10 and 20 on Bauraum configuration 1 with Population size=100



(a) Components positioning. (b) Mean fitness vs generation.

Figure 18: Results of the algorithm on Bauraum configuration 1; Population=100, Generations=20.

Figure 17 shows that from generation 1 a good positioning of the assembly is found with no collisions and small distances. This may be because of the simple Bauraum structure and limits imposed to the design variables at the creation of the initial population. Despite this, the optimization algorithm manages to find improvements to the positioning of the elements and in subsequent generations distances are minimized, as well as the length of the cables. This is not only for the case of the best individual, which is shown in Figure 17, but also for all individuals in the population as can be seen in Figure 18(b). The mean fitness of all the individuals of the population is reduced considerably after some iterations. This means that at the end of the optimization algorithm there will be a fair number of designs that can be considered good solutions.

Another aspect of importance to be tested for the algorithm is its robustness. To test this a more complicated and constrained Bauraum configuration was selected. This bauraum contains openings and holes that makes the positioning of the components more difficult as well as limiting the range of movement of the gearbox.

Figure 19 shows the best designs for the last generation on the complicated Bauraum geometry with a population of 500 individuals. The increase in the population size was found to be a necessary modification in the input in order to get good solutions in a more complex assembly space, however this comes with an increase in the time that it takes to perform the optimization. Figures 20(a) and 20(b) show the optimal solution of the positioning objective after 20 generations and the average fitness of the individuals in each generation respectively for the complicated Bauraum.

The individuals shown corresponding to the best designs of the last generation are different from each other in the positioning and orientation of the bodies. This shows that the algorithm is capable to find a diverse array of solutions or local minimums while still improving the quality of the individuals in each generation, as shown in Figure 20.

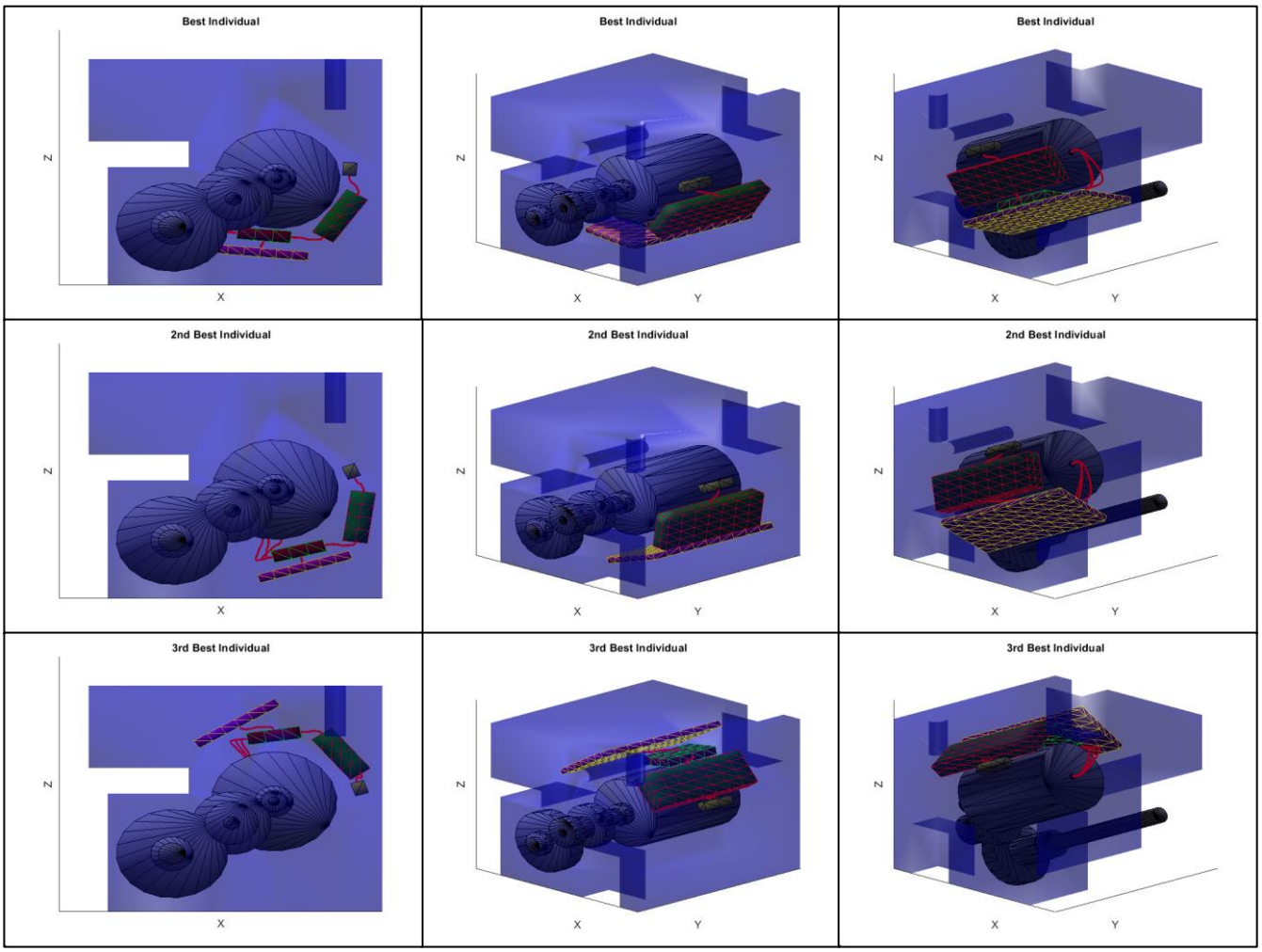
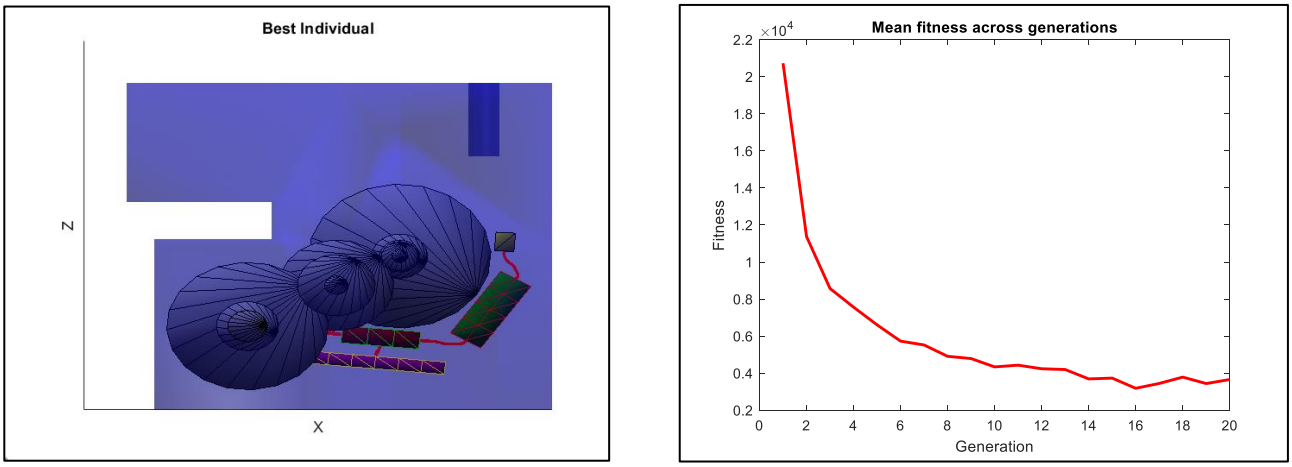


Figure 19: Results showing the best individuals of the last generation (20) on Bauraum configuration 2 with Population=500.



(a) Components positioning. (b) Mean fitness vs generation.

Figure 20: Results of the algorithm on Bauraum configuration 2; Population=500, Generations=20.

As the algorithm progresses, the average fitness decreases, indicating that more and more individuals move towards optimal solutions. It can be concluded that population-based optimization approaches are a good and efficient way to automatize processes such as placement of components, packing, selection of variation etc. These methods are quite flexible and can be modified to adapt to the requirements of the problems.

4 CONCLUSIONS

The optimization algorithm developed in MATLAB fulfilled its objective of finding an optimal positioning of many components in a constrained space. The use of a population-based approach worked well in this implementation because the number of variables was quite big and for some of them there was no clear direction where to move these values in order to get a better solution, like in the angles of the electrical components or the orientation of the gearbox. The solutions found by this algorithm showed improvement across iterations of the procedure, while also improving the average quality of the entire set of individuals in the populations. By the time the optimization procedure was completed, most individuals presented good fitness while still showing differences among them, giving diversity to the generated designs and robustness to the algorithm.

While the population-based function that created new individuals using selection, crossover and mutation was the core of this optimization algorithm, the most demanding tasks were the ones in charge of evaluating the fitness of said individuals, especially when checking for collisions and creating cable connections. For the cases where the assembly space presented more constraints and a bigger size of population was needed, the computation time went from requiring a few minutes to a few hours for completion. This is why it became relevant to find a way to discard unsuccessful individuals without having to create cables by applying a threshold in a pre-calculated fitness or saving the best half of the previous generation along with its fitness values, so each new generation only required to evaluate half of its members.

Overall, it can be concluded that the code implementation presented here succeeded in solving the optimization problem of positioning a gearbox and the associated components inside a constrained space.

5 POSSIBLE FURTHER DEVELOPMENTS

One of the main challenges of this project work was to understand and implement a population-based optimization algorithm to find a solution to the positioning problem. While the algorithm was successfully implemented and a solution was found, the steps of selection, crossover and mutation are performed in a simple way, remaining constant through all the iterations.

A study on the performance of the algorithm and how it can be improved could be a significant addition to this project. This study should check the impact of any change in the optimization parameters like the selection coefficient, the crossover weighing factor, the mutation probability and the mutation amplitude. Also, in this case the penalties assigned to the fitness based on collisions, distance and cable connections were empirically defined based on trial and error. Further studies on these penalty factors and their impact in the solution can lead to an improvement in the algorithm.

References

[1] Bonisoli, E., Velardocchia, M., Moos, S., Tornincasa, S., & Galvagno, E. (2010). Gearbox Design by means of Genetic Algorithm and CAD/CAE Methodologies (No. 2010-01-0895). SAE Technical Paper.

[2] Lyu, N., & Saitou, K. (2005). Topology optimization of multicomponent beam structure via decomposition-based assembly synthesis. J. Mech. Des., 127(2), 170-183.

[3] Li, P., & Shen, Z. (2019, November). 3D line matching model and optimization for rigid pipeline assembly. In IOP Conference Series: Earth and Environmental Science (Vol. 384, No. 1, p. 012104). IOP Publishing.

[4] Ma, Y., Chen, Z., Hu, W., & Wang, W. (2018, August). Packing irregular objects in 3D space via hybrid optimization. In Computer Graphics Forum (Vol. 37, No. 5, pp. 49-59).

[5] Sven (2021). inpolyhedron - are points inside a triangulated volume? (https://www.mathworks.com/matlabcentral/fileexchange/37856-inpolyhedron-are-points-inside-a-triangulated-volume), MATLAB Central File Exchange. Retrieved December 28, 2021.