Quadcopter Design using Genetic Algorithm

Reuben Georgi*

***Abstract*—Selection of Quadcopter components from several options is posed as a multi-objective optimization problem subject to performance and integration constraints. The problem is formulated by considering parameters of components which can be easily found online and is solved using a Genetic Algorithm. A MATLAB app is developed for convenient input of parameters and output of results, including the Pareto frontier, to display Pareto optimal solutions.**

***Index Terms*—Genetic Algorithm, Multi-objective optimization, Quadcopter Design, Pareto Optimality**

I. Introduction and Motivation

Unmanned aerial vehicles and drones have become an increasingly common sight in recent times due to their utility in a variety of different applications. In particular, fabricating and flying drones has become a popular hobby taken up by the general public. One reason why this activity has taken off is due to the wide variety of parts and components which are available for reasonable prices. A commonly seen configuration of drone is the quadcopter, which is a multirotor helicopter that is lifted and propelled by four rotors.

Selecting the best combination of components for one's requirement from the vast number of options available can be a difficult task. The aim of this project is to create a tool that can identify the best combination of components from the available options available using a Genetic Algorithm while considering two important criteria, cost and weight.

A. Genetic Algorithm

A genetic algorithm belongs to a class of algorithms known as evolutionary algorithms which are inspired by certain biological processes. John Holland introduced genetic algorithms (GAs) in 1960 based on the concept of Darwin's theory of evolution and his student David E. Goldberg further extended the concept in 1989 [1].

GAs work with a population of individuals where, each one represents a particular solution to the problem and are usually encoded in some form. The population is evolved, over generations, to produce better solutions to the problem using processes which mimic natural phenomena such as crossover, mutation and selection.

B. Pareto Frontier

Real-world problems usually involve consideration of multiple objectives and often these objectives may conflict with each other. Trade-offs can exist between some objectives, where improving one objective has a detrimental effect on another.

A concept widely used in multi-objective optimization is that of Pareto optimality. A solution is called Pareto optimal, if none of the objective functions can be improved without compromising performance on other objectives. The Pareto front or Pareto frontier is the set of all Pareto optimal solutions and allows the designer to restrict their attention to the set of efficient choices and make tradeoffs within this set, rather than considering the full range of every parameter.

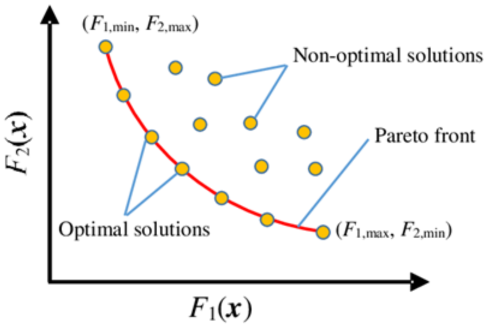


Fig. 1: Pareto Frontier example [2]

II. Problem Formulation

A. Problem Description

While a functioning quadcopter can consist of several unique parts, the following standard components are considered for this problem: Battery, Electronic Speed Controllers (ESC), Motors, Flight controller, Frame and Propellers. The objective of this problem is to find the cheapest and lightest combinations of these components from the available options while considering certain constraints.

During the formulation of this problem, the aim was to use parameters of components that can be readily found on the internet. While most of the following parameters are typically listed with the product, the sources of any parameters obtained from external sources are mentioned in brackets below.

- **Battery**: Cost, Weight, C-rating, Capacity (mAh), Voltage
- **ESC**: Cost, Weight, Current Rating, Maximum Voltage
- **Motor**: Cost, Weight, Data (RPM, Current, Voltage) at 50% and 100% throttle (https://www.miniquadtestbench.com/)
- **Flight Controller**: Cost, Weight
- **Frame**: Cost, Weight, Size
- **Propellers**: Cost, Weight, Diameter, Data (C_T, C_P) at different RPM (https://www.apcprop.com/technical-information/performance-data/)

*The author belongs to the School of Aeronautics and Astronautics, Purdue University

Where, C_T and C_P are coefficients of thrust and power respectively

Making use of these parameters, the following constraints are considered for the optimization problem,

- For a reasonable amount of maneuverability, the maximum thrust that can produced by the quadcopter should be at least twice its total weight,

  $$T_{total} \geq 2W_{total}$$

  T_{total} is found using density ρ (taken to be 1.225 kg/m^3), RPM provided by the motor at 100% throttle n_{100} and propeller diameter D using the formula $T = \rho C_T n_{100}^2 D^4$ [3].

- The Figure of Merit is a measure of the efficiency of the propeller and is the ratio of the ideal power required to hover to the actual power required. The required value is taken as a user input in this project and is calculated as,

  $$F.o.M = \frac{T^{3/2}/\sqrt{2\rho A}}{\rho C_P n_{50}^3 D^5}$$

  where RPM provided by the motor at 50% throttle n_{50} is used with propeller disk area A.

- Peak current drawn from all four motors must not exceed battery's capability, $Capacity(mAh) * C\text{-}rating$.

- The required hovering capability of the quadcopter, measured in minutes, is taken as a user input and calculated as,

  $$t_{hover} = \frac{mAh * 60}{1000 * 4I_{50}}$$

  dividing battery capacity by the number of amps being drawn from the battery at hover.

- Instead of a formal equation relating frame size and propeller size, a Polynomial curve fitted to a guideline table relating the two (https://oscarliang.com/mini-quad-frame-basics/) was used to formulate the related constraints.

- The ESC's max voltage rating must be greater than the maximum voltage drawn by the motor and the voltage drawn at 100% throttle by the motors should be within battery limits.

B. Genetic Algorithm Set Up and Methodology

The GA is set up using the GA550 program from Purdue's AAE550 which is a MATLAB-based genetic algorithm that was originally available from the MathWorks FTP site in the mid-1990s.

This version uses uniform crossover (Crossover Probability, $P_c = 0.5$) with tournament selection. The code also uses a Gray coding scheme and provides a choice between the bit-string affinity and a consecutive generation stopping criteria. The default stopping criterion is a 90% bit-string affinity.

Based on the number of available choices, the length l of a chromosome can be found and used to calculate the population size and the mutation rate using the formulas mentioned below.

The population size: $N_{pop} = 4l$, Mutation rate: $P_m = \frac{l+1}{2N_{pop}l}$

C. Problem Statement

Minimize:

$$f_1(x) = \sum_{i=1}^{6} C_i(x)$$

(Cost)

$$f_2(x) = \sum_{i=1}^{6} W_i(x)$$

(Weight)

Subject to:

$$T_{total} \geq 2W_{total}$$

$$F.o.M \geq M_{desired}$$

$$4I_{peak} \geq Capacity(mAh) * C_{rating}$$

$$t_{hover} \geq t_{desired}$$

$$f(Prop._{size}) - 20mm \geq Frame_{size} \geq f(Prop._{size}) + 20mm$$

$$ESC\ V_{max} \geq Motor\ V_{max}$$

During implementation, all constraints are converted to a standard form ($\frac{value}{limit} - 1 \leq 0$ for upper limit constraints). A weighted sum approach is used here with an exterior step linear penalty function for the fitness function. Each objective is also normalized and made non dimensional.

Fitness function,

$$f(x) = a_1\phi_1(x) + a_2\phi_2(x) + \sum_{i=1}^{n} r_p P(x, g_i(x))$$

where $\phi_i(x) = \frac{f_i(x) - f_i^{min}}{|f_i^{min}|}$, r_p is the penalty multiplier (taken to be 10) and penalty function $P(x, g_i(x))$ for constraint $g_i(x)$ is given by,

$$P(x, g_i(x)) = \begin{cases} 0 & if\ g_i(x) \leq 0 \\ 1 + g_i(x) & else \end{cases}$$

D. MATLAB App

An app was developed using the MATLAB App Designer tool for convenient input of parameters and output of results. After downloading the files (*https://github.com/rageorgi*) to the current folder, a user can run the app by calling the program with the following inputs,

QuadcopterDesignGA(prop_swc,prop_perf,motor_wc,motor_perf,... battery_data,esc_data,frame_data,FC_data);

Where,
prop_swc: Matrix containing propeller sizes, weights and costs
prop_perf: Cell array where each cell contains a matrix with

RPM, C_t and C_p data for a propeller option
motor_wc: Matrix containing motor weights and costs
motor_perf: Matrix containing motor currents, voltages & RPMs at 50% and 100% throttle
battery_data: Matrix containing battery costs, weights, C-ratings, capacities (mAh), voltages
esc_data: Matrix containing ESC costs, weights, current ratings, maximum voltages
frame_data: Matrix containing frame sizes, weights and costs
FC_data: Matrix containing flight controller weights and costs

The "Sample Parts.xlsx" Excel file and the "sample.mat" MATLAB file provide an example of real data input for the app and this data is also used for the results in the next section. The app also takes as input, the desired Figure of Merit, weighting interval (which affects number of points in the Pareto Frontier) and Hovering time in minutes.

III. Results

A. Case 1 (Based on "test_sample.mat")

Inputs: Figure of Merit = 60, Hover Time = 15 mins, Weighting Interval = 0.2

The inputs for this case were made to serve as basic check that the program was working correctly as there were only two options for each component, with the first one being clearly superior. Three runs of the program were made and the results were obtained as expected. Multiple runs were made in order to account for the inherent randomness of a genetic algorithm.

Solution (x)	Mass (Kg)	Cost ($)	F.o.M
111111	0.3896	190.88	46.8

TABLE I: Case 1 (Runs 1-3)

Hover Time (mins)	Max. Thrust (N)
5710.98	8.907

TABLE II: Case 1 contd. (Runs 1-3)

B. Case 2 (Based on "sample.mat")

As mentioned previously, the data used in this case is from real products and represents a proper example of using the program.

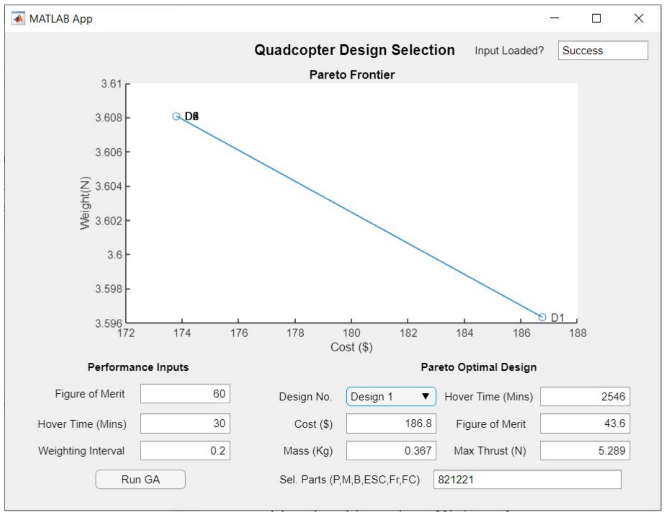
Inputs: Figure of Merit = 60, Hover Time = 30 mins, Weighting Interval = 0.2

a_1	a_2	Solution (x)	Mass (Kg)	Cost ($)
0	1	821221	0.367	186.8
0.2	0.8	811221	0.3682	173.8
0.4	0.6	811221	0.3682	173.8
0.6	0.4	811221	0.3682	173.8
0.8	0.2	811221	0.3682	173.8
1	0	811221	0.3682	173.8

TABLE III: Case 2 (Runs 1-3)

Solution (x)	F.o.M	Hover Time (mins)	Max. Thrust (N)
821221	43.6	2546.4	5.289
811221	46.85	2339	6.057

TABLE IV: Case 2 contd. (Runs 1-3)


MATLAB App

Quadcopter Design Selection — Input Loaded? Success

Performance Inputs

Figure of Merit	60
Hover Time (Mins)	30
Weighting Interval	0.2

Pareto Optimal Design

Design No.	Design 1	Hover Time (Mins)	2546
Cost ($)	186.8	Figure of Merit	43.6
Mass (Kg)	0.367	Max Thrust (N)	5.289

Sel. Parts (P,M,B,ESC,Fr,FC): 821221

Fig. 2: App output with Pareto Frontier

C. Case 3 (Based on "sample_random.mat")

Inputs: Figure of Merit = 60, Hover Time = 30 mins, Weighting Interval = 0.1

This input for this case was manipulated in order to better display some trade-offs between choices and does not reflect any real products in particular. The Pareto Frontier and the results are given below.

a_1	a_2	Solution (x)	Mass (Kg)	Cost ($)
0	1	111211	0.3633	231
0.1	0.9	111212	0.3673	211
0.2	0.8	141112	0.3794	186
0.3	0.7	141122	0.3878	176
0.4	0.6	141122	0.3878	176
0.5	0.5	141122	0.3878	176
0.6	0.4	141122	0.3878	176
0.7	0.3	141142	0.4511	163
0.8	0.2	841142	0.4881	157
0.9	0.1	841142	0.4881	157
1	0	841142	0.4881	157

TABLE V: Case 3

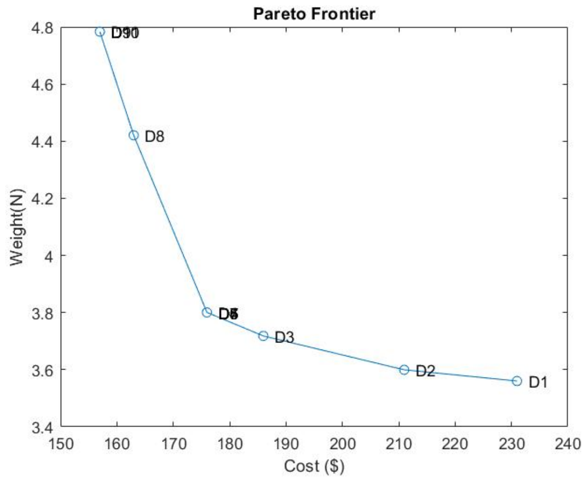


Fig. 3: Case 3 Pareto Frontier

IV. Observations and Conclusions

As seen in the above results, the Genetic Algorithm was successfully used as an aid in quadcopter design which has been posed as a multi-objective optimization problem. However, the values obtained are only approximations to help the design process.

There are more detailed models which can better predict quadcopter performance such as in [4], but they use parameters of the components (ESC resistance for example) which aren't easily found online which would hinder actual hobbyists from using the program. A key improvement for future updates would be to develop better equations relating these common parameters to performance criteria.

Regarding performance criteria such as Figure of Merit which have been posed as constraints, since this application uses discrete variables, there is no guarantee that the constraints will be satisfied. The designer must ultimately determine if the obtained solution is acceptable. Similarly, there is no proof of optimality for the solution obtained from the genetic algorithm as it does not use derivatives or the Kuhn Tucker conditions. However, the results from case 1, doing multiple runs and the inclusion the flight controller component as a sanity check does help in verifying the solution.

Another considerable drawback of the implementation at the moment is that the number of components need to be some power of two. One way to temporarily get around this would be to include dummy options with poor characteristics, preventing them from being selected. While the app in its current state could be useful for a prospective designer, the aim is to keep improving the program and provide a powerful tool for any enthusiast, irrespective of their level of experience.

V. References

[1] Genetic algorithm, Wikipedia, the free encyclopedia , URL: https://en.wikipedia.org/wiki/Genetic_algorithm

[2] Mergos, Panagiotis & Sextos, Anastasios. (2018). Multi-objective optimum selection of ground motion records with genetic algorithms. URL: https://www.researchgate.net/figure/Pareto-optimal-solutions_fig1_329870692

[3] UIUC Propeller Data Site; John B. Brandt, Robert W. Deters, Gavin K. Ananda and Michael S. Selig, URL: https://m-selig.ae.illinois.edu/props/propDB.html

[4] Quan, Quan. (2017). Introduction to Multicopter Design and Control. 10.1007/978-981-10-3382-7.