

Application of Genetic Algorithm for Preliminary Trajectory Optimization

2011-01-2594

Published
10/18/2011

Hugo Pervier, Devaiah Nalianda, Ramon Espi, Vishal Sethi, Pericles Pilidis and
David Zammit-Mangion
Cranfield University

Jean-Michel Rogero and Ricardo Entz
Airbus

Copyright © 2011 SAE International

doi:[10.4271/2011-01-2594](https://doi.org/10.4271/2011-01-2594)

ABSTRACT

The aviation sector has played a significant role in shaping the world into what it is today. The rapid growth of global economies and the corresponding sharp rise in the number of people now wanting to travel on business and for pleasure, has largely been responsible for the development of this industry. With a predicted rise in Revenue Passenger Kilometers (RPK) by over 150% in the next 20 years, the industry will correspondingly be a significant contributor to environmental emissions.

Under such circumstances optimizing aircraft trajectories for lowered emissions will play a critical role amongst various other measures, in mitigating the probable environmental effects of increased air traffic. Aircraft trajectory optimization using evolutionary algorithms is a novel field and preliminary studies have indicated that a reduction in emissions is possible when set as objectives. The paper describes a preliminary study undertaken for the Systems for Green Operation Integrated Technology Demonstrator for the Clean Sky Project (FP7) [1] and uses a customized algorithm NSGAMO2 (a proposed variation of the Non-dominated Sorted Genetic Algorithm II [2]). The paper describes the concept of the algorithm, benchmarking studies undertaken to establish its validity, followed by simple trajectory optimization cases optimized for various multi disciplinary objectives. Results highlighting the tradeoffs between mission fuel burn, mission time and mission NO_x production for a typical short, medium and long range single aisle narrow body aircraft are presented.

INTRODUCTION

The aviation sector has played a significant role in shaping the world into what it is today. The rapid growth of global economies and the corresponding sharp rise in the number of people now wanting to travel on business and for pleasure, has largely been responsible for the development of this industry.

This development however has an environmental cost in the form of aviation emission. To cater for the increased number of air passengers, new routes evolved and that led to a corresponding and rapid rise in air traffic. Emissions from civil aviation are now known to contribute close to 2% of man-made CO₂ into the environment. Even though emissions from aviation have been addressed from the 1960s with the advent of supersonic air travel, in the 1980-90s most regulations primarily addressed NO_x and noise in the vicinity of airports, basically for the benefit of neighboring communities. However considering future predictions for air travel and with global warming on the rise, the requirement of lowering and capping CO₂ emissions have now reached epic proportions.

Literature has therefore indicated that for a sustainable future, significant and continued improvements will need to be developed in the field, to balance the underlying growth within the aviation industry. Under such circumstances optimizing of aircraft trajectories for lowered emissions will play a critical role amongst various other measures, in mitigating the probable environmental effects of increased air traffic.

Aircraft trajectory optimization using evolutionary algorithms is a novel field and preliminary studies have indicated that a reduction in emissions is possible when set as objectives. The paper describes a preliminary study undertaken for the Systems for Green Operation Integrated Technology Demonstrator for the Clean Sky Project (FP7) and uses a customized algorithm NSGAMO2 (further referred as GATAC-GA), which is variation of the Non-dominated Sorted Genetic Algorithm II [2, 3] and extension of work done by Rogero [4]. The paper describes the concept of the algorithm, benchmarking studies undertaken to establish its validity and performance, followed by simple trajectory optimization cases optimized for various multi disciplinary objectives.

The results of the benchmarking study establishes the applicability of the GATAC-GA optimizer as an optimization tool, as it effectively tests the capability of the optimizer to evaluate optimized solutions, in a complex design space which may include convex/concave and discontinuous pareto-optimal fronts.

The paper further discusses the simple trajectory optimization cases in detail with description of the models and the problem constraints considered. Results highlighting the trade-offs between mission fuel burn, mission time and mission NOx production for a typical short, medium and long range mission are finally presented as illustrated cases.

MULTI OBJECTIVE OPTIMIZATION

Literature on the subject of multi objective optimization, suggest various approaches of solving a problem. The basic methodology lies in first finding multiple trade-off optimal solutions with a wide range of values for objectives and to choose one of the obtained solutions using higher level information in the second step. In the first step it is understood that each trade-off solution will correspond to a specific order of importance of the objectives. One approach of solving the problem assumes the user is aware of a *preference factor* among objectives. A simple method would therefore be to form a *composite objective function* as the weighted sum of objectives, where a weight for an objective is proportional to the preference factor/vector assigned to that objective. This method of scalarizing an objective vector into a single composite objective function effectively converts a multi-objective optimization problem into a single objective optimization problem. This procedure is simpler but can be very subjective and will depend also on the higher level information known in the problem. At times this procedure may be used to find multiple trade-off solutions by repeatedly using the procedure using different preference vectors [2].

The main disadvantage of this method is that the process of finding a quantitative relative preference vector is highly subjective and tedious as it may require at times an analysis of non technical, qualitative and experience driven information and therefore unless a reliable and accurate preference vector is utilised, the optimal solution obtained by such methods is highly subjective to a particular user. Classical multi objective optimization methods, which convert multiple objectives to single objective (using a relative preference vector of objectives), use this approach.

The second approach uses the higher level information to evaluate and compare the set of trade off solutions. Based on this information it *chooses one optimal solution* and hence like in the earlier discussed approach does not use the higher level information to **actually search for new solutions**.

Literature therefore suggests that the second approach tends to be more methodical, more practical and less subjective, however if reliable relative preference vectors are available and no requirement of a set of trade-off solutions are needed then the first approach may be considered suitable.

Classical methods are generally classified into **direct search methods** and **gradient based methods**. In direct search methods only the objective functions and constraint values are used in a search process. They are usually slow and are known to require a large number of iterations to achieve convergence. Their advantage however is that it is easy to apply for different problems with little modification to the algorithm. Gradient based methods use the first and/or second order derivatives of the objectives and/or constraints to guide the search process and have the advantage of converging with lesser evaluations, and hence much faster and nearer to an optimal solution. They however have been found disadvantageous when used in discontinuous or non differentiable problems. Classical methods are therefore very quick and can be used to tackle specific problems, but are known to at times get stuck at local optima (or to a sub optimal solution) and may have problems in discrete search spaces.

APPLICATION OF GENETIC ALGORITHMS

The basic idea of using evolutionary concepts in creating a problem solving algorithm was first conceptualized by John Holland and his colleagues from the University of Michigan [2, 5]. It basically uses the principle of 'survival of the fittest and extinction of the weaker species through natural selection'. The salient points of the theory suggest that strong individuals in a population have a greater chance of passing their genes to future generations via reproduction (cross-over) and therefore over a period of time (after many generations) species carrying the correct combination of genes become the dominant population. During the lengthy process of evolution

random changes may occur in genes (mutation) thus changing characteristics of an individual chromosome and its future generations. However, if these processes provide an additional benefit/ advantage in terms of survival or fitness, new species evolve or they are duly eliminated through the process of natural selection.

The Genetic algorithm however replicates approximately the same basic process in finding solutions. The variables used are termed as *Genes*. A set of genes used at any instance form a *chromosome*. The set of chromosomes defines the *population*. The solutions thus calculated using the variables or *genes* form the *raw fitness* of each chromosome in the population. The genetic algorithm sorts the chromosomes out based on their fitness and based on a preset population count, the algorithm eliminates the least fit individuals. Finally the fit individuals selected form a new *generation*.

Amongst the fit individuals (or chromosomes) selected, a further set is randomly selected to form a mating pool and genetic operators are utilized to *cross-over* (essentially reproduction where two or more chromosomes are used to create *offspring*) and *mutate* (wherein the operator introduces a random 'genetic change') the selected chromosome. The crossed-over and mutated offspring are again merged into the population and fitness value of each chromosome is calculated. The process then continues iteratively to form new generations till prefixed criteria, such as maximum fitness possible or maximum generations are reached.

MULTI OBJECTIVE GENETIC ALGORITHM (MOGA)

As discussed in the previous section, genetic algorithms, being a population based approach, are ideal to solve multi objective optimization problems as they effectively search the whole feasible objective space or design space. As the algorithm progresses it will try and find a set of *non-dominated solutions*, that is, each solution dominates another solution in at least one objective. The final non-dominated set will meet the following two conditions; (i) any two solutions of the non-dominated set must be non-dominated with respect to each other, (ii) any solution not in this set, but in the feasible objective space, must be dominated by at least one member of the non-dominated set. This non-dominated set then form the **pareto optimal set or pareto optimal front** (see [Figure 1](#)).

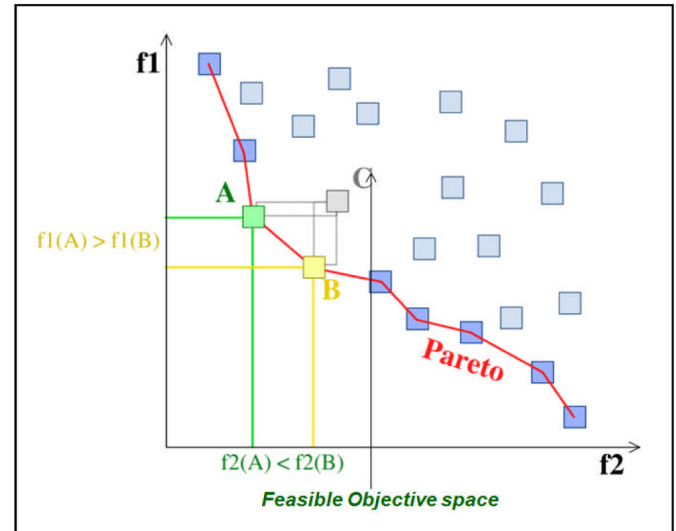


Figure 1. Pareto optimal (non-dominated) front [6]

OBJECTIVES OF MULTI OBJECTIVE OPTIMIZATION

The 'objectives' of multi objective problems may be *non-conflicting* or *conflicting*. The result of the former will be that the cardinality of the pareto optimal set may be one. However if the objectives are conflicting, which they invariably are, then the non-dominated set (or pareto optimal set) will form a defined pareto front with a cardinality higher than one. Therefore the main objective of a multi objective optimization algorithm is to ensure; (i) it finds solutions as close as possible to (or preferably on) the true pareto optimal front, which will be used as an indicator of its accuracy, (ii) it finds solutions as close and as diverse as possible along the true pareto optimal front, which provide the user with a good range of 'trade off' solutions for its objectives (its diversity).

GATAC-GA OPTIMIZER

Concept

NSGA-II or the Non-dominated Sorting Genetic Algorithm is a 'state of the art' genetic algorithm for multi objective optimization. Created by Deb [2], it has been designed for lower computational complexity of non-dominated sorting and has introduced elitism, by which, apart from improving its ability to retain good solutions, once found, it also speeds up the performance of genetic algorithms.

GATAC-GA variant

The genetic algorithm based optimizer incorporated within the framework, however, is based on the variant of the concept of NSGA-II and varies in structure and application of genetic operators. It differs primarily in two respects; (i) it uses a different selection processes to form the mating pool, (ii) it uses an entirely different sequence of genetic operators

(Dynamic Vector Mutate method for mutation and two cross-over methods for cross-over/ reproduction- Tri-Linear cross-over and SBX cross-over) [4].

The optimization algorithm has been coded in the object oriented language JAVA. The algorithm was conceptualized, designed and executed by Airbus France and further benchmarked and tested by Cranfield University, essentially for use in the European Environmental project Clean Sky (FP7) and will be a part of the optimization suite available within the project.

The benchmarking and testing of the algorithm was undertaken using various mathematical functions and simple trajectory optimization case studies, to gauge the performance and for comparison with other methods and commercially available optimizers.

As stated earlier, apart from the optimizer used in this study, two other GA based optimizers have been used for a comparative study, namely an implementation of the NSGA-II (coded using Matlab) available in the Public Domain [7] and MOGA, a genetic algorithm based standard optimisation tool, available within the Matlab optimisation toolbox [8].

BENCHMARKING STUDIES

After creating the GATAC-GA optimizer, a benchmarking and testing exercise was undertaken to establish the validity and the performance of the optimizer. To establish a reference, similar tests were run using optimization algorithms available in Matlab (NSGA-II and MOGA) as described earlier. A key issue when comparing optimizers is the optimization of the parameter settings for each algorithm. This may be achieved by testing through trial and error or through an automated optimization technique. It may however be noted that this has not been considered within the scope of this study, as a comparison has been made primarily for reference of performance and establishing the validity of the obtained optimized results.

The GATAC-GA optimizer used two cross-over methods (SBX Cross-over and Tri-Linear Cross-over) with cross-over ratio of 0.45 for each and a single mutation method (Dynamic Vector mutate [4]) with a mutation ratio of 0.033; this ratio being chosen accordingly to [2] as $1/n$ with n the number of variables considered. The Matlab NSGA-II optimizer used single cross-over and mutation methods (SBX Cross-over and Polynomial Mutation) with ratios of 0.9 and 0.033 respectively. The Matlab MOGA optimizer also used single cross-over and mutation methods (Scattered Crossover and Adaptive Feasible Mutation) with ratios of 0.7 and 0.03 respectively. All optimizers during the benchmarking exercise used a population size of 100.

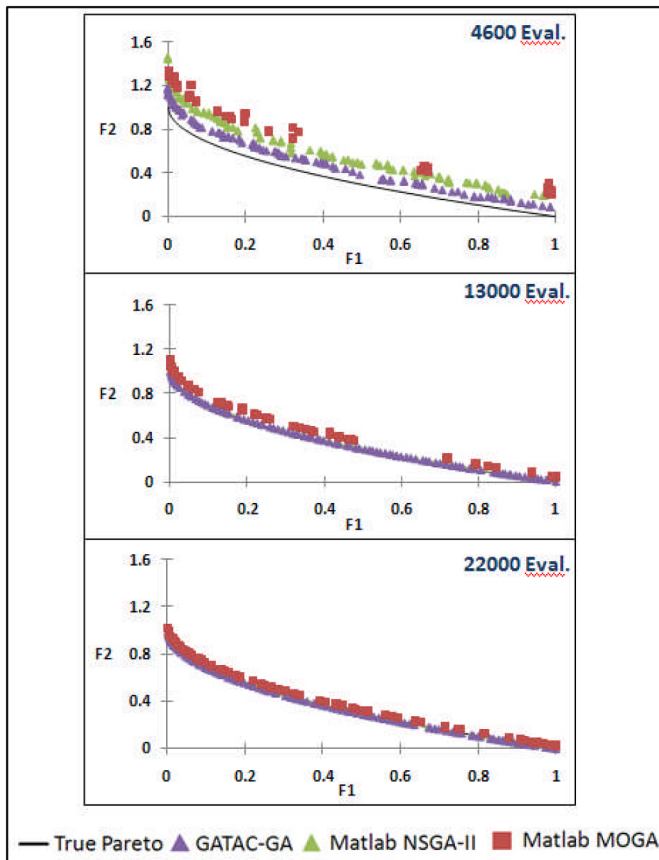
The benchmarking tests were conducted using standard mathematical Zitzler-Deb-Thiele (ZDT) test functions (see [2]). These functions essentially test the multi-objective genetic algorithm ability to handle difficulties along and lateral to a pareto-optimal front and hence test its ability to find a diverse set of solutions converging in close proximity or on the true pareto front. These abilities of the algorithm can be measured using the Diversity metric (Δ) and Convergence metric (γ), as described in referenced literature [2, 3].

Of the ZDT functions recommended in referenced literature, ZDT1, ZDT3 and ZDT6 functions were used for the benchmarking exercise. A description of the functions may be found in referenced literature [2, 3]. ZDT1 and ZDT3 functions both have convex pareto optimal sets. Apart from testing the ability of the optimizer to tackle a large number of decision variables and a convex pareto optimal front, the ZDT3 function also tests its ability to handle a number of discontinuous pareto optimal fronts.

The ZDT6 is considered to be a more complicated function, as it has a concave pareto optimal front, with a non-uniform density and thin set of solutions across and towards the front, respectively.

In order to gauge the performance of the optimizers during the multi-objective optimization routine of each function, three consecutive sets of five runs each were executed and terminated after a prefixed set of evaluations - 4600 (8000 for ZDT6), 13000 and 22000 evaluations. For each run, the convergence metric γ and the diversity metric Δ were computed. The performance of the three optimizers have been tabulated in [Table 1](#) and graphically represented in [Figures 2, 3, 4](#).

ZDT1 is the simplest of the ZDT functions used, and hence it was observed that within 4600 evaluations the optimized solutions had begun converging towards the Pareto-optimal front. It may be observed from [figure 2](#) and [Table 1](#) that after 4600 evaluations the GATAC-GA optimizer was found to perform well amongst the three optimizers. However by 13000 evaluations, the performance of the GATAC-GA optimizer was found to be the best, with a good diversity and evaluated set of solutions close to full convergence. The performance of the Matlab NSGA-II optimiser was also considered to be good and comparable with the GATAC-GA Optimiser as the optimized solutions had converged very close in proximity, to the true Pareto-optimal front with an equally good diversity. After 22000 evaluations no significant improvement was observed in convergence and diversity of the GATAC-GA and Matlab NSGA-II optimizers. The solutions from the Matlab MOGA optimizer were found to improve but would require higher number of evaluations to achieve improved convergence and diversity.



Figures 2. ZDT1 Function - Pareto fronts obtained for 4600, 13000 and 22000 evaluations

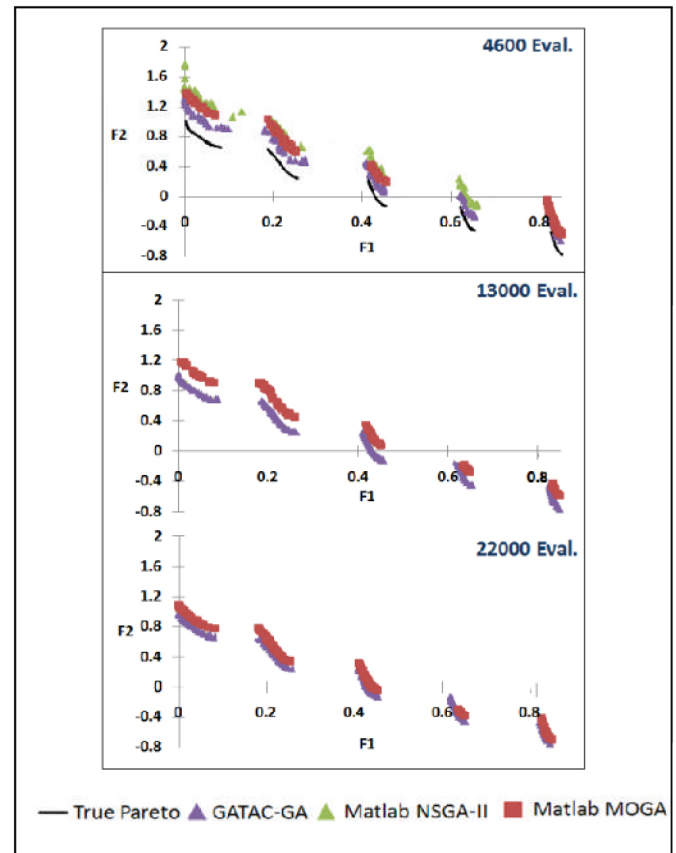


Figure 3. ZDT3 Function - Pareto fronts obtained for 4600, 13000 and 22000 evaluations

As mentioned in the previous section the ZDT3 problem adds the complexity of a discontinuous pareto optimal region. The performance of the three optimizers is as indicated in [Figure 3](#) and [Table 1](#). As observed in the previous case, within 4600 evaluations, the solutions from GATAC-GA optimizer were again observed to have the best convergence and diversity with respect to the true pareto optimal front. However by 13000 evaluations, the performance of the GATAC-GA optimiser and Matlab NSGA-II was similar with solutions close to full convergence and the best diversity. After 22000 evaluations, minimal improvement in performance of the two optimizers was observed. Even though the Matlab MOGA optimizer performance did improve in convergence and diversity, it was not comparable with the other two optimizers.

ZDT6 problem features a non-convex pareto front. [Figure 4](#) and [Table 1](#) illustrate the convergence and diversity information for this function. Due to the complexity of the pareto front significant convergence towards the pareto optimal front was not observed at 4600 like the other functions and hence the first convergence and divergence characteristics were recorded after 8000 evaluations. As observed in the previous two cases GATAC-GA optimizer was found to have performed the best overall. It performed significantly better within 8000 evaluation, with a comparable performance be the Matlab NSGA-II optimizer after 13000 evaluations. After 22000 evaluations there was marginal improvement in Matlab NAGA II optimizer performance. However it may be noted that after 22000 evaluations, owing to the complexity, even though the distribution of solution along the pareto optimal front were comparable with the other function, both the optimizer were unable to achieve a convergence very close to the pareto optimal front and achieved a near optimal set of solution ([figure 4](#)). The performance of the Matlab MOGA was found to be unsatisfactory for this function.