A heuristic-based framework to solve a complex aircraft sizing problem

Céline Badufle[a], Christophe Blondel[a], Thierry Druot[a], Christian Bès[b], Jean-Baptiste Hiriart-Urruty[c]

[a]*Airbus France, 316 route de Bayonne, 31060 Toulouse, FRANCE*
[b]*Laboratoire de Génie Mécanique, Paul Sabatier University, 118 route de Narbonne, 31062 Toulouse cedex 4, FRANCE*
[c]*Institut de Mathématiques, Paul Sabatier University, 118 route de Narbonne, 31062 Toulouse cedex 4, FRANCE*

Abstract

Aircraft sizing studies consist in determining the main characteristics of an aircraft starting from a set of requirements. These studies can be summarized as global constrained optimization problems. The constraints express physical feasibility and the requirements to be satisfied; the objectives are market-driven performances of the aircraft. These optimizations are currently manually conducted as many input data frequently evolve during the study. This work introduced mathematical methods that are useful in a sizing tool to ease, fasten and enhance the aircraft configuration optimization problem. Using genetic algorithms, large amounts of design points satisfying the requirements were rapidly produced, despite some issues inherent to the aircraft model: numerical noise or physically meaningless design points due to the vast design space. Then, multicriteria optimization methods were introduced, as several criteria were considered concurrently. As calculation times became important, the aircraft model was substituted by a surrogate model. Radial basis functions approximated the constraint and the objective functions. Finally, a possible outcome of the integration of these different techniques was proposed in order to yield the engineers a global and operational perception of the design space.

Key words: aircraft sizing, evolutionary algorithms, global and multicriteria optimization, surrogate models

1. Introduction

This study has been performed in the context of aircraft sizing studies, *i.e.* the problem is to define the conceptual and preliminary design of transportation aircraft. Aircraft sizing studies consist in determining the main characteristic parameters of an aircraft, starting from a set of requirements such as range, payload or take-off field length. Such studies lead to the definition of the main aircraft parameters, like fuselage length, wing area, or engine thrust.

In mathematical words, aircraft sizing studies can be summarized as global, multidisciplinary and multicriteria optimization problems under constraints with typically one thousand parameters. Actually, the constraints express the requirements to be satisfied and physical feasibility, and the objectives are market-driven characteristics of the aircraft. These characteristics and performance are assessed from numerical simulations that involve several disciplines, like aerodynamics, structure or noise. Intrinsically, aircraft sizing is a multidisciplinary design optimization (MDO) problem, as defined by the American Institute of Aeronautics and Astronautics (AIAA) in the AIAA White Paper (1991). Moreover, this is also a typical multicriteria optimization problem because some objectives are in conflict (*e.g.* empty weight, fuel burn, noise).

With this kind of problem, the first difficulty engineers face at is to satisfy simultaneously all the constraints. Indeed, most of the requirements that have to be satisfied are nonlinear equations and engineers are working on them as black boxes through numerical simulations. When working on a new problem, engineers start from an already existing aircraft, comparable to the new one to design, and they derive its design parameters until reaching a feasible solution of the new problem.

To solve the aircraft sizing problem, the method proposed here consists in decomposing it to first focus on constraint satisfaction, to ensure feasibility, and then target optimality, *i.e.* perform an optimization, starting from feasible points found during the first step.

Indeed, it appears that it could be interesting to uncouple the research of optimum solutions from the problem of admissible set extraction because of the complexity to exhibit admissible points. Once admissible points are produced, it can be legitimate to ask if it is possible to entirely represent the admissible space. The extraction of large amount of admissible design points is useful to build an analytical description of the admissible set, and thus of the design space, which will also make it easier to perform any optimization within this area of interest.

The method presented in this paper allows to automatically build an analytical description of the design space, the aim being to facilitate the exploration and the exploitation of this domain by any optimization process, and also to help engineers having a critical point of view on the results produced by optimizers.

2. Current design process

The general problem of aircraft sizing that has been treated in this study can be summarized as an optimization problem with typically:

- 15 design variables, like engine thrust, wing area.
- 20 inequality constraints, like take-off field length (TOFL), or approach speed.
- And generally one criterion, like maximum take-off weight (MTOW), or direct operating cost (DOC).

A complete description of the problem can be found in tables 1, 2.

This general problem is not currently treated as a nonlinear programming problem, its generic formulation, because of the multidisciplinary aspects. Indeed, the discipline equations and the interconnections between disciplines make it so complex that it is not possible for classical optimization algorithms to solve it directly numerically (Roskam, 2005). Moreover, optimization is often used within these disciplines, because they need to satisfy some local constraints (Torenbeek, 1982). However, combining the computational tasks into a single optimization problem is impractical, because of the large number of variables and the amount of time required to execute the analyses. Furthermore, even if this would be possible, it would not be used by engineers, because:

- They are cautious with the results automatically produced by optimizers; since the evaluation function is not robust and sometimes fails, it makes the optimizers fail too, or possibly produce degenerated solution for too simple models. Improving the optimizer will not change this situation, but a more robust evaluation function will.
- They need some more information than just finding an optimum, like knowing the results of sensitivity analysis around the optimum point on the objectives and the design variables.

Thus, the optimization is currently manually conducted by engineers, based on experience and knowledge acquired during previous studies, as described below.

The MDO formulation of the problem in the design tool currently used has been formulated as an intermediary formulation between multidisciplinary feasible (MDF) and collaborative optimization (CO), formulations described by Cramer et al. (1994); Alexandrov and Lewis (1999). Indeed, the tool is able to perform an entire analysis on the discipline constraints and the interdisciplinary consistency constraints, and it also performs some local optimization inside disciplines, like for the calculation of the take-off field length (TOFL).

This tool also helps engineers to find feasible design points. With its ability to transform the problem by choosing freely the status of variables (fixed, dependent or free, status defined as in Buckley et al. (1992)), the design constraint values can be imposed to the process, which solves the inverse problem to find the corresponding admissible design variable values.

Once a design point satisfying the requirements is found, the design study can continue using a direct resolution of the aircraft sizing system. Most design studies rely on sequential parametric trade studies in which one or two design variables are changed to examine the effect on the design. Carpet plots (like on Figure 1) can be presented, showing the effect of these variables on each of the system constraints or objectives.

The number of parameters in this type of study is limited by the number of dimensions that can be perceived graphically or by the complexity to collect all information coming from such samplings when there are more than two parameters. When the number of parameters to be optimized is so large that trade studies on the interesting variables are not possible, some kind of alternative optimization is performed. Values of some important parameters are manually changed and then, trade studies are performed again on the modified designs.

This kind of trade studies is not performed so frequently, because of the preparation work that must be done before each sampling. Indeed, some *a priori* choices are made on which trade variables to select, and some evolution rules are imposed to some other degrees of freedom of the study, which are assumed to be second order parameters. Because it is not possible to make vary all the design parameters, the search cannot be exhaustive, which can reduce the optimality of the results.

Sometimes, these trade studies are replaced by inverse resolutions because generally, optimum solutions are located at the intersection of some constraints. Thus, thanks to this knowledge engineers have on this problem, they do not spend time performing trade studies, but directly calculate the configuration which is of their interest.

The most important and difficult part of this work is to define the evolution rules, to make them the most relevant and representative possible. This is the reason why such trade studies are not performed frequently.

This process progresses step-by-step, each aircraft configuration satisfying some particular requirements is kept as a reference configuration, and then, some modifications are applied to it, according to new information on technologies, or on customer needs. This leads to new trade studies, until design parameters freeze; a new reference configuration is found, and so on.

Convergence of the overall process has not a real sense for this kind of study, because there are always new technologies or updates coming from research on the current aircraft program, which can be added to the current future project of aircraft. Thus, it is to the decision-maker to stop the study when he/she considers it is mature enough to start the development phase.

A possibility to take into account multilevel process has also been considered; there are generally two levels of modelisation in conceptual design, but they are rarely represented together in the design environment, mainly because these models do not share the same geometrical representation of the aircraft. Models with different grained precisions are not mixed in an integrated model of aircraft. It could be technically possible, but the engineers are very cautious in the results obtained by coarse-grained models because of the lack of information on the uncertainty related to these models. The general thinking is that coarsed-grained models may produce erroneous results, but they can have a good representation of the variations of the model according to the variable modifications, and sensitivity analyses can give good results using these models.

When multilevel formulation is used, it is done with a lot of precautions. A really simplified model of aircraft is used to calculate the main parameters of the configuration, and these values are given to a more sophisticated model of aircraft, but the transition is made carefully.

Some commercial tools also exist to perform aircraft sizing, like *PIANO 5* (Lissys Ltd, 2009). Such a tool is conceived as a preliminary aircraft sizing and analysis tool, like the one we are currently working with. The main drawbacks are that this tool is not able to perform reverse computation, and it is primarily aimed at conventional, commercial, subsonic aircraft certificated to civil standards. It is not suitable for military, supersonic or unconventional (*e.g.* canard or blended body) configurations.

The aim of the work presented here was to introduce in the current design tool mathematical methods that can make the global optimization process be more automatic in the exploration and the exploitation of the design space. As explained in the introduction, one of the issues in aircraft sizing studies comes from that the design space is large. It can be a hyperbox from two to twenty dimensions. What interest the engineers is to refine it to its most interesting part, increasing chances to find best aircraft configurations.

The methodology described here is composed of three steps:

1. First define the admissible set, because it is difficult to produce feasible points inside the design space. In a real case study, the proportion of admissible points is about 5 among 1 000 000 of points randomly selected.
2. Then introduce a new kind of constraints, named quality constraints, which are actually acceptable margins applied on each criterion, and based on the optimal values of these criteria through acceptable degradation ratios.
3. Finally reduce the admissible set to an interest domain, where not only operational constraints are satisfied but also quality constraints.

Thus, the method consists in progressing step-by-step to reach smaller and smaller subspaces of the design space, the final aim being to find the Pareto front, which most interesting part is included in the interest domain. Moreover, the knowledge of these subspaces of the design space will bring us some information useful to construct an analytical description of the working spaces.

A description of the process proposed here can be found on figure 2.

3. Using evolutionary algorithms

Aircraft sizing activity is basically an inverse problem. We know the performance that the system should achieve and we look for its physical characteristics. The studied system is an aircraft evolving in its environment. It is complex enough to make it impossible to express directly characteristics as a function of performance. For this reason, the core of aircraft sizing is necessarily an iterative process driving an evaluation function that quantifies the performance of a given aircraft configuration. The design space is determined by validity intervals on the design parameters. The domain called the admissible set is the antecedent of the intersection between the image of the allowed design space and the satisfied requirement space.

The necessity to find a simple answer to a complex problem has led many engineers to aggregate their evaluation function into a constrained mono-criterion optimization process (Torenbeek, 1982). The main drawback of this approach is that it gives a limited view on compromise solutions, since it is obvious that the final product is never a mathematical optimum but an alchemic compromise. The multicriteria approach is an answer to the research of compromise, but requirement and performance formulations may change, or at least evolve, along the study lifetime. Finally, it appears that it could be interesting to uncouple the problem of research of compromise solutions from the problem of ensuring constraint satisfaction. The choice of the optimization process has been inspired by the FSQP (Feasible Sequential Quadratic Programming) principle (Zhou et al., 1997), namely to reach the admissible set before performing any optimization. Moreover, the evaluation function should work better close to feasible points, this assumption is due to semi-empirical background.

Looking at the complete optimization problem, evolutionary algorithms seem to be the most adapted methods, because they require little knowledge about the problem, they favor robustness, and they are well-adapted to multicriteria optimization problems (Coello-Coello et al., 2002; Siarry and Michalewicz, 2008).

Thus, in the following section, we will focus on ensuring constraint satisfaction with evolutionary algorithms. The algorithm will make the design points of the population gather and spread step-by-step in the requested domains, *i.e.* the admissible set and the interest domain.

3.1. Ensuring constraint satisfaction

Constraint handling is not so straightforward in evolutionary algorithms, the search operators, mutation and recombination, are “blind” to constraints. Hence, there is no guarantee that if the parents satisfy some constraints, children will satisfy them as well (Eiben, 2001; Mezura-Montes, 2009).

There are many ways to overcome this question of constraint handling, like indirect constraint handling, where constraints are incorporated in the fitness function, or direct handling, where constraints are left as they are and the Genetic Algorithm (GA) is adapted to enforce them. In this study, the research of optimum solutions is uncoupled from the problem of admissible set extraction. Thus, the method consists in decomposing the aircraft sizing problem to first focus on constraint satisfaction and then, perform an optimization without considering constraints anymore, starting from feasible points found in the first step (Venkatraman and Yen, 2005). From now, we focus on the constraint satisfaction problem related to this aircraft sizing problem.

This constraint satisfaction problem (CSP) is treated here as an optimization problem to give an initial structure for the next steps of the global optimization. The aim is to automatically produce large amounts of design points satisfying the requirements, despite the numerical noise introduced by the evaluation function. Moreover, as the design space is vast, due to the large number of degrees of freedom, it contains meaningless design points (geometrically or physically speaking), making the evaluation function sometimes fail.

3.1.1. Existence of solutions

Before starting this study, the demonstration that the admissible set is not empty was not necessary because engineers currently working on the same problem exhibited such a point, satisfying all the requirements. But one point inside the 15-dimension design space is not sufficient to assess the difficulty to produce admissible points. To collect more information about the admissible set shape, the method consists in trying to find other admissible points without using any constraint satisfaction method. This would also give a first guess of the difficulty to reach the admissible set.

The idea is to use a Monte Carlo method by performing a uniform random sampling inside the search interval of each variable, and then, to apply the evaluation function on these design parameters and to check the constraint values. Finally, among 10^6 calculated points,

- 52% could be evaluated,
- 5 satisfied all the constraints (feasible points),
- 190 satisfied all the constraints but one.

Thus, finding admissible points is far from being trivial, and the constraint satisfaction method will have to be adequately fitted.

3.1.2. Method

To solve the CSP, the chosen method was to consider this problem as an optimization problem, using a penalty method. A specific function was deduced from the constraints, applying linear penalties with physically meaningful thresholds on violation of the constraints. Thus, the objective function is a weighted sum of component-wise penalties. Weights are set to ensure homogeneous contributions of all constraints, as explained in Badufle et al. (2005).

Let ε be a criterion on constraint satisfaction: ε reaches its global minimum value (*i.e.* zero) at all admissible design points. The optimization problem to be actually solved is:

$$\min_{x \in X} \varepsilon(F(x)) \quad \text{where} \begin{cases} x \in X \subset \mathbb{R}^n \\ F : X \to Y \subset \mathbb{R}^q \\ \varepsilon : Y \to \mathbb{R} \end{cases} \tag{1}$$

F being the evaluation function, X the design space and Y the objective space.

There are three main requirements to choose the optimization method to solve this CSP:

- It had to be robust against possible evaluation failures,
- It had to manage both continuous and discrete variables,
- At last, it had to produce a homogeneous sampling of the admissible set, *i.e.* a quite uniform distribution of feasible points in the admissible set.

Consequently, for the sake of the good exploration of the design space and of the variety of admissible points, the chosen method to be implemented is the Non-dominated Sorting Genetic Algorithm (NSGA), developed by Srinivas and Deb (1994), because it fulfills the needs listed above. In addition to the standard fitness, this method sorts the population depending on the distance to the closest design point in the population. Consequently, the individuals do not tend to aggregate.

The genetic algorithm implementation was done with the following specifications:

- Each gene represents one degree of freedom, thus each individual is composed of 15 genes.
- The initial population is composed of 50 individuals, whose genes are obtained from uniform randomized sampling inside respective bounds.
- Reproduction is done by mixing genes of two population subsets: the first one is containing the best individuals; the second one is containing a random sample of the current population. Elitism is limited on purpose, to favour robustness. One child is obtained per crossing of two individuals. The number of children is also 50, the crossed population size is thus between 50 and 100 individuals once clones are discarded.
- Selection of one individual is done according to its fitness, which is inversely proportional to its penalty value. To explore uniformly the admissible set and avoid niching effect, the fitness is weighted according to the distance between points in the population.

- Mutation strategy has to be done by a specific function. Considering that $d_x = \varepsilon(F(x))$ is an estimate of the distance to the admissible set, the mutation probability is calculated through a function P, depending on d_x. The function P has to satisfy 3 basic properties:

    1. $P(0) = 0$: if the distance to the admissible set is zero, then the point has reached it, there is no reason to mute any gene. The mutation probability is then 0.
    2. $P(d_x) > 0$ when $d_x \to 0^+$: if d_x is not zero, even in the neighbourhood of 0, the point has to mute to increase chances to reach the admissible set.
    3. $P(d_x) = P_{\max}$ when $d_x \geq d_{\max}$: the mutation probability increases towards a threshold reached at a certain distance from the admissible set. When far enough, being a bit closer or further is not relevant.

  To satisfy these 3 basic properties, the simplest function is a piecewise affine function, which is continuous, except in zero (see figure 3).

- Algorithm convergence is obtained when all population points have reached the admissible set.

The algorithm used to solve our CSP can be seen in table 3.

3.1.3. Numerical results

The first task was to tune the parameters defining the mutation probability function, like the threshold $d_{\max}$ beyond which the probability to mute has reached its maximum value. Among all the possible parameterizations, a trade-off must be found between optimality and variability (see figures 4 and 5).

In test phases, the calculations were stopped after 50 generations. It is worth noting that the initialisation of any computation lasts 1 minute, and the evaluation of one design point takes typically 20 seconds. The calculation time for one generation of 50 points was roughly thirty minutes and it took globally twenty-five hours to evaluate 50 generations. Calculations were computed on an Ultra SPARC III machine at 1.2 GHz.

With the chosen set of mutation function parameters (see figure 5), the algorithm was tested on 10 different initial populations. After 50 generations, in 8 computations among 10, at least one point reached the admissible set, and in 5 computations among the 10, all points in the population were admissible. The chosen parameterization was the one which showed the fastest convergence speed, with a sufficient global search inside the design space.

To assess the efficiency of this way of computing admissible points, this implementation of a genetic algorithm has been compared with a gradient based-method: FSQP. This method was chosen among different softwares performing optimization (some of them are described by Mongeau et al. (2000)) because it is dedicated to problems where the objective evaluation is difficult out of the admissible domain. Moreover, this choice was sensible as FSQP was designed to first exhibit an admissible point, before optimizing the criterion.

FSQP was used in "constraint satisfaction mode" (Zhou et al., 1997), *i.e.* with no objective function. The main problem to deal with was about discrete degrees of freedom. As the algorithm can not manage them, the process was split into two steps. First, FSQP considered that all degrees of freedom are continuous. This process did not make the evaluation function fail. Indeed, all the equations contained in the aircraft model consider that all the variables are continuous. When engineers are dealing with discrete variables, they are always inputs of the equations, like the number of passengers or the number of engines. Thus, it allowed also FSQP to get a gradient relative to these discrete degrees of freedom using finite differences.

After this first pass results, the discrete degrees of freedom values were rounded to the nearest integers, and then, FSQP was run again for another optimization, but this time, discrete variables were fixed to these values and not optimized anymore.

To make an unbiased comparison between the efficiency of the genetic algorithm and FSQP to reach the admissible set, the comparison was made on a random initial population on which the genetic algorithm was run. Then, FSQP performed successively as many constraint satisfactions using as starting point each individual of the initial genetic algorithm population.

The results of the comparison can be seen in the table 4. In this context, genetic algorithms showed more robustness and efficiency than FSQP, as expected.

3.1.4. Conclusion on constraint satisfaction

The methodology presented here automatically produces large amounts of admissible design points in the context of aircraft sizing. An implementation of genetic algorithms dedicated to solve a constraint satisfaction problem succeeds in 50% of the cases to bring the complete population in the admissible set. Moreover, in this aircraft sizing context, genetic algorithms show more robustness but also, more surprisingly, higher productivity than a gold standard dedicated method like FSQP (see table 4).

In addition, this is done in a relatively short time frame, compared with the time engineers need to exhibit a single admissible point.

To explain the failure cases, one can notice that the number of individuals in the current population, nearly 100 individuals including the parents and the children, is small relatively to the 15-dimensional design space. So the genetic algorithm can exhibit a local minimum.

Moreover, all the calculations were stopped after 50 generations, they could have converged if the calculation had lasted longer. But due to limited time, it was not affordable to let the calculations go on until convergence. Increasing the population size and maximizing the population dispersion would probably improve the success rate of the algorithm.

The next step is to reduce once more the design space to a smaller set than the admissible domain, to facilitate the optimizer's task to find the best aircraft configurations.

3.2. Notion of quality constraints

The aim is to define a method to refine the design space to its most interesting part, increasing chances to find best aircraft configurations. The first step of this method was to solve the constraint satisfaction problem related to this aircraft sizing problem.

To refine the admissible set to a smaller subset, the notion of quality of the results was introduced. Quality is based on allowed degradation of the best values of criteria that can be found inside the admissible set. Quality is thus based on the decision-maker needs, wishes and knowledge.

Let C_i^{max} denote the best value of the i^{th} criterion, and R_i denote the degradation ratio, R_i^{max} being the maximum degradation value allowed for the i^{th} criterion. The degradation ratio is defined as in the formula (2):

$$R_i = \frac{|C_i^{max} - C_i|}{C_i^{max}} \leq R_i^{max} \tag{2}$$

where C_i is the current value of the i^{th} criterion.

The relation (2) is similar to a constraint equation. The degradations of the values of each criterion of the current point have to be less than the upper bound given by R_i^{max}. The quality of a point depends on the values of its criteria related to the values of each degradation ratio. Thus, a new type of constraints has been added to the initial problem, called quality constraints.

The Acceptable Domain is defined as the subset of the admissible set where degradation ratios are less than R_i^{max}, for all criteria, *i.e.* where quality constraints are satisfied (Badufle et al., 2006). The figure 6 is illustrating quality constraints for two criteria.

Thus, to define the acceptable domain, the optimal values of each criterion has to be known, so as many mono-criterion optimizations as considered criteria must be performed. The idea is to use FSQP, as it is a fast standard method that is able to handle constraint satisfaction.

The optimizations were started from an initial point located at the barycentre of the admissible population found thanks to genetic algorithms. Thus, the initial point was already admissible; FSQP did not have to put any initial point inside the admissible set, which is one of the main difficulties that occurs when performing an optimization starting from a point randomly chosen among the design space.

The first question was to know if FSQP would have the same problem than when it was configured to produce admissible points, *i.e.* if it would have to calculate points that make the computation fail during the optimization. To assess this failure probability, a Monte Carlo method was used once again, this time on the admissible domain. Among 10^6 calculated points, randomly selected in the admissible domain, 80% could be evaluated.

The probability that FSQP finds a point that makes the computation fail during the optimization is not negligible. Fortunately, each time FSQP ran, the optimization did not stop, because the failing points were computed as intermediary points used to find the descent direction. In this case, when finding failing points, FSQP chose a descent direction that maybe was not optimal. But the amount of failing points in the admissible set is sufficiently small so that FSQP succeeded in finding the optimum. The main issue was the computation time, between 20 to 28 hours for one criterion.

Now that the acceptable domain is defined, refining the design space can continue by gathering the admissible population inside this domain. The idea is to use the same GA than the one developed to find admissible points; this time, new constraints were added to the optimization problem constraints to define the objective function, those defined as quality constraints. The initial population is the admissible population found at the first step of the optimization. After running GA, at each computation, each point of the population inside the acceptable domain has criteria values that are close to the optimum values in the sense that the degradation ratios are small. Thus, the main effort is to produce points satisfying the problem constraints, then satisfying quality constraints is straigthforward.

The aim of the methodology detailed here is to produce large amounts of points that are in the vicinity of optimum points, starting from a huge design space and depending on several criteria to optimize. This methodology proceed step-by-step, by first producing admissible points, then defining what is called the vicinity of optimum points, *i.e.* the acceptable domain, and finally, producing as many points as needed, that are in the vicinity of optimum points.

3.3. Pareto front

In the end, the Pareto front is the last subset of the admissible domain we want to reach. It is the final answer of the multicriteria optimization problem. The Pareto front corresponds to the set of all efficient points (Pareto, 1896). Pareto solutions are such that any improvement in one objective can occur only if at least another objective is degraded. Therefore, it can be stated that no Pareto point is objectively better than another, the choice of one versus another can only be made on the basis of subjective judgment.

To produce some points on the Pareto front, the idea is to keep on working with the implemented GA, NSGA, with some modifications to adapt it to the problem.

The ranking function is used to classify the points according to their Pareto rank, *i.e.* if they are non-dominated, their rank is the smallest, it is assigned to 1, but if they are dominated, their rank is higher than 1. To calculate it, non-dominated points are removed from the current population, and then, the non-dominated points among the remaining points have their rank assigned to 2, and so on.

In this problem, ensuring constraint satisfaction is a complex task, especially with the introduction of quality constraints. Thus, ranking is performed in two steps. The population is split into two subpopulations, depending on constraints. If one point satisfies all the operational and quality constraints, it is included in the first population; if it does not satisfy one of these constraints, it is included in the second population. Then, ranking is performed as described above, the best ranks are set to the non-dominated points of the population which satisfy all the constraints, and then, the process continues with the non-dominated points of the second population, which have their rank assigned to the last rank value of the first population, plus 1. This is a way to ensure feasibility first and optimality in second.

The other ranking function, the fitness function, is the same as previously described (in section 3.1.2, page 11). It includes a dependency on the distance between points in the population.

At each generation, the currently produced non-dominated points are saved in another population containing already produced non-dominated points. The new ones are compared to the others, and only the non-dominated points of this pool will remain for the next generation, as it is done by the SPEA algorithm (Zitzler and Thiele, 1999). Another condition is added in the selection function. One point will remain in the non-dominated population if it is at a distance to its neighbors greater than the minimum one imposed by the decision-maker. This condition ensures that the points of the Pareto front are well-distributed in the design space.

The algorithm used to solve our CSP can be found in table 5.

To assess the capability of this adaptation of NSGA to produce Pareto points, it was tested on some of the test problems found in Deb (1999); Collette and Siarry (2002). Finally, it obtained satisfactory results to find non-convex or discontinuous Pareto fronts (Badufle, 2007).

Generally, this adaptation of a genetic algorithm is efficient in producing Pareto points, when working on generic problems or on this particular design problem. The main issue with this design problem is the computation time. Indeed, it took approximately 90 minutes to compute one generation of 100 points. The only stop condition of the algorithm is the maximum number of generations. Thus, if the computations are stopped after 50 generations, the results only concerning the computation of the Pareto front were obtained after at least three days. Moreover, the improvement between the results obtained at the generation 10 and those obtained at the last generation is not not worth the extra time needed.

In preliminary design, engineers can not afford to wait three days to get the results of one computation, especially at the beginning of the study, when requirements drastically and rapidly evolve. For instance, the number of passengers or the range of the aircraft, and the aerodynamic input data can change several times a month. The problem when willing to obtain a Pareto front is that the computation must process a large amount of calculations. Thus, a mean has to be found to overcome this problem of computation time.

4. Using Response Surfaces

4.1. Motivation

According to Dennis and Torczon (1995), a consistent theme in the engineering optimization literature is that the time and cost required for the detailed analysis of a single design is often so great that it becomes prohibitive to implement a "black-box" optimization approach to the design problem. The point of using approximation techniques is to reduce the number of full or detailed analyses required during the course of the optimization iterations.

Typically, the objective function is expensive to evaluate because there are large numbers of system variables that must be determined for each choice of design parameters before the criteria can be evaluated.

Approximation models have several properties that make them attractive for use with optimization (Simpson, 1998; Sobieski and Kroo, 2000):

- They yield insight into the relationship between input design variables and output responses,
- They provide fast analysis tools for optimization and design space exploration,

- They avoid potential numerical difficulties that can occur (singularity, steps, *etc.*),
- They represent noisy analysis with an inherently smooth model,
- They provide a natural way of implementing coarse-grained parallelization,
- They facilitate the integration of discipline-dependent analysis codes.

There is a standard engineering practice for attacking multidisciplinary optimization problems with expensive evaluation functions f (Booker et al., 1999):

1. Choose a surrogate s for f that is either
   - A simplified physical model of f, or
   - An approximation of f obtained by evaluating f at selected design sites and interpolating or smoothing the function values thus obtained.
2. Minimize the surrogate s on its definition space to obtain x_s.
3. Compute $f(x_s)$ to determine if improvement has been made over the best x found to date.

Many papers deal with the open question of how to manage the interplay between the optimization and the fidelity of the approximation models, the aim being to insure that the process converges towards a solution of the original problem. Some methods, like those described by Dennis and Torczon (1995); Booker et al. (1999), increase the fidelity of the approximation during the optimization, getting it more precise in the vicinity of a minimizer, or when it becomes clear that the approximation is not doing a good job in identifying trends in the objective. The aim is to compromise between the research of a minimizer and the construction of an approximation that gives a reasonable estimation of the behavior of the objective.

In the following section, two approximation techniques are surveyed, response surface approximations and artificial neural network models, the aim being to choose a surrogate model to replace the expensive evaluation function of the problem treated here.

4.2. Selected methods

4.2.1. Polynomials using linear approximation

The idea of using Taylor series developments came from the observation that most of the time, the admissible set is convex and the different optima of the mono-criterion optimizations are located on the boundary of the admissible space, more precisely at points where some constraints are active simultaneously. Thus, to simplify the expression of the constraints in the general optimization problem, *i.e.* $G(x) \leq 0$ in its nonlinear form, the idea is to replace it by a linear inequation of the form $A.x \leq b$.

An easy way to find this approximate expression is to linearize the constraints at some points located on the boundary of the admissible set. The problem becomes now to project some admissible points, obtained when solving the constraint satisfaction problem in the first step of the optimization process, on the boundary of the admissible space.

Still working with genetic algorithms, this time the fitness function depends on the distance of the current point from the boundary of the admissible set, thus from the closest constraint. The way to generate children is unchanged. Only the mutation operator has been modified. The points mutate in the direction of the constraints that move them the farthest away from the barycentre of the admissible point cluster. The distance of the displacement is randomly sorted between zero and the distance to this closest constraint. A minimum distance is also imposed between points in the design space, like it is done in NSGA. The aim is to obtain well-distributed points along the boundary of the admissible space.

Because of the intrinsic random nature of GA operators, it is difficult for the algorithm to find points that exactly reach the boundary of the admissible set. Thus, it is considered that the process has converged when the distance of a particular point from one constraint is smaller than a given small distance. When all points have converged this way, they are projected on the boundary. This minimum distance is a compromise between convergence and well-distributed repartition of points. It has to be not too small to allow the algorithm to converge, but also not too large. Otherwise, the algorithm would manage to converge too fast and would not have enough time to distribute the points along the boundary of the admissible space.

Now having well-distributed points on the boundary of the admissible set, the constraints can be linearized around these points. The gradient is calculated using finite differences, but since this operation is expensive, the constraints are not linearized systematically around every point that has reached it. Thus, for each active constraint, the points located on the extremities of this particular constraint are identified and the constraint is linearized around these points. Then, the values of the other points that have reached the same constraint are calculated using the obtained linearized expressions. If this value is not sufficiently precise according to the decision-maker, then the constraint is linearized once again around this particular point.

To use an optimization technique coming from Linear Programming, the objective function has to be linearized too. Depending on the criterion to optimize, the method succeeds or fails in finding the right step to make the process converge towards the optimum. Indeed, we suppose the failures are due to the gradient values of some criteria that are too close to zero. But as the method works for some other criteria, the failure of the process is not imputable to the methodology.

Since it seems difficult to use a linearization of the constraints to simplify the optimization formulation, another approximation method, Radial Basis Functions networks, was tested because they are easy to implement and to adapt to this particular problem.

4.2.2. Radial Basis Functions

The second kind of surrogate model tested here is radial basis functions, further denoted RBF. This approximation technique is classified in the category of artificial neural networks. This network is formed from three layers of neurons, as described in Krishnamurthy (2003). The activation function depends on the distance of the input x from the corresponding centre c. The network output is obtained by linearly combining the responses of the hidden neurons to the input:

$$\tilde{f}(x) = \sum_{i=1}^{k} w_i \, h\Big(\big\|x - c_i\big\|_2\Big). \tag{3}$$

where:

- The radial functions h are function of the radial distance $\big\|x - c_i\big\|_2$ from node i,
- The w_i are fitting parameters to be determined,
- k is the number of sample or data points with known function values y_i such that $\tilde{f}(x_i) = y_i$.

Finding an approximation using RBF can be viewed as a minimization problem, where the objective function is the error as defined in the following equation:

$$E(\tilde{f}) = \frac{1}{2}\sum_{i=1}^{k} \big(y_i - \tilde{f}(x_i)\big)^2 + \frac{1}{2}\lambda\big\|D\tilde{f}\big\|^2 \tag{4}$$

where:

- D is a differential operator which controls how many of the derivatives should be taken into account to characterize the smoothness of $\tilde{f}$,
- λ is a regularization parameter which controls the tradeoff between attaching strictly to the training patterns and reconstructing a smooth mapping as reflected in the measure of derivatives $\big\|D\tilde{f}\big\|$.

The simplest RBF network is obtained by setting the inputs of all training patterns as centers, $c_i = x_i$, and $\lambda = 0$. In classical RBF based methods, the interpolation of a surface is performed as a linear combination of radial functions as in the relation (3).

For the simplicity of the implementation, each node of the RBF network is taken among the points of all the populations obtained until now, and the parameter λ defined in the equation (4) is assigned to zero, to avoid calculating any gradient of the function.

The network is constructed like in Fritzke (1994), one more basis function is added until the mean error of the approximated function calculated on the known points has reached the decision-maker threshold. The centre of this additionnal basis function is located at the point of largest error in the learning data basis. The figure 7 is illustrating the principle explained above.

4.2.3. Results obtained using RBF

According to Kodiyalam (2001), the learning population on which the network is constructed is really important. The more the points are distributed in all the design space, the more information can be extracted. The risk in using RBF networks is that they can oscillate. Thus, a great part of the construction of the approximation is to choose the learning population.

Tests were done on two different learning data bases, constructed with the populations already calculated. The first learning data basis contained the initial population, with points randomly sorted in the design space, and the points projected on the constraints. The second population contained the lattest plus the set of the admissible points.

When analysing the results obtained on the approximation fidelity with these two learning data bases, it appears that the first set, the one containing less points, is the most efficient as learning data basis. Indeed, the network constructed with the second population oscillated along the active constraints, which is the place where the approximation has to be the most precise.

Relative errors introduced by the approximation model can be seen in table 6 on some of the constraints. As the fidelity with the first approximation was acceptable, the computation could continue, using the network instead of the evaluation function.

The operations done with the approximation are:

- To optimize the criteria separately (to define the quality constraints),
- To project the admissible points inside the acceptable domain,
- And finally to compute the Pareto front.

Optimizing one criterion with FSQP using the approximation takes less than 10 seconds. Then, producing points of the acceptable domain takes between 3 and 5 minutes, depending on the value of the allowed degradations. And concerning the last operation, computing one generation to produce the Pareto front takes less than 15 seconds.

Thus, in half a day time, engineers can compute several hundred of generations to refine the Pareto front. And if finally, they judge the front is not precise enough, they can use the real evaluation function, starting from the front already produced. Moreover, this RBF network can be used as an analytical description of the design space, and sensitivities analyses can easily and efficiently be performed.

5. General conclusion

This study deals with a multicriteria, multidisciplinary and constrained optimization problem coming from an industrial context, aircraft sizing. The problem consists in defining the conceptual and preliminary design of an aircraft, starting from a set of requirements. The main difficulty with this problem comes from the evaluation function, which is implicit and time-consuming, but also from the general context of aircraft sizing. Indeed, requirements and assumptions rapidly evolve during the study.

By using RBF networks to approximate the evaluation function, we build an analytical description of our design space and objectives, that enables engineers to perform more computations in the same time frame and to produce more solutions than what they are used to, *i.e.* not only one solution of a weighted sum of all the criteria to consider, but a Pareto front. Thus, engineers will have the possibility to choose an alternative solution among the Pareto set according to a new selection criterion. The advantage of this methodology is that it is fully automated, the main part of work for the engineers is to define their optimization problem, *i.e.* which variables are degrees of freedom, what are the boundaries of the design space, and which are the constraints and the criteria. In practice, the computations can be launched during the night so that engineers can exploit the results the next day.

Another advantage that was not used here, is that genetic algorithms can easily be parallelized, making the computation even faster.

In future aircraft sizing studies, some new conflicting criteria will have to be taken into account, like community noise, or climate impact. Producing Pareto fronts will bring more flexibility when defining and solving these new aircraft sizing problems.

Acknowledgments

The content of this paper has been provided by a study that has started during the Ph.D. study of C. Badufle (Badufle, 2007).

We thank every person who has contributed in this study, in particular Marylène Duffau, Philippe Mattei, Philippe Escarnot, Jaïme Genty de la Sagne and Serge Bonnet.

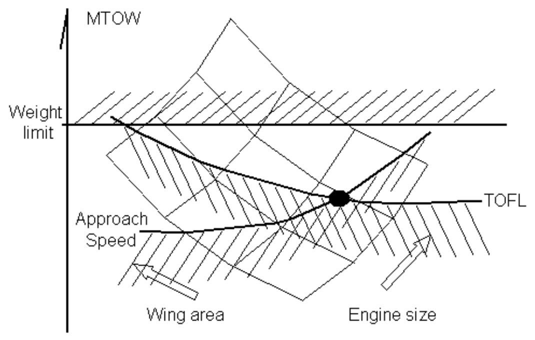

Figure 1: Example of carpet plot

On figure 1, we have two design parameters, engine size and wing area. MTOW, approach speed and TOFL are calculated for several values of these degrees of freedom. The limit value of the constraints, here approach speed and TOFL, are interpolated to be represented on the graph. Then, the minimum value of the criterion, here the MTOW, is found graphically, and the corresponding values of the design parameters are deduced.

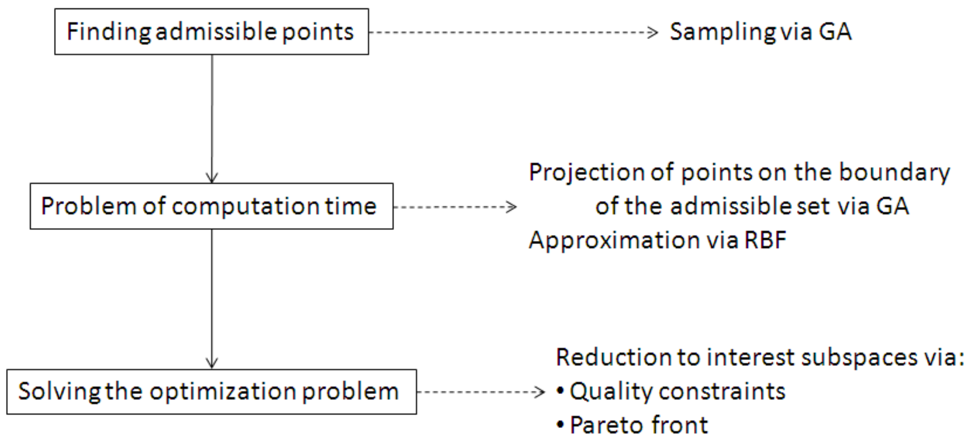
Finding admissible points
Sampling via GA
Problem of computation time
Projection of points on the boundary of the admissible set via GA
Approximation via RBF
Solving the optimization problem
Reduction to interest subspaces via:
• Quality constraints
• Pareto front

Figure 2: Overall process of the proposed resolution method

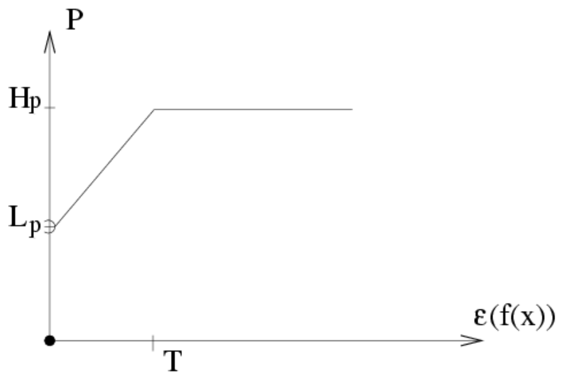

Figure 3: Probability function for mutation strategy

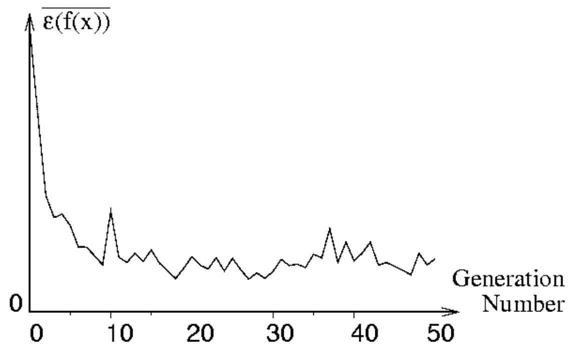

Figure 4: Typical convergence of NSGA when producing admissible points; Hp=50%; Lp=5%; T=1

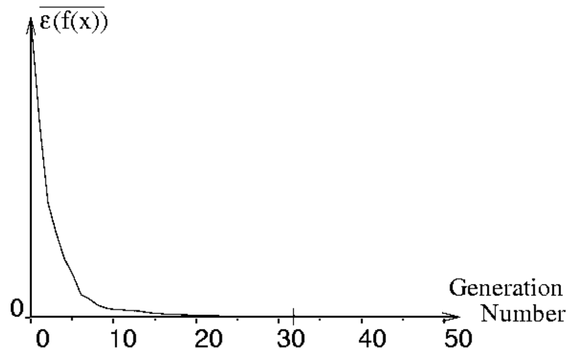

Figure 5: Typical convergence of NSGA when producing admissible points; Hp=5%; Lp=0.5%; T=3

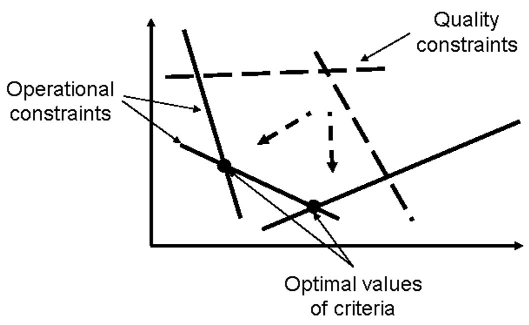

Figure 6: Illustration of quality constraints

On figure 6, we have two degrees of freedom, three operational constraints and two criteria. The descent directions of the criteria are represented by the dashed arrows and the quality constraints by the dashed lines.

As a remark, one can notice that a part of the Pareto front can be included in the boundary of the acceptable domain.

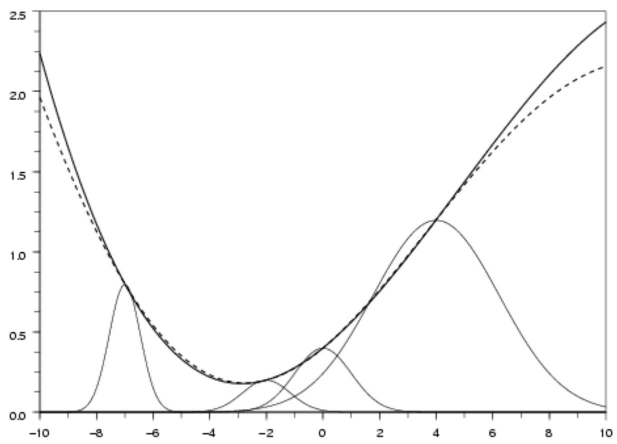

Figure 7: Illustration of RBF network interpolating a polynomial function

On figure 7, the function to interpolate is the black plain line. The approximation function is the dashed line. Four RBF basis functions were used, which are represented on this figure.

Description	Lower bound	Current value	Upper bound
External wing sweep	30 deg	34.4 deg	36 deg
Wing reference area	320 m^2	359.5 m^2	400 m^2
Wing half span	25 m	29 m	33 m
Internal kink spanwise relative position	29%	35%	40%
External kink spanwise relative position	60%	65%	70%
Slat area coefficient	0.5	0.71	1
Additional fuselage frames before wingbox*	-2	0	+2
Number of frames into the wingbox*	8	10	12
Wing root thickness over chord ratio	12%	13.5%	15%
Internal kink thickness over chord ratio	8%	9.1%	10%
External kink thickness over chord ratio	8%	9.2%	10%
Wing tip thickness over chord ratio	8%	9.2%	10%
Engine rubbering factor	0.8	0.85	0.9
Take-off thrust coefficient	0.8	0.92	1
Engine spanwise relative position	30%	38%	40%
Main landing gear spanwise position	4 m	5 m	10 m

Table 1: Design variables of a typical aircraft sizing problem

Variables annoted with the symbol * are discrete variables.

Description	Constraint
Cabin length ratio at 25% of MAC	$= 51.5\%$
Time to climb for nominal mission	≤ 29 min
Climb ceiling	≥ 35000 ft
Buffeting ceiling	≥ 35000 ft
Cruise ceiling	≥ 35000 ft
Approach speed	≤ 140.2 kt
Maximal pitch on ground	≥ 11.6 deg
Regulatory take-off field length	≤ 2800 m
Fuel margin	≥ 1.6
Ground clearance	≥ 0.84 m
Air inlet to door clearance	≥ 2 m
Maximum bank angle on ground	≥ 6 deg
Slat to wing area ratio	$8\% \leq \cdot \leq 9\%$
Aileron to wing area ratio	$3\% \leq \cdot \leq 4\%$
Rear spar clearance for landing gear integration	≥ 0 m
Slat to wing chord ratio at internal kink position	$12\% \leq \cdot \leq 14\%$

Table 2: Constraints of a typical aircraft sizing problem

```
Population initialisation
Population evaluation: $Objectives = \varepsilon(F(x))$
Rank population
While $(\Sigma(Objectives(:)) > 0)$ then (some points do not satisfy at least one constraint)
    Cross population
    Mutate parent population using our probability function $P(d_x)$
    Remove clones from population
    Evaluate children population objectives
    Rank population
    Reduce population
End
```

Table 3: NSGA pseudo code to produce admissible points

	Genetic Algorithms	FSQP
Convergence time (hour)	3.25	5.66
Admissible points (out of 50)	50	16
Success rate (admissible point/initial point)	100%	32%
Efficiency (admissible point/hour)	15.4	2.8

Table 4: Comparison between GA and FSQP in producing admissible points starting from a random population of 50 individuals

```
First population coming from previous computation (we know constraints are satisfied)
Criteria evaluation
Rank population
Initialise non-dominated population
While ($NumberOfGenerations <= NbGenerationMax$) then
    Cross population
    Mutate parent population
    Remove clones from population
    Evaluate children population objectives
        Evaluate constraints
        Evaluate criteria
    Rank population
        First, rank population of points satisfying constraints
        Then, rank population of points not satisfying constraints
    Reduce population, keeping best individuals
    Save non-dominated points
End
```

Table 5: NSGA pseudo code to produce Pareto points

Constraint name	First population		Second population	
	Relative error	Nb of centers	Relative error	Nb of centers
Take-off field length	2.6%	20	16.6%	35
Approach speed	0.08%	13	1.5%	21
Aspect ratio	5.5%	17	6.4%	27
Fuselage fuel ratio	0.3%	9	11.5%	28
Climb speed	4.6%	5	45.1%	28
Scaling factor on engine thrust	0.4%	9	21.9%	23

Table 6: Relative errors in the approximation of some of the constraints. The first population was kept to construct the approximation.

References

AIAA White Paper (1991). Current state-of-the-art in Multidisciplinary Design Optimisation. See also http://endo.sandia.gov/AIAA_MDOTC/sponsored/aiaa_paper.html. prepared by the AIAA Technical Committee for MDO, approved by AIAA Technical Activities Committee.

Alexandrov, N. M. and Lewis, R. M. (1999). Comparative properties of collaborative optimization and other approaches to MDO. Technical Report TR-99-24, NASA/ICASE.

Badufle, C. (2007). *Conceptual aircraft design: towards multiobjective, robust and uncertain optimisation.* PhD thesis, Université Paul Sabatier. http://tel.archives-ouvertes.fr/tel-00367210/fr/.

Badufle, C., Blondel, C., Druot, T., and Duffau, M. (2005). Automatic satisfaction of constraints set in aircraft sizing studies. In Herskovits, J., Mazorche, S., and Canelas, A., editors, *CD-Rom Proceedings of the* 6^{th} *World Congress on Structural and Multidisciplinary Optimization*, Rio de Janeiro. ISSMO. http://www.wcsmo6.org/papers/4081.pdf.

Badufle, C., Blondel, C., Druot, T., and Duffau, M. (2006). Refinement of Design Space in Aircraft Sizing Studies. In *Booklet of abstracts of the International Conference on Mathematics of Optimization and Decision Making*, Pointe-à-Pitre, Guadeloupe. SMAI, *Société de Mathématiques Appliquées et Industrielles*, MODE, *Mathématiques de l'Optimisation et de la Décision*, UAG, *Université Antilles-Guyane.* http://gala.univ-perp.fr/~aussel/CIMODE06/abstracts/badufle.pdf.

Booker, A. J., Dennis, J. E. J., Frank, P. D., Serafini, D. B., Torczon, V., and Trosset, M. W. (1999). A Rigorous Framework for Optimization of Expensive Functions by Surrogates. *Structural Optimization*, 17(1):1–13.

Buckley, M. J., Fertig, K. W., and Smith, D. E. (1992). Design Sheet: An environment for facilitating flexible trade studies during conceptual design. *Aerospace Design Conference, Irvine, California*. AIAA Paper 92-1191.

Coello-Coello, C., Van Veldhuizen, D., and Lamont, G. (2002). *Evolutionary algorithms for solving multi-objective problems*. Kluwer Academic Publishers.

Collette, Y. and Siarry, P. (2002). *Optimisation multiobjectif*. Eyrolles.

Cramer, E. J., Dennis, J. E., J., Frank, P. D., Lewis, R. M., and Shubin, G. R. (1994). Problem formulation for multidisciplinary optimization. *SIAM Journal on Optimization*, 4(4):754–776.

Deb, K. (1999). Multi-objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems. *Evolutionary Computation*, 7(3):205–230.

Dennis, J. and Torczon, V. (1995). Managing approximation models in optimization. In Alexandrov, N. M. and Hussaini, M. Y., editors, *Multidisciplinary Design Optimization, State of the art - Proceedings of the ICASE/NASA Langley Worshop on Multidisciplinary Design Optimization*. SIAM.

Eiben, A. E. (2001). Evolutionary algorithms and constraints satisfaction: Definitions, survey, methodology, and research directions. In Kallel, L., Naudts, B., and Rogers, A., editors, *Theoretical Aspects of Evolutionary Computing*, Natural Computing series, pages 13–30. Springer, Berlin.

Fritzke, B. (1994). Fast learning with incremental RBF networks. *Neural Processing Letters*, 1(1):2–5.

Kodiyalam, S. (2001). Multidisciplinary Aerospace Systems Optimization – AeroSciences (CAS) Project. Technical Report CR-2001-211053, NASA.

Krishnamurthy, T. (2003). Response surface approximation with augmented and compactly supported radial basis functions. In *Proceeding of the 44 th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. AIAA Paper 2003-1748.

Lissys Ltd (2009). Piano 5. see http://www.lissys.demon.co.uk/index.html#find.

Mezura-Montes, E., editor (2009). *Constraint-Handling in Evolutionary Optimization*, volume 198 of *Studies in Computational Intelligence*. Springer-Verlag.

Mongeau, M., Karsenty, H., Rouzé, V., and Hiriart-Urruty, J.-B. (2000). Comparison of public-domain software for black-box global optimization. *Optimization Methods and Software*, 13(3):203–226.

Pareto, V. (1896). *Cours d'économie politique, Rouge.* Lausanne, Switzerland.

Roskam, J. (2005). *Airplane design, Part I to VIII.* Design, Analysis and Research Corporation (DARcorporation).

Siarry, P. and Michalewicz, Z. (2008). *Advances in metaheuristics for hard optimization.* Natural Computing Series. Springer.

Simpson, T. W. (1998). Comparison of Response Surface and Kriging Models in the Multidisciplinary Design of an Aerospike Nozzle. Technical Report TR-98-16, Langley Research Center, NASA.

Sobieski, I. P. and Kroo, I. M. (2000). Collaborative Optimization Using Response Surface Estimation. *AIAA Journal*, 38(10):1931–1938.

Srinivas, N. and Deb, K. (1994). Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248.

Torenbeek, E. (1982). *Synthesis of subsonic airplane design*, chapter General aspects of aircraft configuration development. Delft University Press, Kluwer Academic Publishers.

Venkatraman, S. and Yen, G. (2005). A generic framework for constrained optimization using genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 9(4):424–435.

Zhou, J. L., Tits, A. L., and Lawrence, C. T. (1997). User's Guide for FFSQP Version 3.7: A FORTRAN Code for Solving Constrained Nonlinear (Minimax) Optimization Problems, Generating Iterates Satisfying All Inequality and Linear Constraints. Technical Report TR-92-107r2, Systems Research center, University of Maryland, College Pard, MD 20742.

Zitzler, E. and Thiele, L. (1999). Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271.