Applications of a Multi-Objective Genetic Algorithm to Engineering Design Problems

Johan Andersson

Department of Mechanical Engineering
Linköping University
581 83 Linköping, Sweden
{johan@ikp.liu.se}

Abstract. This paper presents the usage of a multi-objective genetic algorithm to a set of engineering design problems. The studied problems span from detailed design of a hydraulic pump to more comprehensive system design. Furthermore, the problems are modeled using dynamic simulation models, response surfaces based on FE–models as well as static equations. The proposed method is simple and straight forward and it does not require any problem specific parameter tuning. The studied problems have all been successfully solved with the same algorithm without any problem specific parameter tuning. The resulting Pareto frontiers have proven very illustrative and supportive for the decision-maker.

1. Introduction

Many real-world engineering design problems involve simultaneous optimization of several conflicting objectives. In many cases, multiple objective problems are aggregated into one single overall objective function. However, design engineers are often interested in identifying a Pareto optimal set of alternatives when exploring a design space. Pareto optimality is defined as a set where every element is a problem solution for which no other solutions can be better in all design attributes. For the two-dimensional case, the Pareto front is a curve that clearly illustrates the trade-off between the objectives.

The objective of this paper is to present a multi-objective genetic algorithm and describe how it has been applied to a variety of real world applications, without any problem specific parameter tuning. The paper begins by discussing engineering design and its similarities to an ordinary optimization process. We go on to discuss genetic algorithms in general and a multi-objective genetic algorithm (MOGA)in particular. Thereafter, a set of different design problems are studied with the help of the proposed optimization strategy. These problems include design of a hydraulic actuation systems, detail design of a hydraulic pump, a crashworthiness design problem as well as the problem of determining which functionality to include in a mechatronic system. Finally the results are summarized in the conclusions.

1.1. Optimization and engineering design

Engineering design is a special form of problem solving where a set of frequently unclear objectives has to be balanced without violating any given constraints. Furthermore, the design process is an iterative process as have been stated by several authors, e.g. Ulrich and Eppinger [18] and Roosenburg and Eekels [15]. According to Roosenburg and Eekels, the iterative part consists of *analysis, synthesis, simulation, evaluation* and *decision*. For each provisional design the expected properties are compared to the criteria. If the design does not meet the criteria it is modified and evaluated again in the search for the best possible design. From this it could be seen that design is essentially an optimization process, as stated already in 1967 by Simon [17]. Therefore, it seems natural to look upon a design problem as an optimization problem. By employing modern modeling, simulation and optimization techniques, vast improvements could be achieved in design. However, there will always be parts of the design process that require human or inquantifiable judgment that is not suited for automation with any optimization strategy.

Figure 1 below depicts a system design process from [1] where modeling, simulation and optimization are introduced to support and speed up the design process. In the proposed system design process, the iterative part of the design process is formalized and partly automated with the help of an optimization algorithm.

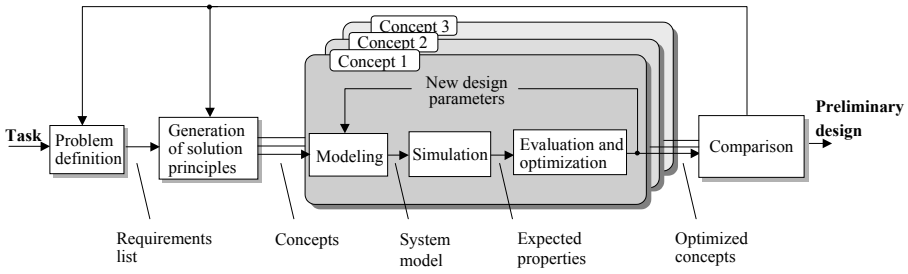


Fig. 1. System design process.

The 'problem definition' in Figure 1 results in a requirements list which is used in order to generate different solution principles/concepts. Once the concepts have reached a sufficient degree of refinement, modeling and simulation are employed in order to predict the properties of particular system solutions. Each solution is evaluated with the help of an objective function, which acts as a figure of merit. Optimization is then employed in order to automate the evaluation of system solutions and to generate new system proposals. The process continues until the optimization is converged and a set of optimal systems are found. One part of the optimization is the evaluation of design proposals. The second part is the generation of new, and hopefully better designs. Thus, optimization consists of both analysis (evaluation) and synthesis (generation of new solutions).

Often the first optimization run does not result in the final design. If the optimization does not converge to a desired system, the concept has to be modified or the problem reformulated, which results in new objectives. In Figure 1 this is visualized

by the two outer loops back to 'generation of solution principles' and 'problem definition' respectively.

Naturally the activity 'generation of solution principles' produces a number of conceivable concepts, which each one is optimized. Thus each concept is brought to maximum performance; optimization thereby provides a solid basis for concept selection. This will be illustrated in a study of hydraulic actuation systems.

One essential aspect of using modeling and simulation is to understand the system we are designing. The other aspect is to understand our expectations on the system, and our priorities among the objectives. Both aspects are equally important. It is essential to engineering design to manage the dialog between specification and prototype. Often simulations confirm that what we wish for is unrealistic or ill conceived. Conversely, they can also reveal that our whishes are not imaginative enough.

2. Genetic algorithms

Genetic algorithms are modeled after mechanisms of natural selection. Each optimization parameter (x_n) is encoded by a gene using an appropriate representation, such as a real number or a string of bits. The corresponding genes for all parameters $x_1,...x_n$ form a chromosome capable of describing an individual design solution. A set of chromosomes representing several individual design solutions comprise a population where the most fit are selected to reproduce. Mating is performed using crossover to combine genes from different parents to produce children. The children are inserted into the population and the procedure starts over again, thus creating an artificial Darwinian environment. For a general introduction to genetic algorithms, see work by Goldberg [8].

Additionally, there are many different types of multi-objective genetic algorithms. For a review of genetic algorithms applied to multi-objective optimization readers are referred to work by Deb [6].

2.1. The proposed method

In this paper the multi-objective struggle genetic algorithm (MOSGA) [4] is used for the Pareto optimization. MOSGA combines the struggle crowding genetic algorithm presented by Grueninger and Wallace [10] with Pareto-based ranking as devised by Fonseca and Fleming [7]. As there is no single objective function to determine the fitness of the different individuals in a Pareto optimization, the ranking scheme presented by Fonseca and Fleming is employed, and the “degree of dominance” in attribute space is used to rank the population. Each individual is given a rank based on the number of individuals in the population that are preferred to it, i.e. for each individual the algorithm loops through the whole population counting the number of preferred individuals. “Preferred to” is implemented in a strict Pareto sense, but one could also combine Pareto optimality with the satisfaction of objective goal levels, as discussed in [7]. The principle of the MOSGA algorithm is outlined below.

Step 1: Initialize the population.
Step 2: Select individuals uniformly from population.
Step 3: Perform crossover and mutation to create a child.
Step 4: Calculate the rank of the new child.
Step 5: Find the individual in the entire population that is most similar to the child. Replace that individual with the new child if the child's ranking is better, or if the child dominates it.
Step 6: Update the ranking of the population if the child has been inserted.
Step 7: Perform steps 2-6 according to the population size.
Step 8: If the stop criterion is not met go to step 2 and start a new generation.

Step 5 implies that that the new child is only inserted into the population if it dominates the most similar individual, or if it has a lower ranking, i.e. a lower "degree of dominance". Since the ranking of the population does not consider the presence of the new child it is possible for the child to dominate an individual and still have the same ranking. This restricted replacement scheme counteracts genetic drifts and is the only mechanism needed in order to preserve population diversity. Furthermore, it does not need any specific parameter tuning. The restricted replacement strategy also constitutes an extreme form of elitism, as the only way of replacing a non-dominated individual is to create a child that dominates it.

The similarity of two individuals is measured using a distance function. The method has been tested with distance functions based upon the Euclidean distance in both attribute and parameter space. A mixed distance function combining both the attribute and parameter distance has been evaluated as well. The result presented in this paper was obtained using an attribute based distance function. As can be seen from the description of the method there are no algorithm parameters that have to be set by the user. The inputs are only: population size, number of generations, genome representation and crossover and mutation methods, as in every genetic algorithm.

3. Design examples

This section describes a set of engineering design problems which have been studied with the proposed method. It shall be pointed out that each problem has been solved without tuning any algorithm parameters. Real parameters are always real encoded whereas for the combinatorial problem we use binary encoding. Furthermore blend crossover has been used for both crossover and mutation of real encoding variables, whereas one point crossover and flip mutation have been used for binary genomes. All problems have been solved with a population size of 40 individuals which has been run for 400 generations.

3.1. Hydraulic Actuation systems

The objects of study for this design problem are two different concepts of hydraulic actuation systems. Both systems consist of a hydraulic cylinder that is connected to a

mass of 1000 kilograms. The objective is to follow a pulse in the position command with a small control error and simultaneously obtain low energy consumption. Naturally, these two objectives are in conflict with each other. The problem is thus to minimize both the control error and the energy consumption from a Pareto optimal perspective.

In the first more conventional system, the cylinder is controlled by a servo valve, which is powered from a constant pressure system. In the second concept, the cylinder is controlled by a servo pump. Thus, the systems have different properties. The valve concept has all that is required for a low control error, as the valve has a very high bandwidth. On the other hand, the valve system is associated with higher losses, as the valve constantly throttles fluid to the tank.

The different concepts have been modeled in the simulation package Hopsan [11]. The models of each component consist of a set of algebraic and differential equations taking aspects such as friction, leakage and non-linearities into account. The system models are depicted in Figure 2.

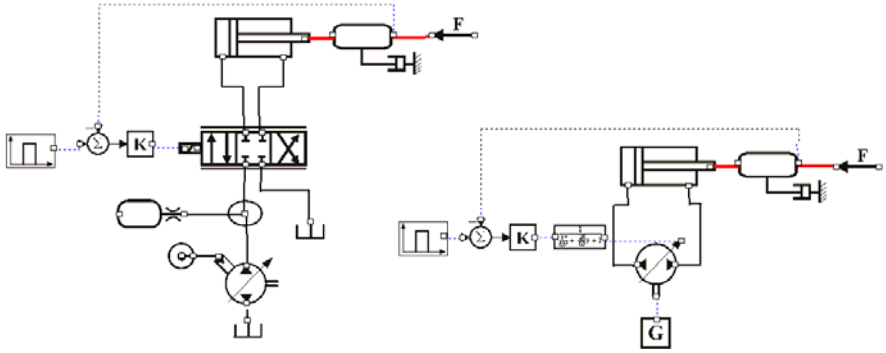


Fig. 2. Two different concepts of hydraulic actuation systems, left is the servo valve system and right is the servo pump system.

The servo valve system consists of the mass and the hydraulic cylinder, the servo valve and a p-controller that is controlling the motion. The servo valve is powered by a constant pressure pump and an accumulator, which keeps the system pressure at a constant level. The optimization parameters are the sizes of the cylinder, valve and the pump, the pressure lever, the feedback gain and a leakage parameter that is necessary to dampen the system. Thus, this problem consists of six optimization parameters and two objectives.

The servo pump concept contains fewer components, the cylinder and the mass, the controller and the pump. A second order low-pass filter is added in order to model the dynamics of the pump. The servo pump system consists of only four optimization parameters.

The optimization is based on component size selection rather then component design, i.e. it is assumed that each component is a predefined entity. As a consequence of this assumption most component parameters are expressed as a function of the component size. Both systems where optimized in order to simultaneously minimize

the control error f_1 and the energy consumption f_2. The control error is obtained by integrating the absolute value of the difference between reference and actual cylinder position, whereas the energy consumption is calculated by integrating the product of flow and the pressure difference over the pump.

As the Pareto optimization searches for all non-dominated individuals, the final population will contain individuals with a very high control error, as they have low energy consumption. It is possible to obtain an energy consumption close to zero if the cylinder does not move at all. However, these solutions are not of interest, as we want the system to follow the pulse. Therefore, a goal level for the control error is introduced. The ranking scheme is modified so that solutions, which are bellow the goal level for the control error are always preferred to solutions that are above it regardless of their energy consumption, as described by Fonseca and Fleming in [7]. In this manner, the population is focused on the relevant part of the Pareto front.

In order to achieve fast systems, and thereby low control errors, large pumps and valves are chosen by the optimization strategy. A large pump delivers more fluid, which enables higher speed of the cylinder. However, bigger components consume more energy, which explains the shape of the Pareto fronts in Figure 3. This problem has been analyzed in more detail in [2]. Furthermore, in [3] the problem was extended to include a mixture of real parameters and selection of valves and cylinders from catalogues.

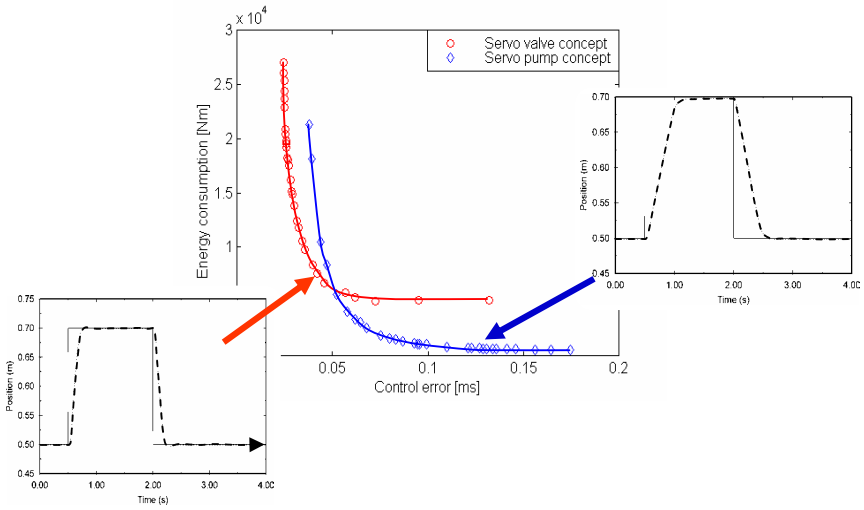


Fig. 3. Pareto front showing the trade-off between energy consumption and control error for the two concepts. Furthermore, the graph on the right shows the simulation result for a slow pulse response, whereas the graph on the left shows a fast pulse response.

In figure 3 the obtained Pareto fronts for the two concepts are plotted in the same graph. In this manner, Pareto optimization constitutes an excellent support for concept selection, as it brings forth the properties of the different concepts. It is evident

that the final design should preferably be on the overall Pareto front, which elucidates when to change between concepts. This is a very illustrative way of presenting the results from a multi-objective optimization. In this particular design example, the servo pump system consumes less energy, and is preferred if a control error larger then 0.05ms is acceptable. The servo valve system on the other hand, is fast but consumes more energy. If lower control error then 0.05ms is desired, the final design should preferably be a servo valve system.

3.2. Detail design of a hydraulic piston pump.

One of the most important origins of noise and vibrations in hydraulic systems is the system pressure ripple. The system pressure ripple is a hydraulic response to introduced flow pulsations, of which the hydrostatic pump is a major source. Consequently, in order to lower the noise from hydraulic systems, the flow transients created by the pump must be reduced. The pump flow ripple constitutes two principal parts; the kinematic flow ripple, due to the limited number of pumping elements, and the compressible flow ripple, due to the compressibility of the fluid. At high pressure levels, the compressible flow ripple is normally the clearly dominating kind. However, with very small design modifications, the compressible flow ripple can be changed considerably, both with regard to amplitude and frequency content. Perhaps the most obvious measure to reduce the compressible flow transient is to equalize the cylinder pressure to the supply port pressure before the cylinder is connected to the supply port.

This is called pre-compression, and produces a rather satisfying flow ripple reduction if designed correctly. There are more refined ways of achieving this pressure equalization, for example the pressure relief groove explained in [13] and the pre-compression filter volume, see [14]. The exact design of these features can rather easily be tuned to minimize flow ripple at a specific operational condition, i.e. a certain displacement angle, rotational speed and discharge and inlet pressure levels. However, as the conditions are changed, the optimum drifts away, implying that a pump optimised for a certain condition may give severe flow ripple at other conditions. In this section a design feature called cross-angle, which reduces the pump's sensitivity to variations in displacement angles is analysed.

The cross-angle is a small (1-4°) fixed displacement angle around the axis perpendicular to the trunnion axis, see [12]. In practice, the cross-angle results in that the additional pre-compression required for achieving optimal pressure equalization is obtained for a wide range of displacement angles.

The pump studied is a seven piston, 40 cm^3, in-line axial piston pump at 1500 rpm with discharge and inlet pressure levels at 250 and 2 bar respectively. The pump has been studied using a very detailed simulation model developed in the HOPSAN simulation environment [11]. The accuracy of the model has been experimentally verified in for example work by Petterson [14]. In the model a large number of different states are available, for example cylinder pressure, flow from each cylinder and piston forces. In the previous design example the time resolution was in milliseconds, in this example it is microseconds.

There are several objectives that have to be considered regarding the minimization of pump related noise. Traditionally, the flow ripple peak-to-peak value has been the obvious objective, which is justified since it correlates well to the system pressure ripple amplitude. However, when running an optimisation, focusing solely on the flow ripple, unreasonable cylinder pressure-peaks may occur. Since the cylinder pressure is directly proportional to the piston forces, this will have a direct impact on the excitation of the pump casting vibrations and thus noise, i.e. the maximal cylinder pressure p_{max} should be below a certain limit p_{lim}. In addition, it is important to avoid cavitation in order to obtain low noise level and a long life of the pump, i.e the minimal system pressure p_{min} should be above a certain value p_{cav}. Thus, the optimisation problem is defined as:

$$
\begin{array}{lll}
\text{Minimize} & f_1(\mathbf{x}) & \textit{discharge flow ripple} \\
\text{Minimize} & f_2(\mathbf{x}) & \textit{inlet flow ripple} \\
\text{Subjected to} & g_1(\mathbf{x}) = p_{\min}(\mathbf{x}) \geq p_{cav} & \textit{cavitation constraint} \\
 & g_2(\mathbf{x}) = p_{\max}(\mathbf{x}) \leq p_{\lim} & \textit{pressure-peak constraint} \\
 & x_i^l \leq x_i \leq x_i^u & \textit{design parameter limits}
\end{array}
\qquad (1)
$$

For this application the cavitation and pressure-peak constraints are added to the objective functions with the help of penalty functions. For each constraint, there is a penalty function that equals to zero if the constraint is not violated. The penalty function then increases exponentially with the degree of constraint violation. The sum of the penalty functions are finally added to both objectives. With this problem formulation a set of non-dominated individuals that do not violate any constraints are identified as shown in Figure 4.

In order to assure a low noise level for a wide range of displacement angles, the simulation model is executed at different displacement angles between zero and maximum displacement for each optimisation iteration. The flow peak-to-peak values for each displacement angle are summed to a total flow pulsation measure.

Furthermore, this problem has also been solved as a single objective problem, with $f_1(\mathbf{x})$ as well as $f_2(\mathbf{x})$ as the single objective using the same constraint formulation. It was then concluded that optimal discharge and inlet performance require different cross angles, as can also be seen in the Pareto front in Figure 4. The single objective problem was solved using the Complex method, see Box [5]. The Complex method is a non-gradient method that has been successfully used in other studies too; see [1] and [12]. Solving the single objective problem yielded the same results as the extreme values on the Pareto front. This is very encouraging since two different optimization methods have given the same optimal results, which increases the confidence that the true optima has really been found. However, as always when using this type of optimization methods, optimality of the final solutions can not be proven.

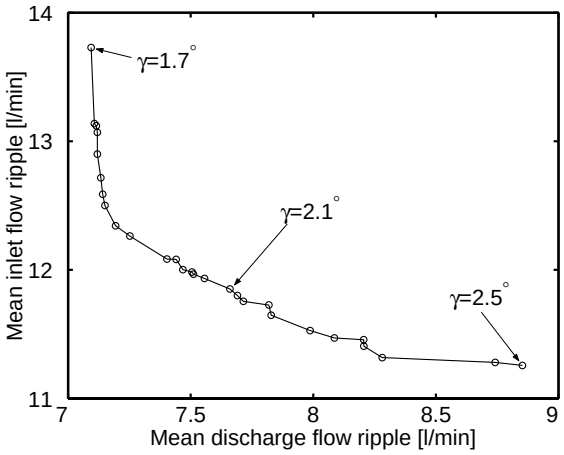
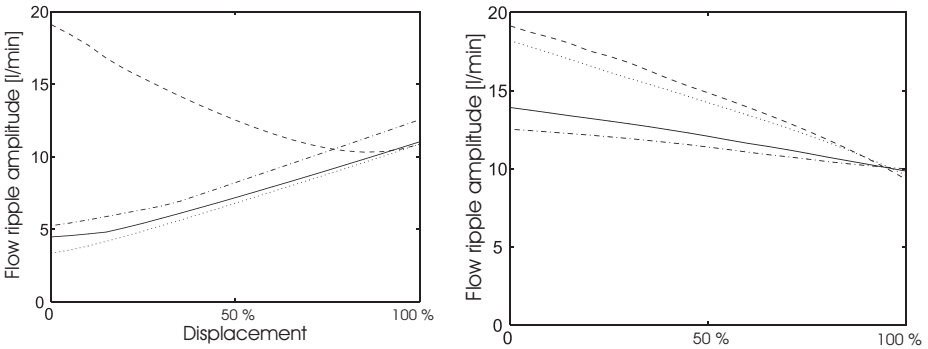


Fig. 4. Pareto front showing the trade-off between optimal discharge and inlet performance.

The multi-objective genetic algorithm identifies the Pareto optimal front that visualizes the trade-off between the two objectives, see Figure 4. Along the x-axis, the mean discharge flow ripple amplitude from simulations at three different displacements (*Dp=100%*, *Dp=50% and Dp=0%*) is displayed, whereas the inlet mean flow ripple is displayed along the y-axis.

With a cross-angle of 1.7°, optimal discharge performance is obtained with mean discharge flow ripple amplitude of approximately 7.1 l/min, while the inlet performance is rather poor with mean flow ripple amplitude of 13.7 l/min. A cross-angle of 2.5°, on the other hand, implies optimal inlet performance with mean flow ripple amplitude of 11.2 l/min, while the discharge mean flow ripple is 8.8 l/min. By selecting a point in the middle of the curve, a fair compromise point where $\gamma = 2.1°$ is obtained. By choosing this point instead of $\gamma = 1.7°$, the mean inlet flow ripple is improved with 1.9 l/min (approximately 14% better), while the discharge flow ripple becomes only 0.5 l/min worse (7.0% worse). If the compromise point is chosen instead of $\gamma = 2.5°$, the gain in discharge flow ripple is 1.2 l/min (approximately 14% better) while the inlet flow ripple is deteriorated with only 0.5 l/min (4.4% worse). Altogether, it can be seen that the total gain from choosing the compromise point instead of the end-points is consequently higher than the loss.

Figures 5(a) and 5(b) show the simulated discharge and inlet flow peak-to-peak value for different displacement angles, for ordinary pre-compression ($\gamma = 0°$,), optimal discharge performance, optimal inlet performance and the compromise point. The diagrams constitute an illustrative way of describing the practical implication of the trade-off between Pareto optimal solutions. As can be seen the compromise point is just slightly worse then the individual optima in both objectives. However, the individual optima are much worse in the other objective. This is evident in figure 5(b) where the compromise solution (solid line) is close to the optimum for inlet performance, whereas the individual optimum for discharge performance has a very poor inlet performance.



(a) Optimal discharge performance

(b) Optimal inlet performance

Fig. 5. Flow ripple sensitivity to changes in displacement angle. Ordinary pre-compression without any cross-angle (γ=0°) is shown as the dashed line. Optimal discharge performance (γ=1.7°) is obtained as the dotted line, while optimal inlet performance (γ=2.5°) is shown by the dash-dotted line. The solid line represents the compromise solution (γ=2.1°).

3.3. Crashworthiness design example

This section presents an example where the multi objective optimization technique is used together with response surface methods in order to support crashworthiness design. Here the conflicting objectives are exemplified by the desire to minimize the intrusion into the passenger compartment area and simultaneously obtain low maximum acceleration during vehicle impact. These two objectives are naturally conflicting, since low acceleration implies large intrusion.

The problem is solved by first creating quadratic response surfaces that captures the global performance of the objectives using a D-optimal experimental design setup. The crash behaviour of the vehicle is studied with a comprehensive FE model, see [15]. The FE model of the vehicle was a sub-model of a complete FE vehicle model consisting of 56.000 shell elements. The vehicle model impacts into a rigid wall with an initial velocity of 30 miles per hour (56 km/h) and the impact is simulated during 100 ms. The impact event is solved using the FE-code LS-DYNA in a LINUX cluster. Sheet thicknesses of parts such as the crash-box, midrail–closingplate, midrail-C-profile, rail-extension and the upper-rail, were used as design parameters. The FE model is evaluated in a set of experimental design points in order to establish quadratic response surfaces for both the intrusion and the acceleration.

Based on the two quadratic response surfaces a Pareto optimization was performed using the multi objective genetic algorithm. The outcome of this optimization is the Pareto front shown in Figure 6. By studying the trade-off among the Pareto optimal solutions, the final design is chosen at a point where a fair compromise between intrusion and maximum acceleration is obtained.

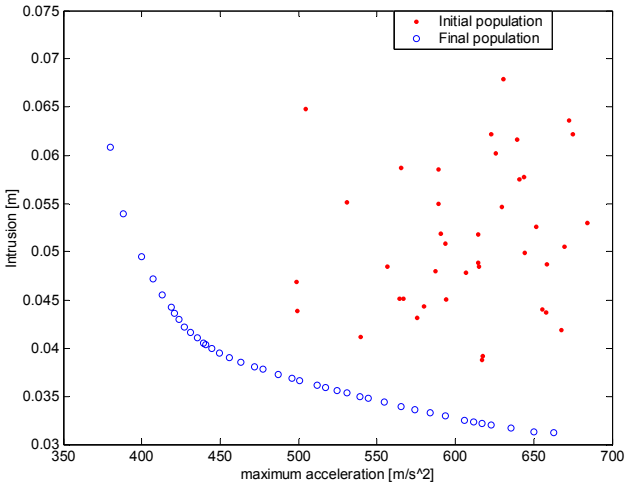


Fig. 6. Pareto front showing the trade-off between intrusion and maximum acceleration in the crashworthiness design problem.

3.4. Optimization of the product specification for mechatronic systems

In this section a method is presented that supports the designer in determining which functionality that should be implemented in a product and which should not. The proposed method identifies the set of customer functions and technical implementations that maximize the possible product profit. The customer functions represent the functionality of the product, and the technical implementations are the hardware and software components needed to realize these functions. For industrial applications, the numbers of possible combinations of customer functions and technical implementations are extremely large.

The purpose of the method is to maximize the possible profit by selecting the optimal functionality to be designed into a mechatronic product with an open architecture. The profit consists of two parameters, value and cost, where the cost is subtracted from the value. The value is the amount the customers are prepared to pay for the set of customer functions in the product and the cost is the sum of the costs of implementing the technical implementations needed to realize the customer functions in the product. The assumption is that each customer function can be implemented separately and adds customer value. This is not to say that the different customer functions are independent since they can share technical implementations.

The customer functions are represented in the customer function vector (**CF**) and the technical implementations in the technical implementation vector (**TI**). A coupling vector $\mathbf{cv_i}$ expresses which technical implementations are necessary in order to realize the customer function CF_i. Each element of the coupling vector could be either one, representing that the corresponding technical implementation are needed, or zero

otherwise. The coupling vectors for all customer functions, $\mathbf{cv_i}, i = 1...m$, make up the coupling matrix **CM**. Thus the problem could be described according to equation (2)

$$\mathbf{CF} = \mathbf{CM} \cdot \mathbf{TI} \qquad \text{i.e.} \begin{bmatrix} CF_1 \\ \cdots \\ \cdots \\ CF_m \end{bmatrix} = \begin{bmatrix} - & \mathbf{cv_1} & - \\ & \cdots & \\ & \cdots & \\ - & \mathbf{cv_m} & - \end{bmatrix} \begin{bmatrix} TI_1 \\ \cdots \\ \cdots \\ TI_n \end{bmatrix} \tag{2}$$

where *m* is the number of potential customer functions and *n* the number of technical implementations. Each different combination of technical implementations yields one possible solution to the problem, or one concept. For a problem with *n* technical implementations there exist 2^n different concepts. A particular concept **X**, is expressed by a vector $\mathbf{X} = [x_1, x_2, ...x_n]$, where x_i can be either one if the technical implementation $\mathbf{TI_i}$ is in the concept or zero otherwise.

In order to calculate the value for a specific concept, the customer functions that are possible to realize with the concept's technical implementations have to be found. The function realization vector, **W**, represents this. **W** is calculated according to equation (3).

$$\mathbf{W}(\mathbf{X}) = \begin{bmatrix} w_1 \\ \cdots \\ \cdots \\ w_m \end{bmatrix}, \text{where } w_i(\mathbf{X}) = \left\lfloor \frac{\mathbf{cv}_i \bullet \mathbf{X}^T}{\mathbf{1}^T \bullet \mathbf{cv}_i{}^T} \right\rfloor \tag{3}$$

The notation $\lfloor a \rfloor$ denotes the largest integer less than or equal to *a*. $\mathbf{cv}_i \bullet \mathbf{X}^T$ represents the number of the necessary technical implementations for customer function *i* that are included in concept **X**. This number is divided by the sum of all functions needed in order to implement customer function *i,* i.e. $\mathbf{1}^T \bullet \mathbf{cv}_i{}^T$, where $\mathbf{1}^T$ is a vector of ones. If **X** contains all functions needed by CF_i this quotient equals 1, otherwise it is lees then one. Thus equation (3) returns 1 only for the customer functions that are implemented by **X**. The total value, of a concept is calculated by summing up the customer value for each customer function realized by concept **X**.

The cost of implementing each technical implementation is represented by the implementation cost vector **IC**, $\mathbf{IC}^T = [ic_1, ic_2, ..., ic_n]$, where each implantation cost, ic_i, is made up of the development, material and production cost. The total cost, *c*, for the concept is thus simply obtained by multiplying **X** and **IC**.

In this simplified model the profit, *p*, for the company is expressed as the value the customer is prepared to pay for a particular concept minus the cost of developing and producing it. The problem could thus be described as to find the concept **X** that maximizes the profit $p(\mathbf{X})$ without exceeding the development budget. The objectives are then to maximize the profit and simultaneously minimize the cost, see equation (4).

$$\begin{aligned} &\max_X \ p(\mathbf{X}) \\ &\min_X \ c(\mathbf{X}) \\ &\text{s.t.} \quad \mathbf{X} = [x_1, x_2, ..., x_n] \\ &\qquad x_i = 0 \vee 1, \forall i \{1, 2, ..., n\} \end{aligned} \tag{4}$$

The method has been evaluated on an industrial case study of active safety system performed at Volvo Cars. The size of the problem is rather impressive, with 51 customer functions and 46 technical implementations, there are $7 \cdot 10^{13}$ different possible solutions to the problem. The resulting Pareto front is shown in figure 7. As can be seen it is possible to increase the profit by increasing the development budget. In reality there is however always a restriction on the development budget. Thus this type of optimization could be used as an argument when negotiating the development budget.

Based on this case study the proposed method shows a substantial profit potential compared to the methods presently used, see Figure 7. The method used today is experts choosing the concepts they "believe in", using the same data but disregarding the sharing of technical implementations by customer functions. This is necessary because it is impossible to grasp all the feasible combinations. The experts might however take other issues into account which are not accounted for in this simple model.

Figure 8 shows the optimal set of customer functions that should be implemented in the product depending on the development budget. As can be seen the number of functions increase as the budget is enlarged. For a more detailed description of this application readers are referred to [9].

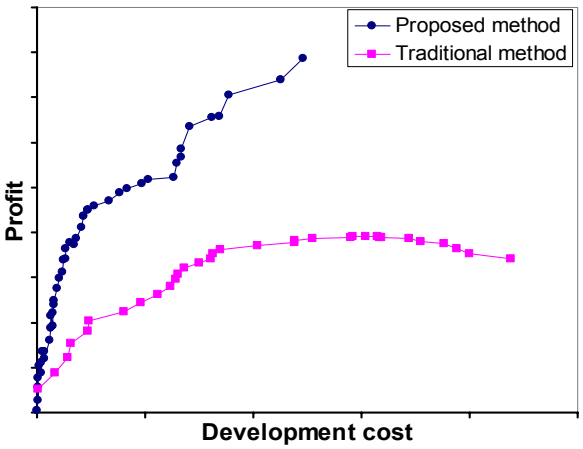


Fig. 7. Pareto fronts obtained by the proposed method, above, and the traditional method, below. As can be seen the profit could be increased by increasing the development budget. An interesting result is that the experts keep selecting functions even when the profit decreases.

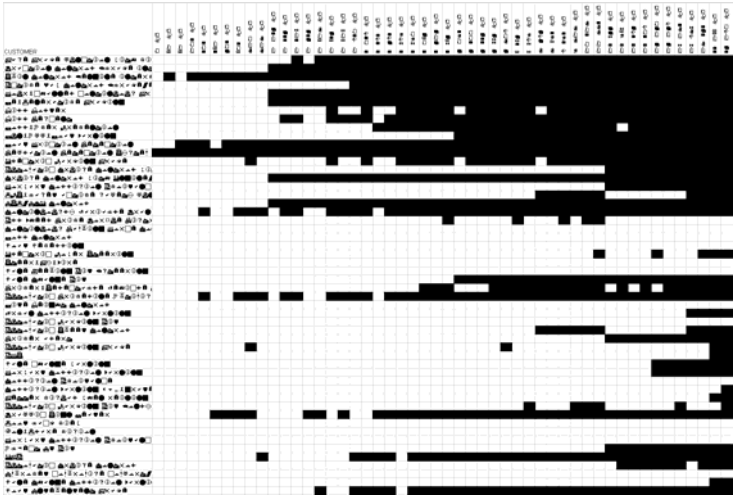


Fig. 8. Visualization of the combinations of customer functions which yield the highest profit for different development budgets. As can be seen the number of functions increases as the budget increases. Due to confidentiality, the names of the functions can not be shown.

4. Discussion and conclusion

This paper has shown how a simple MOGA could be used to support engineering design. The MOGA used in this paper seems very robust and it requires few algorithm specific parameters. The method has been applied to a wide range of engineering optimization problems spanning from detailed design via more comprehensive system design to overall determination of the product specification. The intention of the paper has been to exemplify how a simple MOGA could solve a set of very different real world problems without any problem specific parameter tuning. For a more detailed discussion of the problems studied, the reader is referred to the references listed. Numerical properties of the proposed MOGA could be found in reference [4].

It has also been shown how Pareto optimization could be a useful tool for concept selection. From an engineering perspective the goal of the optimization is not only finding "the optimal" solution, but gaining insight about the properties of the system being designed and the behavior of the system model. Another important lesson is defining the objectives, which forces the designer to make clear what is desired of the system and then to challenge the preferences of the decision-maker by visualizing the trade-off between the objectives. Therefore, what is crucial for a method that should be applied to engineering design optimization is that it is simple and yet robust, and that it could search vast design spaces and produce near optimal solutions that covers the whole Pareto front. The proposed multi-objective genetic algorithm is a good example of such a method.

5. Acknowledgements

The author whishes to thank the fellow researchers; Professor Petter Krus, Christian Grante, Andreas Johansson and Marcus Redhe for sharing their specific problems and their expertise. Part of this work has been financially supported by the Swedish Strategic Research Foundation through the ENDREA program. This support is gratefully acknowledged.

References

1. Andersson J., Multiobjective Optimization in Engineering Design – Application to Fluid Power Systems, Dissertation, Thesis No. 675, Linköping University, Sweden, (2001).
2. Andersson J., Krus P. and Wallace D., "Multi-objective optimization of hydraulic actuation systems", ASME Design Automation Conference, Baltimore, September 11-13, (2000).
3. Andersson J. and Krus P., Multiobjective Optimization of Mixed Variable Design Problems, in Proceedings of 1st International Conference on Evolutionary Multi Criteria Optimization, Zurich, Switzerland, March 7-9, (2001).
4. Andersson J. and Wallace D., Pareto optimization using the Struggle Genetic Crowding Algorithm, Engineering Optimization, vol. 34, No. 6, pp. 623-643, (2002).
5. Box M. J., "A new method of constraint optimization and a comparison with other methods", Computer Journal 8:42-52, (1965).
6. Deb K., Multi-objective Objective Optimization using Evolutionary algorithms, Wiley and Sons Ltd, (2001).
7. Fonseca C. M. and Fleming P. J., "Multiobjective optimization and multiple constraint handling with evolutionary algorithms - Part I: a unified formulation," IEEE Transactions on Systems, Man, & Cybernetics Part A: Systems & Humans, vol. 28, pp. 26-37, (1998).
8. Goldberg D., "Genetic Algorithms in Search and Machine Learning. Reading, Addison Wesley, (1989).
9. Grante C. and Andersson J., Optimisation of Design Specifications for Mechatronic Systems, submitted for publication, (2002).
10. Grüniger T. and Wallace D., Multi-modal optimization using genetic algorithms", Technical Report 96.02, CADlab, Massachusetts Institute of Technology, Cambridge, (1996).
11. Hopsan, , "Hopsan, a simulation package - User's guide", Technical report LiTH-IKP-R-704, Department of Mechanical Engineering, Linköping University, Linköping, Sweden.
12. Johansson A., Andersson J. and Palmberg J.-O., Optimal Design of the Cross-Angle for Pulsation Reduction in Variable Displacement Pumps, In Bath Workshop of Power Transmission and Motion Control, University of Bath, UK, September, (2002).
13. Palmberg J-O., Modelling of flow ripple from fluid power piston pumps. In 2nd Bath International Power Workshop, University of Bath, UK, September, (1989).
14. Pettersson M., Design of fluid power piston pumps, with special reference to noise reduction, PhD thesis, Dissertations No. 394, Department of Mechanical Engineering, Linköping University, Linköping, Sweden, (1995).
15. Redhe M., Simulation based design – structural optimization in early design stages, Licentiate Thesis, Dept. of Mech. Eng., Linköping University, Linköping, Sweden, (2001).
16. Roozenburg N. and Eekels J., Product Design: Fundamentals and Methods, John Wiley & Sons Inc, (1995).
17. Simon H., The Sciences of the Artificial, MIT Press, (1969).
18. Ulrich K. and Eppinger D., Product Design and Development, second edition, McGraw-Hill Companies, (1995).