INTERNATIONAL DESIGN CONFERENCE - DESIGN 2018
https://doi.org/10.21278/idc.2018.0323

A METAHEURISTIC FOR SOLUTION SPACE MODELLING

B. Poulain, T. Naumann, J. Stal-Le Cardinal and J. Anderer

Abstract
To confront current market changes, Set-Based Design (SBD) should help carmakers improve time, cost and quality in early design phase. In a development process using ever more computer experiments, building predictive models of the solution space is key to implementing SBD. The present article proposes a new algorithm to build high quality predictive metamodels efficiently and tests it on several benchmark problems. Promising results are obtained. We believe such models could be used for several purpose in design: implementing SBD, optimisation and generating feasible concepts, among others.

Keywords: computational design methods, solution space engineering, simulation based design, set-based design, early design phase

1. Introduction
The automotive industry undergoes important changes. New competitors challenge established ones, customers and regulations push carmakers towards the development of hybrid and fully electrical vehicles, of connected vehicles while asking for ever more personalisation. German carmakers are under pressure. To remain competitive, they must deal with those challenges while reducing costs and time-to-market and improving quality. This is in particular the case for the early design phase of vehicles, urging design organisations on developing more efficient processes, methods and tools.
Still, point-based design is still a prevalent practice in industrial product development processes. Designers begin by selecting a variety of possible solutions, then choose one for further investigation. Then the latter follows a process of evaluations and modifications till it satisfies every discipline's constraints. For complex products which need multidisciplinary evaluations, the process becomes difficult. The general process is described by Ulrich and Eppinger (2012), where they suggest some solutions to make point-based design more efficient: performing more iterations quickly, through faster and more frequent information exchanges as well as decoupling tasks to avoid iterations, by clearly defining interfaces for interacting components so as to design subsystems independently, in parallel. Ward et al. (1995) describe how U.S. carmakers indeed applied those recommendations to improve their development process, along with the development of concurrent engineering.
On the contrary, set-based design (SBD) does not refine one concept iteratively but consider parameter sets instead. SBD was popularised with a series of articles comparing Toyota's set-based concurrent engineering (SBCE) practice to the point-based design applied by U.S. carmakers (Ward et al., 1995; Sobek II et al., 1999; Ford and Sobek II, 2005). Sobek II et al. (1999) describe three main principles of SBCE: (1) map the design space, (2) integrate by intersection, and (3) establish feasibility before commitment (Figure 1). The process, through a broader exploration of the solution space, seems inefficient, and some aspects of Toyota's practice contradict traditional recommendations for point-based design: iterations and communication between stakeholders are slower, and designers

postpone the choice of critical parameters. Yet, those studies suggest that set-based approaches can offer benefits in terms of reductions in costly design iterations, improved design quality and reduced time-to-market.
However, the aforementioned studies provide little guidance on how to implement SBD. The main tool described are the so-called lessons-learned books that describe the current company capability, including feasibility ranges. The determination of those feasibility ranges is however not described.

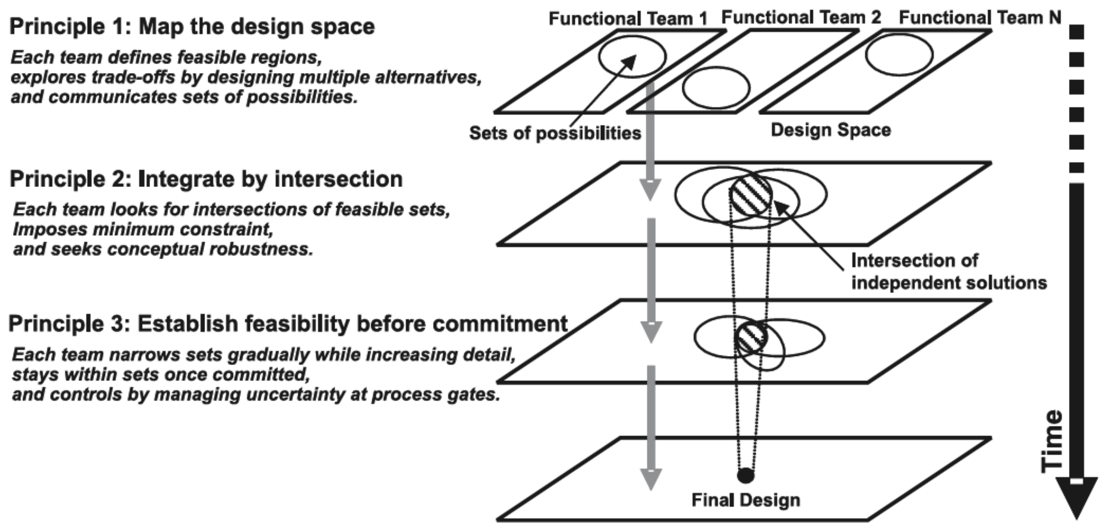
Principle 1: Map the design space
Each team defines feasible regions, explores trade-offs by designing multiple alternatives, and communicates sets of possibilities.
Functional Team 1
Functional Team 2
Functional Team N
Sets of possibilities
Design Space
Principle 2: Integrate by intersection
Each team looks for intersections of feasible sets, Imposes minimum constraint, and seeks conceptual robustness.
Intersection of independent solutions
Principle 3: Establish feasibility before commitment
Each team narrows sets gradually while increasing detail, stays within sets once committed, and controls by managing uncertainty at process gates.
Final Design
Time

Figure 1. Principles of set-based concurrent engineering (Nahm and Ishikawa, 2005)

Furthermore, apart from this lack of guidance, from the authors' experience, it is difficult to implement set-based design in organisations which are mainly used to point-based design. First, knowledge has to be accumulated and formalised on the set of feasible designs, a knowledge which has a different structure from that used in point-based design. This leads to the need of appropriate databases and data structures, such as for instance described in Toepfer and Naumann (2017). Second, designers tend to intuitively prefer to optimise the design of the part they are responsible for, than to explore the overall design space. Besides, organisations seek to reduce time-to-market by accelerating their iteration speed rather than challenging their current methodology.
At a time when Artificial Intelligence techniques and automation are thriving, we would like to use SBD as part of computational design synthesis (CDS) as defined by Campbell and Shea (2014): "CDS is a research area focused on approaches to automating synthesis activities in design. Resulting methods may be fully automated or interactive with the goals of automatically generating a range of alternatives…" We concur with their analysis that "when applied later in the design process, meaningful results are only achievable by interfacing with the computational analysis tools that govern our engineering world, such as those for solving partial-differential field equations or those for solving ordinary-differential equations." Indeed, SBD methods should be able to deal with computationally expensive simulations to present a real interest for industrial applications.
To be more precise, the present paper deals with what Nahm and Ishikawa (2005) dubbed set-based parametric design (SBPD), i.e. when design or performance space are defined by a set of design or performance parameters and their values. Then, evaluation of concepts is made using relevant constraints for each discipline, which may be the results of computationally expensive simulations or optimisations. Methods for implementing set-based parametric design could be beneficial, among others, to the automotive, naval and aeronautical industries.
The rest of the paper is organised as follows: in Section 2 and 3, we present the state-of-the-art of set-based parametric design methods, and state-of-the-art optimisation techniques on which our method is based. Section 4 consists of the description of our algorithm. The evaluation of the developed method on benchmark problems is to be found in Sections 5. Limitations and further developments are discussed in Section 6.

2. State-of-the-art of set-based parametric design methods

The present subsection exposes different methods that were proposed in the past two decades to implement SBPD and especially design space mapping, as well as considerations of their limitations regarding their ability to deal efficiently with simulation data.

A strong research effort has been carried on developing methods that can consider sets, with the possible purpose of mapping the design space, the most well-known ones being fuzzy logics and interval arithmetic, as well as extensions of those. In those tracks, papers such as Yannou et al. (2003), or Trabelsi et al. (2015) present a method based on interval arithmetic and a branch-and-prune algorithm: Constraint satisfaction programming (CSP). CSP in itself is only applicable to problems with explicit analytical expressions with good properties especially that prevent the so-called dependency problem. To tackle the problem of simulation data, Yannou et al. (2003) proposed to use metamodelling techniques to build a metamodel of the constraint functions in order to apply CSP on it. However, CSP presents two difficulties for industrial use. First, to avoid the dependency problem, CSP encourages the use of lower quality metamodels, such as first- or second-order regressions, which are unable to model complex functions. Second, the branch-and-prune algorithm computational cost grows exponentially when the number of design variables increases. Still, the idea of using metamodels for constraint modelling appears to be a promising.

Another technique, proposed by Fender et al. (2017) builds solution boxes (i.e. Cartesian products of design variables' ranges) in which every point satisfies the system's constraints. The method is efficient and precise, yet it is only applicable on linear constraints in order to use linear optimisation algorithms. Those can be obtained either by using first-order regressions or by linearizing the physical problem, but one drawback is the inability to represent complex behaviours accurately. Therefore, the precision of the method is obtained by reducing models' accuracy. This is not satisfying for a general purpose.

While the aforementioned methods assume the use of analytical expressions that can be obtained with metamodels, other authors proposed sampling-based methods to use simulation data without the need for metamodels.

Kizer and Mavris (2014) introduced the Dynamically Constrained SBD methodology. They sample the design space using Design of Computer Experiments (DoCE) methods and then identify the set of feasible points for each constraints to build the convex hull of the feasible space. Then new points near the boundary of the feasible space are sampled, to be able to predict the displacement of the boundary when constraints vary. Though they rely on strong assumptions, namely the fact that constraints define convex feasible spaces, and that one must find several feasible points after the DoCE is performed in order to define a convex hull, the method's ability to deal with constraints perturbations is an important one for industrial practice.

Similarly, Madhavan et al. (2008) propose the SBD Method which is partly based on two levels of Latin Hypercube Sampling (LHS) a DoCE method for which the design space is sampled a first time with LHS, then a second LHS is performed in the neighbourhood of solutions found in the first sample, with the goal to obtain more feasible points. Therefore, that technique suffers from the same drawback as the previous method, which the need for solutions in the first LHS. It is to be expected that for new problems on which designers have little experience or when a large number of design variable is considered, the portion of the solution space in the design space might become negligible. Therefore, this method is not suitable for complex systems design.

Shahan and Seepersad (2009), developed a methodology based on Bayesian networks to map and accumulate knowledge on regions of interest in the design space. This helps designers having a model of the areas that are feasible considering their discipline's constraints. The benefit of these models is that they can be shared between disciplines to find the intersection of the different disciplines' feasible spaces. That ability to share models of feasible spaces or regions of interest is important when multiple teams collaborate. Such models could take the form of solution boxes shared between disciplines, or of more complex models such as Bayesian networks that can be also used for intersection of the disciplines' feasible spaces and can also be used to obtain solution boxes at the system level.

One of the most promising approach found in the literature is the one developed by some BMW researchers, referred as Solution Space Engineering. This approach is related to the two first phases

of SBD (Figure 1). Indeed, after building simplified physical surrogate models, they try to find the largest possible solution box for a given problem. Then intersections can be obtained easily by computing the intersection of boxes (Zimmermann et al., (2017). To achieve this, they created three methods. The first one found in Fender et al. (2016) was already mentioned above. The two other methods are sampling-based. One is semi-manual: starting from a known feasible point, designers sample in an hyperbox (i.e. n-dimensional boxes) in the design space and try to expand that hyperbox iteratively by moving its sides while trying to have no infeasible point in the hyperbox. The other is a stochastic algorithm described and tested on engineering use cases in Zimmermann and von Hoessle (2013), Fender et al. (2014) and Graff et al. (2016). The algorithm starts with searching a feasible point in the design space using classic optimisation algorithm, and then the algorithm sample 100 random points in hyperboxes that move in the design space iteratively. The algorithm stops when a "big" box (the size can be defined by the volume of the box for instance) with 100 feasible points is found, relying on Lehar and Zimmerman's (2012) demonstration to ensure a high probability of having a solution box, independently of the dimension of the solution space. Even though those methods are shown to be effective on high-dimensional problems, one of the requirements is the ability to have very inexpensive physical surrogate models, since a lot of points may be sampled. Zimmermann et al. (2017) mention ~10^4 evaluations for the stochastic algorithm. Much of their research effort has been put on building simplified physical surrogate models.
Considering the methods above, our research question is the following: How can we build metamodels of the solution space with a good quality and a limited number of sample points, when firstly, expensive simulations are used to map the design space, and secondly which can then be used for the semi-manual tool or the stochastic approach? Moreover, it would be interesting for industrial applications that those metamodels can be shared easily, as suggested by Shahan and Seepersad (2009) and also be able to take uncertainty on constraints' levels as suggested by Kizer and Mavris (2014).

3. State-of-the-art of optimisation techniques

Though the presented problem is not an optimisation one, in the sense that the aim is not to optimise an objective function, but rather having a model to predict if a design satisfies constraints or not, recent research streams in optimisation are relevant to our subject. Shan and Wang (2010) carried a survey on modelling and optimisation strategies to solve what they coined High-dimensional, Expensive (computationally), Black-box (HEB) problems. Those problems are indeed the kind of problems encountered during the development of complex systems, since a high number of design variables (more than 20) are to be handled and constraints can be black-box functions, as the result of simulations, measurements or optimisations that can be expensive (minutes, hours, even days of computation time), with no explicit relationship between design variables and simulation results. Among different strategies to tackle those complex problems, one focus is put on Model-Based Design Optimisation (MBDO) techniques. Those techniques consist in building mathematical surrogate-models of the constraint and objective functions with good approximation properties, in order to run well-known optimisation algorithms on those metamodels. This can be done either sequentially or iteratively. In the sequential approach, one first sample points, then build the metamodel and finally run an optimisation on the metamodel. In the so-called direct sampling approach, optimisations' results are used to update the metamodel and re-run optimisations on the updated metamodels, thus improving the model iteratively. This kind of approach are also promising for multiobjective optimisation, as presented in the survey by Tabatabaei et al. (2015). In both cases, those methods are interesting in our context. In fact, one of their aims is to create a good model for constraints in the neighbourhood of either optima or of the Pareto front for single objective and multiobjective optimisation respectively. Since the goal of the present paper is to develop a method which builds a model with a good accuracy of the solution space boundary, MBDO techniques appear to be a sound starting point.
Among the methods explored by optimisation researchers, the COBRA algorithm (Regis, 2014) and its variants presented in Koch et al. (2014, 2015), and Bagheri et al. (2015, 2016a, 2016b) retained our attention. It is able to deal with non-linear constrained problems without using any penalty function. It has also be proven to be efficient on high-dimensional problems such as the Mopta08 benchmark with

124 design variables and 68 constraints (Jones, 2008), using a lower number of evaluations compared to the other state-of-the-art methods. One can also set a margin on the level of constraint fulfilment. This explains why the proposed algorithm to build model is based on COBRA.

4. Metaheuristic description

4.1. General problem

The problem we are dealing with is to build a model predicting if a point in the design space belongs to the solution space or not (Figure 2). We define those terms in the present section. Let $x_i, i = 1, \dots, d$ be the continuous design variables and l_i *and* $u_i, i = 1, \dots, d$ be their lower and upper bounds, respectively. The Design Space (DS) can then be defined as:

$$\Delta = \prod_{i=1}^{d} [l_I,\ u_i] \tag{1}$$

Then we consider inequality constraints, defined as $g_j(x) \leq 0,\ j = 1, \dots, m$. The Solution Space is therefore defined as follows:

$$\Omega = \{x \in \Delta,\ g_j(x) \leq 0,\ j = 1,\ 2,\ \dots,\ m\} \tag{2}$$

Figure 2 illustrate those concepts on a 2 dimensional design problem.

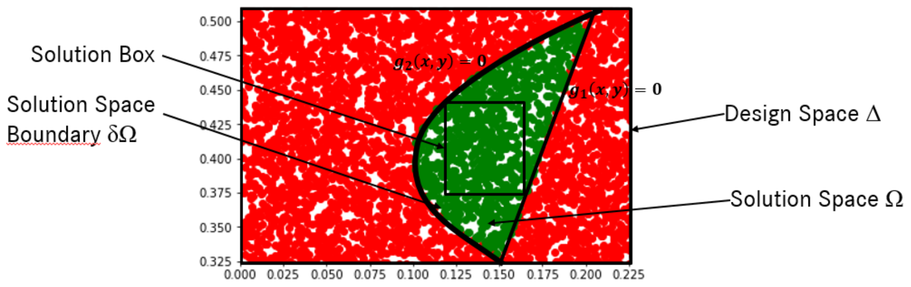

Figure 2. Solution Space Engineering concepts on a 2D design problem

As explained in the previous section, we aim at building a model which predicts if a design satisfies constraints or not. This resolves in creating a model to approximate the following indicator function, i.e. building a binary classifier approximating:

$$1_\Omega(x) = \begin{cases} 1, x \in \Omega \\ 0, x \notin \Omega \end{cases} \tag{3}$$

Equality constraints are not considered here due to the fact that, except for constraints with discrete values, they are likely to reduce the number of dimensions of the solution space, preventing us from finding any solution box in the solution space.

We have two main requirements on the model building process. The first one being that the process should require the lowest number of simulation runs as possible. Indeed, for instance, Jones (2008) explains that for the engineering problem encountered at General Motors, one could hope for a maximum of 60 evaluations per day, meaning it would take a little more than a month just for 1800 evaluations. The second requirement is that the classifier should be of a good quality. We define criteria for our model quality in Section 5.3.

4.2. Algorithm proposal

As mentioned in the literature review, we used COBRA from Regis (2014) as a starting point for our own algorithm. After an initialisation, our algorithm is divided into two phases. Phase I aims at finding a feasible point in the design space. Phase II aims at finding points that can improve the model of the solution space. Our intuition is that points which are well spread in the solution space and which are close to the solution space boundary are likely to improve our metamodel.

The different phases are described in the following subsections. The process is represented in Figure 3.

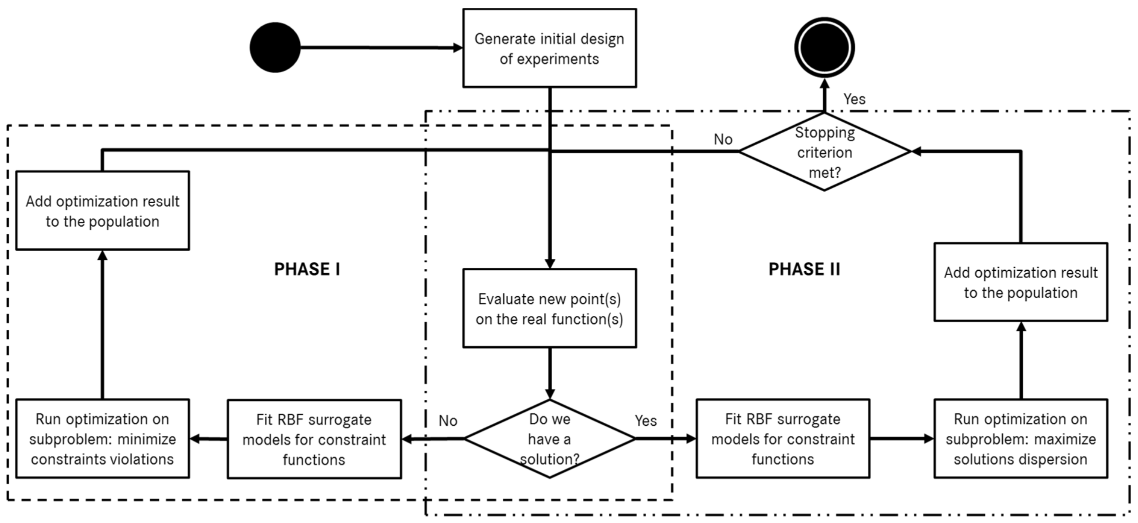
Generate initial design of experiments
Yes
No
Stopping criterion met?
Add optimization result to the population
PHASE I
PHASE II
Evaluate new point(s) on the real function(s)
Add optimization result to the population
Run optimization on subproblem: minimize constraints violations
Fit RBF surrogate models for constraint functions
No
Do we have a solution?
Yes
Fit RBF surrogate models for constraint functions
Run optimization on subproblem: maximize solutions dispersion

Figure 3. Flowchart of the algorithm

4.2.1. Initialisation

After the designers have defined their parameters and their respective ranges, a Design of Computer Experiments (DoCE) is used to generate a sample population to be evaluated. In our case we use a Latin Hypercube Sampling, because it intends to have points spread evenly in the design space. As explained in Regis (2014) the minimum size of the initial sample population shall be higher than $d + 1$, d being the number of design variables. However, Koch et al. (2014) suggest that one may obtain better results with 3*d initial points for optimisation. We decided to keep with the minimum number of points.
After generating an initial sample population by using a DoCE, points are evaluated by computing the constraints for each point with simulations, optimisations or measurements depending on the case. If all points are infeasible, then Phase I starts, otherwise the algorithm moves directly to Phase II.

4.2.2. Phase I

Phase I iterates through the following steps until a feasible point has been found.
First, using the sample population, for each constraint g_i, a metamodel $s^{(i)}$ is built using a Radial Basis Functions (RBF) Network with cubic RBF with a polynomial tail (Powell, 1992). The process for building those metamodels can be found in Regis (2014). Other metamodels could have been used such as Multi-Layered Neural Networks, Support Vector Machines (Cortes and Vapnick, 1995) or Kriging (Krige, 1951). RBF Networks have the advantage of simplicity, even for high-dimensional problems. They can also approximate a function with any arbitrary precision and they bring more information than SVM which are mainly useful for binary data.
Second, an optimisation algorithm is run in order to minimise constraint violations. The optimiser has to handle non-linear constrained optimisation problems. In our case, we use COBYLA, following Koch et al. (2014). The optimisation problem is the following:

$$\min_{x\in\Delta}\begin{cases}\sum_{i=1}^{m}\left[\max\{s^{(i)}(x),0\}\right]^2, if\ \exists i=1,\dots,m, s^{(i)}(x)\geq 0\\ \max\limits_{i=1,\dots,m}\ (s_n^{(i)}(x))\ otherwise\end{cases} \tag{4}$$

Subject to:

$$x \in \Delta \tag{5}$$

$$s^{(i)}(x) + \varepsilon \leq 0, i = 1,\dots,m, \varepsilon \in \mathbb{R}^+ \tag{6}$$

$$\|x - x_j\| \geq \rho, j = 1,\dots,n, \rho \in \mathbb{R}^+ \tag{7}$$

Equation 5 is the constraint for the bounds of the design variables. Equation 6 means the concept has to fulfil for the surrogate constraints. ε is a margin used to guarantee new points are not too close to the

boundary of the solution space. Equation 7 is a requirement on the distance to points already in the sample population, ρ can be changed at each iteration to promote global or local search. See Regis (2014) for more information on the margin and the distance requirement. Since COBYLA needs a starting point, we choose the point in the sample population that minimises the objective function.
The point obtained after the optimisation is then evaluated and added to the sample population.

4.2.3. Phase II

Once a feasible point is found, Phase II begins. This phase is similar to Phase I except for the objective function. Now, the overall goal is to find solutions maximising dispersion. Therefore, we tested our algorithm with the following objective function to maximise.

$$z_1(x) = \frac{1}{|\mathcal{S}_n|} \min_{y \in \mathcal{S}_n} \|x - \mathrm{y}\| \tag{8}$$

It simply consists in maximising the minimum distance to other points in the set of feasible points in the sample population, noted $\mathcal{S}_n$ for the n-th iteration. This objective function aims at having solutions spread evenly in the solution space.
A second option we could not evaluate consists in maximising the product of the eigenvalues of the covariance matrix of the points belonging to $\mathcal{S}_n$, which can be considered as a proxy for the volume of the points in $\mathcal{S}_n$. Since this product can be particularly small, especially in high dimensions, one can use the $\log_{10}$ of this product to avoid approximation errors (9).

$$z_2(x) = log10\left(\prod_{i=1}^{\min(d,|\mathcal{S}_n|)} \lambda_\mathrm{i}\right) = \sum_{i=1}^{\min(d,|\mathcal{S}_n|)} log10(\lambda_\mathrm{i}) \tag{9}$$

It is to be noted that instead of defining the best point to start the optimisation from, we decided to adopt a multistart strategy, starting optimisations from each point in $\mathcal{S}_n$ and selecting the best proposition at the end of this process.
Phase II iterates those optimisations until it reaches a stopping criterion. This stopping criterion could be an evaluation budget, but this is rather arbitrary. Other options can be considered: variation in the value of the objective function (9). When the variation becomes low, the "volume" of the points in $\mathcal{S}_n$ is not changing any more, which would indicate that no more progress is to be expected. Another option could be to use Markov Chain Monte Carlo methods such as the one presented by Martiniani et al. (2016), to measure the volume of the predicted solution space. One could therefore measure the variation in volume of the predicted solution space, which is more representative than solely the "volume" of the points in $\mathcal{S}_n$. However, both stopping criteria were not implemented and tested at the time the present article was being written.

4.2.4. Using the results for solution space modelling

At the end of the process, a metamodel $s^{(i)}$ is obtained for each constraint $\mathrm{g_i}$, and we expect them to have a good precision in the neighbourhood of the solution space. The following approximation for the indicator function of the solution space can then be used:

$$\widehat{1_\Omega}(\mathrm{x}) = \begin{cases} 1, \text{if } \max\limits_{i \in [\![1;m]\!]}(s^{(i)}(x)) < 0 \\ \quad 0, \text{otherwise} \end{cases} \tag{10}$$

This means that if a point satisfies every constraint metamodel, it is predicted to be feasible while if at least one constraint metamodel is not satisfied, then it is considered as infeasible. Moreover, since metamodels are built for every constraint, one can predict which constraints are not satisfied, and have an approximation of the extent of the constraint violations/satisfaction.

5. Evaluation and results on benchmark problems

5.1. Benchmark problems and characteristics

Prior to applying our method on industrial case studies, we desired to ensure that it is working properly on benchmark problems. To do so, we tested our algorithm on benchmark problems used to evaluate non-linear constrained optimisation algorithms found in Regis (2014). Four of those problems are real engineering problems: Welded Beam (WB4), Pressure Vessel Design (PVD4), Speed Reducer (SR7) as well as Gas Transmission Compressor Design (GTCD4). Other problems come from the G-suite problems and a last benchmark "MOPTA08" comes from a real industrial problem from General Motors described by Jones (2008). It is to be mentioned that G2OP is a modified version of G2 for which we took the opposite values of the constraints, so as to reduce the feasibility rate.

Table 1 presents the problems' characteristics. The number of design variables is represented by "Dimension" and varies between 2 and 124. The number of inequality constraints varies between 1 and 68. The feasibility rate is defined as the proportion of solution space in the design space and was measured by evaluating 10^7 randomly sampled points (10^8 points for G2OP and G3MOD, and $3*10^5$ points for MOPTA08) in the design space and computing the proportion of feasible points.

Table 1. Benchmark problems' characteristics and results for Phase I

Problem name	Dimension	Number of constraints	Feasibility rate	Number of failures (/30 runs)	Average number of evaluations	Standard deviation
WB4	4	6	0.003%	1	52	54.3
PVD4	4	3	0.559%	5	8	3.01
SR7	7	11	0.096%	0	10	0.87
Hesse	6	6	20.365%	0	7	1.2
GTCD4	4	1	52.353%	0	5	0
G2OP	10	2	<0.001%	0	23	12.34
G3MOD	20	1	<0.001%	0	23	0.83
G4	5	6	26.991%	0	6	0.47
G5MOD	4	5	0.090%	0	7	1.26
G6	2	2	0.006%	2	10	1.14
G7	10	8	<0.001%	0	39	16.2
G8	2	2	0.864%	0	8	4.76
G9	7	4	0.527%	2	24	12.64
G10	8	6	0.001%	2	21	9.63
G18	9	13	<0.001%	4	67	37.07
G19	15	5	0.004%	0	17	0.4
G24	2	2	44.198%	0	3	0.87
Mopta08	124	68	<0.003%	7	418	170.49

5.2. Evaluation of Phase I

The first evaluation is focused on the ability to find a first solution in the design space. We ran our algorithm 30 times for each problem, and evaluated the performance of our algorithm with 3 criteria. The first one is the number of failures to find a feasible point in the design space after a given number of iterations, which is fixed to 500 for all problems except for MOPTA08 with 1000 evaluations. As we can see in Table 1, the algorithm sometimes fails, without a clear relationship with the number of constraints, design variables or feasibility rate. The functions' complexity plays indeed an important role in the success of our algorithm. We expect that we could improve the success rate for Phase I by implementing the online selection of models proposed by Bagheri et al. (2016) or by changing the starting point for the optimisation algorithm when the algorithm stagnates.

Then, we computed the average number of evaluations to find a first feasible points as well as its standard deviation. There is not much to say on those results, except that they are consistent with those obtained by Regis (2014), with a slightly different implementation. Results are strongly linked to the non-linearity of constraint functions, which has to be considered case by case, explaining the variations in the average and the standard deviation of the number of evaluation to find a first feasible point.

5.3. Evaluation of Phase II

5.3.1. About conventional model building

In order to evaluate our algorithm for Phase II, we compare it to conventional model building (CMB), which consists in generating a sample population with a DoCE method, evaluating those points and then building a model for every constraint. To do so, we used LHS to create a sample, and then used a RBFN with cubic RBF and a polynomial tail as metamodels for every constraint. For each problem, we evaluated conventional (resp. our algorithm's) models built with progressively increasing sample size (resp. number of iterations). We repeated the process 10 times for each problem to compute means and standard deviations for each evaluation criteria described in the next subsection (see also Figure 4).

5.3.2. Evaluation criteria and results

Since $\widehat{1_\Omega}$ is a binary classifier, we can use classic quality measures for its evaluation, and among the possible measures, precision and sensitivity are the two most important for us. Precision corresponds to the proportion of the design points that are really feasible, among the points that are predicted to be feasible. The higher the precision, the more one can trust the model, when it predicts a point to be feasible. Sensitivity, also referred to as probability of detection, corresponds to the ratio between the number of points that are predicted to be and are really feasible, and the total number of feasible points. The higher the sensitivity, the lower number of feasible points the model misses.
To measure precision and sensitivity, we first sampled between $3*10^7$ and 10^7 points randomly, evaluated them with the true functions, and then with the models, either build using CMB, our algorithm. We could thereafter determine the quality of the models by comparing the true results and the predicted results. Since even with that large number of points, some problems did not have a sufficient number of feasible points, we could only evaluate our algorithm on problems listed in Table 2.

Table 2. Evaluation results Phase II

Problem name	**Precision > 90% speed ratio**	**Sensitivity > 90% speed ratio**	**Feasibility rate**	**Solution ratio for precision > 90%**	**Improvement factor on solution ratio**
G24	1.2	0.667	44.2%	51%	1.15
Hesse	1.67	1.67	20.3%	70%	3.4
PVD	2	2	0.559%	43%	77
G5MOD	2	6	0.090%	25%	277
SR7	1	5	0.096%	51%	531
G8	2	2	0.864%	50%	58
GTCD4	30	3	52.3%	70%	1.35
WB4	NA	NA	0.003%	44%	14667
G6	NA	NA	0.006%	15%	2500

For our practice, we think that models with 90% precision and 90% sensitivity are already sufficient for the early design phase. Since we want to compare our algorithm to CMB and since the number of iterations is an important factor of performance of our algorithm, we compared the number of iterations that our algorithm needs to the sample size required by CMB to reach an average of 90% precision and 90% sensitivity. Results are reported in Table 2. To illustrate how to read those results, on GTCD4, our algorithm is 30 times faster to get an average precision superior to 90% than CMB. A remarkable fact is that, except for problem G24, our algorithm is at least as efficient as CMB for precision and sensitivity. Being to create models with 90% precision and sensitivity 2 times faster is already a promising result for industry, as it is the case with most problems. Problems GTCD4, WB4 and G6 demonstrate the possibility of important performance increase thanks to our algorithm. Problems WB4 and G6 are difficult to interpret just with Table 2. Therefore, their results are analysed in the next subsection.

Another important performance factor improved by our algorithm is the number of feasible points in the sample population. Indeed, one can see that the lower the feasibility rate of the problem, the higher the improvement factor on the solution ratio compared to random sampling. This is important to reduce uncertainty about the location of the solution space as well as to build trust in the validity of models. Indeed, conventional models can be built even without having a single feasible point in the sample population. How can one be convinced of their quality in such a situation?

5.3.3. Developed example: WB and G6 problems

Problems WB4 and G6 illustrate the most the interest of our algorithm. Though they are not high-dimensional, they are challenging for CMB techniques. Those difficulties can be observed in Figure 4, comparing the evolution of the performance of models when the number of points in the sample population increases. Points on the graphs represent average values and bars indicate the standard deviation. For G6, the quality of the model increases slowly with the size of the sample but stays far from the 90% objective even with 800 points. Moreover, results were particularly volatile. By contrast, our algorithm reached on average 95% precision and sensitivity with 20 iterations only.

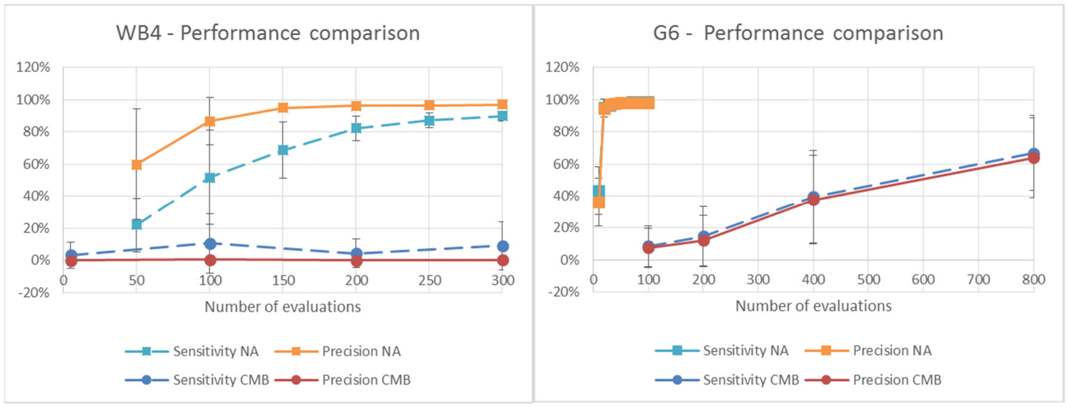

Figure 4. Comparison between Conventional Model Building (CMB) and our New Algorithm (NA) on problems WB4 and G6

CMB methods are even worse for WB4, precision and sensitivity did not reach higher mean values than 11%. Besides, increasing the number of points in the sample population did not seem to have a strong influence on the quality of the model. On the contrary, our algorithm improves steadily the quality of the prediction as the number of iterations increases. Precision increases quickly, reaching 95% already with 150 evaluations, while sensitivity is harder to improve, and reach a mean value of 90% after 300 evaluations. Those results illustrate the promising advantages provided by our algorithm in terms of quality and time, by reducing the number of simulations required to build a high quality model.
One of our current goals is to confirm those results on other problems with very small feasibility rates and possibly with more dimensions. Such problems as G2OP, G3MOD, G7, G10, G18, G19 and Mopta08 are good candidates, though they imply that we develop a more efficient way to measure models' quality than sampling randomly over the whole design space. Another aspect we are working on is the application of our approach on industrial use cases at Daimler. At the time of writing, we are testing and validating our algorithm on the design of side windows of a car, a problem with 25 parameters and 12 constraints obtained with black-box functions. Though the validation process is not finished yet, first results are very encouraging.

6. Conclusion and perspectives

Our literature review revealed that in order to implement SBD in the current automotive industry, where the use of computer experiments is prevalent to evaluate concepts, metamodels predicting if points belong to the solution space are required. We propose a new algorithm based on state-of-the-art optimisation techniques to build such models efficiently and with good quality, which we tested on classic optimisation benchmark problems, obtaining promising results.

Some limitations to the use of our algorithm exist. It can only be used for parametric design, and for the moment with continuous variables only. In our approach, discrete parameters are considered at the configuration level, and then, one model can be created for each configuration (Toepfer et al., 2018). Constraints are assumed to be continuous over the design space, and we make the assumption that simulations are deterministic and return results for every point of the design space. SVMs could be used to model discrete or binary constraints. However, one drawback of SVMs is that they need feasible and infeasible points in their sample which might not always be easy. Also, changing constraint thresholds might imply having to update the model. This could be also be considered in advance by considering a range of constraint thresholds.

Some further improvements are possible to improve our algorithm. Among others, we need to implement the stopping criteria, and we could implement an online selection of models, to adapt the metamodel type for each constraint, as in Bagheri et al. (2016b). Another important theme which has not been considered yet is a way to handle disconnected solution spaces.

Apart from those limitations and potential improvements, our algorithm present several interests. Firstly, one could use it to quickly predict if a design is feasible or not, or which constraints are violated. This, combined to the use of the tool presented in section 2 of Toepfer et al. (2018) helps find solution boxes in the design space and therefore help deal with manufacturing tolerances and have robust designs. Moreover, it could be coupled with generative grammars, with grammars handling discrete parameters while our algorithm could map the solution space for each design. Secondly, our algorithm could be a starting point for optimisation, since it helps considering points that belong only to the solution space. Thirdly, one could sample in the solution space with Markov Chain Monte Carlo methods in order to generate feasible design alternatives, enhancing thus the designers' ideation process.

References

Bagheri, S., Konen, W. and Back, T. (2016a), "Equality constraint handling for surrogate-assisted constrained optimization", *2016 IEEE Congress on Evolutionary Computation (CEC), IEEE*, pp. 1924–1931. https://doi.org/10.1109/CEC.2016.7744023

Bagheri, S., Konen, W. and Back, T. (2016b) "Online selection of surrogate models for constrained black-box optimization", *2016 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE*, pp. 1–8. https://doi.org/10.1109/SSCI.2016.7850206

Bagheri, S., Konen, W., Emmerich, M. and Bäck, T. (2015) "Solving the G-problems in less than 500 iterations: Improved efficient constrained optimization by surrogate modeling and adaptive parameter control", *arXiv preprint*, arXiv:1512.09251.

Campbell, M.I. and Shea, K. (2014), "Computational design synthesis", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Vol. 28 No. 3, pp. 207–208. https://doi.org/10.1017/S0890060414000171

Cortes, C., Vapnik, V. (1995), "Support-vector networks", *Machine Learning*, Vol. 20 No. 3, pp. 273–297. https://doi.org/10.1007/BF00994018

Fender, J., Duddeck, F. and Zimmermann, M. (2017), "Direct computation of solution spaces", *Structural and Multidisciplinary Optimization*, Vol. 55 No. 5, pp. 1787–1796. https://doi.org/10.1007/s00158-016-1615-y

Fender, J., Graff, L., Harbrecht, H. and Zimmermann, M. (2014), "Identifying Key Parameters for Design Improvement in High-Dimensional Systems With Uncertainty", *Journal of Mechanical Design*, Vol. 136 No. 4, pp. 041007. https://doi.org/10.1115/1.4026647

Ford, D.N. and Sobek II, D.K. (2005), "Adapting Real Options to New Product Development by Modeling the Second Toyota Paradox", *IEEE Transactions on Engineering Management*, Vol. 52 No. 2, pp. 175–185. https://doi.org/10.1109/TEM.2005.844466

Graff, L., Harbrecht, H. and Zimmermann, M. (2016), "On the computation of solution spaces in high dimensions", *Structural and Multidisciplinary Optimization*, Vol. 54 No. 4, pp. 811–829. https://doi.org/10.1007/s00158-016-1454-x

Jones, D.R. (2008), "Large-scale multi-disciplinary mass optimization in the auto industry", *Modeling and Optimization: Theory and Application (MOPTA) Conference, Ontario, Canada, August 20, 2008*.

Kizer, J.R. and Mavris, D.N. (2014), "Set-based design space exploration enabled by dynamic constraint analysis", *29th Congress of the International Council of the Aeronautical Sciences, St. Petersburg, Russia, September 7-12, 2014*.

Koch, P., Bagheri, S., Foussette, C., Krause, P., Bäck, T. and Konen, W. (2014), "Constrained optimization with a limited number of function evaluations", *Proceedings. 24. Workshop Computational Intelligence, Dortmund, November 27-28, 2014*, KIT Scientific Publishing, pp. 237-256.

Koch, P., Bagheri, S., Konen, W., Foussette, C., Krause, P. and Bäck, T. (2015), “A New Repair Method For Constrained Optimization”, *GECCO '15 Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, ACM Press, New York, pp. 273–280. https://doi.org/10.1145/2739480.2754658

Krige, D.G. (1951), A statistical approach to some basic mine valuation problems on the Witwatersrand”, *Journal of the Southern African Institute of Mining and Metallurgy*, Vol. 52 No. 9, pp. 201–203.

Lehar, M. and Zimmermann, M. (2012), “An inexpensive estimate of failure probability for high-dimensional systems with uncertainty”, *Structural Safety*, Vol. 36–37, pp. 32–38. https://doi.org/10.1016/j.strusafe.2011.10.001

Madhavan, K., Shahan, D., Seepersad, C.C., Hlavinka, D.A. and Benson, W. (2008), “An Industrial Trial of a Set-Based Approach to Collaborative Design”, *ASME 2008 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference Volume 1: 34th Design Automation Conference, Parts A and B, Brooklyn, New York, USA, August 3–6, 2008*, pp. 737–747. https://doi.org/10.1115/DETC2008-49953

Nahm, Y.-E. and Ishikawa, H. (2006), “Novel space-based design methodology for preliminary engineering design”, *The International Journal of Advanced Manufacturing Technology*, Vol. 28 No. 11-12, pp. 1056–1070. https://doi.org/10.1007/s00170-004-2463-2

Powell, M.J. (1992), “The theory of radial basis function approximation”, In: Light W. (Ed.), *Advances in Numerical Analysis, Vol. II: Wavelets, Subdivision Algorithms and Radial Functions*, University Press, Oxford, UK, pp. 105-210.

Regis, R.G. (2014), “Constrained optimization by radial basis function interpolation for high-dimensional expensive black-box problems with infeasible initial points”, *Engineering Optimization*, Vol. 46 No. 2, pp. 218–243. https://doi.org/10.1080/0305215X.2013.765000

Shahan, D. and Seepersad, C.C. (2009), “Bayesian Networks for Set-Based Collaborative Design”, *ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Volume 5: 35th Design Automation Conference, Parts A and B, San Diego, California, USA, August 30–September 2, 2009,* pp. 303–313. https://doi.org/10.1115/DETC2009-87541

Shan, S. and Wang, G.G. (2010), “Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions”, *Structural and Multidisciplinary Optimization*, Vol. 41 No. 2, pp. 219–241. https://doi.org/10.1007/s00158-009-0420-2

Sobek II, D.K., Ward, A.C. and Liker, J.K. (1999), “Toyota’s principles of set-based concurrent engineering”, *Sloan management review*, Vol. 40 No. 2.

Tabatabaei, M., Hakanen, J., Hartikainen, M., Miettinen, K. and Sindhya, K. (2015), “A survey on handling computationally expensive multiobjective optimization problems using surrogates: non-nature inspired methods”, *Structural and Multidisciplinary Optimization*, Vol. 52 No. 1, pp. 1–25. https://doi.org/10.1007/s00158-015-1226-z

Toepfer, F. and Naumann, T. (2017), “Parameter Management, a Novel Approach in Systems Engineering”, *Research into Design for Communities, Volume 1. ICoRD 2017. Smart Innovation, Systems and Technologies, Vol. 65*, Springer, Singapore, pp. 383–395. https://doi.org/10.1007/978-981-10-3518-0_34

Toepfer, F., Naumann, T., Anderer, J. and Vajna, S. (2018), “Integrating the Knowledge about Functional Interdependencies into a Parameter Management Approach”, *Proceedings of the DESIGN 2018 / 15th International Design Conference, Dubrovnik, Croatia, May 21–24, 2018*, The Design Society, Glasgow.

Trabelsi, H., Yvars, P.-A., Louati, J. and Haddar, M. (2015), “Interval computation and constraint propagation for the optimal design of a compression spring for a linear vehicle suspension system”, *Mechanism and Machine Theory*, Vol. 84, pp. 67–89. https://doi.org/10.1016/j.mechmachtheory.2014.09.013

Ulrich, K.T. and Eppinger, S.D. (2012), *Product design and development*, 5th ed., McGraw-Hill Irwin, New York.

Ward, A.C., Liker, J.K., Cristiano, J.J. and Sobek II, D.K. (1995), “The second Toyota paradox: How delaying decisions can make better cars faster”, *Long Range Planning*, Vol. 28 No. 4, p. 129. https://doi.org/10.1016/0024-6301(95)94310-U

Yannou, B., Simpson, T.W. and Barton, R.R. (2003), “Towards a Conceptual Design Explorer Using Metamodeling Approaches and Constraint Programming”, *ASME 2003 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Volume 2: 29th Design Automation Conference, Parts A and B, Chicago, Illinois, USA, September 2–6, 2003*, pp. 605–614. https://doi.org/10.1115/DETC2003/DAC-48766

Zimmermann, M. and von Hoessle, J.E. (2013), “Computing solution spaces for robust design”, *International Journal for Numerical Methods in Engineering*, Vol. 94 No. 3, pp. 290–307. https://doi.org/10.1002/nme.4450

Zimmermann, M., Königs, S., Niemeyer, C., Fender, J., Zeherbauer, C. et al. (2017), “On the design of large systems subject to uncertainty”, *Journal of Engineering Design*, Vol. 28 No. 4, pp. 233–254. https://doi.org/10.1080/09544828.2017.1303664

Benjamin Poulain, Engineer
CentraleSupélec, LGI
9 Rue Joliot Curie, 91190 Gif-sur-Yvette, France
Email: benjamin.poulain@student.ecp.fr