
This version is available at https://strathprints.strath.ac.uk/52251/

Strathprints is designed to allow users to access the research output of the University of Strathclyde. Unless otherwise explicitly stated on the manuscript, Copyright © and Moral Rights for the papers on this site are retained by the individual authors and/or other copyright owners. Please check the manuscript for details of any other licences that may have been applied. You may not engage in further distribution of the material for any profitmaking activities or any commercial gain. You may freely distribute both the url (https://strathprints.strath.ac.uk/) and the content of this paper for research or private study, educational, or not-for-profit purposes without prior permission or charge.

Any correspondence concerning this service should be sent to the Strathprints administrator: strathprints@strath.ac.uk
ANALYSIS OF TWO ALGORITHMS FOR MULTI-OBJECTIVE MIN-MAX OPTIMIZATION

Simone Alicino  
Mechanical and Aerospace Engineering  
University of Strathclyde, Glasgow, UK  
simone.alicino@strath.ac.uk

Massimiliano Vasile  
Mechanical and Aerospace Engineering  
University of Strathclyde, Glasgow, UK  
massimiliano.vasile@strath.ac.uk

Abstract  This paper presents two memetic algorithms to solve multi-objective min-max problems, such as the ones that arise in evidence-based robust optimization. Indeed, the solutions that minimize the design budgets are robust under epistemic uncertainty if they maximize the Belief in the realization of the value of the design budgets. Thus robust solutions are found by minimizing with respect to the design variables the global maximum with respect to the uncertain variables. A number of problems, composed of functions whose uncertain space is modelled by means of Evidence Theory, and presenting multiple local maxima as well as concave, convex, and disconnected fronts, are used to test the performance of the proposed algorithms.

Keywords: Multi-objective optimization, worst-case scenario design, evidence-based robust optimization.

1. Introduction

Worst-case scenario problems arise whenever a performance index, or cost function, has to be optimal with respect to a design vector \( \mathbf{d} \), and at the same time robust against an uncertain vector \( \mathbf{u} \). This class of problems is common in several fields, such as game theory, decision making, robust control, risk analysis, and robust design. For instance, the lower expectation in the realization of the value of a particular performance index for a model of a system can be defined as the degree of belief that
Algorithm 1 Min-max optimization via restoration

1: Initialize archive $A_u = \{u_1\}$, and set $i = 1$
2: while the stopping condition is not met do
3: Compute $d_i = \arg \min_{d \in D} \left\{ \max_{u \in A_u} f(d, u) \right\}$
4: Compute $u_{i+1} = \arg \max_{u \in U} f(d_i, u)$
5: Add $u_{i+1}$ to the archive $A_u$
6: $i \leftarrow i + 1$
7: end while
8: Return $\{d_i, u_{i+1}\}$.

one has in a certain proposition being true, given the available evidence. In the framework of imprecise probabilities, it can be seen as a lower bound to the cumulative distribution function of classical probability theory. Its use is therefore interesting in engineering design, as it gives the lower limit of the confidence that the design budgets under uncertainty will be below a given threshold. In this framework both epistemic and aleatory uncertainties can be treated even when no exact information on the probability distribution associated to an uncertain quantity is available. Stochastic variables and associated probability are replaced by a multivalued mapping from a collection of subsets of an uncertain space $U$ into a lower expectation (Belief function in the case of Evidence Theory). The main drawback of the use of multivalued mappings is that the computation of the lower expectation, i.e. the Belief, has a complexity that is exponential with the number of uncertain variables. Recently, some strategies were proposed in [1] to obtain an estimation of the maximum Belief with a reduction of the computational cost. The approach starts by translating an optimization under uncertainty into a single or multi-objective min-max problem equivalent to a worst-case scenario optimization problem. Several methods have been proposed to address single-objective min-max problems, especially using evolutionary approaches [2, 3], and metamodels [4–6]. For the multi-objective case, a gradient-based approach is presented in [7]. An interesting approach is based on the procedure proposed in [6, 8] for single-objective problems, and exploited in [9] for interval multi-objective linear programming. Such procedure is based on an iterative minimization over the design space and subsequent restoration of the global maximum over the uncertain space as shown in Algorithm 1. The stopping condition can be the achievement of a desired accuracy, or a maximum number of function evaluations, for example. In this paper we present a multi-objective version of Algorithm 1 implemented in an algorithm called
MACSminmax. MACSminmax employs MACS2 and IDEA at steps 3 and 4, respectively. Another algorithm, MACS$\nu$, is presented in this paper and compared to MACSminmax. MACS$\nu$ is a variant of MACS2 containing heuristics to deal with min-max problems. The paper starts with a brief introduction to Evidence Theory and its use in the context of robust design optimization in Section 2. Section 3 introduces the two memetic algorithms, MACSminmax and MACS$\nu$. Section 4 finally presents the results on some test cases.

2. Evidence-Based Robust Design Optimization

Evidence Theory [10] allows to adequately model both epistemic and aleatory uncertainty when no information on the probability distributions is available. For instance, during the preliminary design of an engineering system, experts can provide informed opinions by expressing their belief in an uncertain parameter $u$ being within a certain set of intervals. The level of confidence an expert has in $u$ belonging to one of the intervals is quantified by using a mass function generally known as Basic Probability Assignment (bpa). All the intervals form the so-called frame of discernment $\Theta$, which is a set of mutually exclusive elementary propositions. The power set of $\Theta$ is called $U = 2^\Theta$, or the set of all the subsets of $\Theta$ (the uncertain space in the following). An element $\theta$ of $U$ that has a non-zero bpa is called focal element. When more than one parameter is uncertain, the focal elements are the result of the Cartesian product of all the elements of each power set associated to each uncertain parameter. The bpa of a given focal element is then the product of the bpa of all the elements in the power set associated to each parameter. All the pieces of evidence completely in support of a given proposition form the cumulative belief function $Bel$, defined as follows:

$$Bel(A) = \sum_{\forall \theta_i \subseteq A} m(\theta_i)$$  \hspace{1cm} (1)

where $A$ is the proposition about which the Belief is evaluated. For example, the proposition can be expressed as:

$A = \{ u \in U \mid f(u) \leq \nu \}$  \hspace{1cm} (2)

where $f$ is the outcome of the system model and the threshold $\nu$ is the desired value of a design budget. It is important to note that the set $A$ can be disconnected or present holes, likewise the focal elements can be disconnected or partially overlapping. This introduces discontinuities in the search space, making the problem more difficult to solve.

An engineering system to be optimized can be modelled as a function $f : D \times U \subseteq \mathbb{R}^{m+n} \rightarrow \mathbb{R}$. The function $f$ represents the model of the
system budgets (e.g. power budget, mass budget, etc.), and depends on some uncertain parameters \( u \in U \) and design parameters \( d \in D \), where \( D \) is the available design space and \( U \) the uncertain space. What is interesting for the designers is the value of the function \( f \) for which \( Bel = 1 \), i.e. it is maximum. This value of the design budget is the threshold \( \nu_{\text{max}} \) above which the design is certainly feasible, given the current body of evidence. If \( q \) objective functions exist, then the following problem can be solved without considering all the focal elements:

\[
\nu_{\text{max}} = \min_{d \in D} F = \min_{d \in D} \max_{u \in U} [f_1(d, u), \ldots, f_q(d, u)]^T
\]  

(3)

Problem in (3) is a multi-objective min-max over the design space \( D \) and the uncertain space \( \bar{U} \), where \( \bar{U} \) is a unit hypercube collecting all the focal elements in a compact set with no overlapping or holes. The transformation between \( U \) and \( \bar{U} \) is given by:

\[
x_U = \frac{(b_{U,i}^u - b_{U,i}^l)}{(b_{\bar{U},i}^u - b_{\bar{U},i}^l)} x_{\bar{U},i} + \frac{(b_{U,i}^l - b_{\bar{U},i}^l)}{(b_{U,i}^u - b_{\bar{U},i}^l)} b_{\bar{U},i}
\]  

(4)

where \( b_{U,i}^u \) and \( b_{U,i}^l \) (resp. \( b_{\bar{U},i}^u \) and \( b_{\bar{U},i}^l \)) are the upper and lower boundaries of the \( i \)-th hypercube to which \( x_U \) (resp. \( x_{\bar{U},i} \)) belongs.

3. **Multi-Objective Min-Max Memetic Optimization**

Problem (3) searches for the minimum of the maxima of all the functions over \( \bar{U} \) and represents an example of worst-case scenario design optimization. The maximum of every function is independent of the other functions and corresponds to a different uncertain vector. Therefore, all the maxima can be computed in parallel with \( q \) single-objective maximizations. The maximization of each function is performed by running a global optimization over \( \bar{U} \) using Inflationary Differential Evolution (IDEA). The minimization over \( D \) is performed by means of MACS2. IDEA [11] is a population-based memetic algorithm for single-objective optimization. It hybridizes Differential Evolution and Monotonic Basin Hopping in order to simultaneously improve local convergence and avoid stagnation. MACS2 [12] is a memetic algorithm for multi-objective optimization based on a combination of Pareto ranking and Tchebycheff scalarization. The search for non-dominated solutions is performed by a population of agents which combine individualistic and social actions. The initial population is randomly generated in the search domain. Individualistic actions perform a sampling of the search space in a neighborhood of each agent. Then, subsets of the population perform social
actions aiming at following particular descent directions in the criteria space. Social agents implement a Differential Evolution operator and assess the new candidate solutions using Tchebycheff scalarization. Current non-dominated solutions are then stored in an archive. Both social and individualistic actions make use of a combination of the population and the archive.

In a classical minimization problem two solutions $d_1$ and $d_2$ are ranked according to which one gives the lower value of the function. In the minimization loop of a min-max problem, the same can be done only if the maximization loop has returned the actual global maxima $\hat{u}_1$ and $\hat{u}_2$. However, this is usually not true. Therefore a mechanism of cross-check such that also $(d_1, u_2)$ and $(d_2, u_1)$ are evaluated is needed in order to increase the probability that each maximization identifies the global maximum, and correctly rank two solutions.

3.1 MACS$\nu$

MACS$\nu$ (Algorithm 2) is the min-max variant of MACS2. It endows MACS2 with special heuristics to increase the probability of finding the global maxima in $\bar{U}$. More in detail, a CROSS-check (lines 7, 18, and 28) compares the values of the objective functions for a newly generated design vector in the trial populations $P_t$ (line 7) and $P_s$ (line 18) against the function values of a solution already archived in $A$ (indicated with subscript arch in Algorithm 2). In addition, the cross-check performs a local search or a simple function evaluation in the inner maximization loop depending on whether the location of the maxima changes or not, respectively, for different design vectors. After the cross-check, a MIN-MAX SELECTION (lines 11 and 22) compares the population $P$ with the new candidate populations $P_t$ (line 11) and $P_s$ (line 22) and selects the design vectors to attribute to the next generation according to the following rule: If $d$ (resp. $u$) is unchanged, the old $u$ (resp. $d$) is replaced with the new one, if it yields a higher (resp. lower) value of the objective function; if both $d$ and $u$ are different, the new vectors will replace the old ones. At the end of the algorithm, and at the last iteration, a VALIDATION (line 24) mitigates the possibility that the cross-check operators assign the same incorrect $u$ to all $d$ vectors in the population and archive. This is done by starting from the minimum value of the first objective in the archived Pareto front, and performing a global search in the uncertain space. If the new uncertain vector gives a higher value of the function, then it replaces the old one. This operation is repeated for the elements in the archived Pareto front until there is no more variation in their value.
Algorithm 2 MACS\(\nu\)

1: Initialize population \(P\), archive \(A = P\), \(n_{feval} = 0, \epsilon = 0.7, \delta = 10^{-6}\)
2: \(\textbf{while } n_{feval} < n_{feval,max} \textbf{ do}\)
3: \(\text{Run individualistic moves and generate trial population } P_t\)
4: \(\textbf{for all } d \in P_t \textbf{ do}\)
5: \(\text{for all } d_{arch} \in A \textbf{ do}\)
6: \(\text{if } d > d_{arch} \textbf{ then}\)
7: \(\text{CROSS-CHECK}(P_t, A)\)
8: \(\textbf{end if}\)
9: \(\textbf{end for}\)
10: \(\textbf{end for}\)
11: \(\text{MIN-MAX SELECTION}(P, P_t)\)
12: \(\text{Update } P \text{ and } A\)
13: \(Z \leftarrow \|F_{max_{arch}}^u - F_{min_{arch}}^u\|\)
14: \(\text{Run social moves and generate candidate population } P_s\)
15: \(\textbf{for all } d \in P_s \textbf{ do}\)
16: \(\text{for all } d_{arch} \in A \textbf{ do}\)
17: \(\text{if } d > d_{arch} \textbf{ or } \|F(d) - F(d_{arch})\| > \epsilon Z \textbf{ then}\)
18: \(\text{CROSS-CHECK}(P_s, A)\)
19: \(\textbf{end if}\)
20: \(\textbf{end for}\)
21: \(\textbf{end for}\)
22: \(\text{MIN-MAX SELECTION}(P, P_s)\)
23: \(\text{Update } P \text{ and } A\)
24: \(\text{VALIDATION}(A)\)
25: \(\textbf{for all } d \in P \textbf{ do}\)
26: \(\text{for all } d_{arch} \in A \textbf{ do}\)
27: \(\text{if } d > d_{arch} \textbf{ or } d < d_{arch} \textbf{ then}\)
28: \(\text{CROSS-CHECK}(P, A)\)
29: \(\text{else if } \|F(d) - F(d_{arch})\| < \delta \textbf{ then}\)
30: \(\text{Replace } u \in P \text{ with } u \in A\)
31: \(\textbf{end if}\)
32: \(\textbf{end for}\)
33: \(\textbf{end for}\)
34: \(\textbf{end while}\)

3.2 MACSminmax

MACSminmax (Algorithm 3) is a min-max memetic algorithm inspired by the procedure of Algorithm 1. This is the main difference with MACS\(\nu\). In MACSminmax, for each agent of the minimization the best function value is computed with respect to an archive \(A_u\) of candidate
Algorithm 3 MACSminmax

1: Initialize archive $A_u = \{u_1\}$, $n_{feval} = 0$
2: while $n_{feval} < n_{feval,max}$ do
3: Run MACS2 to compute $d_{min} = \arg \min_{d \in D} \max_{u \in A_u} f(d, u)$ and associated $f_d$
4: Add $d_{min}$ to the archive $A_d$
5: for all $d_{min} \in A_d$ do
6: for all $l \in \{1, \ldots, q\}$ do
7: Run IDEA to compute $u_{max}^l = \arg \max_{u \in U} f^l(d_{min}, u)$ and associated $f^l_u$
8: if $f^l_u > f^l_d$ then
9: Add $u_{max}^l$ to the archive $A_u$
else
10: Evaluate function to find $u_{max}^l = \arg \max_{u \in A_u} f^l(d_{min}, u)$
end if
11: end for
12: end for
13: end while
14: for all $d_{min} \in A_d$ do
15: for all $l \in \{1, \ldots, q\}$ do
16: Run local search to refine $u_{max}^l \in A_u$ associated to $d_{min}$
17: end for
18: end for
19: Return non-dominated $d_{min}$ and associated $u_{max}^l$.

For uncertain vectors (line 3). The archive $A_u$ is composed of the results of a global maximization or a simple function evaluation, depending on which one of the two gives the higher function value (as explained above, it is not guaranteed that the maximization finds the global maximum), for each design vector contained in another archive $A_d$ of candidate solutions (lines 5 to 14). Thus, each element in the archive $A_u$ corresponds to an element in the archive $A_d$. This is so if the global maxima change for different design vectors. If they do not change, the archive $A_u$ is composed of only one element. At the end of the main loop, a local search is run for each element of the archive $A_d$ in order to refine the accuracy of the elements in the archive $A_u$. Finally, because the archive $A_d$ is filled with batches of solutions given in output by MACS2, the solutions are non-dominated only inside each batch. Therefore a further dominance check is necessary to find the non-dominated solutions among the batches.
Interesting is a comparison between MACSminmax and MACS\(\nu\). In MACS\(\nu\) a maximization is run for every agent of the minimization, whereas in MACSminmax each agent of the minimization is cross-checked with the archive of candidate uncertain vectors through a function evaluation. However, it is worth noting that the evaluation, in MACSminmax, of each \(d\) against an archive \(A_u\) of candidate uncertain vectors, as well as the update of \(A_u\) for each element of an archive \(A_d\) of candidate design vectors, is equivalent to the cross-checks implemented in MACS\(\nu\). Furthermore, the local search in MACSminmax after the main loop is similar to the validation procedure in MACS\(\nu\), where a global search is run starting from the extrema of the Pareto front. Finally, in terms of balance between exploration (social moves) and exploitation (individualistic moves) of the search space, both MACS\(\nu\) and MACSminmax employ the same search algorithms, MACS2 and IDEA, therefore they are equivalent so long as the parameters (population, \(F\), \(C_R\)) are set to the same values.

4. Test Cases

MACS\(\nu\) and MACSminmax were tested on the six bi-objective and one tri-objective test cases reported in Table 1, where \(n\) is the dimension of the design vector \(d\), as well as the uncertain vector \(u\) – therefore the total dimension of the test cases is \(2n\) – and \(n_{\text{eval,max}}\) is the maximum number of function evaluations, i.e. the termination condition for the algorithms. The test cases are composed of the functions in Table 2. The functions are easily scalable and present very challenging landscapes, with multiple maxima that can change significantly with the design vector. Function MV10, in particular, is characterized by having the maxima located on top of multiple sharp, steep peaks. Note also that the test cases present several types of Pareto fronts, convex, concave, linear, and disconnected. The uncertain vector \(u\) is assigned the \(bpa\) structure reported in Table 3. The uncertain intervals present holes and overlappings, that introduce discontinuities in the uncertain space. The reference solution, i.e. the real front in Figures 1, was computed by merging the results of 200 runs of the same problems solved by means of MACS\(\nu\) with the results of 200 runs of MACSminmax.

From a sensitivity analysis on total number of agents (5, 10, 20) vs. subset of social agents (1/3, 1/2, 1), and \(F\) (0.1, 0.5, 1, 2) vs. \(C_R\) (0.1, 0.5, 0.9) for MACS2 and IDEA resulted that the best settings were: 200\(n\) function evaluations for both MACS2 and IDEA, for 10 agents for MACS2, half of which perform the social actions, 5 agents for IDEA, and \(F = 1\) and \(C_R = 0.1\) for both MACS2 and IDEA. The sensitivity
Table 1. Test cases.

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Functions</th>
<th>d</th>
<th>n</th>
<th>( n_{\text{feval, max}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC1</td>
<td>( f_1 = \text{MV1}, f_2 = \text{MV3} )</td>
<td>([1, 5]^n)</td>
<td>2</td>
<td>2E5</td>
</tr>
<tr>
<td>TC2</td>
<td>( f_1 = \text{MV2}, f_2 = \text{MV8} )</td>
<td>([0, 3]^n)</td>
<td>8</td>
<td>1E6</td>
</tr>
<tr>
<td>TC3</td>
<td>( f_1 = \text{MV2}, f_2 = \text{EM1} )</td>
<td>([1, 5]^n)</td>
<td>8</td>
<td>1E6</td>
</tr>
<tr>
<td>TC4</td>
<td>( f_1 = \text{MV8}, f_2 = \text{MV9} )</td>
<td>([1, 3]^n)</td>
<td>2</td>
<td>4E5</td>
</tr>
<tr>
<td>TC5</td>
<td>( f_1 = \text{MV8}, f_2 = \text{EM1} )</td>
<td>([1, 5]^n)</td>
<td>4</td>
<td>1E6</td>
</tr>
<tr>
<td>TC6</td>
<td>( f_1 = \text{MV10}, f_2 = \text{MV9} )</td>
<td>([-4, 2\pi]^n)</td>
<td>1</td>
<td>1E5</td>
</tr>
<tr>
<td>TC7</td>
<td>( f_1 = \text{MV2}, f_2 = \text{MV8}, f_3 = \text{EM1} )</td>
<td>([1, 5]^n)</td>
<td>4</td>
<td>1E6</td>
</tr>
</tbody>
</table>

Table 2. Test functions.

<table>
<thead>
<tr>
<th>ID</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>MV1</td>
<td></td>
</tr>
<tr>
<td>MV2</td>
<td></td>
</tr>
<tr>
<td>MV3</td>
<td></td>
</tr>
<tr>
<td>MV8</td>
<td></td>
</tr>
<tr>
<td>MV9</td>
<td></td>
</tr>
<tr>
<td>MV10</td>
<td></td>
</tr>
<tr>
<td>EM1</td>
<td></td>
</tr>
</tbody>
</table>

analyses were run for test case TC4 with a total of 2E5 function evaluations, and the results assessed in terms of success rate of finding the global maximum, as well as convergence \( M_{\text{conv}} \) and spreading \( M_{\text{spr}} \) as per definition in [13]. The same settings were used in all the test cases.
Table 4 summarizes the success rates of finding the global maxima, as well as convergence and spreading of MACSminmax in comparison to MACS$\nu$. The results are the average performances obtained from the 200 runs needed to achieve a confidence interval of 95% on the success rate being within a ±5% interval containing its estimated value [11]. Columns $maxf_1$, $maxf_2$ and $maxf_3$ contain the maximization success rates computed with an accuracy of $10^{-4}$ with respect to the actual maxima, columns $M_{conv}$ and $M_{spr}$ contain the mean value of $M_{conv}$ and $M_{spr}$ respectively, and columns $p_{conv}/t_{conv}$ and $p_{spr}/t_{spr}$ contain the success rate of computing a front which convergence and spreading are below the thresholds $t_{conv}$ and $t_{spr}$ also contained in the columns after the ‘/’ symbol. MACSminmax attains performances similar to MACS$\nu$, with excellent success rates for almost all the test cases. For TC2, TC3 and TC7, MACSminmax provides a significantly better spreading (2.0, 0.3, and 2.1) than MACS$\nu$ (16.1, 7.5, and 9.3). Note also that TC2 and TC3 are the test cases with the higher dimension, 16, whereas TC7 has the highest number of objectives, 3. Moreover, for TC5 MACSminmax has a significantly higher success rate for the maximization of the second objective (87.6% against 54.1%): in function EM1 the global maximum has a jump for a certain value of $d$. This makes the global maximum been tracked more effectively with the global search implemented in MACSminmax than with the local search of MACS$\nu$. However, for TC5 average converge and spreading computed by MACSminmax, and their success rates, are worse than for MACS$\nu$. MACS$\nu$ also performs better at finding a front for TC6 which spreading is below a threshold equal to 2. In conclusion, MACSminmax has equal or better capability in the maximization in the uncertain space, and also in terms of convergence and spreading, than MACS$\nu$, which in turns performed slightly better in two cases. On one hand, such rather equivalent performances of the two algorithms can be explained by the fact that they have equivalent balance between exploration and exploitation, as explained in subsection 3.2. On the other hand, the better performance of MACSminmax on some of the test cases can be due to more effective archiving, cross-check and validation mechanisms, which are the only aspects that differentiate MACS$\nu$ and MACSminmax. The Pareto fronts for the seven test cases are shown in Figure 1. As a comparison, the fronts computed by means of MACS$\nu$ and of MACSminmax are displayed. One can see that MACSminmax performs as well as MACS$\nu$ at identifying the true Pareto front for all the test cases. In addition, it is worth noting that TC4 presents a deceptive front, as the bottom-right portion of it has a multitude of dominated fronts above it. This resulted to be a very difficult part for both MACS$\nu$ and MACSminmax to identify.
Two multi-objective min-max memetic algorithms, MACSminmax and MACSν have been presented and compared in this paper. MACSν is a variant of MACS2 endowed with cross-checks, and selection and validation mechanisms to properly maximize the subproblem. MACSminmax makes use of an iterative restoration of the global maxima in the uncertain space. Despite the different procedures, the two strategies imple-
ment similar cross-checks. The two algorithms have been tested on seven scalable test cases that present several types of Pareto fronts. Results show that both MACSminmax and MACSν are able to achieve similar very good performances, in terms of finding the global maxima in the uncertain space and the true Pareto front. However, MACSminmax performed significantly better in terms of spreading in the two test cases with the highest dimension and the one with three objectives. Multi-objective min-max optimization algorithms find applicability to worst-case scenario problems, such as evidence-based robust engineering design.

References