A Memetic Approach to the Solution of Constrained Min-Max Problems

Gianluca Filippi
Mechanical and Aerospace Engineering
University of Strathclyde
Glasgow, UK
g.filippi@strath.ac.uk

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

***Abstract*—This paper proposes a novel memetic algorithm for the solution of constrained min-max problems that derive from the optimal design of complex systems under worst-case conditions. In this context the maximisation of a quantity of interest over the space of uncertain variables is required to identify the worst-case scenario (or worst-case solution under uncertainty). An optimal design vector is then identified such that the worst-case value of the quantity of interest is minimised. In the most general case, both maximisation and minimisation are subject to strict feasibility constraints. The ultimate goal of the minimisation problem is to identify the design solution that is feasible for all possible values of the uncertain parameters.**

***Index Terms*—worst case scenario, min-max, epistemic uncertainty, benchmark**

I. Introduction

One aspect of Resilience Engineering is the design of systems that are robust against uncertainty of different nature. Different sources of uncertainty are possible: uncertainty on model definition, measurement noise, manufacturing and fabrication errors, human error, numerical error, etc [1]. All these forms of uncertainty are commonly classified into groups: *aleatory* and *epistemic*. The former group collects irreducible uncertainties while the latter group collects uncertainty due to lack of knowledge and/or subjective probability. Uncertainty can be reduced to aleatory as information increases.
An engineering system can be optimised using a model for the *Quantity of Interest* (QoI) f that is a function of both decision (or design) variables $\mathbf{d}$ and uncertain parameters $\mathbf{u}$:

$$f = f(\mathbf{d}, \mathbf{u}), \tag{1}$$

with $\mathbf{d} \in D$ and $\mathbf{u} \in U$, where D is a design/decision space and U the uncertainty space. Considering, without loss of generality, a minimisation problem, we can now define the worst-case scenario as the uncertainty vector $\mathbf{u}^*$ such that f attains the maximum value in U. The resulting unconstrained min-max problem then reads as:

$$\min_{\mathbf{d}\in D} \max_{\mathbf{u}\in U} f(\mathbf{d}, \mathbf{u}) \tag{2}$$

A simple approach to solve Eq. (2) is described in [2] as *Best Replay*: for a fixed value $\mathbf{u}^*$, $\mathbf{d}_{min}$ is evaluated minimising $f(\mathbf{d}, \mathbf{u}^*)$; then $\mathbf{d}_{min}$ is fixed and f is maximised over $\mathbf{u}$ and this two optimisation steps are alternated until convergence is achieved. Another simple approach is in Eq. (3) where the maximum of f over $\mathbf{u}$ is minimised over $\mathbf{d}$:

$$\begin{aligned} &\min_{\mathbf{d}\in D} f(\mathbf{d}) \\ &s.t. \\ &f(\mathbf{d}) = \max_{\mathbf{u}\in U} f(\mathbf{d}, \mathbf{u}) \end{aligned} \tag{3}$$

It has been proven, however, that the *Best Replay* approach often does not converge or it cycles through wrong design candidates. On the other side, the direct approach proposed in Eq. (3) is too computationally expensive. Thus, a lot of effort has gone into developing methods that solve the min-max problems with an affordable computational cost: a number of papers have been published about mathematical programming [3]–[13] and heuristic methods [14]–[17]. Mathematical programming approaches require strong assumptions on the nature of the function f and tend to be tailored on specific problems. On the other hand heuristic approaches appear to be more flexible. In particular, evolutionary-based algorithms represent a promising alternative to mathematical programming methods. In the existing literature, only few papers could be found that have explored how to deal with constraints in bi-level optimisation. Most of them need to start from some strong assumptions on the nature of constraints and cost functions and have been developed for constrained bi-level problems and not specifically for the treatment of min-max problems [18], [19].

This paper proposes $MacMinMax$, an extension of the algorithm presented in [20] to treat min-max problems under strict constraints. In this case the goal is to find a design solution $\mathbf{d}^*$ that is feasible for all values of the uncertain variables $\mathbf{u}$ and minimises the worst case value of the quantity of interest f. In formulas this constrained min-max problem reads:

$$\begin{aligned} &\min_{\mathbf{d}\in D} \max_{\mathbf{u}\in U} f(\mathbf{d}, \mathbf{u}) \\ &s.t. \\ &\max_{\mathbf{u}\in U} C(\mathbf{d}, \mathbf{u}) \leq 0 \end{aligned} \tag{4}$$

The paper is structured as follows. Section II presents the proposed memetic algorithm for the solution of constrained min-max problems. Section III introduces the experimental benchmark on which the algorithm is tested where the results

This research has been developed with the partial support of the H2020 MCSA ITN UTOPIAE grant agreement number 722734.

are shown in Eq. (III-B). Section IV, finally, comments the performance of the algorithm at identifying feasible and robust solutions.

II. A Memetic Min-max Approach

The algorithm proposed in this paper is inspired by the procedure, alternating *optimisation* and *restoration* loops, suggested by Shimizu and Aiyoshi [21], [22]. Following this idea, other methods [20], [23] were developed to solve the unconstrained problem in Eq. (2). $MacMinMax$, first introduced in [24], solves, instead, the more general constrained min-max problem defined in Eq. (4). The original contribution of this paper consists in a new benchmark to test constrained min-max algorithms and the extensive testing of $MacMinMax$. $MacMinMax$ is a bi-level optimisation algorithm where the *optimisation* loop (upper or outer level) minimises the function f over the decision vector $\mathbf{d}$ considering the worst condition $\mathbf{u}_A$ in an subset A_u of the uncertain space ($A_u \subset U$):

$$\min_{\mathbf{d}\in\mathbf{D}} \Big(\max_{\mathbf{u}_A \in A_u} f(\mathbf{d}, \mathbf{u}_A) \Big). \tag{5}$$

The *restoration loop* (lower or inner level), instead, maximises the objective function f over the uncertainty vector $\mathbf{u}$ for the design vector $\mathbf{d}^*$ fixed in the optimisation loop:

$$\max_{\mathbf{u}\in U} f(\mathbf{d}^*, \mathbf{u}). \tag{6}$$

Both loops are considered under the constraint function C. Note that in the case of a vector of constraint functions $[C_1, ..., C_{nc}]$ we consider the scalar:

$$C(\mathbf{d}, \mathbf{u}) = \sum_{i=1}^{nc} \max(C_i(\mathbf{d}, \mathbf{u}), 0). \tag{7}$$

The resulting algorithm proceeds as follows:

1) [Optimisation] Given a set of maxima in $A_u = A_f \bigcup A_c$, from the *restoration* loop, solve the constrained minimisation problem:

$$\begin{array}{l} \min_{\mathbf{d}\in D} \max_{\mathbf{u}\in A_u} f(\mathbf{d}, \mathbf{u}) \\ s.t. \\ \max_{\mathbf{u}\in A_u} C(\mathbf{d}, \mathbf{u}) \leq 0 \end{array} \tag{8}$$

   The solution $\mathbf{d}^*$ is then saved in an archive A_d.
2) [Restoration] Given the solution of problem (8), $\mathbf{d}^*$, solve the two maximisation problems:

$$\begin{array}{l} \max_{\mathbf{u}\in U} f(\mathbf{d}^*, \mathbf{u}) \\ s.t. \\ C(\mathbf{d}^*, \mathbf{u}) \leq 0 \end{array} \tag{9}$$

$$\max_{\mathbf{u}\in U} C_i(\mathbf{d}^*, \mathbf{u}) \qquad \forall i \in [1, ..., nc]^T \tag{10}$$

   The solution of Eq. (9), $\mathbf{u}_{af} = \arg\max_{\mathbf{u}\in\mathbf{U}} f(\mathbf{d}^*, \mathbf{u})$ s.t. $C(\mathbf{d}^*, \mathbf{u}) \leq 0$, from the space of the feasible maxima of f is added to the archive A_f. The solutions of Eq. (10), $\mathbf{u}_{ac,i} = \arg\max_{u\in U} C_i(\mathbf{d}^*, \mathbf{u}) \ \ \forall i$ from the space of the maxima violation of the constraint C are stored in the archive A_c if $\max_{\mathbf{u}\in U} C_i(\mathbf{d}^*, \mathbf{u}) > 0$. The archive $A_u = A_f \cup A_c$ is then used in the upper loop to evaluate the new $\mathbf{d}^*$. Note that Eq. (10) has to be understood as a maximisation for every constraint function in C and not as a vector optimisation. This approach pushes the optimiser to find design solutions that are feasible for all values of the uncertain variables. If a feasible solution cannot be found, the constraints are relaxed by defining the new constraint $C^* = C + \epsilon$ with ϵ the minimum constraint violation over U.

The optimisation and restoration loops are repeated one after the other for a prescribed number of iterations and all $\mathbf{d}^*$ and associated maxima in A_u are stored in a global archive A_g (for the relationship between the archives please refer to Fig. 1). The global archive is then used to perform a cross-check of the solutions. Given a finite number of iterations, one might obtain a solution $\mathbf{d}^*$ associated to non-globally optimal value of $\mathbf{u}_e \in A_u$. In order to mitigate this problem one can evaluate f and C taking multiple pairs $\mathbf{d}^*, \mathbf{u}_e$ taken from the archive A_g.

The overall procedure is summarised in Algorithm 1 where, without loss of generality, a single constraint C is considered. First, the design vector $\bar{\mathbf{d}}$ is initialised and two optimisations over the uncertain domain U are run keeping fixed $\bar{\mathbf{d}}$ (line 1): a constrained maximisation of f and a maximisation of the constraint violation C. The archives - A_f, A_c, A_d - are initialised (line 2). Then the inner and outer loops are alternated until the maximum number of iterations is reached (lines 3-22). In particular, the archive A_d of the design vectors $\mathbf{d}$ is updated after each outer loop (line 6) while the archives A_f of the uncertainty vectors $\mathbf{u}_{af}$ - from the maximisation of the objective function - and A_c of the vectors $\mathbf{u}_{ac}$ - from the maximisation of the constraint violation - are updated after each inner loop (respectively in lines 10 and 13) if they are not already saved in the archives. During the last loops of the algorithm the relaxation procedure could be activated if the condition expressed in line 13 is satisfied: a fixed number of iterations - arbitrarily lower than the maximum allowed - has to be reached and none of the solution saved in the whole archive $A_u = A_f \cup A_c$ has to be feasible in all the uncertainty domain U. If this happens, a small violation ϵ of the constraint C is accepted and increased as long as a feasible solution is obtained. The relaxation procedure is helped by the elimination from the archive A of all the vectors $\mathbf{u}$ previously saved with a constraint violation smaller than the actual ϵ (line 17).
The archive $A_u = A_f \cup A_c$ is a growing set: at each i-th iteration (lines 6 and 10 of Algorithm 1) $0 \leq n_{u,i} \leq 1+nc$ new vector(s) are included. Therefore, the number of crosschecks that has to be performed in the outer loop (for both f and C) at the k-th iteration for each candidate solution $\mathbf{d}$ is $\sum_{i=1}^{k} n_{u,i}$. Then, while the cost of the inner loop is constant with the number of iterations, the outer loop becomes more and more costly. However, this increment of the number of function evaluations allows a better exploration of the space of the feasible maxima of f, reduces the risk of the red queen effect [20] and reduces the total number of iterations the algorithm needs to converge.

Algorithm 1 Constrained min-max

1: Initialise $\bar{\mathbf{d}}$ at random and run $\mathbf{u}_{af} = \arg\max f(\bar{\mathbf{d}}, \mathbf{u})$ s.t. $C(\bar{\mathbf{d}}, \mathbf{u}) \leq 0$ and $\mathbf{u}_{ac} = \arg\max_{u \in U} C(\bar{\mathbf{d}}, \mathbf{u})$
2: $A_f = \{\mathbf{u}_{af}\}$; $A_c = \{\mathbf{u}_{ac}\}$; $A_d = \emptyset$
3: **while** $N_{fval} < N_{fval}^{max}$ **do**
4: *Outer loop*:
5: $\mathbf{d}_{min} = \arg\min_{d \in D}\{\max_{\mathbf{u} \in A_f \cup A_c} f(\mathbf{d}, \mathbf{u})\}$ s.t. $\max_{u \in A_f \cup A_c} C(\mathbf{d}, \mathbf{u}) \leq 0$
6: $A_d = A_d \cup \{\mathbf{d}_{min}\}$
7: *Inner loop*:
8: $\mathbf{u}_{af} = \arg\max_{\mathbf{u} \in \mathbf{U}} f(\mathbf{d}_{min}, \mathbf{u})$ s.t. $C(\mathbf{d}_{min}, \mathbf{u}) \leq 0$
9: $\mathbf{u}_{ac} = \arg\max_{u \in U} C(\mathbf{d}_{min}, \mathbf{u})$
10: $A_f = A_f \cup \{\mathbf{u}_{af}\}$
11: **if** $N_{fval} < N_{fval}^{relaxation} \vee$ $\exists \mathbf{d} \in A_d$ t.c. $\max_{u \in U} C(\mathbf{d}, \mathbf{u}) \leq 0$ **then**
12: **if** $\max_{u \in U} C(\mathbf{d}_{min}, \mathbf{u}) > 0$ **then**
13: $A_c = A_c \cup \{\mathbf{u}_{ac}\}$
14: **end if**
15: **else**
16: update ϵ
17: $A = \{A \setminus \mathbf{u}_a \mid C(\mathbf{d}_{min}, \mathbf{u}) \leq \epsilon\}$
18: **if** $\max_{u \in U} C(\mathbf{d}_{min}, \mathbf{u}) > \epsilon$ **then**
19: $A_c = A_c \cup \{\mathbf{u}_{a,C}\}$
20: **end if**
21: **end if**
22: **end while**

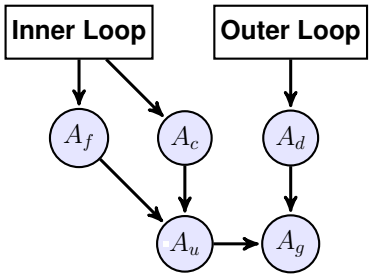

Fig. 1. Diagram of the relationships between the archives in Algorithm 1.

III. Testing Procedure

The approach to the solution of constrained min-max problems, $MacMinMax$, presented in this paper, is here applied to the benchmark in TABLE I and TABLE III using the testing procedure explained in Algorithm 2 that is a generalisation of what presented in [27]. Each problem $p_{i,j}$ corresponds to a combination of the objective function TC-$i(\mathbf{d}, \mathbf{u})$ as in TABLE I and a constraint function TCC-$j(\mathbf{d}, \mathbf{u})$ as in TABLE III: $p_{i,j} = TC\text{-}i \wedge TCC\text{-}j$. Problems are solved with $MacMinMax$ for different numbers of maximum function evaluations N_{feval}^{max} and each combination of $p_{i,j}$ and N_{feval}^{max} is repeated $n = 100$ times (line 3 of Algorithm 2). The solutions, finally, are compared with the exact min-max presented in TABLE II and TABLE IV. As suggested in [27], the *Success Rate* SR is used for the comparative assessment of the algorithm performance instead of best value, mean, and variance. SR is defined as the ratio $\frac{j_s}{n}$ where j_s is the index of performance as described in lines 5-10 of Algorithm 2. The number of successes of $MacMinMax$ on the generic $p_{i,j}$ depends on the tolerances tol_f and tol_u - on the objective function solution f and on the uncertain vector $\mathbf{u}$ respectively - and on the constraint C satisfaction. The condition is given in line 8 and it depends on the errors $\delta_f^{p_{i,j},k}$, $\delta_u^{p_{i,j},k}$ (with references in TABLES II and IV) and on $\nu_c^{p_{i,j},i}$ as described in lines 5-7. $f_{opt}(p_{i,j})$ in line 5 and $\mathbf{u}_{opt}(p_{i,j})$ in line 6 are the reference solutions for the problem $p_{i,j}$. $\mathbf{d}_{i,j,k}$, and $\mathbf{u}_{i,j,k}$ in lines 5-6 are the solution vectors calculated with Algorithm 1 at the k-iteration. $\bar{\mathbf{u}}_{i,j,k}$ in line 7 is, finally, the uncertain vector that maximise the constraint C violation with the design solution $\mathbf{d}_{i,j,k}$. tol_u is necessary to verify the convergence on the maximisation in the inner loop (restoration in section II) and then to avoid counting as success solution an $f_i(\mathbf{d}_{i,j,k}, \mathbf{u}_{i,j,k})$ close to $f_{opt}(p_{i,j})$ that is coming from a lucky combination of a wrong maximisation and a wrong minimisation in the outer loop (optimisation in section II).

Algorithm 2 Testing Procedure

1: Set for each problem p the maximum number of function evaluation N_{feval}^{max};
2: set $j_s = 0$;
3: Run n times with N_{feval}^{max} the algorithm $MacMinMax$ on each problem p;
4: **for** k = [1,2,.., n] **do**
5: Compute $\delta_f^{p_{i,j},k} = |f_{opt}(p_{i,j}) - f_i(\mathbf{d}_{i,j,k}, \mathbf{u}_{i,j,k})|$;
6: Compute $\delta_u^{p_{i,j},k} = ||\mathbf{u}_{opt}(p_{i,j}) - \mathbf{u}_{i,j,k}||$;
7: Compute $\nu_c^{p_{i,j},k} = C(\mathbf{d}_{i,j,k}, \bar{\mathbf{u}}_{i,j,k})$;
8: **if** $\nu_c^{p_{i,j},k} \leq 0 \wedge \delta_f^{p_{i,j},k} < tol_f \wedge \delta_U^{p_{i,j},k} < tol_u$ **then**
9: $j_s = j_s + 1$
10: **end if**
11: **end for**
12: $SR = \frac{j_s}{n}$

A. Benchmark

Objective functions from TC-1 to TC-6 are convex-concave test functions defined in Chapter5 of [25] and used in [23]. Functions from TC-7 to TC-12 are selected from [16], [17], [26]. TC-13 is a modification of the Rastrigin function where half of the variables are design parameters and the others are epistemic uncertainties. The dimensions of TC from TC-1 to TC-12 range from $dim_d = 1$ and $dim_u = 1$ up to $dim_d = 4$ and $dim_u = 3$; dimensions of TC-13 can go from 1 to infinite and is here kept up to $dim_d = 3$ and $dim_u = 3$. Reference solutions for $p_{i,1}$ and $p_{i,2}$ are presented in TABLE II: for the unconstrained test cases from TC-1 to TC-12 they can be found in [16], [17], [23], [25], [26] while for the unconstrained TC-13 the min-max can be easily evaluated from the minimum of the Rastringin function. Design d and uncertainty u parameters, indeed, are not coupled and then the solution of $\min_d \max_u TC\text{-}13(d, u)$ can be found decoupling the problem: for $dim_d = dim_u = 1$, the solution is shown in

TABLE I
TEST CASES FOR THE OBJECTIVE FUNCTION f

name	objective function TC
TC-1	$f(d,u) = 5(d_1^2 + d_2^2) - (u_1^2 + u_2^2) + d_1(-u_1 + u_2 + 5) + d_2(u_1 - u_2 + 3)$
TC-2	$f(d,u) = 4(d_1 - 2)^2 - 2u_1^2 + d_1^2 u_1 - u_2^2 + 2d_2^2 u_2$
TC-3	$f(d,u) = d_1^4 u_2 + 2d_1^3 u_1 - d_2^2 u_2(u_2 - 3) - 2d_2(u_1 - 3)^2$
TC-4	$f(d,u) = -\sum_{i=1}^{3}(u_i - 1)^2 + \sum_{i=1}^{2}(d_i - 1)^2 + u_3(d_2 - 1) + u_1(d_1 - 1) + u_2 d_1 d_2$
TC-5	$f(d,u) = -(d_1 - 1)u_1 - (d_2 - 2)u_2 - (d_3 - 1)u_3 + 2d_1^2 + 3d_2^2 + d_3^2$
TC-6	$f(d,u) = u_1(d_1^2 - d_2 + d_3 - d_4 + 2) + u_2(-d_1 + 2d_2^2 - d_3^2 + 2d_4 + 1) + d_3(2d_1 - d_2 + 2d_3 - d_4^2 + 5) + 5d_1^2 + 4d_2^2 + 3d_3^2 + 2d_4^2 - \sum_{i=1}^{3} u_i^2$
TC-7	$f(d,u) = (d_1 - 5)^2 - (u_1 - 5^2)$
TC-8	$f(d,u) = \min(3 - 0.2d_1 + 0.3u_1, 3 + 0.2d_1 - 0.1u_1)$
TC-9	$f(d,u) = \frac{sin(d_1 - u_1)}{\sqrt{d_1^2 + u_1^2}}$
TC-10	$f(d,u) = \frac{cos(\sqrt{d_1^2 + u_1^2})}{\sqrt{d_1^2 + u_1^2} + 10}$
TC-11	$f(d,u) = 100(d_2 - d_1^2)^2 + (1 - d_1)^2 - u_1(d_1 + d_2^2) - u_2(d_1^2 + d_2)$
TC-12	$f(d,u) = (d_1 - 2)^2 + (d_2 - 1)^2 + u_1(d_1^2 - d_2) + u_2(d_1 + d_2 - 2)$
TC-13	$f(d,u) = A(dim_{\mathbf{d}} + dim_{\mathbf{u}}) + \sum_{i=1}^{N}(d_i^2 + u_i^2 - A\big[\cos(2\pi d_i) + \cos(2\pi u_i)\big] - 5$

TABLE II
REFERENCE SOLUTIONS FOR THE TEST CASES IN TABLE I, WITHOUT CONSTRAINTS

Test Function	D	U	Reference d	Reference u	f min-max
TC-1	[-5; 5]^2	[-5; 5]^2	-0.4833	0.0833	-1.6833
			-0.3167	-0.0833	
TC-2	[-5; 5]^2	[-5; 5]^2	1.6954	0.7186	1.4039
			-0.0032	-0.0001	
TC-3	[-5; 5]^2	[-3; 3]^2	-1.1807	2.0985	-2.4688
			0.9128	2.666	
TC-4	[-5; 5]^2	[-3; 3]^3	0.4181	0.709	-0.1348
			0.4181	1.0874	
				0.709	
TC-5	[-5; 5]^3	[-1; 1]^3	0.1111	0.4444	1.345
			0.1538	0.9231	
			0.2	0.4	
TC-6	[-5; 5]^4	[-2; 2]^3	-0.2316	0.6195	4.543
			0.2228	0.3535	
			-0.6755	1.478	
			-0.0838		
TC-7	[0; 10]	[0;10]	5	5	0
TC-8	[0; 10]	[0;10]	0	0	3
TC-9	[0; 10]	[0;10]	10	2.1257	9.7794×10^{-2}
TC-10	[0; 10]	[0;10]	7.0441	10	4.2488×10^{-2}
TC-11	[-0.5; 0.5]×[0; 1]	[0;10]^2	0.5	0	0.25
			0.25	0	
TC-12	[-1; 3]^2	[0;10]^2	1	Any	1
			1	Any	
TC-13	[-5.14; 5.14]^{N_d}	[-5.14; 5.14]^{N_u}	0	$\pm$ 4.5230	$A(N_d + N_u) - 10N_d + 30.3533N_u - 5$
			...	...	
			0	$\pm$ 4.5230	
	N_d = 1	N_u = 1			35.3533
	N_d = 2	N_u = 2			75.7066
	N_d = 3	N_u = 3			116.0599
	N_d = 4	N_u = 4			156.4132

Fig. 6 where $d = 0$ and $u \in [-5.14, 5.14]^T$ is given by the maximisation of $TC\text{-}14(d,u)_{d=0} = C + u^2 - A\cos(2\pi u)$.

Constraint functions TCC-1, TCC-2 and TCC-4 depends on the design vector **d** only. In particular TCC-1 (Fig. 2) introduces a narrow concave area for the min-max solution, TCC-2 (Fig. 3) is multi-modal and TCC-4 presents plateau areas for both feasible and unfeasible regions. TCC-1, TCC-2 and TCC-4 introduce difficulties in the convergence but they do not move the exact min-max of the unconstrained TCs in TABLE II.

Functions TCC-3 and TCC-5, instead, depend on both design **d** and uncertain **u**. Also they force the solution to move from the unconstrained one in TABLE II to TABLE IV.

TCCs functions are scalable and are translated by means of

TABLE III
TEST CASES FOR THE CONSTRAINT FUNCTIONS C

name	constraint function TCC
TCC-1	$C(d) = \begin{cases} \sum_{i=1}^{n-1} \sqrt{A_1^2 - (d_1 - x_0)^2} + B_{1,i} - d_{i+1} & \text{if } d_1 \geq x_0 \text{ - R} \\ \sum_{i=1}^{n-1} \sqrt{A_1^2 - (d_1 - x_0 + 2A_1)^2} + B_{1,i} - d_{i+1} & \text{else} \end{cases}$
TCC-2	$C(d) = A_2 \cdot n + \sum[d_i^2 - A_2 \cdot cos(2\pi d_i)] - B_2$
TCC-3	$C(d,u) = \sum_{i=1}^{N} \left[\max\left(0, d_i + u_i - A_7\right)\right]$
TCC-4	$C(d,u) = \begin{cases} 1 & \text{if any } \lvert d_i - d_i^* \rvert \leq \nu_9 \\ -1 & \text{else} \end{cases}$
TCC-5	$C(d,u) = \begin{cases} -1 & \text{if } \lvert d - d^* \rvert \leq C \\ A_{14}(\lvert\lvert d \rvert\rvert + \lvert\lvert u \rvert\rvert) + \sum_{i=1}^{N}(d_i^2 + u_i^2 - A_{14}\left[\cos(2\pi d_i) + \cos(2\pi u_i)\right] - 5 & \text{else} \end{cases}$

TABLE IV
REFERENCE SOLUTIONS FOR THE TEST CASES IN TABLE I,WITH CONSTRAINTS THAT MOVE THE MIN-MAX

TC	TCC	Reference d	Reference u	f min-max		
				$dim_{d,u} = 1$	$dim_{d,u} = 2$	$dim_{d,u} = 3$
TC-13	3	[-4.140 ... -4.140]	[-4.5229 ... -4.5229]	56. 118	117.2373	178.3559
TC-13	5	[-0.9950 ... -0.9950]	[± 4.5229 ... ± 4.5229]	36.3482	77.6965	119.0447

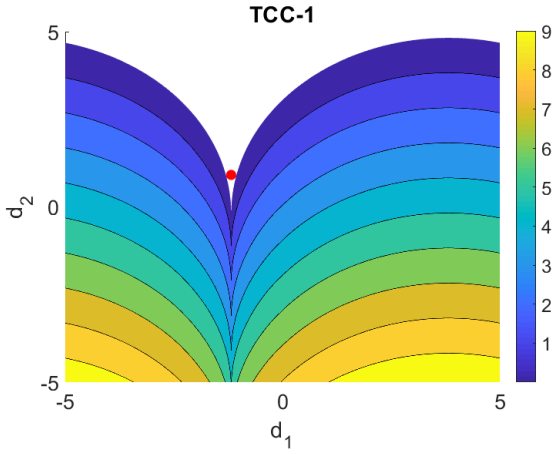

Fig. 2. Contour of TCC-1 where the white area is the feasible domain and min-max solution (red point) for $p_{6,1}$.

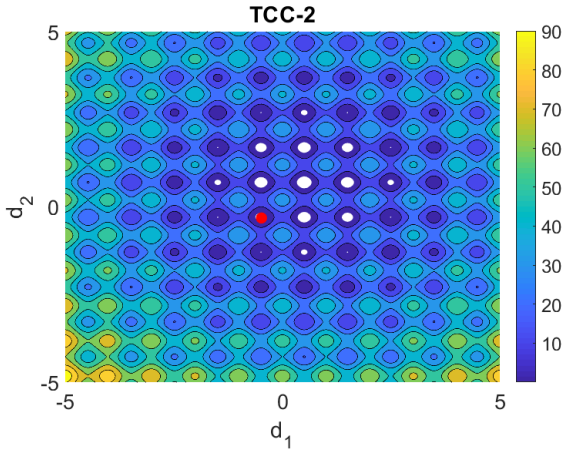

Fig. 3. Contour of TCC-2 where the white area is the feasible domain and min-max solution (red point) for $p_{1,2}$

coefficients A_i, B_i and C_i, as written in TABLE III, in order to be applied to the different TCs in TABLE II. TCC-1 and TCC-2 requires at least 2 elements in the design vector and then they are applied to test functions from TC-1 to TC-6, TC-11 and TC-12.

The optimiser used for both *optimisation* and *restoration* loops is $MP\text{-}AIDEA$ [28]. $MP\text{-}AIDEA$ has been used with one and three populations. Parameters have been set as follows: the number of agents for each population is equal to the number of variables ($dim_{\mathbf{d}}$ in the outer loop and $dim_{\mathbf{u}}$ in the inner loop); the maximum number of local restart is iun = 20 for one population and it is adaptive for 3 populations; the crossover probability, CR, and the differential weight, F are self adapted; the size of the convergence box is ρ= 0.25; the distance from the cluster centres for the global restart is δ_{global} = 0.1 and the dimension of the bubble for the local restart, if not adapted, is δ_{local} = 0.1.

B. Test Results

The convergence of $MacMinMax$ has been tested with different values of N_{feval}^{max} that results from the combination of the number of function evaluation in the inner loop N_f^{in} and in the outer loop N_f^{out}. Different tolerances tol_f and tol_u, also, have been considered. TABLE IX reports some results with 1 and 3 populations. Fig.5 shows an example of success rate for an increasing number of function calls, for the case of problem $p_{6,1}$. In the figure, for each N_{feval}^{max} the best SRs - over all combinations of N_f^{in} and N_f^{out} - is evaluated with the three criteria in line 8 of Algorithm 2.

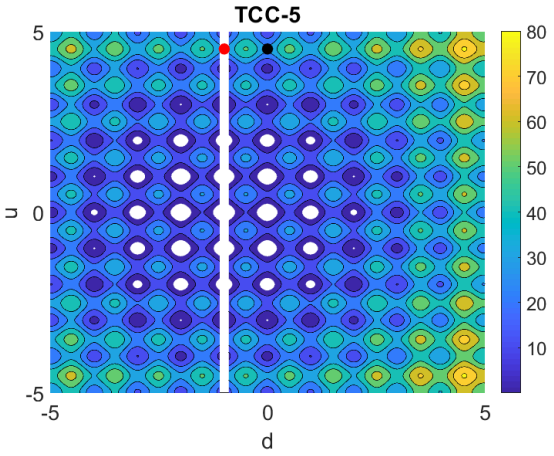

Fig. 4. Contour of TCC-5 where the white area is the feasible domain. The min-max solution of TC-13 without TCC-5 is the black point while the red point is the feasible solution for $p_{13,5}$

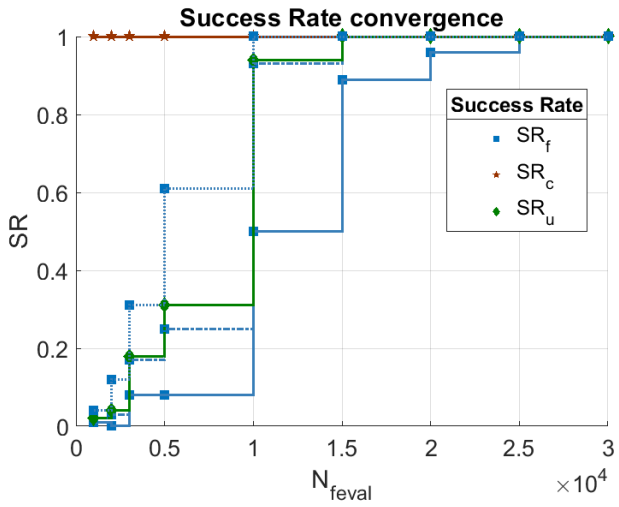

Fig. 5. Success Rates for problem $p_{6,1}$ (TC-6 and TCC-1) with different N_{feval}^{max}, evaluated from the three tolerances described in Algorithm 2 - tol_f, tol_c and tol_u - independently. In particular, SR_f has been evaluated with three values of tol_f.

In TABLES V, VI, VII, VIII, IX, X, XI and XII it is shown that the higher both N_f^{in} and N_f^{out} are, the more accurate are the maxima and minima evaluated at each optimisation-restoration loop but the overall cost of the min-max algorithms increases accordingly. If the total cost is kept fixed an increase of N_f^{in} and N_f^{out} does not necessarily lead to an improvement of the solution. In Fig. 7, for example, for $N_f^{in} = N_f^{out} = 200$ the algorithm has a poor success rate for all function evaluations. When the number of function evaluations of the inner and outer levels is increased to 300, the success rate improves rather quickly. When the number of function evaluations of the inner and outer loops is increased even further, the success rate progressively decreases, for a constant number of total function evaluations. By setting N_f^{in} too small, the convergence is not guaranteed even with a very high N_f^{out} and usually the solution is underestimated. On the other hand for a low N_f^{out}, the algorithm stagnates at different local minima - if TC-i and/or TCC-j are multimodal - and it does not converge, or converges slowly, even with an high N_f^{in}.

IV. Conclusion

In this paper we have presented a novel method for the solution of the constrained min-max problem. The algorithm was extensively tested on a new benchmark of objective and constraint functions with a variety of features that can be encountered in real-life applications. Results show that the algorithm we proposed is generally successful at identifying the constrained min-max solution with a limited number of calls to objective functions and constraints. The unconstrained version of the algorithm proposed in this paper was already proven to be efficient and reliable compared to existing evolutionary and non-evolutionary algorithms. We argue that also this constrained version is equally reliable and efficient given the good success rate displayed on most of the test functions.

Future developments will include the use of surrogate models to further reduce the calls to objective and constraints functions.

TABLE V
RESULTS FOR TCC-1

TC	#feval tol_f = 0.001	SR$_{TC}$	#feval tol_f = 0.01	SR$_{TC}$	#feval tol_f = 0.1	SR$_{TC}$
			tol_u = 0.01			
1	3000	95	3000	95	3000	95
2	30000	97	30000	97	30000	97
3	30000	88	30000	88	30000	88
4	20000	99	20000	99	20000	99
5	10000	99	10000	99	10000	99
6	50000	90	50000	90	50000	90
11	1500	97	1000	90	1000	99
12	5500	92	4000	95	2500	97
			tol_u = 0.1			
1	3000	95	3000	99	2000	97
2	15000	92	10000	95	5000	95
3	20000	96	10000	93	10000	94
4	15000	94	10000	99	10000	100
5	10000	99	5000	99	2000	91
6	40000	92	25000	98	20000	95
11	1500	97	1000	90	1000	100
12	5500	92	4000	95	2500	97

TABLE VI
RESULTS FOR TCC-1 AND TC-13

dim	#feval tol_f = 0.001	SR$_{TC}$	#feval tol_f = 0.01	SR$_{TC}$	#feval tol_f = 0.1	SR$_{TC}$
			tol_u = 0.01			
1	5000	100	5000	100	1000	94
2	20000	94	20000	95	15000	90
3						
			tol_u = 0.1			
1	5000	100	5000	100	5000	100
2	20000	94	20000	95	15000	90
3						

TABLE VII
RESULTS FOR TCC-2

TC	#feval tol_f = 0.001	SR_{TC}	#feval tol_f = 0.01	SR_{TC}	#feval tol_f = 0.1	SR_{TC}
			tol_u = 0.01			
1	15000	95	15000	96	15000	96
2	30000	98	30000	98	30000	98
3	90000	71	90000	71	90000	71
4	30000	94	30000	94	30000	94
5	30000	100	30000	100	30000	100
6	100000	13	100000	13	100000	13
11	2500	94	2500	97	1500	91
12	7000	78	7000	87	7000	89
			tol_u = 0.1			
1	15000	95	10000	100	5000	91
2	20000	91	20000	98	10000	95
3	90000	89	50000	98	40000	93
4	30000	98	15000	98	10000	98
5	30000	100	10000	91	10000	98
6	100000	23	100000	74	90000	90
11	2500	94	2500	97	1500	91
12	7000	78	7000	87	6000	90

TABLE VIII
RESULTS FOR TCC-2 AND TC-13

dim	#feval tol_f = 0.001	SR_{TC}	#feval tol_f = 0.01	SR_{TC}	#feval tol_f = 0.1	SR_{TC}
			tol_u = 0.01			
1	1000	99	1000	99	1000	100
2	10000	95	10000	96	10000	96
3	90000	92	90000	93	60000	90
			tol_u = 0.1			
1	1000	99	1000	99	1000	100
2	10000	95	10000	96	10000	96
3	90000	92	90000	93	60000	90

TABLE IX
RESULTS FOR TCC-3 AND TC-13

dim	#feval tol_f = 0.001	SR_{TC}	#feval tol_f = 0.01	SR_{TC}	#feval tol_f = 0.1	SR_{TC}
			1 population MP-AIDEA			
			tol_u = 0.01			
1	30000	68	5000	70	20000	71
2	80000	30	50000	73	70000	76
3	70000	3	90000	24	80000	36
			tol_u = 0.1			
1	30000	68	5000	70	20000	71
2	80000	30	50000	73	70000	76
3	100000	3	90000	24	80000	36
			3 population MP-AIDEA			
			tol_u = 0.01			
1	5000	94	5000	96	5000	96
2	100000	38	100000	86	90000	90
3	80000	3	80000	11	80000	32
			tol_u = 0.1			
1	5000	94	5000	96	5000	96
2	100000	38	100000	86	90000	90
3	80000	3	80000	11	80000	32

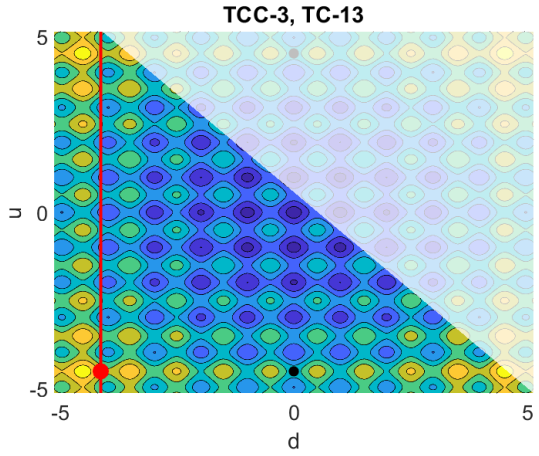

Fig. 6. problem $p_{13,3}$ with $dim_d = dim_u = 1$. The contour corresponds to TC-13 while the area in transparency is the set of the unfeasible configurations (d,u). The black points correspond to the solution of the unconstrained problem. The solution is feasible for any realisation of u if $d \in$ [-5.14 5.14]. The red point is the solution for the constrained min-max problem.

TABLE X
RESULTS FOR TCC-4 WITH $\nu = 0.01$

TC	#feval tol_f = 0.001	SR_{TC}	#feval tol_f = 0.01	SR_{TC}	#feval tol_f = 0.1	SR_{TC}
			tol_u = 0.01			
1	10000	95	10000	95	10000	95
2	30000	95	30000	95	30000	95
3	50000	99	50000	99	50000	99
4	30000	100	30000	100	30000	100
5	10000	96	10000	96	10000	96
6	50000	96	50000	96	50000	96
7	1000	100	1000	100	1000	100
8	4500	76	4500	76	4500	76
9	6000	19	6000	19	6000	19
10	5000	53	5000	53	5000	53
11	5500	88	6000	93	5500	96
12	4000	91	3500	92	3000	94
			tol_u = 0.1			
1	10000	95	10000	95	10000	95
2	30000	95	30000	95	30000	95
3	30000	98	30000	99	30000	99
4	30000	100	30000	100	30000	100
5	10000	96	10000	96	10000	96
6	40000	96	40000	96	40000	96
7	1000	100	1000	100	1000	100
8	4500	76	4500	76	4500	76
9	6000	19	6000	19	6000	19
10	5000	54	5000	54	5000	54
11	5500	88	6000	93	5500	96
12	4000	91	3500	92	3000	94

REFERENCES

[1] H. G. Beyer and B. Sendhoff, "Robust optimisation - A comprehensive survey", Comp. Methods Appl. Mech. Emgrg. 196 (2007) 3190-3218.

[2] B. Rustem, "Algorithms for Nonlinear Programming and Multiple Objective Decisions", Wiley, Chichester (1998).

[3] R. W. Chaney, "A method of centers algorithm for certain minimax problems", Volume 22, Issue 1, pp 202-226 (1982).

[4] R. Klessig and E. Polak, "A method of feasible directions using function approximations with applications to min max problems", Journal of Mathematical Analysis and Applications,vol: 41 (3) pp: 583-602 (1973).

[5] V. Panin, Linearisation method for continuous minmax problem", Springer, Volume 17, Issue 2, pp 239243, March 1981.

[6] Y. M. Danilin, V. M. Panin, and B. N. Pshenichnyi, "On the Shannon Capacity of a graph", Notes Control and Information Sci, vol: 23 (30) pp: 51-57 (1982).

[7] VF Damyanov, VN Malozemov (1974) Wiley, New York.

[8] D. Agnew, "Improved minimax optimization for circuit design", IEEE Transactions on Circuitsand Systems , vol. 28, no. 8, 1981.

TABLE XI
RESULTS FOR TCC-4 WITH $\nu = 0.1$

TC	#feval tol_f = 0.001	SR_{TC}	#feval tol_f = 0.01	SR_{TC}	#feval tol_f = 0.1	SR_{TC}
			tol_u = 0.01			
1	10000	100	10000	100	10000	100
2	15000	99	15000	99	15000	99
3	40000	99	40000	99	40000	99
4	15000	95	15000	95	15000	95
5	10000	100	10000	100	10000	100
6	30000	96	30000	96	30000	96
7	1000	100	1000	100	1000	100
8	3000	95	3000	95	3000	95
9	5500	61	5500	61	5500	61
10	5000	94	5000	94	5000	94
11	1500	100	1500	100	1000	96
12	5000	88	3500	90	3000	97
			tol_u = 0.1			
1	10000	100	10000	100	5000	99
2	15000	99	15000	100	15000	100
3	30000	99	20000	99	20000	100
4	15000	98	10000	96	10000	97
5	10000	100	5000	100	5000	100
6	30000	100	15000	98	15000	100
7	1000	100	1000	100	1000	100
8	3000	95	2000	95	2000	95
9	6000	79	5500	84	5500	84
10	5000	97	5000	97	5000	97
11	1500	100	1500	100	1000	96
12	5000	88	3500	90	3000	97

TABLE XII
RESULTS FOR TCC-5 AND TC-13

dim	#feval tol_f = 0.001	SR_{TC}	#feval tol_f = 0.01	SR_{TC}	#feval tol_f = 0.1	SR_{TC}
			tol_u = 0.01			
1	10000	100	10000	100	10000	100
2	70000	94	50000	90	50000	91
3	90000	49	90000	49	90000	49
			tol_u = 0.1			
1	10000	100	10000	100	10000	100
2	70000	94	50000	90	50000	91
3	90000	49	90000	49	90000	49

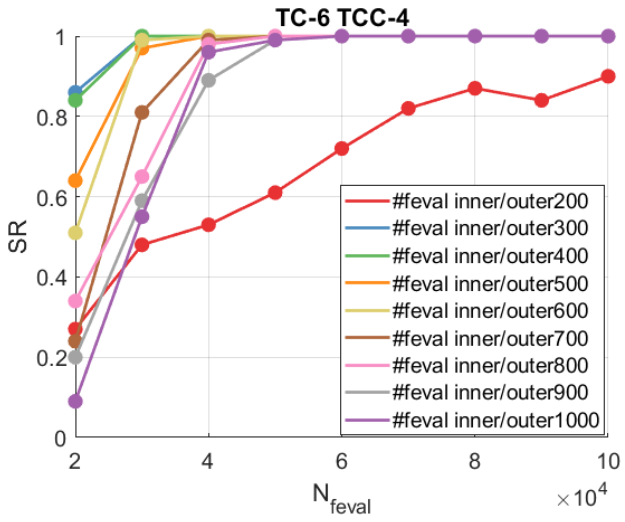

Fig. 7. Convergence of the Success Rate for problem $p_{6,4}$ with an increasing number of $N_f^{in} = N_f^{out}$.

[9] J. S.H. Wang and W. W.M. Dai, "Transformation of min-max optimization to least-square estimation and application to interconnect design optimization," Proceedings of ICCD '95 International Conference on Computer Design. VLSI in Computers and Processors, Austin, TX, USA, 1995, pp. 664-670.

[10] B. Lu, Y. Cao, M. Yuan and J. Zhou, "Reference variable methods of solving minmax optimization problems", Journal of Global Optimization, 2008 vol: 42 (1) pp: 1-21.

[11] M. A. Sainz, P. Herrero, J. Armengol and J. Vehi, "Continuous minimax optimization using modal intervals", J. Math. Anal. Appl, 2008 vol: 339 pp: 18-30.

[12] Y. Feng, L. Hongwei, Z. Shuisheng and L. Sanyang, "A smoothing trust-region Newton-CG methodfor minimax problem", Applied Mathematics and Computation, 2008 vol: 199 (2) pp: 581-589.

[13] P. Parpas and B. Rustem, "An Algorithm for the Global Optimization of a Class of ContinuousMinimax Problems", Journal of Optimization Theory and Applications, 2009 vol: 141 (2) pp:461-473.

[14] T. M. Cavalier, W. A. Conner, E. del Castillo and S. I. Brown, "A heuristic algorithm for minimax sensor location in the plane", European Journal of Operational Research , vol. 183,no. 1, pp. 4255, 2007.

[15] D. Ahr and G. Reinelt, "A tabu search algorithm for the minmax k-chinese postman problem", Computers and Operations Research , vol. 33, no. 12, pp. 3403-3422, December 2006.

[16] A.M. Cramer, S.D. Sudhoff and E.L. Zivi, "Evolutionary algorithms for minimax problems in robust design". IEEE Trans. Evolut. Comput. 13(2), 444-453 (2009)

[17] R.I. Lung and D. Dumitrescu, "A new evolutionary approach to minimax problems". In: Proceedings of the 2011 IEEE Congress on Evolutionary Computation, New Orleans, USA, pp. 1902-1905 (2011)

[18] A. Sinha, Z. Lu, K. Deb and P. Malo, "Bilevel Optimization based on Iterative Approximation of Multiple Mappings".

[19] A. Sinha, P. Malo, K. Deb, "A Review on Bilevel Optimization: From Classical to Evolutionary Approaches and Applications".

[20] M. Vasile, "On the solution of min-max problems in robust optimisation". In: The EVOLVE 2014 International Conference, A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computing, 2014-07-01 - 2014-07-04, Jian-Guo Hotel (2014).

[21] K. Shimizu and E. Aiyoshi, "Necessary conditions for min-max problems and algorithms by a relaxation procedure", IEEE Trans. Autom. Control25 (1), 6266 (1980).

[22] K. Shimizu, Y. Ishizuka and J.F. Bard, "Non-differentiable and Two-level Mathematical Programming", Springer US (1997)

[23] J. Marzat, E. Walter and H. P. Lahanier, "Worst-case global optimisation of black-box functions through Kriging and relaxation", Journal of Global Optimization, 2013.

[24] G. Filippi, M. Marchi, M. Vasile and P. Vercesi, "Evidence-Based Robust Optimisation of Space Systems with Evidence Network Models", In 2018 IEEE Congress on Evolutionary Computation (CEC) (pp. 18).

[25] B. Rustem and M. Howe, "Algorithms for Worst-Case Design and Applications to Risk Management", Princeton University Press, Princeton, NJ (2002).

[26] A. Zhou, Q. Zhang, "A surrogate-assisted evolutionary algorithm for minimax optimization". IEEE Congress on Evolutionary Computation, Barcelona, Spain, pp. 17 (2010)

[27] M. Vasile, E. Minisci and M. Locatelli, "An Inflationary Differential Evolution Algorithm for Space Trajectory Optimization", IEEE Transaction on Evolutionary Computation, vol. 15, no. 2, April 2011.

[28] M. Di Carlo, M. Vasile and E. Minisci, "Multi-population adaptive inflationary differential evolution", BIOMA (2014).