Optimization and Engineering
https://doi.org/10.1007/s11081-021-09613-3

RESEARCH ARTICLE

Global solution of constrained min-max problems with inflationary differential evolution

Gianluca Filippi[1] · **Massimiliano Vasile**[1]

Received: 31 July 2020 / Revised: 19 January 2021 / Accepted: 1 March 2021

Abstract
This paper proposes a method for the solution of constrained min-max problems. The method is tested on a benchmark of representative problems presenting different structures for the objective function and the constraints. The particular min-max problem addressed in this paper finds application in optimisation under uncertainty when the constraints need to be satisfied for all possible realisations of the uncertain quantities. Hence, the algorithm proposed in this paper search for solutions that minimise the worst possible outcome for the objective function due to the uncertainty while satisfying the constraint functions in all possible scenarios. A constraint relaxation and a scalarisation procedure are also introduced to trade-off between objective optimality and constraint satisfaction when no feasible solutions can be found.

Keywords Constrained min-max · Worst case scenario · Optimisation · Uncertainty · Constraint relaxation · Scalarisation

Abbreviations

AOCS	Attitude and orbit control system
CEdS	Complex engineered system
CPSS	Chebychev/Pascoletti-Serafini scalarisation
CrDE	Crowding differential evolution
DE	Differential evolution
ECS	Epsilon-Constraint scalarisation
EGO	Efficient global optimisation
GA	Genetic algorithm

✉ Gianluca Filippi
g.filippi@strath.ac.uk

Massimiliano Vasile
massimiliano.vasile@strath.ac.uk

[1] University of Strathclyde, James Weir Building, 75 Montrose St, Glasgow G1 1XJ, Scotland

Published online: 02 April 2021

MP-AIDEA	Multi-population adaptive inflationary differential evolution algorithm
OBDH	On board data handling
PSO	Particle swarm optimisation
PSS	Pascoletti-Serafini scalarissation
QoI	Quantity of interest
TTC	Telemetry, tracking and command system
WCS	Weighted chebyshev scalarisation
WSS	Weighted-Sum scalarisation

1 Introduction

A min-max optimisation problem aims at minimising, with respect to a vector $\mathbf{d}$ defined in some space D, the maximum value of a given cost function with respect to a vector $\mathbf{u}$, different from $\mathbf{d}$. In this paper we are interested in a class of constrained min-max problems where the constraints have to be always satisfied for all values of $\mathbf{u}$ in a given set U.

Min-max problems appear in a wide range of applications. They can be found in the optimal sequence of moves for a computer player in famous board games: the Egyptian ancient game Seega (Abdelbar et al. 2003), the game of checkers (Hughes 2005) and chess (Shannon 1950). They are used to formulate the design for robustness of systems affected by uncertainty: design of electric circuits (Agnew 1981), design of on-line controllers (Sebald and Schlenzig 1994), of aerodynamic shapes (Ong et al. 2006), of the location of sensors and pursuit-evasion games between missile and target (Kim and Tahk 2001).

Due to their wide applicability, they can appear in different forms. They can be seen indeed as a special case of *bi-level* optimisation problems where two agents quantify their utility with opposite fitness functions. From a game theoretic point of view min-max problems are two-players zero-sum games where the optimal strategy brings the two agents to a Nash equilibrium. From an engineering point of view min-max problems can be seen as deterministic approaches to make decisions under uncertainty. In this case one can understand the engineering problem as a zero-sum game where the two antagonist players are respectively the designer, handling the decision variables, and Nature, handling the uncertainty variables.

Mathematical Programming (Agnew 1981; Chaney 1982) has been widely used to solve min-max problems. An overview of classical discrete approaches can be found in Aissi et al. (2008). The use of Mathematical Programming, however, requires, strong assumptions on the nature of the problem and tends to be problem specific.

On the other hand heuristic approaches (Cramer et al. 2009; Lung and Dumitrescu 2011; Cavalier et al. 2007) and in particular evolutionary based algorithms (Cramer et al. 2009; Lung and Dumitrescu 2011; Laskari et al. 2002; Zhou and Zhang 2010) appear to be more flexible. For example one can find a form of Genetic Algorithm (GA) in Cramer et al. (2009), a special version of the Differential evolution (DE) called Crowding Differential Evolution (CrDE) combined with the Nash ascendancy

relation in Lung and Dumitrescu (2011) and a form of Particle Swarm Optimisation (PSO) in Laskari et al. (2002).

As stated in Zhou and Zhang (2010) we can divide the class of evolutionary approaches for min-max problems in three branches. The first one includes problems with a discrete set of possible scenarios U. In Laskari et al. (2002) U is by definition discrete and small, in Cavalier et al. (2007), initially continuous, it is instead discretised by a uniform grid while in Cramer et al. (2009) it is reduced to be discrete by a random selection. The second branch considers a continuous space U and directly solves a sequence of nested problems (Agnew 1981; Sebald and Schlenzig 1994). This approach could be computationally intractable if no strategy is implemented to alleviate the cost. Finally, a third category uses a form of co-evolution optimisation (Kim and Tahk 2001; Cramer et al. 2009; Tahk and Sun 2000; Shi and Krohling 2002) where two separate populations evolve simultaneously with a predator-prey interaction in D and U separately each one being competitor and environment to the other. As demonstrated in Jensen (2001) however most of the co-evolutionary algorithms for min-max have to satisfy a symmetric assumption known as Issac's condition (Tamer Basar 1982). A class of problems that can be properly modelled under this assumption exists (Tahk and Sun 2000; Barbosa 1999) and for them the co-evolution approach is proved to converge to the global solution (Power 1998). However this is not the general case and new methods to overcome this limitation are presented in Branke and Rosenbusch (2008).

In the interest of completeness, we include also the *Best Replay* approach cited by Marzat et al. (2013) where two optimisation problems are alternated until convergence, one minimising over D and the other maximising over U. It has been proven, however, that this approach often does not converge or it cycles through wrong design candidates, problem known as the Red Queen Effect.

An other research line, applicable to any previously presented method, deals with the reduction of the computational complexity. It makes use of surrogates which approximate the fitness function (Lung and Dumitrescu 2011; Zhou and Zhang 2010; Marzat et al. 2013). In particular, Zhou and Zhang (2010), Marzat et al. (2013) consider response surfaces within an Efficient Global Optimisation (EGO) where they alternate the minimisation in D of the surrogate of the maxima in U and the maximisation in U of the real function for the obtained optimal solution in D.

When it comes to constraint handling, in the existing literature, only few papers could be found that have explored how to deal with constraints in min-max 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 (Sinha et al. 2020, 2018). In Kim and Tahk (2001) the duality between primal constrained min-max problem and dual unconstrained min-max problem with the Lagrange multipliers is demonstrated for both separable and non-separable constraints under the Issac's condition. The constrained problem is then translated in the unconstrained one and a co-evolutionary algorithm is finally used to converge to the saddle point. Other works instead, still dealing with both min-max problem and constraint handling, go in the opposite direction. They indeed translate a single-level constrained problem in an

unconstrained min-max problem and use the co-evolution approach to solve it (Tahk and Sun 2000; Shi and Krohling 2002).

This paper proposes a novel algorithm for the solution of a class of constrained min-max problems. The proposed algorithm is inspired by the procedure suggested by Shimizu and Aiyoshi (1980) and further elaborated in Zhou and Zhang (2010), Marzat et al. (2013). The idea is to alternate a *minimisation* and a *restoration* (or maximisation) process. As the alternation of these two processes progresses, we incrementally build a discrete representation of the space of the maxima in U. This representation is then used in the minimisation process to converge to an approximation of the desired min-max solution. This mechanism takes memory of the previous solutions avoiding the optimiser to follow the same path again and again. For this reason the proposed algorithm does not suffer from the Red queen Effect. The main novelty is the extension of this logic to the handling of constraints that need to be satisfied for all maxima in a given set. The paper proposes also a constraint relaxation and a scalarisation techniques that allow convergence to a solution in the case the given set U is too restrictive. A multi-population inflationary differential evolution algorithm is proposed to address both the constrained minimisation and the maximisation.

The paper is structured as follows. First we introduce the constrained min-max problems of interest. We then present the general algorithm together with a complexity analysis. The paper then introduces a benchmark of synthetic functions with increasing complexity. Finally the proposed method is tested on a real application case of space systems engineering under epistemic uncertainty.

2 Problem statement

This paper is concerned with the following class of single-objective constrained min-max problems:

$$\begin{aligned} &\min_{\mathbf{d}\in D} \max_{\mathbf{u}\in U} f(\mathbf{d},\mathbf{u}) \\ &s.t. \\ &c_i(\mathbf{d},\mathbf{u}) \leq 0 \quad \forall \mathbf{u} \in U, \forall i \in I_c = [1,...,s]^T \end{aligned} \tag{1}$$

where f is the objective function and c_i is the i-th constraint function. Both f and c_i are defined on the space $D \times U$ and depend on a vector of design (or decision) variables $\mathbf{d} \in D \subset \mathcal{R}^n$ and a vector of uncertain (or environmental) variables $\mathbf{u} \in U \subset \mathcal{R}^m$. The solution $\mathbf{d}_{\text{opt}}$ of Eq. (1) has two properties: it satisfies all the constraint functions c_i over the whole uncertain domain U and minimises the worst realisation of the objective function f over U. Furthermore, we assume that both f and c_i are locally $\mathcal{C}^2$. This assumption can be relaxed if the local search can handle non-differentiable or discontinuous problems.

3 A memetic single objective constrained min-max approach

The proposed algorithm is a bi-level optimisation procedure based on the alternation of a minimisation and a restoration step. The minimisation step searches for a global solution to the constrained min-max problem:

$$\begin{aligned} &\min_{\mathbf{d}\in D} \max_{\mathbf{u}_{af}\in \bar{A}_{uf}} f(\mathbf{d}, \mathbf{u}_{af}) \\ &s.t. \\ &\max_{\mathbf{u}_{ac}\in \bar{A}_{uc}} \max_{i\in I_c} c_i(\mathbf{d}, \mathbf{u}_{ac}) \leq 0. \end{aligned} \tag{2}$$

While the restoration step searches for a solution to the following two global maximisation problems, given the solution $\bar{\mathbf{d}}$ coming from problem (2):

$$\begin{aligned} &\max_{\mathbf{u}\in U} f(\bar{\mathbf{d}}, \mathbf{u}) \\ &s.t. \\ &\max_{i\in I_c} c_i(\bar{\mathbf{d}}, \mathbf{u}) \leq 0 \end{aligned} \tag{3}$$

$$\max_{\mathbf{u}\in U} \max_{i\in I_c} c_i(\bar{\mathbf{d}}, \mathbf{u}) \tag{4}$$

The two archives $\bar{A}_{uf}$ and $\bar{A}_{uc}$ contain respectively the solutions of problem (3) and (4). By iteratively alternating the minimisation and restoration steps, one fills the two archives with the maxima of problems (3) and (4). Thus we can say that problem (2) searches for an optimal $\mathbf{d}$ over a discrete representation of the space of the maxima of problems (3) and (4).

This mechanism was first proposed for unconstrained problems by Shimizu and Aiyoshi (1980) and Marzat et al. (2013). A solution approach using a memetic algorithm was then introduced in Vasile (2014) and the constrained version was first formulated in Filippi and Vasile (2019). In this paper we provide five main contributions: (i) a detailed algorithmic presentation of a memetic solution approach, (ii) a constrained relaxation strategy, (iii) a scalarisation strategy, (iv) an algorithmic complexity analysis and (v) an extensive benchmark on which our proposed solver of constrained min-max problems can be tested.

The solution approach is summarised in Algorithms 1–8 and explained in the following subsections with the help of the flow diagrams in Figs. 1 and 2 and the examples in Figs. 3, 5 and 6.

3.1 Initialisation

The algorithm is initialised either by selecting a random value $\bar{\mathbf{d}}$ or with a given first guess and then by searching for a first solution to problems (3) and (4). These first solutions are then saved into the two archives A_{uf} and A_{uc}. Problems (3) and (4) are global optimisation problems in general. In the following we will propose the use of a memetic global optimiser which combines Differential Evolution with a multi-restart mechanism and local search. Given the global nature of the search for a solution, one needs to set the amount of computational resources allocated to the solution of each of the minimisation and maximisation problems. We quantify this resource in terms of objective function evaluations. More specifically, with reference to problem (3) the maximum number of function evaluations $n_{\text{feval,max}}$ is the number of calls to the objective function $f(\bar{\mathbf{d}}, \mathbf{u})$ while for problem (4) we count the function calls to $\max_i c_i(\bar{\mathbf{d}}, \mathbf{u})$. Once the maximum number of function evaluations per sub-

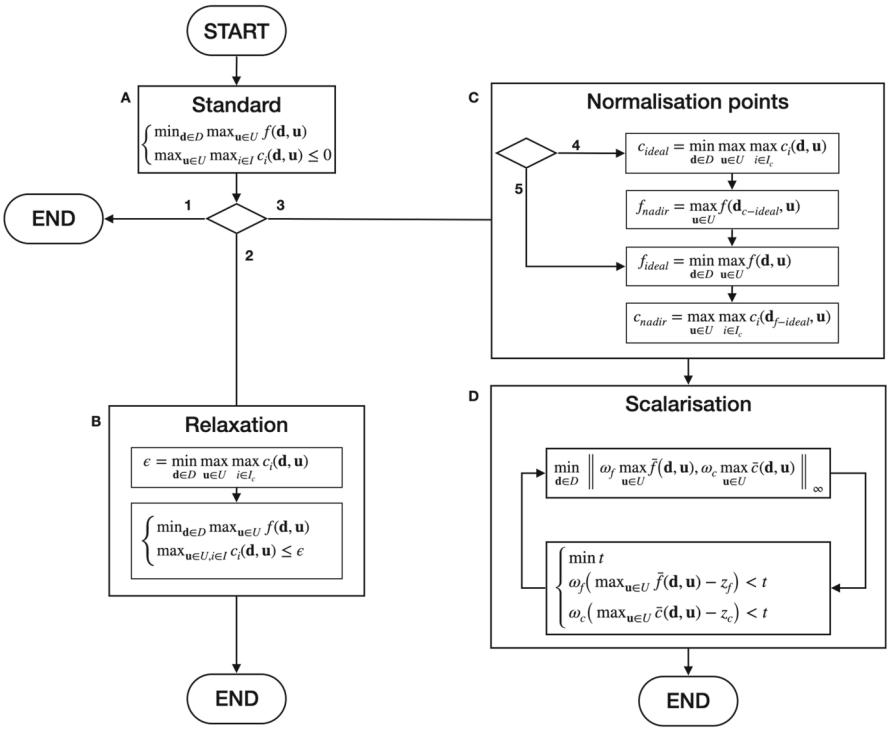

Fig. 1 Flow diagram of the constrained min-max alternative approaches described in Sects. 3.1, 3.2 and 3.4 – 3.7 and summarised in Algorithms 1–8. In particular, this diagram describes the sequence of optimisation problems applied for the standard approach and its alternative strategies: the constraint relaxation and the scalarisation. The first optimisation problem is defined in block A and it refers to the general constrained min-max in Eq. (1). If a solution exists for this problem, no further analysis is required (link 1). If differently, two alternatives are given. The first (link 2) brings to the relaxation strategy (block B). Here an unconstrained min-max problem is solved to find ϵ. The relaxed constrained min-max problem is then solved (Sect. 3.4). Link 3 instead brings to the scalarisation strategy (Sect. 3.5). Block C is first solved to find the reference points in the Pareto front (ideal and nadir). Finally, the scalarisation procedure is activated (block D)

problem are defined one needs to define the maximum number of iterations $n_{\text{loop,max}}$ of minimisation/restoration.

The steps for the initialisation are summarised in Algorithm 1. Line 1 presents how to define the problem in order to be solved by Algorithm 2: the required information on objective function $f(\mathbf{d}, \mathbf{u})$, set of constraint functions $\mathbf{c}(\mathbf{d}, \mathbf{u})$, dimensions n_D and n_U of the design and uncertain vectors $\mathbf{d} \in D \subset \mathcal{R}^{n_D}$ and $\mathbf{u} \in U \subset \mathcal{R}^{n_U}$ and the bounds of each variable in $\mathbf{d}$ and $\mathbf{u}$. Line 2 defines the constraints on the computational resources for the single sub-problems in Eqs. (2) – (4). They are the maximum number of function evaluations $n^{\text{outer}}_{\text{feval,max}}$ for problem (2), $n^{\text{inner}}_{\text{feval,max}}$ for problem (3) and $n^{\text{inner},c}_{\text{feval,max}}$ for problem (4). Line 3 defines the threshold σ_{stop} below which Algorithm 2 is considered to have converged. Line 4 defines the limit on the computational cost for the whole procedure (1) with its maximum number of function evaluations $n_{\text{fval,max}}$ and maximum number of iterations $n_{\text{loop,max}}$ between minimisation and restoration.

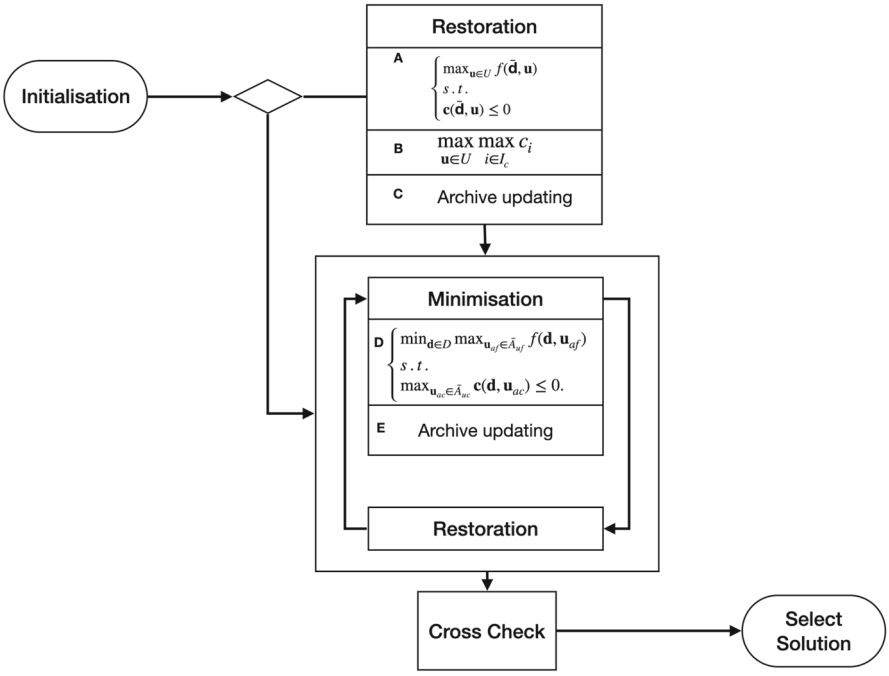

Fig. 2 Flow diagram of the constrained min-max algorithm for the standard approach in block A in Fig. 1. The relaxation and scalarisation alternatives (blocks B,C and D) follow however the same logic. Algorithms 2, 1, 4, 7 and 8 and Sects. 3.1, 3.2, 3.6 and 3.7 explain in detail all the represented blocks. Each of them lists the main operations that are performed: the optimisation problems and the archives updating. The algorithm is initialised. Then there is the iteration between blocks 'Restoration' and 'Minimisation'. Finally the 'Cross Check' is performed and the solution is chosen

Line 5 defines the optimisation algorithms and its parameter settings that are used to solve the sub-problems in Eq. (2) – (4). On line 6 the initial design vector is defined by the user or initialised randomly with a hyper-cube sampling procedure. In the case some important uncertain scenarios in U or design configurations in D are already known, the initial archives can be seeded with these scenarios, otherwise they are initialised as empty sets (line 7). Similarly, it is done for the archives relative to functions f and $\mathbf{c}$ (line 8). Lines 9 and 10 initialise the counters of the number of function evaluation n_{feval} and the iteration n_{loop} between minimisation and restoration processes. The initial accepted constraint violation ϵ (line 11) is initialised to zero.

3.2 Minimisation-restoration loop

The main algorithm is summarised in Algorithm 2. Using the first design guess $\bar{\mathbf{d}}$ from the initialisation, Eqs. (3) and (4) are solved in parallel (lines 4 and 5) within the first restoration, or inner, loop (lines from 2 – 8). With $\bar{\mathbf{d}}$ fixed, the former equation evaluates the feasible worst case condition of f while the latter determines the worst constraint violation. In the following we will adopt a multi-population version of Infla-

Algorithm 1 Initialisation

1: Define the problem: objective function f, constraint function c, dimension (n_D and n_U), lower and upper bounds for both design space D and uncertain space U.
2: Define computational limits to the sub-problems: $n^{\text{inner},f}_{\text{fval,max}}$, $n^{\text{inner},c}_{\text{feval,max}}$, $n^{\text{outer}}_{\text{fval,max}}$
3: Define algorithm convergence threshold σ_{stop}.
4: Define computational limits to the whole algorithm: $n_{\text{fval,max}}$ and $n_{\text{loop,max}}$
5: Define optimiser(s) and its (their) parameters setting for all optimisation steps
6: Define/initialise design vector $\bar{\mathbf{d}}$ (latin hypercube sample or input)
7: Define/initialise vectors archives: A_{uf}, A_{uc}, A_d, A_u (∅ or input)
8: Define/initialise functions archives: A_f, A_{cf}, A_c (∅ or input)
9: Initialise number of function evaluation $n_{\text{feval}} = 0$
10: Initialise number of loops $n_{\text{loop}} = 0$
11: Initialise accepted violation $\epsilon = 0$

tionary Differential Evolution (Carlo et al. 2019) to solve Eqs. (3) and (4). In this case multiple solutions are returned. The solution $\mathbf{u}_{af}$ is selected following Algorithm 3. In particular, if none of the outputs from the optimiser is feasible (no elements in no one of the populations), the uncertain vector with the lowest constraint violation is chosen (lines 2 – 3 of Algorithm 3). Otherwise, within the set of feasible solutions, the uncertain vector with the highest value of f is selected (line 5 of Algorithm 3). The uncertain vectors solutions are then stored in the archives as illustrated in Algorithm 4. Consider first lines 9 – 11 of Algorithm 4: the uncertain vector solutions $\mathbf{u}_{af}$ corresponding to the feasible maximum of f for the fixed $\bar{\mathbf{d}}$ is saved in A_{uf}, the corresponding $f(\bar{\mathbf{d}}, \mathbf{u}_{af})$ and $\max_{i\in I_c} c_i(\bar{\mathbf{d}}, \mathbf{u}_{af})$ are saved in A_f and A_{cf} respectively. Consider now lines 12 and 13 of Algorithm 4: the uncertain vector solutions $\mathbf{u}_{ac}$ that maximises the constraint violation for the same fixed design vector is saved in A_{uc}, while the corresponding $\max_{i\in I_c} c_i(\bar{\mathbf{d}}, \mathbf{u}_{ac})$ is saved in A_c. Finally, on lines 14 and 15 of Algorithm 4 the archives $\bar{A}_{uf}$ and $\bar{A}_{uc}$ are created by removing from A_{uf} and A_{uc} all the repeated elements (with a selected tolerance of 1e-8 on the euclidean distance between two elements in the archive).

The main loop is then started (line 9 of Algorithm 2 and Eq. (2)) with an alternation of the minimisation, or outer, step (lines 10, 11 and 12) and restoration, or inner, step (lines from 13 to 17). The latter has been already described. A further check is, however, performed in lines 1 to 8 of Algorithm 4 before the updating procedures of the archives. If the condition in line 2 holds, indeed, the inner loop in Eq. (3) has failed. Then the solution generated in this loop is discarded and the one of the previous outer loop is maintained. This condition follows the same criterion used in Algorithm 3 by giving priority to the constraint satisfaction. Similarly the condition in line 6 for Eq. (4).

The outer step addresses the solution of the constrained minimisation in Eq. (4). Also for this minimisation loop we propose the use of a multi-population version of Inflationary Differential Evolution (Carlo et al. 2019). For each new design vector generated by the optimiser the cost function and constraints are evaluated for all the $\mathbf{u}$ vectors in $\bar{A}_{uf}$ and $\bar{A}_{uc}$ and the worst cost function and constraint violation values are retained (line 11). Note that in some cases it is desirable to run a local search every time a vector in the two archives $\bar{A}_{uf}$ and $\bar{A}_{uc}$ is evaluated (see for example Vasile (2014)). This added local search significantly increases the computational cost

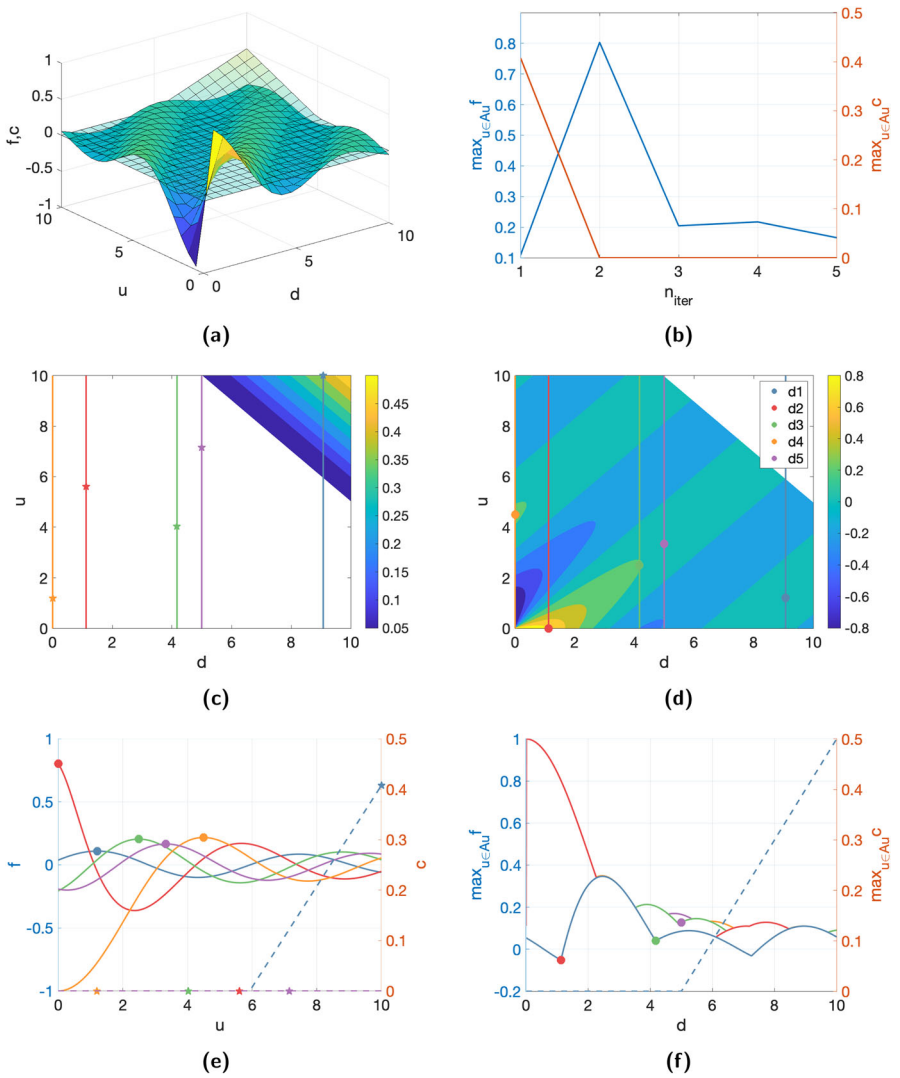

Fig. 3 Algorithm 2 applied to test case MWP10&GFC1. Sub-figure **a** shows the test case's characteristics by plotting both the objective function f and the constraint function c. Sub-figure **b** shows the convergence of both the worst case conditions max f and max c for the design solutions found at each iteration. Sub-figures **c** and **d** plot the functions c and f respectively in the coupled space $D \times U$, all the design solutions at the different iterations (vertical lines) and the corresponding worst case for f and c (dots and stars). The white areas correspond to feasible solutions $c \leq 0$ in (**c**) and unfeasible solutions f s.t. $c > 0$ in (**d**). Sub-figure **e** represents, for each explored design configuration, the corresponding f (continuous lines) and c (dotted lines). Sub-figure **f**, represents the space of the maxima of f and c over the design space

of each single evaluation of the outer loop but in some cases improves convergence to the point that the overall cost of the algorithm is reduced. For this reason we inserted this option in the algorithm although it is not tested in this paper. It was, however,

tested, for the case of unconstrained min-max problems, in a previous work by the authors Vasile (2014). At the end of the minimisation process, the archive A_d of the design configurations is updated with the new design solution $\bar{\mathbf{d}}$.

If the condition in line 18 of Algorithm 2 holds, the constraint relaxation strategy described in Algorithm 5 is activated. An alternative option, the trade-off strategy in Algorithm 6, can instead be activated if condition in line 21 is true. These alternatives can be visualised in Fig. 1. If the former condition holds, problem A is stopped, the relaxation step in the first block in B is performed finding ϵ and finally the constrained min-max problem is updated as in the second block in B. If instead the second condition hold, block A is stopped and then blocks C and D are solved.

The minimisation and restoration steps are alternated until condition in line 9 is satisfied. In particular, the iteration holds until the number of calls to the function has not reached the maximum allowed ($n_{\text{feval}} < n_{\text{feval,max}}$), the number of iterations is below the upper bound ($n_{\text{loop}} < n_{\text{loop,max}}$) and the solutions saved in archives have not converged ($\max_A(\sigma) > \sigma_{\text{stop}}$) where $\max_A(\sigma)$ is the maximum standard deviation between all the archives in the last 3 iterations. The violation of at least one of these conditions corresponds to the termination criterion.

Then, the cross-check between all the design vectors archived in A_d is performed (line 26). The cross-check procedure is explained in Sect. 3.6 and summarised in Algorithm 7. Finally, the solution is chosen following Algorithm 8 (line 28).

An example of the application of Algorithm 2 to the two functions MWP10 and GFc1 (see Sect. 4.1), without constraint relaxation and trade-off can be visualised in Fig. 3. In sub-figures c and d an initial design guess $\hat{\mathbf{d}}$ and each new design $\bar{\mathbf{d}}$ proposed by the minimisation step are represented as vertical lines. The corresponding worst case scenario for f and $\mathbf{c}$ are plotted, with the same colour, as dots and stars respectively. Sub-figure e shows sections of f and $\mathbf{c}$ over U for different design configurations. Sub-figure f shows how the space of the maxima of f and $\mathbf{c}$ as it appears to the minimisation process. In particular, sub-figure (c) shows that the algorithm is able to find at the first iteration, for this test case, the subspace $\hat{D} \in D$ in which the design solutions are feasible for any uncertain scenario. The first design guess $\hat{\mathbf{d}}$ (blue line) finds the worst scenario for $\mathbf{u} = 10$. In b indeed, the maximum constraint violation is brought to zero at the second iteration and all the other iterations are then used to minimise the worst case of f working within the design domain $\hat{D}$.

3.3 Memetic strategy

When problems in Eqs. (2) to (4) within the min-max Algorithm 2 are global optimisation problems we propose the use of the memetic optimisation algorithm Multi–Population Adaptive Inflationary Differential Evolution Algorithm (MP-AIDEA) (Di Carlo et al. 2019). It combines the Darwinian evolution of a number of populations of candidates through a Differential Evolution (DE) strategy with the Lamarckian evolution of the best agent in each converged population through a local search algorithm. In particular, the local refinement is performed if the converged solution is not in the basin of attraction of previous local minima. A number of local restart are then performed before globally restarting.

Algorithm 2 Constrained min-max

Initialisation: Algorithm 1
if $\bar{A}_{uf} = \emptyset \wedge \bar{A}_{uc} = \emptyset$ **then**
  Initialisation loop:
    Run $\mathbf{u}_{af} = \arg\max_{\mathbf{u}\in U} f(\hat{\mathbf{d}}, \mathbf{u})$ s.t. $\max_{i\in I_c} c_i(\hat{\mathbf{d}}, \mathbf{u}) \leq \epsilon$
    Run $\mathbf{u}_{ac} = \arg\max_{\mathbf{u}\in U} \max_i c_i(\hat{\mathbf{d}}, \mathbf{u})$
    if multiple outputs, choose best $\mathbf{u}_{af}$ as in Algorithm 3
    Update archives as in Algorithm 4
end if
while $n_{\text{feval}} < n_{\text{feval,max}} \wedge n_{\text{loop}} < n_{\text{loop,max}} \wedge \max_A(\sigma) > \sigma_{\text{stop}}$ **do**
  Minimisation loop:
    $\bar{\mathbf{d}} = \arg\min_{\mathbf{d}\in D}\{\max_{\mathbf{u}_a\in\bar{A}_{uf}} f(\mathbf{d}, \mathbf{u}_a)\}$ s.t. $\max_{\mathbf{u}_a\in\bar{A}_{uc}} \max_{i\in I_c} c_i(\mathbf{d}, \mathbf{u}_a) \leq \epsilon$
    Update global archive $A_d = A_d \cup \{\bar{\mathbf{d}}\}$
  Restoration loop:
    Run $\mathbf{u}_{af} = \arg\max_{\mathbf{u}\in\mathbf{U}} f(\bar{\mathbf{d}}, \mathbf{u})$ s.t. $\max_{i\in I_c} c_i(\bar{\mathbf{d}}, \mathbf{u}) \leq \epsilon$
    Run $\mathbf{u}_{ac} = \arg\max_{\mathbf{u}\in U} \max_i c_i(\bar{\mathbf{d}}, \mathbf{u})$
    if multiple outputs, choose best $\mathbf{u}_{ac}$: Algorithm 3
    Update archives: Algorithm 4
  if relaxation flag $\wedge$ not convergence $\wedge$ satisfy limits on n_{feval} and n_{loop} **then**
    Stop Algorithm 2 and apply the constraint relaxation strategy: Algorithm 5
  end if
  if trade-off flag $\wedge$ convergence on $\mathbf{d}$, $f_{\max}$, $c_{\max}$ $\wedge$ satisfy limits on n_{feval} and n_{loop} **then**
    Stop Algorithm 2 and apply the trade-off strategy: Algorithm 6
  end if
end while
for all $\mathbf{d} \in A_d$ **do**
  Cross-check: Algorithm 7
end for
Select which solution $[\mathbf{d}_{\text{opt}}, \mathbf{u}_{\text{opt}}]$ to return: Algorithm 8

Algorithm 3 Solution Selection - Inner loop

for all output $\mathbf{u}_{af}$ of the restoration level **do**
  if $\nexists \mathbf{u}_{af} \rightarrow \max_{i\in I_c} c_i(\bar{\mathbf{d}}, \mathbf{u}_{af}) \leq \epsilon$ **then**
    select $\mathbf{u}$ with minimum violation
  else
    select between the feasible $\mathbf{u}$ the one with the highest objective function f.
  end if
end for

MP-AIDEA depends on the following parameters: the maximum number of function evaluation $n_{\text{feval,max}}$, the number of populations n_{pop}, the number of agents in each population N_{pop}, the convergence threshold ρ of DE, and the radius of the global restart bubble δ_{global}.

An example of a run of MP-AIDEA with $n_{\text{pop}} = 2$ for problem *MWP-1&GFc-1* is in Fig. 4 which shows, for the evolving populations, the alternations of DE steps together with the local refinements. For this test case the MATLAB function solver *fmincon* has been used. Given the constraint on the maximum number of function evaluations, the algorithm is able to perform only one local restart in each population. Colours red, blue and green refer to the first population while colours orange, yellow and brown refer to the second population. As shown in the figure, first the two populations are initialised randomly and evolved independently until convergence is

Algorithm 4 Archive Updating

if in the main loop **then**
  if $(f_{\text{out}} > f_{\text{in}} \wedge c_{\text{out}} < 0) \vee (0 < c_{\text{out}} < c_{\text{in}})$ **then**
    update $\mathbf{u}_{a,f} = \mathbf{u}$ from outer loop
  end if
  if $c_{\text{out}} > c_{\text{in}}$ **then**
    $\mathbf{u}_{a,c} = \mathbf{u}$ from outer loop
  end if
end if
$A_{uf} = A_{uf} \cup \{\mathbf{u}_{af}\}$
$A_f = A_f \cup \{f(\bar{\mathbf{d}}, \mathbf{u}_{af})\}$
$A_{cf} = A_{c,f} \cup \{\max_i c_i(\bar{\mathbf{d}}, \mathbf{u}_{af})\}$
$A_{uc} = A_{uc} \cup \{\mathbf{u}_{ac}\}$
$A_c = A_c \cup \{\max_i c_i(\bar{\mathbf{d}}, \mathbf{u}_{ac})\}$
$\bar{A}_{uf} = A_{uf} \setminus$ repeated solutions
$\bar{A}_{uc} = A_{uc} \setminus$ repeated solutions

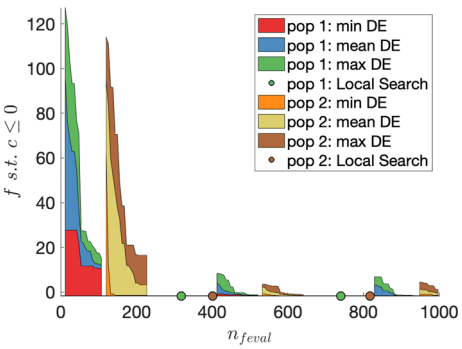

Fig. 4 Convergence of MP-AIDEA with two populations in the outer loop of Algorithm 2 for test problem *MWP1&GFc1*. Coloured areas represent the convergence of the differential evolution steps (different set of colours for different populations) while dots represents optimal solutions of the local search

achieved (being the parameter ρ one of the termination criteria). In particular the red (orange) area represent the evolution of the best agent in the population while the blue (yellow) represent the mean value and the green (brown) the worst agent. From the two converged solutions a local search is performed with $fmincon$ until the green and brown points are obtained. From these two local minima then the two populations are locally reinitialised. The overall process is then repeated till convergence.

3.4 Constraint relaxation strategy

The *min-max* problem proposed in this paper imposes quite stringent conditions on the satisfaction of the constraints as constraints need to be satisfied for all possible values of $\mathbf{u} \in U$. It is, therefore, possible that no solution $\mathbf{d}_{\text{opt}}$ is feasible in all U. Since we are interested in the worst case solution for both constraints and objective function, when no feasible $\mathbf{d}$ is possible we introduce an automatic relaxation of the constraints.

In order to find a set $X \subset D$ that is feasible for all $\mathbf{u} \in U$, we first solve the following minmax problem:

$$\min_{\mathbf{d}\in D} \max_{\mathbf{u}\in U} \max_{i\in I_c} c_i(\mathbf{d}, \mathbf{u}) \tag{5}$$

Problem (5) minimises the maximum violation of the constraints and returns a solution vectors $\mathbf{d}_{\min,c}$ and $\mathbf{u}_{\min,c}$. Vectors $\mathbf{d}_{\min,c}$ satisfies the constraint:

$$\mathbf{c}_\epsilon \leq 0 \tag{6}$$

for all $\mathbf{u} \in U$, where:

$$\mathbf{c}_\epsilon = \mathbf{c} - \epsilon \tag{7}$$

and

$$\epsilon = \max_{i\in I_c} c_i(\mathbf{d}_{\min,c}, \mathbf{u}_{\min,c}) \tag{8}$$

The relaxation strategy is explained in Algorithm 5. If condition on line 18 of Algorithm 2 is satisfied then Algorithm 5 is triggered. Once in Algorithm 5, until condition in line 2 holds, problem (5) is solved (see lines 3 – 8 of Algorithm 5), with the iteration between the following minimisation

$$\min_{\mathbf{d}\in D} \max_{\mathbf{u}_{ac}\in \bar{A}_{uc}} \max_{i\in I_c} c_i(\mathbf{d}, \mathbf{u}_{ac}), \tag{9}$$

and restoration step

$$\max_{\mathbf{u}\in U} \max_{i\in I_c} c_i(\bar{\mathbf{d}}, \mathbf{u}) \tag{10}$$

This is an unconstrained min-max formulation where the optimised function is the vector of constraints $\mathbf{c}$. The solution at convergence is the minimum over D of the worst constraint violations in U. In line 10 that value is associated to the relaxation parameter ϵ. In this way the algorithm is re-conducted to Eq. (5) by means of the relaxation of the constraint violation and Eq. (1) is translated in:

$$\begin{aligned} &\min_{\mathbf{d}\in D} \max_{\mathbf{u}\in U} f(\mathbf{d}, \mathbf{u}) \\ &s.t. \\ &c_i(\mathbf{d}, \mathbf{u}) \leq \epsilon \quad \forall \mathbf{u} \in U, \forall i \in I_c = [1, ..., n]^T \end{aligned} \tag{11}$$

3.5 Scalarisation strategy

This subsection presents a different point of view that is based on the parameter-based scalarisation approach (Kasimbeyli et al. 2019) for the solution of Eq. (1). It is here supposed that the decision maker, who is the final user of this methodology, is able to assign preference weights to the different Quantity of Interests (QoIs) involved in the problem under analysis. The scalarisation approach then has been preferred to the direct multi-objective optimisation because, for a given set of weights, it reduces the

Algorithm 5 Constraint Relaxation

1: Inherit vectors from Algorithm 2
2: **while** not convergence of relaxation **do**
3: *Minimisation loop*:
4: $\bar{\mathbf{d}} = \arg\min_{d \in D}\{\max_{\mathbf{u}_{ac} \in \bar{A}_{uc}} \mathbf{c}(\mathbf{d}, \mathbf{u}_{ac})\}$ with the cross-check as in Algorithm 7.
5: *Restoration loop*:
6: Run $\mathbf{u}_{ac} = \arg\max_{\mathbf{u} \in U} \max_i c_i(\bar{\mathbf{d}}, \mathbf{u})$
7: if multiple outputs, choose best $\mathbf{u}_{ac}$: Algorithm 3
8: Update archives A_{uc} and A_c: Algorithm 4
9: **end while**
10: set $\epsilon = \max_{\mathbf{u} \in U} \max_i c_i(\bar{\mathbf{d}}, \mathbf{u})$
11: Restart Algorithm 2

computational complexity. Problem (11) indeed can be seen as an Epsilon-Constraint Scalarisation (ECS) (Filippi and Vasile 2020; Haimes et al. 1971) formulation of the bi-objective min-max problem:

$$\min_{\mathbf{d} \in D} \left[\max_{\mathbf{u} \in U} f(\mathbf{d}, \mathbf{u}), \max_{\mathbf{u} \in U} \max_{i \in I_c} c_i \right] \tag{12}$$

Thus one could be interested in a trade-off between f and ϵ and accept larger relaxations of the constraints in favour of a better objective. Note that in the context of uncertainty quantification this implies accepting a higher probability of violating the constraints in favour of a better cost function.

We now introduce the assumption that a preference vector $\boldsymbol{\omega} = [\omega_f, \omega_c]^T$ can be defined a priori and explain the scalarisation approach summarised in Algorithm 6.

In the first part (line 2 – 10) the reference points c_{ideal}, f_{nadir}, f_{ideal} and c_{nadir} are calculated. c_{ideal} is the minimum (best) over D of the worst case constraint violations in U:

$$c_{\text{ideal}} = \min_{\mathbf{d} \in D} \max_{\mathbf{u} \in U} \max_{i \in I_c} c_i(\mathbf{d}, \mathbf{u}) \tag{13}$$

and it is equal to the relaxed constraint ϵ. For the corresponding design vector $\mathbf{d}_{c\text{-ideal}}$ the worst scenario for the objective function is f_{nadir} (lines 3 to 8):

$$f_{\text{nadir}} = \max_{\mathbf{u} \in U} f(\mathbf{d}_{c\text{-ideal}}, \mathbf{u}) \tag{14}$$

In line 9 the unconstrained min-max problem

$$f_{\text{ideal}} = \min_{\mathbf{d} \in D} \max_{\mathbf{u} \in U} f(\mathbf{d}, \mathbf{u}) \tag{15}$$

is solved to define the best design configuration $\mathbf{d}_{f\text{-}ideal}$ that minimises the worst scenarios of the objective function f regardless the constraint violation. $\mathbf{d}_{f\text{-}ideal}$ is then used in line 10 to calculate the corresponding worst condition for the constraint violation:

$$c_{\text{nadir}} = \max_{\mathbf{u} \in U} \max_{i \in I} c_i(\mathbf{d}_{f\text{-ideal}}, \mathbf{u}) \tag{16}$$

An example of the reference points for a generic Pareto front applied to the min-max problem is in Fig. 5 where $\mathbf{z}^{\text{ideal}} = [f_{\text{ideal}}, c_{\text{ideal}}]$ and $\mathbf{z}^{\text{nadir}} = [f_{\text{nadir}}, c_{\text{nadir}}]$. In the second part (lines 13 – 27) it is instead described the scalarisation procedure. Nadir and ideal points are here used to normalise f and c:

$$\bar{f} = \frac{f - f_{\text{ideal}}}{f_{\text{nadir}} - f_{\text{ideal}}} \tag{17}$$

$$\bar{c} = \frac{\max_{i \in I_c} c_i - c_{\text{ideal}}}{c_{\text{nadir}} - c_{\text{ideal}}}. \tag{18}$$

Three different methods were considered in this study: ECS, Weighted-Sum Scalarisation (WSS) (Gass and Saaty 1955) and Chebychev/Pascoletti-Serafini Scalarisation (CPSS) where the last one is a smooth combination of Weighted Chebyshev Scalarisation (WCS) (Bowman 1976) and Pascoletti-Serafini Scalarissation (PSS) (Bo et al. 2007). A comparison of the different approaches can be found in Filippi and Vasile (2020). The difference in Algorithm 6 is only within the minimisation loop (lines 13 – 22).

The WSS solves the minimisation over the design space D of the weighted sum of the worst case scenarios for f and c in the respective archives (line 18):

$$\min_{\mathbf{d} \in D} \left[\omega_f \max_{\mathbf{u} \in \bar{A}_{uf}} \bar{f}(\mathbf{d}, \mathbf{u}) + \omega_c \max_{\mathbf{u} \in \bar{A}_{uc}} \bar{c}(\mathbf{d}, \mathbf{u}) \right] \tag{19}$$

The CPSS solves two different problems. During the DE step of Inflationary Differential Evolution, the algorithm addresses the following problem:

$$\min_{\mathbf{d}} \max_{i} \omega_i \bar{z}_i \tag{20}$$

where we have introduced the weight vector $\boldsymbol{\omega} = [\omega_f, \omega_c]^T$ and the objective vector $\bar{\mathbf{z}} = [\max_{\mathbf{u}_{af} \in A_{uf}} \bar{f}(\mathbf{d}, \mathbf{u}_{af}) - z_f, \max_{\mathbf{u}_{ac} \in A_{uc}} \bar{c}(\mathbf{d}, \mathbf{u}_{ac}) - z_c]^T$. During the local search step of Inflationary Differential Evolution, instead, the CPSS addresses the following constrained minimisation problem:

$$\begin{cases} \min_{\mathbf{d} \in D} t \\ t < 0 \\ \omega_f \left(\max_{\mathbf{u}_{af} \in A_{uf}} \bar{f}(\mathbf{d}, \mathbf{u}_{uf}) - z_f \right) < t \\ \omega_c \left(\max_{\mathbf{u}_{ac} \in A_{uc}} \bar{c}(\mathbf{d}, \mathbf{u}_{ac}) - z_c \right) < t \end{cases} \tag{21}$$

where z_f and z_c are the values of $\bar{f}$ and $\bar{c}$ respectively for the best solution obtained from the DE step in Eq. (20) and from which the local search is initialised. For all three methods, the restoration problem solves the two problems:

$$\max_{\mathbf{u} \in U} f(\bar{\mathbf{d}}, \mathbf{u}) \tag{22}$$

$$\max_{\mathbf{u} \in U} \max_{i \in I_c} c_i(\bar{\mathbf{d}}, \mathbf{u}) \tag{23}$$

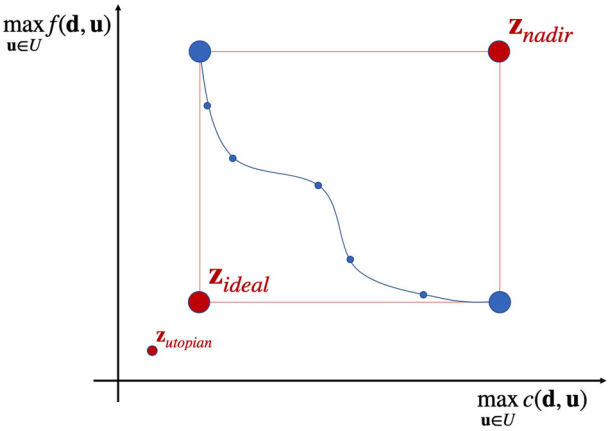

Fig. 5 A generic Pareto front for the min-max problem. In this case the functions max f and max c are considered as conflicting objectives. The ideal $\mathbf{z}_{\text{ideal}}$, nadir $\mathbf{z}_{\text{nadir}}$ and utopian $\mathbf{z}_{\text{utopian}}$ points are represented. They are theoretic points that collapse the extreme behaviour of the different solutions in the Pareto front. $\mathbf{z}_{ideal}$ is the combination of the best solutions for the different objectives. $\mathbf{z}_{\text{nadir}}$ represents instead the worst possible combination of points. $\mathbf{z}_{\text{utopian}}$ is finally defined by means of an ϵ from $\mathbf{z}_{\text{ideal}}$

For the given preference vector, the optimal worst case condition for f and the optimal relaxation of the constraint ϵ are given, at convergence, by Eqs. (22) and (23).

The three possible scalarisation procedures ECS, WSS and CPSS can be used to reconstruct the Pareto front for the minimisation trade-off between the worst case scenarios max f and max $\mathbf{c}$. Fig. 6 shows a comparison of the three methods, where the combination of the objective function $GFf1$ in Table 1 and the constraint function $GFc1$ in Table 2 has been chosen as representative test case. We used MP-AIDEA with the following settings: maximum number of function evaluation $n_{\text{feval,max}} = 3000$, number of populations $n_{\text{pop}} = 2$, number of agents in the population $N_{\text{pop}} = 5$, dimension of the bubble for the global restart $\delta_{\text{global}} = 0.1$ and DE threshold for each population $\rho = 0.1$. The whole constrained min-max algorithm has then been run until convergence. For ECS the algorithm has been run for 60 different thresholds ranging between 1 and 6.2 (calculated minimum and maximum constraint violation.) For WSS and CPSS, instead, the trigonometric weights

$$\begin{aligned} w_f &= \frac{\cos\theta}{\cos\theta+\sin\theta} \\ w_c &= \frac{\sin\theta}{\cos\theta+\sin\theta} \end{aligned} \tag{24}$$

have been used letting θ varying from 0 to $\frac{\pi}{2}$ and using a discretisation of 60 interval as well. Fig. 6 shows that WSS is not capable of finding optimal Pareto points in the non convex part of the Pareto front (Das and Dennis 1997) and also, using equally spaced weights, it finds non-equally spaced points in the front. The ECS strategy gives better results. The best performance is, however, obtained with the CPSS. Indeed it allows the user to express a preference through the selection of the weights and, as shown in Fig. 6, with the same number of simulations (60 different descent directions for the

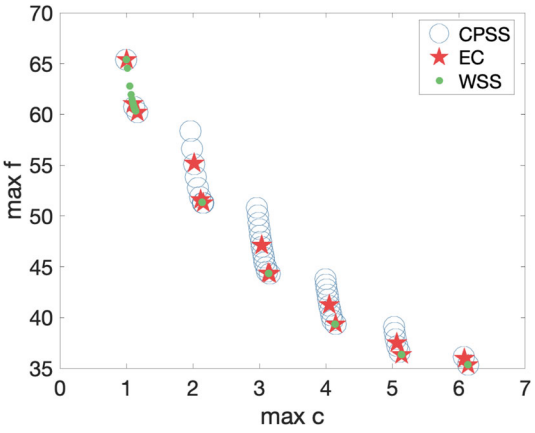

Fig. 6 Pareto front corresponding to the trade-off between the two conflicting goals max f and max c. The results refer to test case $GFf1\&GFc1$. In particular, the ϵ-constraint (EC) approach has been applied with different thresholds while the Chebychev/Pascoleti Serafini (CPSS) and the Weighted-Sum (WSS) scalarisations have been applied with different preference vectors

CPSS and epsilon values for the ECS) it is able to find more Pareto optimal solutions than ECS. For these reasons CPSS was chosen and implemented in Algorithm 2

3.6 Cross-check

All the optimisation problems in the minmax algorithm require the identification of a global maximum or a global minimum. Since it is proposed to use a memetic algorithm it is possible that some of the maxima or minima in the archive are only locally optimal. Note that the use of a deterministic global optimiser would remove this problem but would introduce a tractability problem due to the potential NP-hard nature of some optimisation problems.

In order to mitigate the occurrence of local minima/maxima in the archives we introduce a cross-check of the solutions following the procedure explained in Algorithm 7. It is performed for each design vector $\bar{\mathbf{d}}$ that can be proposed by the optimiser during the minimisation step and at the end of the whole algorithm (respectively in line 9 and 19 of Algorithm 2). Referring to Algorithm 7, lines 1 – 7 regard the objective function f while lines from 8 – 14 regard the constraint function $\mathbf{c}$. In both cases, for a given $\bar{\mathbf{d}}$ objectives and constraints are evaluated for all the $\mathbf{u}$ vectors in the archives $\bar{A}_{uf}$ and $\bar{A}_{uc}$. We also introduced an option (through *local flag*) to run a local search from each new pair $[\bar{\mathbf{d}}, \mathbf{u}]$. This option slows down Algorithm 2 but improves the quality of the solution if the functions present nested minima/maxima. Finally, line 15, retains the worst values of f and $\mathbf{c}$ for the archives $\bar{A}_{uf}$ and $\bar{A}_{uc}$ for each $\bar{\mathbf{d}}$.

3.7 Selection of output solution

After the termination criterion in Algorithm 2 is applied and the cross-check over the archives is performed (line 21), the solution for the min-max problem is selected following Algorithm 8. In particular, if a feasible subset $\hat{A}_d$ of the archive A_d of the design vectors exists (line 1) the selected solution vector is the one, within $\hat{A}_d$,

Algorithm 6 Scalarisation Strategy

1: Inherit vectors from Algorithm 2
2: *Normalisation points*:
3: **if** Algorithm 2 not converged **then**
4: run relaxation strategy in Algorithm 5
$c_{\text{ideal}} = \epsilon = \min_{\mathbf{d}\in D} \max_{\mathbf{u}\in U} \max_{i\in I_c} c_i(\mathbf{d}, \mathbf{u})$
5: $f_{\text{nadir}} = \max_{\mathbf{u}\in U} f(\mathbf{d}_{c\text{-ideal}}, \mathbf{u})$
6: **else**
7: c_{ideal} and f_{nadir} from Algorithm 2.
8: **end if**
9: run Algorithm 2 for the unconstrained problem on f:
$f_{\text{ideal}} = \min_{\mathbf{d}\in D} \max_{\mathbf{u}\in U} f(\mathbf{d}, \mathbf{u})$
10: $c_{\text{nadir}} = \max_{\mathbf{u}\in U} \max_{i\in I} c_i(\mathbf{d}_{f\text{-ideal}}, \mathbf{u})$
11: *Scalarisation step*:
12: **while** satisfy limits on $n_{fval}, n_{loop} \wedge$ not convergence **do**
13: *Minimisation loop*:
14: **if** Weighted-sum scalarisation **then**
15: $\min_{\mathbf{d}\in D} \left[\omega_f \max_{\mathbf{u}\in \bar{A}_{uf}} \bar{f}(\mathbf{d}, \mathbf{u}) + \omega_c \max_{\mathbf{u}\in \bar{A}_{uc}} \bar{c}(\mathbf{d}, \mathbf{u})\right]$
with the cross-check as in Algorithm 7,
$\bar{f}$ and $\bar{c}$ defined in Eqs. (17) and (18)
16: **else if** Chebyshev Pascoletti-Serafini scalarisation **then**
17: during the Differential Evolution step:
18: $\min_{\mathbf{d}\in D} \left\| \omega_f(\max_{\mathbf{u}\in A_{uf}} \bar{f}(\mathbf{d}, \mathbf{u}), \omega_c \max_{\mathbf{u}\in A_{uc}} \bar{c}(\mathbf{d}, \mathbf{u}) \right\|_\infty$
19: during the Local Search step:
20: $\min_{\mathbf{d}\in D, t<0} t$
s.t.
$\omega_f\left(\max_{\mathbf{u}_{af}\in A_{uf}} \bar{f}(\mathbf{d}, \mathbf{u}_{af}) - z_f\right) < t$
$\omega_c\left(\max_{\mathbf{u}_{ac}\in A_{uc}} \bar{c}(\mathbf{d}, \mathbf{u}_{uc}) - z_c\right) < t$
$\bar{f}$ and $\bar{c}$ defined in Eqs. (17) and (18)
21: **end if**
22: Update global archive $A_d = A_d \cup \{\bar{\mathbf{d}}\}$
23: *Restoration loop*:
24: Run $\mathbf{u}_{af} = \arg\max_{\mathbf{u}\in \mathbf{U}} f(\bar{\mathbf{d}}, \mathbf{u})$
25: Run $\mathbf{u}_{ac} = \arg\max_{\mathbf{u}\in U} \max_{i\in I} c_i(\bar{\mathbf{d}}, \mathbf{u})$
26: if multiple outputs, choose best $\mathbf{u}_{ac}$: Algorithm 3
27: Update archives: Algorithm 4
28: **end while**
29: **for** all $\mathbf{d} \in A_d$ from phase 2 **do**
30: Cross-check: Algorithm 7
31: **end for**
32: Select which solution $[\mathbf{d}_{\text{opt}}, \mathbf{u}_{\text{opt}}]$ to return: Algorithm 8

minimising the worst value of f (line 2). If, on the other hand, $\hat{A}_d$ is an empty set, the design vector that minimises the constraint violation is selected (line 4).

3.8 Computational complexity

The computational cost of Algorithm 2 is measured in terms of number of function calls. With reference to the minimisation step, the counter $n_{\text{feval}}^{\text{outer}}$ takes into account, for both the constrained and the unconstrained min-max problems, the calls to $\max_{\mathbf{u}\in \bar{A}_{uf}} f(\mathbf{d}, \mathbf{u}_{af})$ in Eq. (2). The same criterion, then, holds for the constraint

Algorithm 7 Cross-Check

1: **for** all elements $\mathbf{u}_{af} \in \bar{A}_{uf}$ **do**
2: **if** local flag **then**
3: Compute local maximum $f(\bar{\mathbf{d}}, \mathbf{u}_a^*)$ s.t. $\max_{i \in I_c} c_i(\bar{\mathbf{d}}, \mathbf{u}_a^*) \leq \epsilon$ from $\mathbf{u}_{af}$
4: **else**
5: Compute $f(\bar{\mathbf{d}}, \mathbf{u}_a)$ s.t. $\max_{i \in I_c} c_i(\mathbf{d}, \mathbf{u}_{af}) \leq \epsilon$
6: **end if**
7: **end for**
8: **for** all elements $\mathbf{u}_{ac} \in \bar{A}_{uc}$ **do**
9: **if** local flag **then**
10: Compute local maximum $\max_{i \in I_c} c_i(\bar{\mathbf{d}}, \mathbf{u}_a^*)$ from $\mathbf{u}_{ac}$
11: **else**
12: Compute $\max_{i \in I_c} c_i(\bar{\mathbf{d}}, \mathbf{u}_{ac})$
13: **end if**
14: **end for**
15: For each $\bar{\mathbf{d}}$ Save worst vectors $\mathbf{u}_{af}$ and $\mathbf{u}_{ac}$ in the archives $\bar{A}_{uf}$ and $\bar{A}_{uc}$.

Algorithm 8 Select Solution - Output

1: **if** $\hat{A}_d = \{\mathbf{d} | \max_{\mathbf{u}_{ac} \in A_{uc}} \max_{i \in I_c} c_i(\mathbf{d}, \mathbf{u}_{ac}) \leq \epsilon\} \neq \emptyset$ **then**
2: take $\mathbf{d} \in \hat{A}_d$ that minimise $\max_{\mathbf{u}_{af} \in A_{uf}} f(\mathbf{d}, \mathbf{u}_{af})$
3: **else**
4: take $\mathbf{d} \in A_d$ that minimise $\max_{\mathbf{u}_{ac} \in A_{uc}} \max_{i \in I_c} c_i(\mathbf{d}, \mathbf{u}_{ac})$
5: **end if**

relaxation step in Eq. (9) and for the two trade-off steps in Eqs. (15) and (19). It has to be noted that, as the algorithm proceeds in the search of the global optimum solution, the archives $\bar{A}_{uf}$ and $\bar{A}_{uc}$ of the uncertainty vectors increase progressively in dimension. Each minimisation step explores a maximum number of possible design configurations which is limited by the input parameter $n_{\text{feval,max}}^{\text{outer}}$. However, due to the growth of the archives of the solutions coming from the restoration step, each evaluation of the minimisation loop becomes increasingly more expensive.

With reference to the maximisation step, instead, the cost of the two separate problems in Eqs. (3) and (4) has to be considered. For the former $n_{\text{feval}}^{\text{inner},f}$ counts the number of calls to the objective function $f(\bar{\mathbf{d}}, \mathbf{u}_{af})$ and it is limited by the input $n_{\text{feval,max}}^{\text{inner},f}$. This holds true also for the two steps in the trade-off strategy in Eqs. (16) and (22) and for the relaxation step in Eq. (10) where the function c is considered instead of f. For the latter, $n_{\text{feval}}^{\text{inner},c}$ counts the number of calls to $\max_i c_i(\hat{\mathbf{d}}, \mathbf{u})$ in Eq. (4) where the input $n_{\text{feval,max}}^{\text{inner},c}$ is the upper limit.

Finally, the parameters $n_{\text{feval,max}}$ and $n_{\text{loop,max}}$ give an upper limit on the whole cost of Algorithm 2. Note that $n_{\text{feval,max}}$ and $n_{\text{loop,max}}$ represent an upper limit because, as it is shown in line 9 of Algorithm 2, we use an additional termination criterion that looks at the convergence of the solutions in the archives.

The computational complexity of the different parts of the overall algorithm is as follows:

1. *Local Search*: the local search uses the Matlab *fmincon* function. All the alternative algorithms (*interior-point*, *trust-region-reflective*, *sqp*, *sqp-legacy*, *active-set*) can be selected. We use here *interior-point* that works well with both large sparse and

small dense problems. The complexity is $\mathcal{O}(n_D^3)$ or $\mathcal{O}(n_U^3)$ depending on which step between minimisation and restoration is considered, where n_D is the design and n_U the uncertain vector's dimension.

2. *Adaptation of CR and F*: The DE parameters CR and F in MP-AIDEA are auto-adapted for each element of each population. For n_{pop} evolving populations with N_{pop} agents, the complexity is $\mathcal{O}(n_{\text{pop}} N_{\text{pop}} n_D^2)$ and $\mathcal{O}(n_{\text{pop}} N_{\text{pop}} n_U^2)$ for inner and outer problem respectively (Di Carlo et al. 2019).
3. *Restart mechanisms*. The populations evolve with a differential evolution DE approach which is restarted, locally and globally, a number of times. The local restart has a cost proportional to $n_{\text{pop}} N_{\text{pop}}$. The cost for the global restart, instead, has a component related to the clustering procedure $\mathcal{O}(N_{\text{pop}} n_D n_{\text{iter}})$ or $\mathcal{O}(n_{LM} n_U^2 n_{\text{iter}})$ with n_{iter} the required number of iteration for the clustering and n_{LM} the number of local minima, and a component related to the verification that the new population is far from the clusters $\mathcal{O}(N_{\text{pop}} n_{LM})$ (Di Carlo et al. 2019).
4. *Outer-Loop*. During the minimisation step there is a cost related to the testing of each design vector suggested by the optimiser in combination with all the uncertain vectors saved in the archives $\bar{A}_{uf}$ and $\bar{A}_{uc}$ and there is a cost due to the selection of their maximum. In both cases the complexity is $\mathcal{O}(||\bar{A}_{uf}|| n_{\text{fval,max}}^{\text{outer}}) \leq \mathcal{O}(n_{\text{loop}} n_{\text{fval,max}}^{\text{outer}})$ for the objective function f and $\mathcal{O}(||\bar{A}_{uc}|| n_{\text{fval,max}}^{\text{outer}}) \leq \mathcal{O}(n_{\text{loop}} n_{\text{fval,max}}^{\text{outer}})$ for the constraint function c.
5. *Cross-Check*. As in the outer loop there is here a cost for the cross-check and a cost for the selection of the maxima. In both cases it is: $\mathcal{O}(||A_d|| ||\bar{A}_{uf}||) \leq \mathcal{O}(n_{\text{loop}}^2)$ and $\mathcal{O}(||A_d|| ||\bar{A}_{uc}||) \leq \mathcal{O}(n_{\text{loop}}^2)$ for f and c respectively because each design vector $\mathbf{d} \in A_d$ is considered.
6. *Select Solution*. After the final cross-check, the archives are updated and the set of design vectors $\hat{A}_d$ feasible in all the uncertain domain can be defined. The min-max solution is selected following Algorithm 8 sorting the feasible solutions f ($\mathcal{O}(||\hat{A}_d||)$) or minimising the constraint violation c ($\mathcal{O}(||A_d||)$).

4 Testing procedure

Algorithm 2 has been tested on the benchmark described and explained in this section. Each test case is a combination of an objective function f and a constraint function c. Depending on the mathematical features of each problem, a local optimiser or a global optimiser have been used for the three problems in Eqs. (2) to (4). The criteria used to choose the right optimiser is explained in Sect. 4.3.

Given the stochastic nature of MP-AIDEA, each optimisation for each problem has been repeated 100 times. Results are then reported in Section 5. For the evaluation of the algorithm's performance, the *Success Rate SR* is used instead of best value, mean, and variance. The SR was suggested in Vasile et al. (2011) for a generic problem $\min_f$ and a generic algorithm. It is here generalised to consider also the handling of constraints. The definition of SR is in Sect. 4.2.

Table 1 test cases for the objective function f

ID	objective functions
MWP-1	$5(d_1^2+d_2^2)-(u_1^2+u_2^2)+d_1(-u_1+u_2+5)+d_2(u_1-u_2+3)$
MWP-2	$4(d_1-2)^2-2u_1^2+d_1^2u_1-u_2^2+2d_2^2u_2$
MWP-3	$d_1^4u_2+2d_1^3u_1-d_2^2u_2(u_2-3)-2d_2(d_1-3)^2$
MWP-4	$-\sum_{i=1}^{3}(u_1-1)^2+\sum_{i=1}^{2}(d_1-1)^2+u_3(d_2-1)+u_1(d_1-1)+u_2d_1d_2$
MWP-5	$-(d_1-1)u_1-(d_2-2)u_2(d_3-1)u_3+2d_1^2+3d_2^2d_3^2$
MWP-6	$u_1(d_1^2-d_2+d_3-d_4+2)+u_2(-d_1+2d_2^2-d_3^2+2d_4+1)+d_3(2d_1-d_2+2d_3-d_4^2+5)+5d_1^2+4d_2^2+3d_3^2+2d_4^2-\sum_{i=1}^{3}u_i^2$
MWP-7	$2d_1d_5+3d_4d_2+d_5d_3+5d_4^2+5d_5^2-d_4(u_4-u_5-5)+d_5(u_4-u_5+3)+\sum_{i=1}^{3}(u_i(d_i^2-1))-\sum_{i=1}^{5}u_i^2$
MWP-8	$(d_1-5)^2-(u_1-5)^2$
MWP-9	$\min(3-0.2d_1+0.3u_1, 3+0.2d_1-0.1u_1)$
MWP-10	$\frac{\sin(d_1-u_1)}{\sqrt{d_1^2+u_1^2}}$
MWP-11	$\frac{\cos(\sqrt{d_1^2+u_1^2})}{\sqrt{d_1^2+u_1^2}+10}$
MWP-12	$100(d_2-d_1^2)^2+(1-d_1)^2-u_1(d_1+d_2^2)-u_2(d_1^2+d_2^2)$
MWP-13	$(d_1-2)^2+(d_2-1)^2+u_1(d_1^2-d_2)+u_2(d_1+d_2-2)$
GFf-1	$10(n_D+n_U)+\sum_{i=1}^{n_D}(d_i^2+u_i^2-10\big[\cos(2\pi d_i)+\cos(2\pi u_i)\big]-5$
GFf-2	$\sum[(\mathcal{R}_{d_{i+1},u_{i+1}}(d_i)-5)^2-((\mathcal{R}_{d_{i+1},u_{i+1}}(u_i)-5)^2]$

Table 2 test cases for the constraint functions c

name	constraint functions
GFc-1	$\sum d_i+u_i+K$
GFc-2	$\max\left[0,\sum d_i+u_i+K\right]$
GFc-3	$\begin{cases}0 & \text{if } \max_i(d_i-d_i^{opt})\le 0.1\\ 1 & \text{else}\end{cases}$
GFc-4	$\begin{cases}0 & \text{if } \max_i(d_i-d_i^{opt})\le 0.1\\ 30n+\sum[x_i^2-30\cos(2\pi x_i)]-30 & \text{else}\end{cases}$
GFc-5	$\begin{cases}0 & \text{if } \max_i(d_i-d_i^{opt})\le 0.1\\ 30n+\sum[x_{i,\mathcal{R}}^2-30\cos(2\pi x_{i,\mathcal{R}})]-30 & \text{else}\end{cases}$
GFc-6	$\left[\sum(x_i-x_i^{opt})\sum(d_i^2-2u_i)\sum(d_i-d_i^{opt})\right]^{(2/7)}$
GFc-7	$1-AC/BD$
GFc-8	$\sum[(d_{i,\mathcal{R}}-5)^2-(u_{i,\mathcal{R}}-5)^2]$

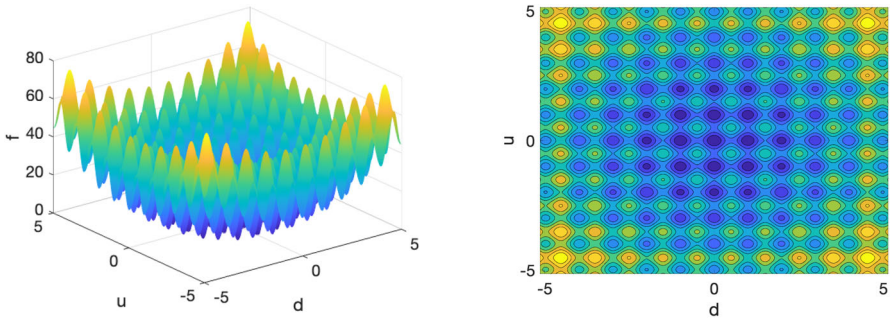

Fig. 7 GFf-1 in the case $n_D = n_U = 1$

4.1 Benchmark

The equations of f and c are listed in Tables 1 and 2 respectively. The constraint functions c are more extensively presented in the following Sects. 4.1.1 – 4.1.10 and visualised in Figs. 7, 8, 9, 10, 11, 12 and 13 for the case $n_D = n_U = 1$. Table 3 lists lower and upper bounds, dimensions and optimal solutions for the unconstrained problems in Table 1. The same solutions holds also for the constraint min-max problems for which the constraint does not change the global optimum. Table 4 presents instead the reference solutions for the constraint min-max problems for which the constraint function changes the position of the global optimum.

MWP-1,2,...,7 are convex-concave test functions taken from chapter 5 of Baxter et al. (2008). Objective functions *MWP-8,...,11* are first introduced in Power (1998) and then used also in Cramer et al. (2009), Lung and Dumitrescu (2011), Zhou and Zhang (2010) while *MWP-12,13* are instead selected from Cramer et al. (2009), Lung and Dumitrescu (2011), Zhou and Zhang (2010). They have been used all together in Marzat et al. (2013) as benchmark for the unconstrained min-max problem. Functions *GFf-1,2* and *GFc-1,2,...,8* have been specifically designed for the testing of Algorithm 2, given the lack of a benchmark in the literature for the constrained version of the min-max problem.

Both f and c are designed to include different structures that can be encounter in practice (Jamil and Yang 2013). In particular, they exhibits the following features:

1. *Modality*: number of local optima that try to trap the algorithm in the wrong peak.
2. *Basin* or plateau: a relatively steep decline that surrounds a large area. There is no information to drive the algorithm.
3. *Valley*: similar to the basin but it is narrow area.
4. *Non separability*: property related to the coupling between parameters. Non-separable functions are in general more difficult to optimise.
5. *Dimensionality*: property related to the number of parameters or dimension of the problem. The search space increases with the dimension, increasing then also its difficulty.

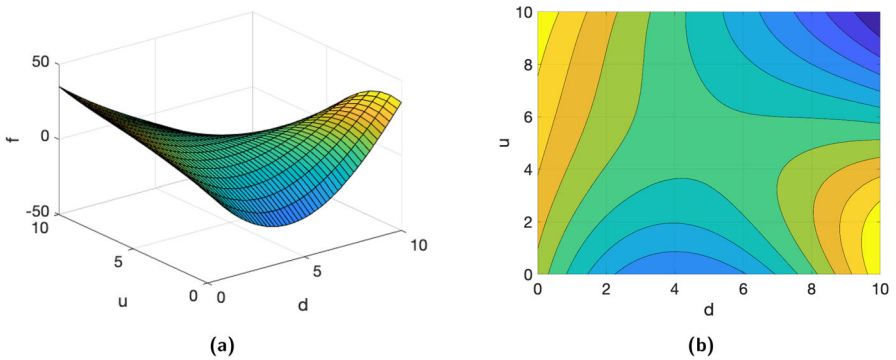

Fig. 8 GFf-2 in the case $n_D = n_U = 1$

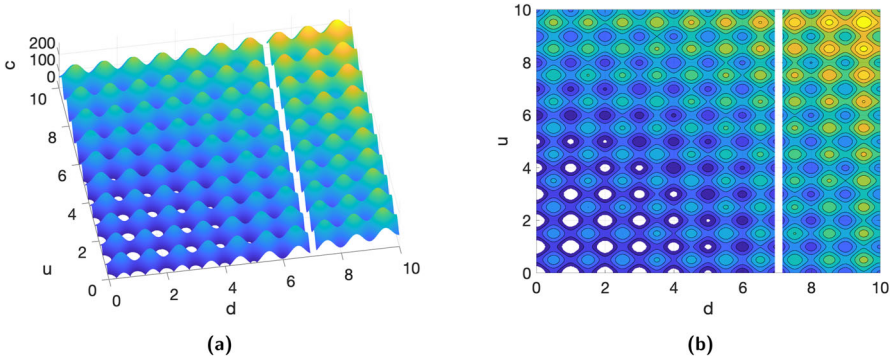

Fig. 9 GFc-4 applied to MWP-11 in the case $n_D = n_U = 1$. Feasible areas are white

6. *Non differentiability*: cuspids, corners, tangents and discontinuities are features that make functions non differentiables in some points. Some of the constraint functions present cuspids, corners and discontinuity. In particular discontinuity is an abrupt change in the function values. Discontinuities are classified in jump, infinite, removable, endpoint, or mixed. Some of the constraint functions c present jump discontinuities.

MWP-9 is not differentiable. *MWP-1,8,9,12,13* are uni-modal in both D and U. On the other hand *MWP-4,5,6,7,10,11* are multi-modal in both D and U. *MWP-2,3* are multi-modal in D only. The new test cases GFf-1, 2 and GFc-1, ..., 8 are explained in the following. They depend on the components d_i of the design vector $\mathbf{d}$, the components u_i of the uncertainty vector $\mathbf{u}$ and the combined vector $\mathbf{x} = [\mathbf{d}, \mathbf{u}]^T$.

4.1.1 GFf-1

GFf-1 is a modifications of the Rastrigin function where half of the variables are design parameters and the others are uncertain variables.

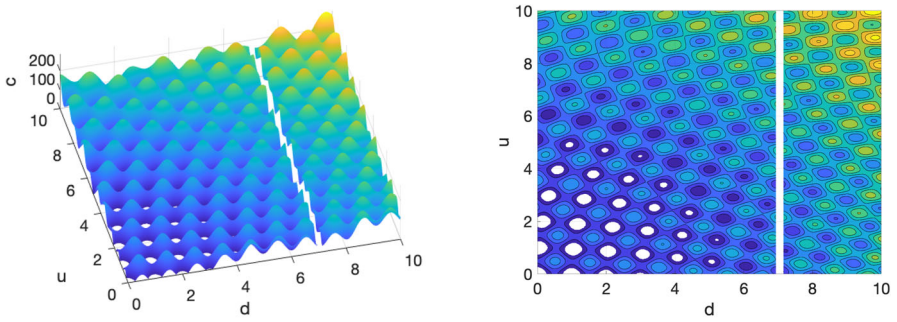

Fig. 10 GFc-5 applied to MWP-11 in the case $n_D = n_U = 1$. Feasible areas are white

$$c(\mathbf{d}, \mathbf{u}) = 10(n_D + n_U) + \sum_{i=1}^{n_D} (d_i^2 + u_i^2 - 10\big[\cos(2\pi d_i) + \cos(2\pi u_i)\big] - 5 \quad (25)$$

It is continuous, differentiable, scalable, without valleys and basins, and highly multi-modal with hundreds of local peaks.

4.1.2 GFf-2

GFf-2 is a variation of the saddle-point function *MWP-8*:

$$c(\mathbf{d}, \mathbf{u}) = \sum_{k=1}^{n} [(d_{k,\mathcal{R}} - 5)^2 - ((u_{k,\mathcal{R}} - 5)^2] \quad (26)$$

where both components $d_{k,\mathcal{R}}$ and $u_{k,\mathcal{R}}$ are obtained rotating d_k and u_k respectively by the angle

$$\theta_k = \begin{cases} \pi/8 + 1^k d_{k+1}/20 + u_{k+1}/20 & \text{if } k < n \\ \pi/8 + 1^k d_k/20 + u_k/20 & \text{else} \end{cases} \quad (27)$$

GFf-2 is continuous, differentiable, non-separable, scalable, without valleys and basins, and uni-modal.

4.1.3 GFc-1

GFc-1 is a hyper-plane and it is a linear function in both $\mathbf{d}$ and $\mathbf{u}$:

$$c(\mathbf{d}, \mathbf{u}) = \sum_{i=1}^{n} d_i + u_i + K \quad (28)$$

where $K = -\sum_i d_i - \sum_i u_{u,i} - 0.05$ with $u_{u,i}$ the upper bound for the i-th uncertain variable. *GFc-1* is continuous, differentiable, separable, scalable, without valleys and basins, and uni-modal.

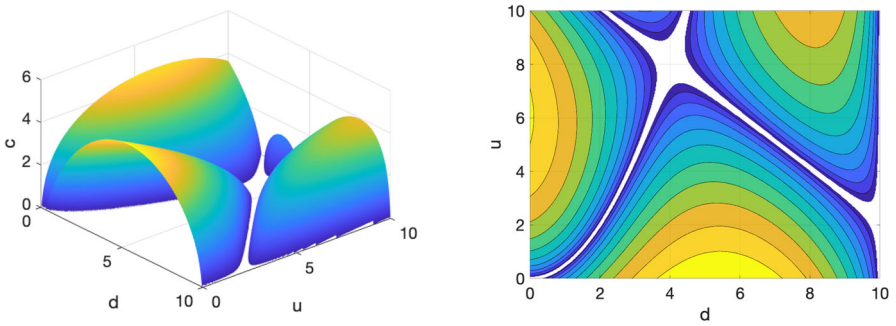

Fig. 11 GFc-6 applied to MWP-10 in the case $n_D = n_U = 1$. Feasible areas are white

4.1.4 GFc-2

GFc-2 is a modification of *GFc-1*. It is a continuous piece-wise linear function where the feasible region is a plateau. It is the intersection of two hyper-planes, the second being at the border between of feasible and infeasible regions.

$$\max\left[0, \sum_{i=1}^{n} d_i + u_i + K\right] \tag{29}$$

with K as in Eq. (28). *GFc-2* is continuous, non differentiable, separable, scalable, with a plateau, without valleys and uni-modal.

4.1.5 GFc-3

In *GFc-3* there are a jump discontinuities, valleys and plateaus. The feasible area is a narrow multidimensional circle. The function is not differentiable, scalable and uni-modal:

$$c(\mathbf{d}, \mathbf{u}) = \begin{cases} 0 & \text{if } \max_i (d_i - d_i^{opt}) \le 0.1 \\ 1 & \text{else} \end{cases} \tag{30}$$

4.1.6 GFc-4

GFc-4 is a modification of the Rastrigin function where a jump discontinuity is introduced:

$$c(\mathbf{d}, \mathbf{u}) = \begin{cases} 0 & \text{if } \max_i (d_i - d_i^{opt}) \le 0.1 \\ 30n + \sum_{i=1}^{n} [x_i^2 - 30\cos(2\pi x_i)] - 30 & \text{else} \end{cases} \tag{31}$$

It is highly multi-modal, discontinuous, not differentiable with valleys and plateaus, separable and scalable.

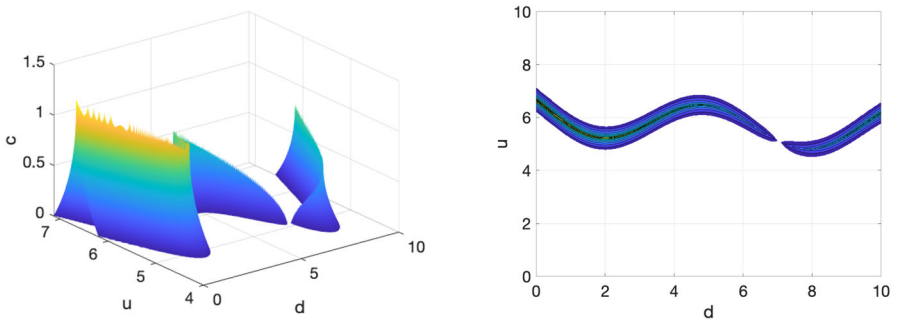

Fig. 12 GFc-7am applied to MWP-10 in the case $n_D = n_U = 1$. Feasible areas are white

4.1.7 GFc-5

GFc-5 is a modifications of Eq. (31). Here a rotation of the vectors **d** and **u** is also introduced.

$$c(\mathbf{d},\mathbf{u}) = \begin{cases} 0 & \text{if } \max_i (d_i - d_i^{opt}) \leq 0.1 \\ 30n + \sum_{k=1}^{n} [x_{k,\mathcal{R}}^2 - 30\cos(2\pi x_{k,\mathcal{R}})] - 30 & \text{else} \end{cases} \tag{32}$$

The rotated components $d_{k,\mathcal{R}}$ and $u_{k,\mathcal{R}}$ are given by the angle $\theta_k = d_i + 2u_i$.

GFc-5 is discontinuous, not differentiable, with valleys and plateaus, scalable, separable and multi-modal.

4.1.8 GFc-6

GFc-6 is a multi-dimensional peak function with high coupling between D and U. It is unfeasible in most of the domain while it is satisfied only in few narrow non linear valleys varying with **d**.

$$c(\mathbf{d},\mathbf{u}) = \left[\sum_{i=1}^{2n} (x_i - x_i^{opt}) \sum_i (d_i^2 - 2u_i) \sum_i (d_i - d_i^{opt}) \right]^{(2/7)} \tag{33}$$

It presents very narrow non-linear valleys. It is continuous, locally non differentiable, without plateaus, scalable, non separable and multi-modal.

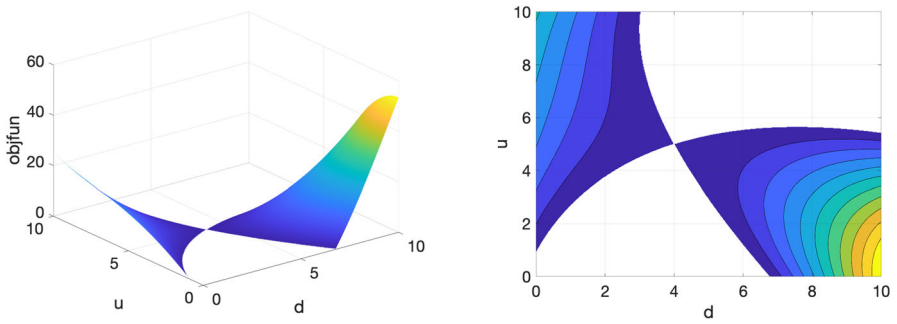

Fig. 13 GFc-8 applied to MWP-10 in the case $n_D = n_U = 1$. Feasible areas are white

4.1.9 GFc-7

GFc-7 is a multi-dimensional peak functions with high coupling between D and U and narrow unfeasible areas varying with **d**.

$$\begin{aligned} A &= \sum_{i=1}^{n_D}((3/2d_i - (d_{u,i} - d_{opt,i})/2)(\sin(d_i)/d_i) - d_{opt,i}/2)^{1/5} \\ &\quad + \sum_{i=1}^{n_U}((3/2u_i - (u_{u,i} - u_{opt,i})/2)(1 - u_{opt,i}/2)^{1/5} \\ B &= \sum_{i=1}^{n_D}(d_u(\sin(d_{u,i})/d_{u,i}) - d_{opt,i}/2)^{1/5} \\ &\quad + \sum_{i=1}^{n_U}(u_u(1 - u_{opt,i}/2)^{1/5} \\ C &= (d_1 - d_{opt,1})^{2/5} \\ D &= (d_{u,1} - d_{opt,1})^{2/5} \\ c(\mathbf{d}, \mathbf{u}) &= 1 - AC/BD \end{aligned} \tag{34}$$

where $d_{u,i}$ and $u_{u,i}$ are the upper bounds for d_i and u_i respectively. *GFc-7* has very narrow non-linear peaks with large plateaus. It is continuous, locally non differentiable (for the cuspids), without valleys, scalable, separable and multi-modal in D.

4.1.10 GFc-8

GFc-8 is a rotated versions of *MWP-8*:

$$c(\mathbf{d}, \mathbf{u}) = \sum_{i=1}^{n}[(d_{i,\mathcal{R}} - 5)^2 - (u_{i,\mathcal{R}} - 5)^2] \tag{35}$$

where $d_{i,\mathcal{R}}$ and $u_{i,\mathcal{R}}$ are obtained from d_i and u_i respectively with the rotation angle in Eq. (27). It is non-separable, scalable, continuous, differentiable and uni-modal and without valleys and plateaus.

4.2 Success rate

The Success Rate (SR) is adopted here for the performance assessment of Algorithm 2. Its definition is given in Algorithm 9 for a generic algorithm $\mathcal{A}$ applied to a generic constrained problem CP on the $D \times U$ space. It is defined as the ratio $\frac{j_s}{n}$ between the index of performance j_s and the number of independent experiments n.

First, all the parameters required by Algorithm 2 are fixed (refers to the initialisation in Algorithm 1). The following parameters are then defined: the number of repetition of the experiments n, the tolerances tol_f, tol_d and tol_u on the solution error for the objective function f, the design vector $\mathbf{d}$ and the uncertain vector $\mathbf{u}$ respectively. The formula for SR is in line 15. It depends on the tolerances and on the errors δ_c^k, δ_f^k, δ_d^k and δ_u^k with respect to the reference solutions f_{ref}, $\mathbf{d}_{\text{ref}}$ and $\mathbf{u}_{\text{ref}}$ in Tables 3 and 4. In particular, δ_c^k depends on the uncertain vector $\mathbf{u}_{\text{opt},c}$ that is the worst for the constraint function c while δ_f^k depends on the vector $\mathbf{u}_{\text{opt},f}$ that make worst the objective functions f. δ_u is necessary to verify the convergence on the maximisation in the inner loop (restoration in Sect. 3) and then to avoid counting as success solution an f_{opt}^k close to f_{ref} that is coming from a lucky combination of a wrong maximisation and a wrong minimisation in the outer loop (optimisation in Sect. 3).

Algorithm 9 Success Rate

1: Define the parameters for algorithm A to solve the constrained problem CP;
2: Define the number of repetition n;
3: Define tolerances tol_f, tol_d and tol_u;
4: Initialise the index of performance $j_s = 0$;
5: **for** k = [1,2,.., n] **do**
6: Run A on CP with the defined settings;
7: Compute $f_{\text{opt}}^k \leftarrow A(CP(\mathbf{d}, \mathbf{u}))$;
8: Compute $c_{\text{opt}}^k \leftarrow A(CP(\mathbf{d}, \mathbf{u}))$;
9: Compute $\mathbf{d}_{\text{opt}}^k$: optimal solution for the design vector
10: Compute $\mathbf{u}_{\text{opt}}^k$: optimal solution for the uncertain vector
11: Compute $\delta_f^k = |f_{ref} - f_{\text{opt}}^k|$;
12: Compute $\delta_d^k = ||\mathbf{d}_{ref} - \mathbf{d}_{\text{opt}}^k||$;
13: Compute $\delta_u^k = ||\mathbf{u}_{ref} - \mathbf{u}_{\text{opt}}^k||$;
14: Compute $\delta_c^k = \max(0, c_{\text{opt}}^k)$;
15: **if** $\delta_c^k \leq 0 \wedge \delta_f^k < tol_f \wedge \delta_d^k < tol_d \wedge \delta_u^k < tol_u$ **then**
16: $j_s = j_s + 1$
17: **end if**
18: **end for**
19: $SR = \frac{j_s}{n}$

Table 3 Reference solutions for the test cases in Table 1

ID	D	U	d min-max	u min-max	f min-max
MWP-1	$[-5; 5]^2$	$[-5; 5]^2$	−0.4833	0.0833	−1.6833
			−0.3167	−0.0833	
MWP-2	$[-5; 5]^2$	$[-5; 5]^2$	1.6954	0.7186	1.4039
			−0.0032	−0.0001	
MWP-3	$[-5; 5]^2$	$[-3; 3]^2$	−1.1807	2.0985	−2.4688
			0.9128	2.666	
MWP-4	$[-5; 5]^2$	$[-3; 3]^3$	0.4181	0.709	−0.1348
			0.4181	1.0874	
				0.709	
MWP-5	$[-5; 5]^3$	$[-1; 1]^3$	0.1111	0.4444	1.345
			0.1538	0.9231	
			0.2	0.4	
MWP-6	$[-5; 5]^4$	$[-2; 2]^3$	−0.2316	0.6195	4.543
			0.2228	0.3535	
			−0.6755	1.478	
			−0.0838		

Table 3 continued

ID	D	U	d min-max	u min-max	f min-max
MWP-7	$[-5; 5]^5$	$[-3; 3]^5$	1.4252	0.5156	−6.3509
			1.6612	0.8798	
			1.2585	0.2919	
			−0.9744	0.1198	
			−0.7348	−0.1198	
MWP-8	[0; 10]	[0;10]	5	5	0
MWP-9	[0; 10]	[0;10]	0	0	3
MWP-10	[0; 10]	[0;10]	10	2.1257	9.7794×10^{-2}
MWP-11	[0; 10]	[0;10]	7.0441	10	4.2488×10^{-2}
MWP-12	$[-0.5; 0.5]\times[0; 1]$	$[0;10]^2$	0.5	0	0.25
			0.25	0	
MWP-13	$[-1; 3]^2$	$[0;10]^2$	1	Any	1
			1	Any	
GFf-1	$[-5.14; 5.14]^{n_D}$	$[-5.14; 5.14]^{n_U}$	0	$\pm$ 4.5230	$10(n_D + n_U) - 10n_D +$
			...	...	$+30.3533n_U - 5$
			0	$\pm$ 4.5230	
	$n_D = 2$	$n_U = 2$			75.7066
GFf-2	$[0; 10]^{n_D}$	$[0;10]^{n_U}$	$[5]^{n_D}$	$[5]^{n_U}$	0

Table 4 Reference solutions for the test cases in Table 1 with the constraint changing the global optimum

ID	n_D	n_U	d min-max	u min-max	f min-max
GFf-2 & GFc-8	10	10	$[4]^{10}$	$[6.1712]^{10}$	1.4261
	20	20	$[4]^{20}$	$[6.1712]^{20}$	2.8522
	30	30	$[4]^{30}$	$[6.1712]^{30}$	4.2784
	40	40	$[4]^{40}$	$[6.1712]^{40}$	5.7045
	50	50	$[4]^{50}$	$[6.1712]^{50}$	7.1306

4.3 Algorithm settings

An important feature of the proposed approach is its modularity in the sense that any optimiser can be plugged in and used for the single optimisation problem in Eqs. (2) – (4). To enhance efficiencies of Algorithm 2, then, the right combination of optimisation solvers should be selected. An optimal choice would require a prior knowledge of the main features of a given problem. For complex multi-modal functions, we suggest the use of the memetic optimisation solver MP-AIDEA because it has shown to be efficient and effective, on average, on a wide range of problems mixing different characteristics. For continuous uni-modal functions we use instead the Matlab *fmincon* solver with an interior-point scheme. We give here the parameter settings of MP-AIDEA that have been used for all tests. The number of agents for each population N_{pop} and the maximum number of function evaluations were set to be respectively $N_{\text{pop}} = \max[5, n_D]$, $n^{\text{outer}}_{\text{feval,max}} = 500 n_D$, $n^{\text{inner,f}}_{\text{feval,max}} = 500 n_U$ and $n^{\text{inner,c}}_{\text{feval,max}} = 500 n_U$. The dimension of the bubble for the global restart is $\delta_{\text{global}} = 0.1$, the number of populations is $n_{\text{pop}} = 2$ and the convergence threshold of DE is $\rho = 0.25$.

5 Results

The results are presented and explained in this section. In particular, four sets of test have been performed. In the first, Algorithm 2 has been combined with the optimiser *fmincon*, while in the other cases MP-AIDEA has been used. First we consider one uni-modal problem. The performance of the algorithm is assessed increasing the dimension of the problem. Then we consider the worst-case complexity analysis on the benchmark presented in Sect. 4.1 with a wide variety of difficulties. A complexity analysis on the algorithm convergence is then presented for a selected test case for different problem dimensions. Finally we apply Algorithm 2 to solve a realistic engineering problem: the design for robustness of a communication satellite.

5.1 Uni-modal test problem

For the first set of results, the test case used is given by the combination of the objective function *GFf-2* and the constraint function *GFc-8*. They are both continuous,

Table 5 Success rates of *GFf-2* and *GFc-8* for different problem dimensions (rows) and limits on the maximum number of function evaluations (columns). Optimiser: fmincon. $\delta_d = \delta_u = \delta_f = 0.1, \delta_c = 0$

dim	2e5	4e5	6e5	8e5	1e6	2e6	3e6	4e6	5e6	6e6	7e6
10×10	0.03	0.95	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
20×20	–	–	–	–	0.28	1.00	1.00	1.00	1.00	1.00	1.00
30×30	–	–	–	–	–	0.14	0.92	1.00	1.00	1.00	1.00
40×40	–	–	–	–	–	0.7	0.8	0.34	0.79	0.97	1.00
50×50	–	–	–	–	–	–	–	–	0.04	0.16	0.30

differentiable, unimodal and non-separable. With these features a local optimiser is sufficient to solve Eqs. (2)– (4) at each iteration. The constraint function c admits only one feasible design vector, which is different from the unconstrained optimum of *GFf-2*. The local optimiser we used in this test is *fmincon*. The test functions are devised to be scalable with a predictable value of the exact min-max solution. Results are collected in Table 5 for a number of function evaluations up to $7e6$. The table shows up to dimension $n_D = 40$ and $n_U = 40$ the algorithm can achieve $SR = 1$ within the maximum number of function evaluations. For $n_D = 50$ and $n_U = 50, 7e6$ is not enough and the best result is a success rate of 30%.

5.2 Multi-modal test problems

For the second set of experiments, Tables 6, 7, 8, 9, 10, 11 and 12 collect the results for all the test cases given by the combination of objective functions f from Table 1 and constraint functions c from Table 2. The last two columns of each table, $n_{\text{iter,min}}$ and $n_{\text{iter,max}}$, collect the minimum and maximum number of loops for which the algorithm achieves $SR = 1$ (rows with the symbol $-$ correspond to problems for which $SR = 1$ has not been obtained for any of the 100 runs). For almost all the problems Algorithm 2 converges to the correct solution with an $SR = 1$ within the maximum number of function evaluations. For some problems (namely *GFF1-GFC1*, *MWP10-GFC4*, *MWP11-GFC5*, *MWP11-GFC2*) few of the runs did not converge to the correct minimum of f but the SR is still reasonably high.

5.3 Convergence complexity

Sections 5.1 and 5.2 show the performance of Algorithm 2 with respect to the worst-case computational complexity where an upper bound on the number of function evaluations ($n_{\text{feval,max}}$) is fixed for each optimisation step of the approach: $n^{\text{outer}}_{\text{feval,max}}$, $n^{\text{inner,f}}_{\text{feval,max}}$ and $n^{\text{inner,c}}_{\text{feval,max}}$. It is here interesting to show an other computational complexity analysis that is related to the order of magnitude of the number of function evaluations needed to converge to the optimal solution. In particular, Table 13 summarises the results for the test case $GFf\text{-}1\&GFc\text{-}1$. This test case has been selected as representative because it is scalable in both design D and uncertain U spaces

Table 6 GFc-1, MP-AIDEA, 2 populations, $\delta_d = \delta_u = \delta_f = 0.1$, $\delta_c = 0$

	1000	5000	10000	50000	100000	200000	500000	1000000	$n_{iter,min}$	$n_{iter,max}$
MWP-1	0.00	0.00	0.56	1.00	1.00	1.00	1.00	1.00	6.00	9.00
MWP-2	0.00	0.00	0.40	0.96	1.00	1.00	1.00	1.00	6.00	29.00
MWP-3	0.00	0.00	0.04	1.00	1.00	1.00	1.00	1.00	9.00	18.00
MWP-4	0.00	0.00	0.00	0.88	1.00	1.00	1.00	1.00	11.00	18.00
MWP-5	0.00	0.00	0.00	1.00	1.00	1.00	1.00	1.00	8.00	10.00
MWP-6	0.00	0.00	0.00	0.56	1.00	1.00	1.00	1.00	10.00	22.00
MWP-7	0.00	0.00	0.00	0.00	0.00	0.10	1.00	1.00	19.00	19.00
MWP-8	0.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	6.00	6.00
MWP-9	0.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	11.00	12.00
MWP-10	0.00	0.02	0.02	1.00	1.00	1.00	1.00	1.00	13.00	29.00
MWP-11	0.00	0.14	0.20	1.00	1.00	1.00	1.00	1.00	11.00	29.00
MWP-12	0.00	0.00	1.00	1.00	1.00	1.00	1.00	1.00	6.00	7.00
MWP-13	0.00	0.00	0.08	1.00	1.00	1.00	1.00	1.00	29.00	29.00
GFf-1	0.00	0.00	0.68	0.98	0.98	0.98	0.98	0.98	–	–

Table 7 GFc-2, MP-AIDEA, 2 populations, $\delta_d = \delta_u = \delta_f = 0.1$, $\delta_c = 0$

	1000	5000	10000	50000	100000	200000	500000	1000000	$n_{iter,min}$	$n_{iter,max}$
MWP-1	0.00	0.00	0.54	1.00	1.00	1.00	1.00	1.00	6.00	9.00
MWP-2	0.00	0.00	0.50	1.00	1.00	1.00	1.00	1.00	6.00	15.00
MWP-3	0.00	0.00	0.02	1.00	1.00	1.00	1.00	1.00	9.00	15.00
MWP-4	0.00	0.00	0.00	0.82	1.00	1.00	1.00	1.00	8.00	14.00
MWP-5	0.00	0.00	0.00	1.00	1.00	1.00	1.00	1.00	7.00	9.00
MWP-6	0.00	0.00	0.00	0.64	1.00	1.00	1.00	1.00	9.00	15.00
MWP-7	0.00	0.00	0.00	0.00	0.00	0.08	1.00	1.00	19.00	19.00
MWP-8	0.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	6.00	6.00
MWP-9	0.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	8.00	8.00
MWP-10	0.00	0.02	0.02	1.00	1.00	1.00	1.00	1.00	12.00	29.00
MWP-11	0.00	0.14	0.18	0.98	0.98	0.98	0.98	0.98	–	–
MWP-12	0.00	0.00	1.00	1.00	1.00	1.00	1.00	1.00	6.00	7.00
MWP-13	0.00	0.00	0.14	1.00	1.00	1.00	1.00	1.00	7.00	9.00
GFf-1	0.00	0.00	0.62	1.00	1.00	1.00	1.00	1.00	28.00	29.00

Table 8 GFc-3, MP-AIDEA, 2 populations, $\delta_d = \delta_u = \delta_f = 0.1$, $\delta_c = 0$

	1000	5000	10000	50000	100000	200000	500000	1000000	$n_{iter,min}$	$n_{iter,max}$
MWP-1	0.00	0.00	0.82	1.00	1.00	1.00	1.00	1.00	6.00	9.00
MWP-2	0.00	0.00	0.44	1.00	1.00	1.00	1.00	1.00	7.00	16.00
MWP-3	0.00	0.00	0.00	0.94	1.00	1.00	1.00	1.00	9.00	21.00
MWP-4	0.00	0.00	0.10	1.00	1.00	1.00	1.00	1.00	7.00	11.00
MWP-5	0.00	0.00	0.00	1.00	1.00	1.00	1.00	1.00	7.00	9.00
MWP-6	0.00	0.00	0.00	0.68	1.00	1.00	1.00	1.00	8.00	15.00
MWP-7	0.00	0.00	0.00	0.00	0.02	0.54	1.00	1.00	18.00	19.00
MWP-8	0.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	6.00	6.00
MWP-9	0.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	6.00	18.00
MWP-10	0.00	0.02	0.02	1.00	1.00	1.00	1.00	1.00	12.00	29.00
MWP-11	0.00	0.16	0.28	1.00	1.00	1.00	1.00	1.00	10.00	29.00
MWP-12	0.00	0.00	1.00	1.00	1.00	1.00	1.00	1.00	6.00	7.00
MWP-13	0.00	0.00	0.34	1.00	1.00	1.00	1.00	1.00	7.00	10.00
GFf-1	0.00	0.00	0.60	0.96	1.00	1.00	1.00	1.00	19.00	29.00

Table 9 GFc-4, MP-AIDEA, 2 populations, $\delta_d = \delta_u = \delta_f = 0.1$, $\delta_c = 0$

	1000	5000	10000	50000	100000	200000	500000	1000000	$n_{iter,min}$	$n_{iter,max}$
MWP-1	0.00	0.00	1.00	1.00	1.00	1.00	1.00	1.00	7.00	18.00
MWP-2	0.00	0.00	0.56	1.00	1.00	1.00	1.00	1.00	8.00	29.00
MWP-3	0.00	0.00	0.00	0.94	1.00	1.00	1.00	1.00	11.00	29.00
MWP-4	0.00	0.00	0.02	0.98	1.00	1.00	1.00	1.00	8.00	29.00
MWP-5	0.00	0.00	0.00	1.00	1.00	1.00	1.00	1.00	7.00	13.00
MWP-6	0.00	0.00	0.00	0.62	1.00	1.00	1.00	1.00	11.00	19.00
MWP-7	0.00	0.00	0.00	0.00	0.02	0.22	1.00	1.00	19.00	19.00
MWP-8	0.00	0.96	1.00	1.00	1.00	1.00	1.00	1.00	6.00	29.00
MWP-9	0.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	7.00	14.00
MWP-10	0.00	0.68	0.90	0.98	0.98	0.98	0.98	0.98	–	–
MWP-11	0.00	0.64	0.92	1.00	1.00	1.00	1.00	1.00	9.00	29.00
MWP-12	0.00	0.00	1.00	1.00	1.00	1.00	1.00	1.00	6.00	12.00
MWP-13	0.00	0.00	0.68	1.00	1.00	1.00	1.00	1.00	7.00	29.00
GFf-1	0.00	0.00	0.54	0.96	1.00	1.00	1.00	1.00	13.00	29.00

Table 10 GFc-5, MP-AIDEA, 2 populations, $\delta_d = \delta_u = \delta_f = 0.1, \delta_c = 0$

	1000	5000	10000	50000	100000	200000	500000	1000000	$n_{iter,min}$	$n_{iter,max}$
MWP-1	0.00	0.00	0.82	1.00	1.00	1.00	1.00	1.00	6.00	28.00
MWP-2	0.00	0.00	0.76	1.00	1.00	1.00	1.00	1.00	7.00	29.00
MWP-3	0.00	0.00	0.00	0.96	1.00	1.00	1.00	1.00	12.00	29.00
MWP-4	0.00	0.00	0.02	1.00	1.00	1.00	1.00	1.00	8.00	29.00
MWP-5	0.00	0.00	0.00	1.00	1.00	1.00	1.00	1.00	7.00	27.00
MWP-6	0.00	0.00	0.00	0.38	1.00	1.00	1.00	1.00	11.00	20.00
MWP-7	0.00	0.00	0.00	0.00	0.00	0.04	0.96	1.00	19.00	19.00
MWP-8	0.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	6.00	29.00
MWP-9	0.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	8.00	15.00
MWP-10	0.00	0.66	0.86	1.00	1.00	1.00	1.00	1.00	8.00	29.00
MWP-11	0.00	0.90	0.96	1.00	1.00	1.00	1.00	1.00	10.00	29.00
MWP-12	0.00	0.00	1.00	1.00	1.00	1.00	1.00	1.00	6.00	7.00
MWP-13	0.00	0.00	0.22	1.00	1.00	1.00	1.00	1.00	7.00	29.00
GFf-1	0.00	0.00	0.58	0.96	1.00	1.00	1.00	1.00	6.00	29.00

Table 11 GFc-6, MP-AIDEA, 2 populations, $\delta_d = \delta_u = \delta_f = 0.1, \delta_c = 0$

	1000	5000	10000	50000	100000	200000	500000	1000000	$n_{iter,min}$	$n_{iter,max}$
MWP-1	0.00	0.00	1.00	1.00	1.00	1.00	1.00	1.00	7.00	20.00
MWP-2	0.00	0.00	0.96	1.00	1.00	1.00	1.00	1.00	7.00	24.00
MWP-3	0.00	0.00	0.02	1.00	1.00	1.00	1.00	1.00	11.00	22.00
MWP-4	0.00	0.00	0.04	1.00	1.00	1.00	1.00	1.00	9.00	23.00
MWP-5	0.00	0.00	0.06	1.00	1.00	1.00	1.00	1.00	7.00	18.00
MWP-6	0.00	0.00	0.00	0.48	1.00	1.00	1.00	1.00	11.00	22.00
MWP-7	0.00	0.00	0.00	0.00	0.04	0.62	1.00	1.00	19.00	19.00
MWP-8	0.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	6.00	14.00
MWP-9	0.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	10.00	25.00
MWP-10	0.00	0.98	1.00	1.00	1.00	1.00	1.00	1.00	7.00	16.00
MWP-11	0.00	0.68	0.94	1.00	1.00	1.00	1.00	1.00	7.00	20.00
MWP-12	0.00	0.00	1.00	1.00	1.00	1.00	1.00	1.00	6.00	14.00
MWP-13	0.00	0.00	1.00	1.00	1.00	1.00	1.00	1.00	7.00	18.00
GFf-1	0.00	0.00	0.64	1.00	1.00	1.00	1.00	1.00	6.00	29.00

Table 12 GFc-7, MP-AIDEA, 2 populations, $\delta_d = \delta_u = \delta_f = 0.1, \delta_c = 0$

	1000	5000	10000	50000	100000	200000	500000	1000000	$n_{iter,min}$	$n_{iter,max}$
MWP-1	0.00	0.00	0.60	1.00	1.00	1.00	1.00	1.00	7.00	14.00
MWP-2	0.00	0.00	0.14	1.00	1.00	1.00	1.00	1.00	11.00	23.00
MWP-3	0.00	0.00	0.00	1.00	1.00	1.00	1.00	1.00	11.00	16.00
MWP-4	0.00	0.00	0.00	0.94	1.00	1.00	1.00	1.00	10.00	18.00
MWP-5	0.00	0.00	0.00	1.00	1.00	1.00	1.00	1.00	7.00	9.00
MWP-6	0.00	0.00	0.00	0.22	1.00	1.00	1.00	1.00	11.00	16.00
MWP-7	0.00	0.00	0.00	0.00	0.00	0.18	1.00	1.00	19.00	19.00
MWP-8	0.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	6.00	6.00
MWP-9	0.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	6.00	18.00
MWP-10	0.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	6.00	18.00
MWP-11	0.00	0.08	0.14	1.00	1.00	1.00	1.00	1.00	9.00	27.00
MWP-12	0.00	0.02	1.00	1.00	1.00	1.00	1.00	1.00	6.00	11.00
MWP-13	0.00	0.00	0.26	1.00	1.00	1.00	1.00	1.00	7.00	10.00
GFf-1	0.00	0.00	0.82	1.00	1.00	1.00	1.00	1.00	6.00	29.00

Table 13 Problem GFf-1&GFc-1. Convergence complexity

dim	$\overline{n}^{out,f}_{feval}$	$\overline{n}^{in,f}_{feval}$	$\overline{n}^{in,c}_{feval}$	$\overline{n}_{feval}$
2	667.1	493.5	162.5	1323.1
3	2631.8	1232.2	200.5	4064.5
4	9607.1	2229.9	241.0	12078.0
5	17405.2	3151.2	254.5	20810.9
6	74085.8	13971.8	645.0	88702.6
7	86616.2	16354.2	992.6	103963.0
8	184190.9	29084.7	1337.4	214613.0
9	211740.5	43107.5	1591.9	256439.9
10	234045.3	51374.4	1765.1	287184.8

and it is also one of the most difficult within the proposed benchmark. Each row in Table 13 corresponds to a different problem dimension: $n = n_D = n_U = 2, 3, ..., 10$. The columns represent the average costs at convergence over 10 repetitions for the constrained minimisation in the outer loop in Eq. (2) ($\overline{n}^{out,f}_{feval}$), the constrained maximisation in the inner loop in Eq. (3) ($\overline{n}^{in,f}_{feval}$), the maximisation of the constraint function in the inner loop in Eq. (2) ($\overline{n}^{in,c}_{feval}$) and finally the average cost for the whole algorithm ($\overline{n}_{feval}$). The quantities $\overline{n}^{out,f}_{feval}$, $\overline{n}^{in,f}_{feval}$ and $\overline{n}^{in,c}_{feval}$ have been determined averaging the sum, for the different algorithm's iterations, of the number of function evaluations at convergence. The optimiser MP-AIDEA has been used. In order to assure convergence in each optimisation step in Algorithm 2, the number of populations has been set equal to the problem dimension $n_{pop} = n$ (the problem is highly multi-modal) and the maximum allowed number of function evaluations of each step has been fixed at $n^{\text{outer}}_{\text{feval,max}} = n^{\text{inner,f}}_{\text{feval,max}} = n^{\text{inner,c}}_{\text{feval,max}} = 1e4n$ while for the whole algorithm it is $n_{\text{feval,max}} = 2e6n$. The remaining input parameters for MP-AIDEA have been fixed as in Sect. 4.3.

5.4 Space system design

The min-max approach in Algorithm 2 is finally tested on the design for robustness of a Complex Engineered System (CEdS) under uncertainty. The system under analysis is an observation spacecraft and the goal of the mission is the fire detection within a belt centred at the latitude of 50 deg. The spacecraft is modelled as the network shown in Fig. 14 where the nodes correspond to its subsystems and the links to the coupling between them. The mathematical models that have been used for the nodes are a modification of the ones the authors extensively presented in Filippi et al. (2019). The differences are described in the following and are in the explicit definition of a node for the orbital dynamic and in the payload subsystem. Design and uncertain variables are listed in Tables 14 and 15 respectively.

Within the orbit node, considering a circular Low Earth Orbit (LEO), the altitude h, inclination i, the minimum elevation angle ϵ_{min} at which the ground station is

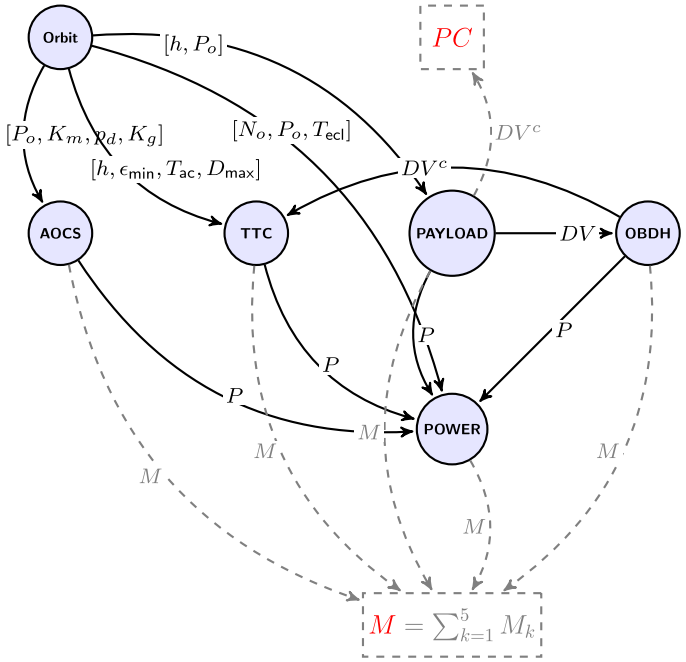

Fig. 14 Representation of the spacecraft as a complex system. The two quantities of interest are the mass of the M and the percent of coverage are PC for the payload

Table 14 Spacecraft model - design parameters

Design parameter	Symbol	Units	id	LB	UB	Sub-system
Altitude	h	km	d_1	1000	1400	Orbit
Min elevation angle ground station	$\epsilon_{\min}$	deg	d_2	15	20	Orbit
Inclination	i	deg	d_3	0	90	Orbit
Width for square detector	d	μm	d_4	20	40	Payload
Quality factor for imaging	Q	–	d_5	0.5	2	Payload
Operating wavelength	λ	μm	d_6	3	6	Payload
Obdh type	τ_{obdh}	–	d_7	0	1	Obdh
Compression factor	C	–	d_8	0.2	0.6	Obdh
Slew angle	sl	deg	d_9	10	60	Aocs
Time for slew manuvers	t_{sl}	s	d_{10}	10	20	aocs
Frequency	f	GHz	d_{11}	7	10	ttc
Modulation	β	–	d_{12}	0	1	ttc
Amplifier type	τ_{amp}	–	d_{13}	0	1	ttc
Cell type	τ_{cell}	–	d_{14}	0	1	Power
Bus voltage	V_{bus}	V	d_{15}	3	5	Power
Allowed bus drop	V_{drop}	V	d_{16}	1	3	Power

Table 15 Spacecraft model - uncertain parameters

Uncertain parameter	Symbol	Units	id	LB	UB	Sub-system
Magnetic latitude	lat_{m}	deg	u_1	0	10	Orbit
Maximum incidence angle	$IA_{\max}$	deg	u_2	60	80	Payload
Max ground sampling distance	$Y_{\max}$	m	u_3	60	80	Payload
Δ Mass	Δ_{m}	%	u_4	0	20	obdh
Δ Power	Δ_{p}	%	u_5	0	20	obdh
Antenna efficincy	η_{ant}	–	u_6	0.6	0.9	ttc
Antenna gain	G_{ant}	dB	u_7	1	5	ttc
Mass distribution network	m_{rfdn}	kg	u_8	0.1	0.5	ttc
Cell packing efficiency	η_{pack}	–	u_9	0.8	0.9	Power
Harness mass factor	k_{harn}	%	u_{10}	1	10	Power
Worst case angle of incidence	θ	deg	u_{11}	20	40	Power
Reflectance factor	q	–	u_{12}	0.5	0.7	aocs
Residual dipole	m	Am^2	u_{13}	0.0005	0.0015	aocs
Delta inertia	δI	–	u_{14}	2	10	aocs

able to see the orbiting satellite and the magnetic latitude lat_m are used to evaluate the coupling variables with the Attitude and Orbit Control System (AOCS), the Telemetry, Tracking and Command System (TTC), the Power System and the Payload System (Fig. 14). These coupling variables are: the period of each orbit P_o, the number of orbits N_o the satellite perform due to the shift of the longitude of the ascending node, the time of eclipse for each orbit T_{ecl}, the dynamic pressure p_{dyn}, the mean Earth magnetic field strength K_m, the gravitational field K_g, the maximum distance to the target $D_{\max}$, and the access time T_{ac} (or total time in view) between the target and the satellite, where the target is the ground station at 22 deg of latitude used for down-link and up-link. With exception of T_{ac} the formulas can be found in Filippi et al. (2019). Instead, considering that the satellite ground track is determined by the inclination i and by the longitude of the ascending node L_{node} and that the latter increases by 360 deg in 1346 min (the rotation of the Earth relative to the stars), T_{ac} is calculated considering the total number of orbits i that happens during this period $T_{\mathrm{ac}} = \sum_i T_{\mathrm{ac},i}$. Following Wertz et al. (1999), that describes the motion of the satellite as seen from a point on the Earth (the ground station), $T_{\mathrm{ac},i}$ is evaluated as:

$$T_{\mathrm{ac},i} = \frac{P}{180deg} \arccos \frac{\cos \lambda_{\max,i}}{\cos \lambda_{\min,i}} \tag{36}$$

with $\lambda_{\min,i}$ and $\lambda_{\max,i}$ the minimum and maximum Earth Central Angle for the i-th orbit.

The Payload System is an infrared camera that is used to detect possible fires and its target is the belt at 50 deg of latitude. Within the payload node the model's parameters are h (shared with the orbit node), P_o (coupling parameter), the width for square detector d, the quality factor Q, the operating wavelength λ, the maximum incidence

angle of the instrument $IA_{\max}$ and the maximum ground sampling distance $Y_{\max}$. The model evaluates the following coupling variables: the data volume DV shared with On Board Data handling (OBDH) and the power requirement P shared with the Power System. The model evaluates also the payload mass and the percentage of coverage area PC of each orbit during which the payload target is seen. In particular, PC is calculated following Wertz et al. (1999) as function of $\lambda_{\max}$, i and the latitude of the target $Lat = 50\ deg$:

$$PC = \begin{cases} 0 & \text{if } Lat > \lambda_{\max} + i \\ \phi_1/180 & \text{if } i + \lambda_{\max} > Lat > i - \lambda_{\max} \\ (\phi_1 - \phi_2)/180 & \text{if } i - \lambda_{\max} > Lat > 0 \end{cases} \tag{37}$$

where

$$\cos\phi_1 = \frac{-\sin\lambda_{\max} + \cos i \sin Lat}{\sin i \cos Lat} \tag{38}$$

and

$$\cos\phi_2 = \frac{\sin\lambda_{\max} + \cos i \sin Lat}{\sin i \cos Lat} \tag{39}$$

The remaining couplings between nodes are the compressed data volume DV^c that OBDH send to TTC for down-link to the ground station and the power requirements P of all the nodes (orbit excluded) that the Power sub-system has to make available. Finally, the global outputs of the network are the overall mass M of the satellite, sum of the masses of the components, and the percent coverage PC of payload target land. In the optimisation framework, M is considered to be the performance indicator while PC is the constraint to be satisfied. This mission design problem is translated into the following constrained min-max problem:

$$\begin{aligned} &\min_{\mathbf{d}\in D} \max_{\mathbf{u}\in U} M(\mathbf{d}, \mathbf{u}) \\ &s.t. \\ &PC(\mathbf{d}, \mathbf{u}) \geq \nu \quad \forall \mathbf{u} \in U \end{aligned} \tag{40}$$

In order to explore the conflict between f and c, the corresponding Pareto front has been reconstructed. We want to apply here the main min-max method presented in Algorithm 2. The algorithm has then been repeated using 30 different values for the threshold ν through an ECS approach.

The results of Eq. (40) are shown in Fig. 15 for which the optimiser MP-AIDEA has been used with the setting specified in Section 4.3. In particular, Fig. 15a presents the Pareto front reconstructed for the different values of ν, while the shape of the front can be understood looking at Fig. 15b,c,d. There are indeed two different geographical targets on the Earth for the defined mission: the ground station (22 deg of latitude) used for up-link and down-link by TTC and the area that has to be monitored by the payload for possible fire detection (50 deg of latitude).

These two targets are quantified in the respectively node's models by T_{ac} and PC, the latter being the constraint function c and the former having an high impact on the final mass M of the overall spacecraft that is the objective function f. The most

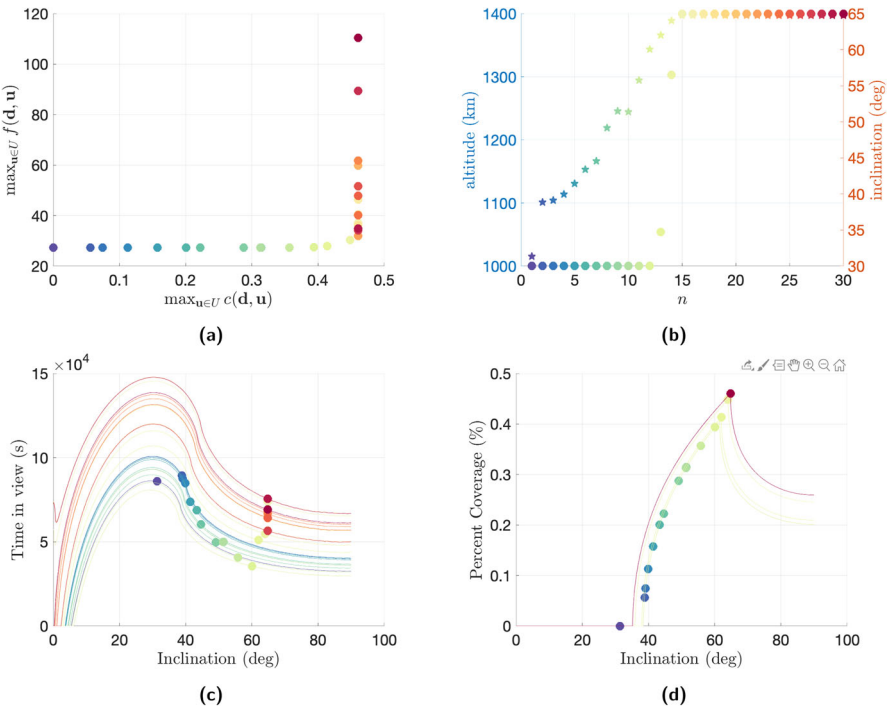

Fig. 15 Optimal Pareto points for the spacecraft design problem in Eq. (40) calculated with Algorithm 2 and applying the EC approach, running 30 optimisations with different thresholds v in the constraint function c. Sub-figure a shows the Pareto Front representing the tension between max c and max f. Sub-figures b,c,d explain the shape of the Pareto Front. The most important design parameters leading the trade-off between f and c are the altitude $H = d_1$ and the inclination $I = d_3$. Sub-figure (b) shows the increase of altitude (points) and inclination (stars) for the different solutions in sub-figure a. Sub-figure c presents the time in view between satellite and the ground station for a series of revolutions as function of the inclination. Sub-figure d finally shows the percent of land coverage by the payload

influential design parameters with regard to PC and T_{ac} in the trade-off within the set of optimal Pareto points are the altitude $h = d_1$ and the inclination $i = d_3$. Fig. 15b shows their optimal values while moving in the front, while Fig. 15c,d show finally the corresponding values of T_{ac} and PC. For low values of v in the constraint function (left side of the Pareto front) the design solution selects the orbit inclination that maximise the amount of time for the link between the spacecraft' antenna and ground station. This configuration reduces the overall mass M at the expense of the capacity of detect fires (PC). As v increases, the solutions becomes sub-optimal for Fig. 15c while maximises the area in Fig. 15d.

6 Conclusion

The paper has presented a new algorithm for the solution of a class of constrained min-max problems. The class of min-max problems emerges naturally from the need

to make robust decisions under uncertainty in the case in which constraints need to be always satisfied. The method is based on the alternation of a minimisation and a restoration step. This scheme is fairly optimiser agnostic and we demonstrated its applicability even in the case a simple gradient search is used. For the case in which the min-max solution requires the global exploration of a complex solution space we have proposed the use of a memetic approach based on Inflationary Differential Evolution. Our complexity analysis has revealed that the algorithm is overall of polynomial complexity with maximum exponent equal to 2.

The combination of the proposed solution strategy and memetic global optimiser 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 propose is successful at identifying the constrained min-max solution with a limited number of calls to objective functions and constraints. Such solution minimises the worst case realisation of the objective function in the uncertain space while guaranteeing its feasibility in all the possible scenarios. The benchmark is complemented by a real case of robust optimisation of a space systems.

In the case in which a feasible solution in all the uncertain domain could not be found, we proposed a constraint relaxation procedure to automatically adapt the admissible region. Finally we proposed a trade-off approach between unconstrained min-max solution and constraint satisfaction based on a combination of Chebyshev and Pascoletti-Serafini scalarisation. This approach is promising for cases in which a user defined relaxation of the constraint is possible as it allows one to explore the optimal trade-off curve between optimality and reliability. The use of this approach will be further developed in future work.

Acknowledgements The work in this paper was supported by the H2020-MSCA-ITN-2016 UTOPIAE, grant agreement 722734.

References

Abdelbar AM, Ragab S, Mitri S (2003) Applying Co-Evolutionary Particle Swam Optimization to the Egyptian Board Game Seega. Proceedings of the First Asian-Pacific Workshop on Genetic Programming, pp 9–15

Agnew D (1981) Improved minimax optimization for circuit design. IEEE Trans Circuits Syst 28:791–803

Aissi H, Bazgan C, Vanderpooten D (2008) Min-max and min-max regret versions of combinatorial optimization problems: a survey. Eur J Op Res 197:427–438

Barbosa HJ (1999) A coevolutionary genetic algorithm for constrained optimization. In: Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999, vol 3, pp 1605–1611

Baxter R, Hastings N, Law A, Glass EJ (2008) Algorithms for worst-case design and applications to risk management, vol 39

Bo RI, Grad SM, Wanka G (2007) A general approach for studying duality in multiobjective optimization. Math Methods Op Res 65(3):417–444

Bowman VJ (1976) On the relationship of the Tchebycheff norm and the efficient frontier of multiple-criteria objectives. lecture notes in economics and mathematical systems (operations research). In: Thiriez HZS (ed) Multiple criteria decision making. Springer-Verlag, Berlin

Branke J, Rosenbusch J (2008) New approaches to coevolutionary worst-case optimization. Springer, Berlin Heidelberg

Carlo MD, Vasile M, Greco C, Epenoy R (2019) Aas 19-285 Robust Optimisation of Low-Thrust Interplanetary Transfers Using Evidence Theory, Aas, pp 1–20

Cavalier TM, Conner WA, del Castillo E, Brown SI (2007) A heuristic algorithm for minimax sensor location in the plane. Eur J Op Res 183(1):42–55. https://doi.org/10.1016/j.ejor.2006.10.055

Chaney RW (1982) A method of centers algorithm for certain minimax problems. Math Programm 22:202–226

Conner W. Comparison of Evolutionary Algorithms on the Minimax Sensor Location Problem, tech. rep

Cramer A, Sudhoff S, Zivi E (2009) Evolutionary algorithms for minimax problems in robust design. IEEE Trans Evolut Comput 13:444–453

Das I, Dennis JE (1997) A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems. Struct Optim 14(1):63–69

Di Carlo M, Vasile M, Minisci E (2019) Adaptive multi-population inflationary differential evolution. Soft Comput 24(5):3861–3891

Filippi G, Vasile M, Krpelik D, Korondi PZ, Marchi M, Poloni C (2019) Space systems resilience optimisation under epistemic uncertainty. Acta Astronautica 165(May):195–210

Filippi G, Vasile M (2019) A Memetic Approach to the Solution of Constrained Min-Max Problems. In: 2019 IEEE Congress on Evolutionary Computation, CEC 2019 - Proceedings, pp 506–513

Filippi G, Vasile M (2020) Inflationary Differential Evolution for Constrained Multi-Objective Optimisation Problems. In: International Conference on Bioinspired Optimisation Methods and Their Application, (Bruxelles)

Gass S, Saaty T (1955) The computational algorithm for the parametric objective function. Naval Res Logist Q 2(1–2):39–45

Haimes YY, Lasdon LS, Wismer DA (1971) On a bicriterion formation of the problems of integrated system identification and system optimization. IEEE Trans Syst, Man Cybern SMC 1(3):296–297

Hughes EJ (2005) Checkers using a co-evolutionary on-line evolutionary algorithm. 2005 IEEE Congress on Evolutionary Computation, IEEE CEC 2005. Proceedings, vol 2, pp 1899–1905

Jamil M, Yang X-S (2013) A literature survey of benchmark functions for global optimization problems. J Math Modell Numer Optim 4(2):150–194

Jensen MT (2001) A New Look at Solving Minimax Problems with Coevolution. In: 4th Metaheuristics International Conference

Kasimbeyli R, Ozturk ZK, Kasimbeyli N, Yalcin GD, Erdem BI (2019) Comparison of some scalarization methods in multiobjective optimization: comparison of scalarization methods. Bull Malays Math Sci Soc 42:1875–1905

Kim J, Tahk MJ (2001) Co-evolutionary computation for constrained min-max problems and its applications for pursuit-evasion game. Proceedings of the IEEE Conference on Evolutionary Computation, ICEC 2:1205–1212

Laskari EC, Parsopoulos KE, Vrahatis MN (2002) Particle swarm optimization for minimax problems. Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002, vol 2, pp 1576–1581

Lung RI, Dumitrescu D (2011) A new evolutionary approach to minimax problems, 2011 IEEE Congress of Evolutionary Computation. CEC 2011:1902–1905

Marzat J, Walter E, Piet-Lahanier H (2013) Worst-case global optimization of black-box functions through Kriging and relaxation. J Glob Optim 55(4):707–727

Ong YS, Nair PB, Lum KY (2006) Max-min surrogate-assisted evolutionary algorithm for robust design. IEEE Trans Evolut Comput 10(4):392–404

Power J (August 1998) Basic Research in Computer Science, Power, no

Sebald AV, Schlenzig J (1994) Minimax design of neural net controllers for highly uncertain plants. IEEE Trans Neural Netw 5(1):73–82

Shannon CE (1950) Philosophical Magazine Series 7 XXII. Programm Computer Play Chess, Philos Magaz Ser 7314(41):256–275

Shi Y, Krohling RA (2002) Co-evolutionary particle swarm optimization to solve min-max problems. Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002, vol 2, pp 1682–1687

Shimizu K, Aiyoshi E (1980) Necessary conditions for min-max problems and algorithms by a relaxation procedure. IEEE Trans Autom Control 25:62–66

Sinha A, Malo P, Deb K (2018) A review on bilevel optimization: from classical to evolutionary approaches and applications. IEEE Trans Evolut Comput 22:276–295

Sinha A, Lu Z, Deb K, Malo P (2020) Bilevel optimization based on iterative approximation of multiple mappings. J Heuristics 26:151–185

Tahk MJ, Sun BC (2000) Coevolutionary augmented lagrangian methods for constrained optimization. IEEE Trans Evolut Comput 4(2):114–124

Tamer Basar GJO (1982) Dynaminc noncooperative game theory. Academic Press, New York

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

Vasile M, Minisci E, Locatelli M (2011) An inflationary differential evolution algorithm for space trajectory optimization. IEEE Trans Evolut Comput 15:267–281

Wertz WJ, Larson JR (1999) Space mission analysis and design. Kluwer Academic Publishers, New York

Zhou A, Zhang Q (2010) A surrogate-assisted evolutionary algorithm for minimax optimization. 2010 IEEE World Congress on Computational Intelligence, WCCI 2010–2010 IEEE Congress on Evolutionary Computation. CEC 2010:6