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

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

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

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

In Section II, the problem formulation and the motivation for introducing this algorithm are presented along with a discussion of related literature, as well as an overview of the memetic algorithm. Section III introduces the experimental benchmark on which the algorithm is tested where the results are evaluated and compared to other optimisation methods. The paper is structured as follows. Section II presents the proposed memetic algorithm for the solution of constrained min-max problems. Section III introduces the experimental benchmark on which the algorithm is tested where the results are evaluated and compared to other optimisation methods.

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

I. INTRODUCTION

One aspect of Resilience Engineering is the design of systems that are robust against uncertainty of different nature. Different sources of uncertainty are possible: uncertainty on model definition, measurement noise, manufacturing and fabrication errors, human error, numerical error, etc [1]. All these forms of uncertainty are commonly classified into groups: *aleatory* and *epistemic*. The former group collects irreducible uncertainties while the latter group collects uncertainty due to lack of knowledge and/or subjective probability. Uncertainty can be reduced to aleatory as information increases. An engineering system can be optimised using a model for the quantity of interest over the space of uncertain variables is required to identify the worst-case scenario (or worst-case solution under uncertainty). An optimal design vector is then identified such that the worst-case value of the quantity of interest is minimised. In the most general case, both maximisation and minimisation are subject to strict feasibility constraints. In this case the goal is to find a design that is feasible for all possible values of the uncertain parameters.

In formulas this constrained min-max problem reads:

\[
\min_{\mathbf{d} \in \mathcal{D}} \max_{\mathbf{u} \in \mathcal{U}} f(\mathbf{d}, \mathbf{u})
\]

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

\[
\min_{\mathbf{d} \in \mathcal{D}} f(\mathbf{d})
\]

\[
\text{s.t.}
\]

\[
\max_{\mathbf{u} \in \mathcal{U}} f(\mathbf{d}, \mathbf{u})
\]

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

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

\[
\min_{\mathbf{d} \in \mathcal{D}} \max_{\mathbf{u} \in \mathcal{U}} f(\mathbf{d}, \mathbf{u})
\]

\[
\text{s.t.}
\]

\[
\max_{\mathbf{u} \in \mathcal{U}} C(\mathbf{d}, \mathbf{u}) \leq 0
\]

The paper is structured as follows. Section II presents the proposed memetic algorithm for the solution of constrained min-max problems. Section III introduces the experimental benchmark on which the algorithm is tested where the results...
are shown in Eq. (III-B). Section IV, finally, comments the performance of the algorithm at identifying feasible and robust solutions.

II. A MEMETIC MIN-MAX APPROACH

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

$$\min_{d \in D} \left( \max_{u_A \in A_u} f(d, u_A) \right).$$  \hspace{1cm} (5)

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

$$\max_{u \in U} f(d^*, u).$$ \hspace{1cm} (6)

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

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

The resulting algorithm proceeds as follows:

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

$$\min_{d \in D} \max_{u_A \in A_u} f(d, u) \quad s.t. \quad \max_{u \in U} C(d, u) \leq 0$$ \hspace{1cm} (8)

The solution $d^*$ is then saved in an archive $A_d$.

2) [Restoration] Given the solution of problem (8), $d^*$, solve the two maximisation problems:

$$\max_{u \in U} f(d^*, u) \quad s.t. \quad C(d^*, u) \leq 0$$ \hspace{1cm} (9)

$$\max_{u \in U} C_i(d^*, u) \quad \forall i \in [1, ..., nc]$$ \hspace{1cm} (10)

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

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

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

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

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

III. TESTING PROCEDURE

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

Algorithm 2 Testing Procedure

1: Set for each problem \( p \) the maximum number of function evaluation \( N_{\text{max}} \).
2: set \( j_s = 0; \)
3: Run \( n \) times with \( N_{\text{max}} \) the algorithm \( \text{MacMinMax} \) on each problem \( p; \)
4: for \( k = [1, 2, \ldots, n] \) do
5:   Compute \( \delta_p \) as in lines 5-6 of Algorithm 2
6:   Compute \( \nu_p \) as in lines 5-6 of Algorithm 2
7:   if \( \nu_p \leq 0 \) then \( j_s = j_s + 1 \)
8: end if
9: end for
10: \( SR = \frac{J_s}{n} \)

A. Benchmark

Objective functions from TC-1 to TC-6 are convex-concave test functions defined in Chapter 5 of [25] and used in [23]. Functions from TC-7 to TC-12 are selected from [16], [17], [26]. TC-13 is a modification of the Rastrigin function where half of the variables are design parameters and the others are epistemic uncertainties. The dimensions of TC from TC-1 to TC-12 range from \( dim_d = 1 \) and \( dim_u = 1 \) up to \( dim_d = 4 \) and \( dim_u = 3 \); dimensions of TC-13 can go from 1 to infinite and is here kept up to \( dim_d = 3 \) and \( dim_u = 3 \). Reference solutions for \( p_{1,1} \) and \( p_{1,2} \) are presented in TABLE II: for the unconstrained test cases from TC-1 to TC-12 they can be found in [16], [17], [23], [25], [26] while for the unconstrained TC-13 the min-max can be easily evaluated from the minimum of the Rastrigin function. Design \( d \) and uncertainty \( u \) parameters, indeed, are not coupled and then the solution of \( \text{min}_{d} \text{max}_{u} TC-13(d, u) \) can be found decoupling the problem: for \( dim_d = dim_u = 1 \), the solution is shown in
Fig. 6 where \( d = 0 \) and \( u \in [-5, 5]^2 \) is given by the maximisation of \( TC-14(d, u)_{d=0} = C + u^2 - A \cos(2\pi u) \).

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

Functions \( TCC-3 \) and \( TCC-5 \), instead, depend on both design \( d \) and uncertain \( u \). Also they force the solution to move from the unconstrained one in TABLE II to TABLE IV. \( TCCs \) functions are scalable and are translated by means of
TABLE III
TEST CASES FOR THE CONSTRAINT FUNCTIONS C

<table>
<thead>
<tr>
<th>name constraint function TCC</th>
<th>C(d) = [ \begin{cases} \sum_{i=1}^{n-1} \sqrt{A_i^2 - (d_i - x_0)^2} + B_{1,i} - d_{i+1} &amp; \text{if } d_1 \geq x_0 - R \ \sum_{i=1}^{n-1} \sqrt{A_i^2 - (d_i - x_0 + 2A_1)^2} + B_{1,i} - d_{i+1} &amp; \text{else} \end{cases} ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCC-1</td>
<td>( C(d_1) = A_2 \cdot n + \sum(d_i^2 - A_2 \cdot \cos(2\pi d_i)) - B_2 )</td>
</tr>
<tr>
<td>TCC-2</td>
<td>( C(d, u) = \sum_{i=1}^{N} \left[ \max(0, d_i + u_i - A_7) \right] )</td>
</tr>
<tr>
<td>TCC-3</td>
<td>( C(d, u) = \begin{cases} 1 &amp; \text{if any }</td>
</tr>
<tr>
<td>TCC-4</td>
<td>( C(d, u) = \begin{cases} 1 &amp; \text{if any }</td>
</tr>
<tr>
<td>TCC-5</td>
<td>( C(d, u) = \begin{cases} -1 &amp; \text{if }</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>TC</th>
<th>TCC</th>
<th>Reference d</th>
<th>Reference u</th>
<th>dim(_{d,u}) = 1</th>
<th>dim(_{d,u}) = 2</th>
<th>dim(_{d,u}) = 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC-13</td>
<td>3</td>
<td>[-4.140 ... -4.140]</td>
<td>[-4.5229 ... -4.5229]</td>
<td>56.118</td>
<td>117.2373</td>
<td>178.3559</td>
</tr>
<tr>
<td>TC-13</td>
<td>5</td>
<td>[-0.9950 ... -0.9950]</td>
<td>[-4.5229 ... ± 4.5229]</td>
<td>36.3482</td>
<td>77.6965</td>
<td>119.0447</td>
</tr>
</tbody>
</table>

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

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

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

The optimiser used for both optimisation and restoration loops is MP-AIDEA [28]. MP-AIDEA has been used with one and three populations. Parameters have been set as follows: the number of agents for each population is equal to the number of variables (\( dim_d \) in the outer loop and \( dim_u \) in the inner loop); the maximum number of local restart is \( \text{un} = 20 \) for one population and it is adaptive for 3 populations; the crossover probability, \( CR \), and the differential weight, \( F \) are self adapted; the size of the convergence box is \( \rho = 0.25 \);

the distance from the cluster centres for the global restart is \( \delta_{global} = 0.1 \) and the dimension of the bubble for the local restart, if not adapted, is \( \delta_{local} = 0.1 \).

B. Test Results

The convergence of MacMinMax has been tested with different values of \( N_{\text{maxfeval}} \) that results from the combination of the number of function evaluation in the inner loop \( N_{f}^{in} \) and in the outer loop \( N_{f}^{out} \). Different tolerances \( tol_f \) and \( tol_u \), also, have been considered. TABLE IX reports some results with 1 and 3 populations. Fig.5 shows an example of success rate for an increasing number of function calls, for the case of problem \( p_{6,1} \). In the figure, for each \( N_{\text{maxfeval}} \), the best SHs - over all combinations of \( N_{f}^{in} \) and \( N_{f}^{out} \) - is evaluated with the three criteria in line 8 of Algorithm 2.
usually the solution is underestimated. On the other hand for a low \( N_{f}^{\text{in}} \), the algorithm stagnates at different local minima - if \( TC-i \) and/or \( TCC-j \) are multimodal - and it does not converge, or converges slowly, even with an high \( N_{f}^{\text{in}} \).

### IV. Conclusion

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

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

### TABLE V

<table>
<thead>
<tr>
<th>( \text{TCC-1} )</th>
<th>#feval &amp; SR( \text{TFC} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( tol_f = 0.001 )</td>
<td>( tol_f = 0.01 )</td>
</tr>
<tr>
<td>( tol_u = 0.01 )</td>
<td>( tol_u = 0.1 )</td>
</tr>
<tr>
<td>1</td>
<td>3000 &amp; 95</td>
</tr>
<tr>
<td>2</td>
<td>30000 &amp; 97</td>
</tr>
<tr>
<td>3</td>
<td>30000 &amp; 88</td>
</tr>
<tr>
<td>4</td>
<td>20000 &amp; 99</td>
</tr>
<tr>
<td>5</td>
<td>10000 &amp; 99</td>
</tr>
<tr>
<td>6</td>
<td>50000 &amp; 90</td>
</tr>
<tr>
<td>11</td>
<td>1500 &amp; 97</td>
</tr>
<tr>
<td>12</td>
<td>5500 &amp; 92</td>
</tr>
</tbody>
</table>

### TABLE VI

<table>
<thead>
<tr>
<th>( \text{TCC-1 and TCC-13} )</th>
<th>#feval &amp; SR( \text{TFC} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( tol_f = 0.001 )</td>
<td>( tol_f = 0.01 )</td>
</tr>
<tr>
<td>( tol_u = 0.01 )</td>
<td>( tol_u = 0.1 )</td>
</tr>
<tr>
<td>1</td>
<td>5000 &amp; 100</td>
</tr>
<tr>
<td>2</td>
<td>20000 &amp; 94</td>
</tr>
<tr>
<td>3</td>
<td>20000 &amp; 94</td>
</tr>
</tbody>
</table>

In TABLES V, VI, VII, VIII, IX, XI and XII it is shown that the higher both \( N_{f}^{\text{in}} \) and \( N_{f}^{\text{out}} \) are, the more accurate are the maxima and minima evaluated at each optimisation-restoration loop but the overall cost of the min-max algorithms increases accordingly. If the total cost is kept fixed an increase of \( N_{f}^{\text{in}} \) and \( N_{f}^{\text{out}} \) does not necessarily lead to an improvement of the solution. In Fig. 7, for example, for \( N_{f}^{\text{in}} = N_{f}^{\text{out}} = 200 \) the algorithm has a poor success rate for all function evaluations. When the number of function evaluations of the inner and outer level is increased to 300, the success rate improves rather quickly. When the number of function evaluations of the inner and outer loops is increased even further, the success rate progressively decreases, for a constant number of total function evaluations. By setting \( N_{f}^{\text{in}} \) too small, the convergence is not guaranteed even with a very high \( N_{f}^{\text{out}} \).
TABLE VII
RESULTS FOR TCC-2

<table>
<thead>
<tr>
<th>TC</th>
<th>#f</th>
<th>#f</th>
<th>#f</th>
<th>#f</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>tol = 0.001</td>
<td>tol = 0.01</td>
<td>tol = 0.1</td>
<td>tol = 0.1</td>
</tr>
<tr>
<td>1</td>
<td>15000</td>
<td>95</td>
<td>15000</td>
<td>96</td>
</tr>
<tr>
<td>2</td>
<td>30000</td>
<td>98</td>
<td>30000</td>
<td>98</td>
</tr>
<tr>
<td>3</td>
<td>90000</td>
<td>71</td>
<td>90000</td>
<td>71</td>
</tr>
<tr>
<td>4</td>
<td>30000</td>
<td>94</td>
<td>30000</td>
<td>94</td>
</tr>
<tr>
<td>5</td>
<td>30000</td>
<td>100</td>
<td>30000</td>
<td>100</td>
</tr>
<tr>
<td>6</td>
<td>100000</td>
<td>13</td>
<td>100000</td>
<td>13</td>
</tr>
<tr>
<td>11</td>
<td>2500</td>
<td>94</td>
<td>2500</td>
<td>97</td>
</tr>
<tr>
<td>12</td>
<td>7000</td>
<td>78</td>
<td>7000</td>
<td>87</td>
</tr>
</tbody>
</table>

TABLE VIII
RESULTS FOR TCC-2 AND TCC-3

<table>
<thead>
<tr>
<th>dim</th>
<th>#f</th>
<th>#f</th>
<th>#f</th>
<th>#f</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>tol = 0.001</td>
<td>tol = 0.01</td>
<td>tol = 0.1</td>
<td>tol = 0.1</td>
</tr>
<tr>
<td>1</td>
<td>1000</td>
<td>99</td>
<td>1000</td>
<td>98</td>
</tr>
<tr>
<td>2</td>
<td>1000</td>
<td>95</td>
<td>1000</td>
<td>96</td>
</tr>
<tr>
<td>3</td>
<td>9000</td>
<td>92</td>
<td>9000</td>
<td>93</td>
</tr>
<tr>
<td></td>
<td>tol = 0.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1000</td>
<td>99</td>
<td>1000</td>
<td>98</td>
</tr>
<tr>
<td>2</td>
<td>1000</td>
<td>95</td>
<td>1000</td>
<td>96</td>
</tr>
<tr>
<td>3</td>
<td>9000</td>
<td>92</td>
<td>9000</td>
<td>93</td>
</tr>
</tbody>
</table>

TABLE IX
RESULTS FOR TCC-3 AND TCC-4

<table>
<thead>
<tr>
<th>dim</th>
<th>#f</th>
<th>#f</th>
<th>#f</th>
<th>#f</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>tol = 0.001</td>
<td>tol = 0.01</td>
<td>tol = 0.1</td>
<td>tol = 0.1</td>
</tr>
<tr>
<td></td>
<td>tol = 0.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>30000</td>
<td>68</td>
<td>30000</td>
<td>70</td>
</tr>
<tr>
<td>2</td>
<td>80000</td>
<td>30</td>
<td>80000</td>
<td>32</td>
</tr>
<tr>
<td>3</td>
<td>70000</td>
<td>3</td>
<td>90000</td>
<td>24</td>
</tr>
</tbody>
</table>

REFERENCES


