 CMES, vol.18, no.3, pp.183-221, 2007

Weight Optimization of Skeletal Structures with Multi-Point Simulated Annealing

L. Lamberti[1,2] **and C. Pappalettere**[1,3]

Abstract: This paper presents a novel optimization algorithm for minimizing weight of skeletal structures. The algorithm–denoted as MPISA (Multi Point Improved Simulated Annealing)–utilizes a multi-level simulated annealing scheme where different candidate designs are compared simultaneously. This is done in purpose to increase computational efficiency and to minimize the number of exact structural analyses.
MPISA is tested in three complicated design problems of skeletal structures: (i) sizing optimization of a planar bar truss under five independent loading conditions including 200 design variables; (ii) sizing-configuration optimization of a cantilevered bar truss including 81 design variables; (iii) sizing-configuration optimization of a frame structure including 84 design variables.
The new algorithm is compared to another multi-level simulated annealing algorithm and to gradient based optimizers recently presented in literature or included in commercial software.
Numerical results demonstrate the very high efficiency of MPISA which achieved weight savings ranging between 250 and 350 kg in all test problems.

Keyword: Simulated annealing (SA), multi-point random search, structural optimization, skeletal structures.

1 Introduction

Global optimization of structures is a complicated task which may often entail a considerable number of exact structural analyses. Gradient based optimizers (GBO) can find optimum designs rather quickly but there is no guarantee they reached the true global optimum. Global optimization may be attempted by performing multi-start gradient based optimizations from different initial designs chosen randomly. This is done, for instance, in design of stiffened panels under combined loads, subjected to stress, displacement and buckling constraints (Bushnell, 1996). Advantages and limitations of this approach applied to the preliminary design of reusable launch vehicle tanks under multiple loading conditions are extensively discussed in Lamberti et al. (2003).

Non-gradient based optimizers (NGBO) search the optimum in regions of design space that are significantly larger than in the GBO case. The most straightforward way to avoid the recursive gradient evaluations entailed by sensitivity analysis is to use heuristic criteria that add (remove) material to those parts of the structure where stress values go beyond (below) the allowable limit. This approach, based on the idea of Evolutionary Structural Optimization (Xie and Steven, 1997), has been followed, for instance, by Tapp et al. (2004) in topology optimization of sandwich structures with composite face-sheets. Lightweight structures could be designed yet with a reasonably low computational effort.

The other approach followed in NGBO is to generate a certain number of random trial designs in order to cover each possible combination of optimization variables that can be considered in the design space. A variety of optimization algorithms inspired to biology, evolution theory and physics have been proposed in literature. Genetic Algorithms (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) and Simulated Annealing (SA) are certainly the most popular methods amongst this class of optimiza-

[1] Dipartimento di Ingegneria Meccanica e Gestionale, Politecnico di Bari, Viale Japigia 182–70126–BARI–Italy.
[2] Email: lamberti@poliba.it
[3] Email: c.pappalettere@poliba.it

tion techniques.

Genetic Algorithms (Goldberg, 1989) rely on concepts of genetics and Darwinian survival of the fittest. The design is represented by a combinatorial set, called chromosomes and each component of the set is called a gene. Chromosomes and genes are generated by three predefined rules of evolution: reproduction (rebirth or duplication), cross-over (generation) and mutation. Thus, one must define the population size, rate of reproduction, cross-over pattern, percentage of mutation, etc. After many generations, the design represented by the most popular chromosomes indicates the optimal design. Two recent examples of application of GA-related techniques in structural optimization problems are reported in Baumann and Kost (2005) and Fedelinski and Gorski (2006) who, respectively, determined the optimum shape of grid structures and stiffened panels under static and dynamic loads.

The ACO (Dorigo and Stuzle, 2004) and PSO (Clerc, 2006) algorithms are specially suited for problems where the location and value of global optimum may change with time. These methods attempt to mimic interactions between individuals of insect colonies or bird/fish swarms. If one individual sees a desirable path to go (for food, protection, etc.), the rest of colony/swarm will be able to follow quickly even if they are not in a direct connection with the leading individuals. On the other hand, in order to facilitate the exploration of the search space, each insect/particle should have a certain level of "craziness" or randomness in their movement. Therefore, the optimization problem consists in determining the best path or the optimal positions of the colony/swarm ensemble.

An ACO-based algorithm has been applied successfully by Aymerich and Serra (2006) to the determination of the optimal stacking sequence in composite laminates subjected to different combinations of in-plane and out-of-plane loads and in presence of different constraint conditions. Schutte et al. (2005) have proven that the PSO technique can solve difficult biomechanical problems such as identification of human movements or estimation of muscular actions or/and other internal forces where design variables may have very different length scales or units.

It should be noted that the canonical forms of both ACO and PSO algorithms include a considerable level of heuristics. Therefore, the performance of these algorithms may be very sensitive to modifications introduced *ad hoc* to deal with the specific optimization problem that has to be solved (see discussion reported in Engelbrecht, 2005).

Simulated annealing (Kirkpatrick et al., 1983; Haftka and Gurdal, 1992; Rao, 1996) has been developed from the statistical thermodynamics to simulate the behavior of the atomic arrangements in liquid or solid material during the annealing process. Lowering the temperature of the melted material, the material reaches the lowest energy level (globally stable condition). SA includes a rather simple optimization strategy. A trial design is randomly generated and problem functions are evaluated at that point. If the trial point is infeasible, it is rejected and a new trial point is evaluated. If the trial point is feasible and the cost function is smaller than the current best record, then the point is accepted and the best record is updated. If the trial point is feasible but the cost function is higher than the best value, then the point is accepted or rejected based on a probabilistic criterion which estimates if design may improve in the next function evaluations. In order to compute probability, a parameter called the temperature is utilized. In the optimization problem, temperature can be a target value (estimated) for the cost function corresponding to a global minimizer. Initially, a larger target value is selected. As the trials progress, the target value is reduced based on a cooling schedule. The acceptance probability steadily decreases to zero as the temperature is reduced.

SA has been widely utilized in structural optimization problems because of its inherent simplicity and ability to find the global optimum even if there are many design variables. However, the basic SA algorithm has often been modified in order to improve convergence behavior and reduce the number of exact structural analyses. For instance, Shim and Manoochehri (1997) solved non-linear shape optimization problems by con-

ducting the annealing search on the approximate problem formed through linearization of stress constraints. To ensure design feasibility, a correction factor adjusted stress terms based on the maximum ratio of the change in the linearized stress to the change in the actual stress values.

Pantelides and Tzan (2000) used SA for optimizing structural systems subjected to dynamic loads. They included in the annealing procedure a sensitivity analysis in order to identify which design variables must be modified to decrease global displacements.

Researchers have proposed many different schemes for generating randomly new trial designs. Yu Chen and Su (2002) and Blachut (2003) pointed out that two different random generation mechanisms can be used in SA: *1-directional ("local") search* where design variables are perturbed one at a time; *multi-directional ("global") search* where all design variables are perturbed simultaneously. The latter strategy allows to increase the convergence speed but may fail in finding the global optimum. On the other hand, 1-directional search may result in too high computation times.

More recently, Luo and Tang (2005) have employed a neighborhood search function for producing tentative solutions by using the mutation strategy of GA as reference. Erdal and Sonmez (2005) have proposed an SA algorithm where a set of current configurations is used rather than just one trial design. However, their algorithm included too many heuristics. Finally, Higginson et al. (2005) have coded a parallel SA algorithm where each different size of the neighborhood search radius - set randomly - is analyzed by a single processor.

Generally speaking, a good compromise between global optimization capability and computational speed is an algorithm able to explore large fractions of design space using also gradient information in order to speed up the design process. Genovese et al. (2005) demonstrated the validity of this approach in structural design and reverse engineering problems by implementing a multi-level optimization algorithm based on Simulated Annealing. The ISA - *Improved Simulated Annealing* - algorithm is an advanced optimization code integrating different search strategies into a multi-level annealing process. Global search is performed when the nominal design at the beginning of the current annealing cycle is feasible. Trial designs lie on descent directions. Local search is performed only when global search failed or intermediate designs ended up infeasible. Superiority of ISA over classical annealing algorithms comes from the fact that ISA always attempts to generate trial designs which may guarantee design improvements or at least limit oscillations in cost. ISA was successfully applied in sizing and configuration structural optimization problems with up to 200 design variables and 3500 non-linear constraints. Remarkably, ISA was averagely twice as fast as classical annealing and required less structural analyses.

The structure of ISA is now briefly recalled. If the design at the beginning of a cooling cycle is feasible, a new trial design point P_{TR} is defined by perturbing simultaneously all optimization variables. Let $\boldsymbol{X}_{OPT}$ be the current best record corresponding to the design point P_{OPT}. The new trial search direction $\boldsymbol{S}_{TR}$ is limited in the design space by points P_{OPT} and P_{TR}: the components of $\boldsymbol{S}_{TR}$ are the perturbations taken for each design variable. Since the total change in cost is the sum of cost changes yield by perturbing design variables one at a time, ISA defines the descent direction $\boldsymbol{S}_{TR}$ by imposing the condition $\boldsymbol{S}_{TR}^T \nabla W(\boldsymbol{X}_{OPT}) < 0$. This task involves gradient evaluations for the cost function.

If the optimizer enters in an infeasible region, design is perturbed by taking movements along directions where constraint violation may be reduced. In order to steer the design back to a feasible domain, constraint functions are linearized and trial designs are evaluated in the approximate model. The trial design that violates linearized constraints the least is taken as the starting point for a new search. This strategy reduced computational cost of optimization in case of infeasible trial points. Reliability and accuracy of the approximate model are ensured by a trust region model.

Improvement routines using quadratic approxi-

mations of cost function and constraints are activated if the trial point is better than the current best record but violates constraints. Finally, the cooling schedule changes adaptively in the optimization process based on the improvement in design achieved in the current annealing cycle.

In spite of its advanced formulation and the proven numerical efficiency, ISA has two major limitations. In the first place, each new gradient evaluation leads to defining only one new descent direction $\boldsymbol{S}_{TR}$ and, hence, only one new trial design. Furthermore, no information are available on whether $\boldsymbol{S}_{TR}$ is the descent direction for which the improvement in cost is actually the largest.

In order to overcome these limitations, ISA is re-formulated more rationally in this paper. The new optimization algorithm MPISA - where the acronym stands for *Multi Point Improved Simulated Annealing* - maintains the same overall architecture of ISA and is yet based on the combination of multidimensional and one-dimensional annealing search. However, MPISA now builds a search domain Ω, centered about each current best record, including a set of descent directions and not only one $\boldsymbol{S}_{TR}$ vector. Step size along each descent direction is adjusted by means of a trust region scheme. Non-descent directions are also included in the Ω domain provided that they satisfy a probabilistic criterion of acceptance which might yield later effective improvements in design.

Then, MPISA searches for the descent direction $\boldsymbol{S}_{TR}$ which actually yields the largest improvement or the minimum increase in cost. Furthermore, line search is performed in order to determine the optimal step size along each direction included in the Ω domain.

The quadratic model included in the improvement routine originally implemented in ISA is now replaced by a cubic approximation which does not require any constraint linearizations. This strategy further reduces the number of exact structural analyses.

Multi-point search strategy is activated in MPISA also in case of infeasible intermediate designs. This allows to approach more quickly the boundaries of the constraint domain with respect to the original algorithm ISA.

Finally, MPISA includes a strategy for handling constraint domain non-convexity. This is very important when local annealing search has to be performed.

These new features implemented in MPISA certainly augment the potentiality of simulated annealing with respect to the other NGBO algorithms - GA, ACO and PSO - discussed previously. In fact, the present code now deals with a "pool" of candidate designs rather than with a single trial design like it usually occurs in classical SA and even happened in ISA. Each candidate design is connected to the set ("pool") of neighboring designs lying on the $\boldsymbol{S}_{TR}$ directions included the Ω domain.

Therefore, MPISA becomes conceptually very similar to the biologically inspired optimization algorithms - GA, ACO and PSO - that are all based on the existence of some relationship between parent and child (GA) or between single individuals (leaders) and other members of the colony/swarm ensemble (ACO, PSO). However, MPISA has a definite strength-point in the fact that biologically inspired algorithms include a considerable level of heuristics which may strongly affect their performance.

The MPISA code has successfully been tested in three weight minimization problems of skeletal structures: (i) sizing optimization of a planar 200 bar truss including 200 design variables; (ii) combined sizing-configuration optimization of a 45-element cantilevered bar truss including 81 design variables; (iii) combined sizing- configuration optimization of a frame comprised of 45 I-beam elements including 84 design variables. Constraints on cross section geometry, nodal displacements, element stress and buckling are considered.

Remarkably, weight savings between 250 and 350 kg with respect to ISA and other referenced algorithms have been achieved in all test cases.

The present paper is organized as follows. After the Introduction section, MPISA is described in Section 2 providing a pseudo-code of the new optimization algorithm. Section 3 presents the op-

timization problems chosen as test cases. Results of the optimization runs are discussed in Section 4. Finally, Section 5 summarizes the work done and the main findings of this study.

2 The MPISA Optimization Algorithm

The non-linear optimization problem can be formulated as:

$$\begin{cases} \min W(x_1,x_2,\ldots,x_N) \\ G_k(x_1,x_2,\ldots,x_N) \le 0 \\ x_j^l \le x_j \le x_j^u \end{cases} \quad \begin{cases} j=1,\ldots,N \\ k=1,\ldots,NC \end{cases} \tag{1}$$

where:

- $(x_1,x_2,\ldots,x_N)$ are the N design variables;
- $W(x_1,x_2,\ldots,x_N)$ is the objective function;
- $G_k(x_1,x_2,\ldots,x_N)$ are the NC inequality constraint functions;
- x_j^l and x_j^u are the lower and upper bounds of the j^{th} design variable.

The pseudo code of MPISA is now provided in order to make potential users able to code the new algorithm on computers. A detailed flow chart of the new optimization algorithm is shown in Fig. 1.

Step 1. Start the optimization process. Set initial design and temperature.

Choose the initial design vector $\boldsymbol{X}_o(x_{1,o},x_{2,o},\ldots,x_{N,o})$ and set it as the current best record $\boldsymbol{X}_{OPT}$. Define the corresponding point P_{OPT} in the design space. Say W_{OPT} the cost function value computed at P_{OPT}.

In general, the initial design $\boldsymbol{X}_o$ should be feasible and far enough from constraint domain boundaries in order to explore a zone of design space approximately centered about P_{OPT} with no risk to generate infeasible points that might bias the optimum design search since the very beginning of the optimization process. This strategy serves also to reduce the number of constraint evaluations eventually done in the infeasible region when design variables are perturbed one by one.

Set the K counter of cooling cycles as $K=1$ and choose the limit number of cooling cycles as K_{LIM}=100.

Set the I_{GLOB} counter of the global annealing cycles eventually performed by MPISA in a cooling cycle as I_{GLOB}=0. Let I_{LIM}=5 be the limit number of global cycles.

If the optimization problem is constrained and the starting design is feasible, set the initial temperature T_o as about 10% of the initial cost. If there are "soft" non-linear constraints or the optimization problem is unconstrained, use a very large value of T_o. These two strategies are practically equivalent because the cost of a "very feasible" initial design (i.e., located very far away from the constraint domain boundaries) is usually much more than 10 times as large as the final optimum cost.

If the starting point violates constraints, set the initial temperature T_o as 10 times the cost corresponding to the first feasible trial design $P_{FEA,1}$ generated by MPISA. The rationale behind this strategy is the following. In order to steer the design back to a feasible region, MPISA perturbs the nominal design by taking movements along a series of directions whose components depend on constraint gradients (see Step 8). If some of these directions are not descent, the cost $W_{FEA,1}$ computed at $P_{FEA,1}$ will be larger than the cost W_o computed at the starting point. In such a case, MPISA will try to reduce $W_{FEA,1}$ in order to minimize the cost function $W(\boldsymbol{X})$. If $P_{FEA,1}$ is pretty far away from the constraint domain boundary, the corresponding cost $W_{FEA,1}$ will be much higher than the final optimum cost and hence MPISA will easily reduce $W(\boldsymbol{X})$ by performing global annealing cycles (see Step 3). Conversely, if the $P_{FEA,1}$ point is close to constraint boundaries, the cost $W_{FEA,1}$ may be comparable to the minimum cost. Hence, MPISA will have to evaluate an acceptance/rejection probability function in order not to get stuck in local minima. Since this is done in the very early stages of the optimization process, the temperature T_o should be high enough to ensure exploration of a sufficiently large zone of the design space.

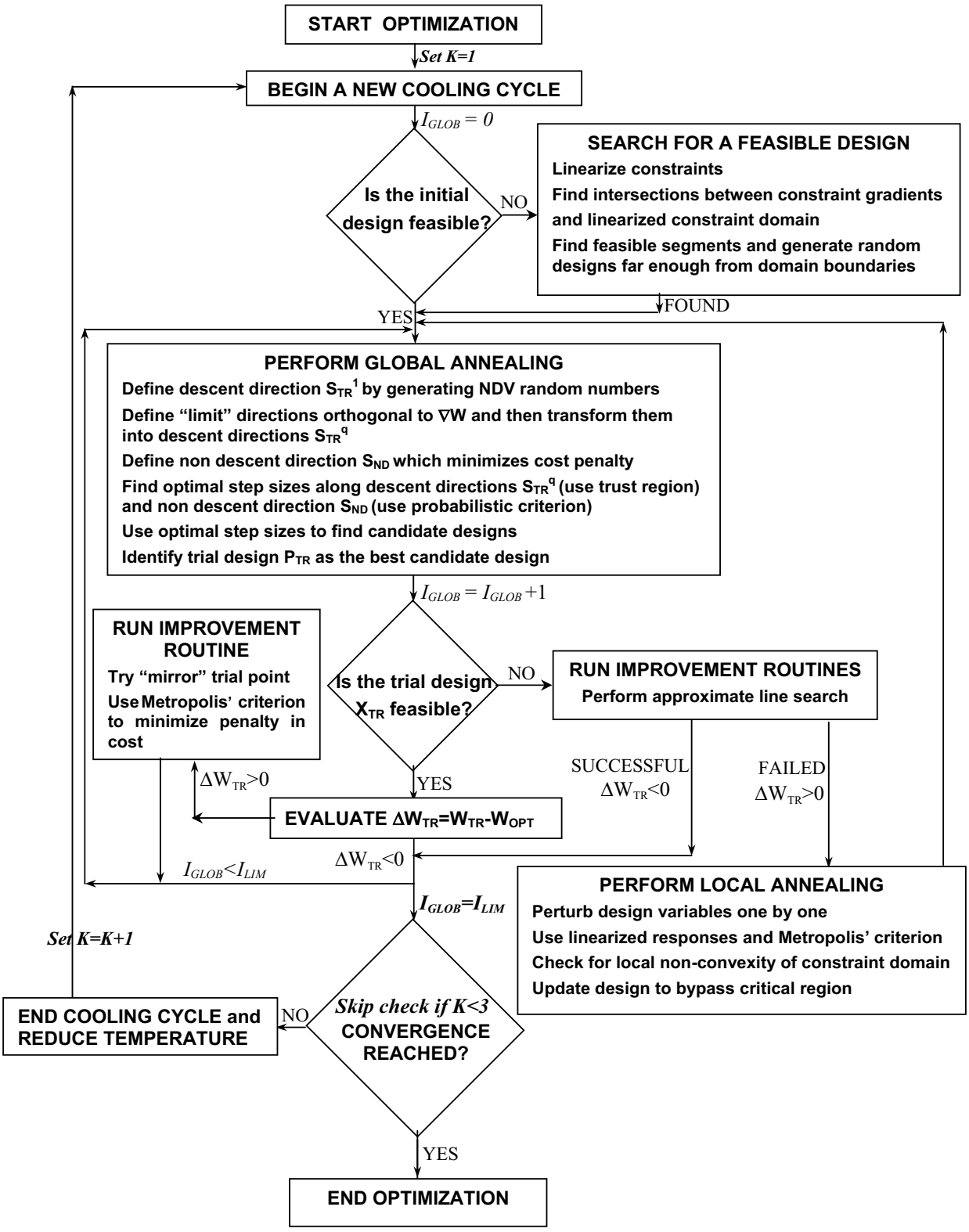


Figure 1: Flow chart of the MPISA algorithm

Step 2. Check on design feasibility.

Evaluate non-linear constraints at the current best record $\boldsymbol{X}_{OPT}$. If constraints are satisfied execute Step 3. Conversely, if constraints are violated execute Step 8.

Step 3. Global annealing by perturbing all design variables simultaneously. Definition of the search domain Ω.

Set $I_{GLOB} = I_{GLOB} + 1$. Evaluate the gradient $\overline{\nabla} W(\boldsymbol{X}_{OPT})$ of the cost function $W(\boldsymbol{X})$ at the P_{OPT} point. Perturb randomly each design variable $x_j (j = 1,\ldots,N)$ so that $(\partial W/\partial x_j)\Delta x_j < 0$. Each movement Δx_j is calculated as follows:

$$\begin{aligned} \partial W/\partial x_j > 0 &\Rightarrow \Delta x_j = -\left(x_j^u - x_j^l\right)\cdot N_{RND,j}\cdot \mu_j \\ \partial W/\partial x_j < 0 &\Rightarrow \Delta x_j = \left(x_j^u - x_j^l\right)\cdot N_{RND,j}\cdot \mu_j \end{aligned} \tag{2}$$

The $N_{RND,j}$ parameter in expression (2) is a random number chosen in the interval (0,1) for the j^{th} variable. Each weighting coefficient μ_j is defined as $\left|\partial W/\partial x_j\right| / \left|\left|\overline{\nabla} W(\boldsymbol{X}_{OPT})\right|\right|$. The purpose of μ_j is to adjust the Δx_j movement based on the contribution that the j^{th} sensitivity gives to the magnitude of cost function gradient. Design variables are changed following their order sequentially.

The Δx_jmovements are taken as the components of the first descent direction $\boldsymbol{S}_{TR}^1$ defined in the search process. That is, we define the vector $\boldsymbol{S}_{TR}^1(\Delta x_1, \Delta x_2, \ldots, \Delta x_N)$. Therefore, the $\boldsymbol{S}_{TR}^1$ vector is the diagonal of the Ω_1 domain defined by the Δx_j movements. Figure 2 shows the domain Ω_1 for the simple case of an optimization problem including two design variables.

It can be seen that each movement included in $\boldsymbol{S}_{TR}^1$yields an improvement in cost. Ideally, the $\boldsymbol{S}_{TR}^1$ direction should have coincided with the direction defined by the $-\overline{\nabla} W(\boldsymbol{X}_{OPT})$ vector in order to perturb design by moving along the steepest descent direction (i.e., along the negative gradient of cost function). However, the random nature of the generation process of the Δx_j movements may lead to significant misalignment between $-\overline{\nabla} W(\boldsymbol{X}_{OPT})$ and $\boldsymbol{S}_{TR}^1$. Furthermore, it

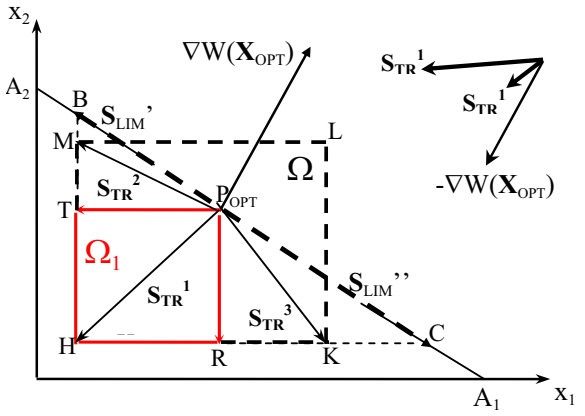


Figure 2: Definition of search domain Ω for feasible designs and special cases where $\boldsymbol{S}_{TR}^1$ may lead to marginal improvements in cost

may happen that the total step $\left|\left|\boldsymbol{S}_{TR}^1\right|\right|$ along $\boldsymbol{S}_{TR}^1$ is much smaller than $\left|\left|-\overline{\nabla} W(\boldsymbol{X}_{OPT})\right|\right|$ (see detail in the top right corner of Fig. 2).

In order to overcome these limitations inherently carried by the original code ISA developed by the present authors, MPISA now defines many other descent directions. Consequently, portions of design space are added to the Ω_1 domain so to extend the search region. Figure 2 shows that any descent direction has to lie below the $\overline{A_1A_2}$ segment which is orthogonal to the gradient vector $\overline{\nabla} W(\boldsymbol{X}_{OPT})$. Therefore, two "limit directions" $\boldsymbol{S}_{LIM}{}'$ and $\boldsymbol{S}_{LIM}{}''$ lying on the $\overline{A_1A_2}$ segment must be defined. Direction $\boldsymbol{S}_{LIM}{}'$–limited by points P_{OPT} and B–has one known component equal to Δx_1 and one unknown component Δx_2^{lim} in the x_2-direction. Direction $\boldsymbol{S}_{LIM}{}''$–limited by points P_{OPT} and C–has one known component equal to Δx_2 and one unknown component Δx_1^{lim} in the x_1-direction.

As is clear, $\boldsymbol{S}_{LIM}{}'$ and $\boldsymbol{S}_{LIM}{}''$ must satisfy the conditions of orthogonality stated as $\left(\boldsymbol{S}_{LIM}{}'\right)^T \left[\overline{\nabla} W(\boldsymbol{X}_{OPT})\right] = 0$ and $\left(\boldsymbol{S}_{LIM}{}''\right)^T \left[\overline{\nabla} W(\boldsymbol{X}_{OPT})\right] = 0$. Hence, unknown movements can easily be determined by solving

the following linear system:

$$\begin{cases} \Delta x_1 \dfrac{\partial W}{\partial x_1}(X_{OPT}) + \Delta x_2^{\lim} \dfrac{\partial W}{\partial x_2}(X_{OPT}) = 0 \\ \Delta x_1^{\lim} \dfrac{\partial W}{\partial x_1}(X_{OPT}) + \Delta x_2 \dfrac{\partial W}{\partial x_2}(X_{OPT}) = 0 \end{cases} \quad (3)$$

However, movements $\Delta x_1^{\lim}$ and $\Delta x_2^{\lim}$ must be compatible with design variable bounds. Therefore MPISA resets those movements for which such requirement is not satisfied. For instance, for the j^{th} variable, the corresponding movement $\Delta x_j^{\lim}$ is reset as $\left(x_{OPT}^j - x_j^l\right)$ or $\left(x_j^u - x_{OPT}^j\right)$, respectively, if $\Delta x_j^{\lim} < 0$ or $\Delta x_j^{lim} > 0$.

Movements $\Delta x_1^{\lim}$ and $\Delta x_2^{\lim}$ so determined have to be adjusted in order to transform the limit directions $\boldsymbol{S}_{LIM}{}'$ and $\boldsymbol{S}_{LIM}$" into the descent directions $\boldsymbol{S}_{TR}^2$ and $\boldsymbol{S}_{TR}^3$ for which it holds $(\boldsymbol{S}_{TR}^2)^T \left[\overline{\nabla} W(\boldsymbol{X}_{OPT})\right] < 0$ and $\left(\boldsymbol{S}_{TR}^3\right)^T \left[\overline{\nabla} W(\boldsymbol{X}_{OPT})\right] < 0$. Interestingly, the process of resetting too large movements to satisfy side constraints also leads to define descent directions.

Say Δx_1^{exp} and Δx_2^{exp} the movements which serve to expand the Ω_1 domain so to include as many descent directions as possible. These new movements are defined in MPISA using another set of random numbers ζ chosen in the interval (0,1). That is:

$$\begin{cases} \Delta x_1^{\exp} = \xi_1 \Delta x_1^{\lim} \\ \Delta x_2^{\exp} = \xi_2 \Delta x_2^{\lim} \end{cases} \quad (4)$$

In the case portrayed by Fig. 2, $\boldsymbol{S}_{LIM}{}'$ is transformed into the descent direction $\boldsymbol{S}_{TR}^2\left(\Delta x_1, \Delta x_2^{exp}\right)$ while $\boldsymbol{S}_{LIM}$" is transformed into the descent direction $\boldsymbol{S}_{TR}^3\left(\Delta x_1^{exp}, \Delta x_2\right)$. Once movements Δx_1^{exp} and Δx_2^{exp} have been defined, the position of the last vertex (L for the case of Fig. 2) of the search domain Ω also is univocally defined.

The process described above can be generalized for optimization problems with N design variables: N limit directions $\boldsymbol{S}_{LIM,r}$ $(r = 1,\ldots,N)$ are to be determined. Each direction $\boldsymbol{S}_{LIM,r}$ will include N_p known components - denoted as M_ℓ - and N-N_p unknown components - denoted as M_u. The former set of movements can be defined using Eq. (2) while the latter is the set of movements yet to be determined. Hence, MPISA solves a system of N linear equations each of which is derived by choosing a new combination of known and unknown movements. The rth equation included in the linear system is obtained by imposing the orthogonality condition $(\boldsymbol{S}_{LIM,r})^T \left[\overline{\nabla} W(\boldsymbol{X}_{OPT})\right] = 0$. Implementation tricks allow us to solve efficiently the linear system thus formed. For instance, each time a limit direction $\boldsymbol{S}_{LIM,r}$ has been completely defined, the components of the opposite direction -$\boldsymbol{S}_{LIM,r}$ are searched.

Say $\boldsymbol{M}_u$ the vector including the new movements $\Delta x_j^{\lim}$ derived from the conditions of orthogonality (for the case of Fig. 2, it is $\boldsymbol{M}_u\left(\overline{RC}, \overline{TB}\right)$. Each movement $\Delta x_j^{\lim}$ must be resized in order to define the set of descent directions $\boldsymbol{S}_{TR}$:

$$\Delta x_j^{\exp} = \xi_j \Delta x_j^{\lim} \quad (j = 1,\ldots,N) \qquad (4 \text{ mod.})$$

Each movement Δx_k^{exp} is stored as a component of the $\boldsymbol{M}_u^{exp}$ vector (for the case of Fig. 2, it occurs $\boldsymbol{M}_u^{exp}\left(\overline{RK}, \overline{TM}\right)$. Finally, limit directions $\boldsymbol{S}_{LIM,r}$ are replaced by the descent directions $\boldsymbol{S}_{TR}^q$ each of which includes some of the movements stored in $\boldsymbol{M}_u^{exp}$.

Say Δx_j^{fin} the movements associated with the descent directions. They may be in turn the Δx_j movements determined with Eq. (2) or the Δx_j^{exp} movements. As is clear, the former will be the case of a trial design lying on the first descent direction $\boldsymbol{S}_{TR}^1$ defined in Step 3.

It can be seen from Fig. 2 that the descent directions $\boldsymbol{S}_{TR}^1$, $\boldsymbol{S}_{TR}^2$ and $\boldsymbol{S}_{TR}^3$ are the diagonals of the search domain Ω while the last diagonal $\overline{P_{OPT}L}$ is not a descent direction. Therefore, in the case of N design variables, there will be $(2^N - 1)$ descent directions $\boldsymbol{S}_{TR}^q$ and 1 non-descent direction $\boldsymbol{S}_{ND}$.

Figure 2 clearly shows that the step size along the non-descent direction $\boldsymbol{S}_{ND}$ is automatically set once step sizes on each descent direction have been defined. Say Δx_j^{ND} $(j = 1,\ldots,N)$ the movements defining the non-descent direction $\boldsymbol{S}_{ND}$.

Step 4. Refinement of the search domain Ω with trust region scheme and probability criterion.

Step sizes along the $(2^N - 1)$ descent directions $\boldsymbol{S}_{TR}^q$ and the non-descent direction $\boldsymbol{S}_{ND}$ are ad-

justed in order to guarantee the highest probability of improving cost function in the current annealing cycle.

Each descent direction $\boldsymbol{S}_{TR}^{q}(q = 1,\ldots,2^N - 1)$ defined in Step 3 satisfies the $(\boldsymbol{S}_{TR}^{q})^T \left[\overline{\nabla} W(\boldsymbol{X}_{OPT})\right] < 0$ inequality where the internal product represents the reduction in cost which would be achieved by moving along the $\boldsymbol{S}_{TR}^{q}$ direction by the step $||\boldsymbol{S}_{TR}^{q}||$. However, if the cost function is non-linear the actual change in cost ΔW includes also the contributions of 2^{nd} order sensitivities (stored by the Hessian matrix $[H]$) and of higher order terms. Therefore, the actual change in cost ΔW can be expressed by Eq. (5):

$$\Delta W = \left(\boldsymbol{S}_{TR}^{q}\right)^T \left[\overline{\nabla} W\left(\boldsymbol{X}_{OPT}\right)\right] + \left(\boldsymbol{S}_{TR}^{q}\right)^T \left[H(\boldsymbol{X}_{OPT})\right] \left(\boldsymbol{S}_{TR}^{q}\right)/2 + \text{H.O.T.} \quad (5)$$

The above expression can be rewritten for a step size δS_{TR}^{q} taken along the corresponding descent direction $\boldsymbol{S}_{TR}^{q}$. It follows:

$$\Delta\left(W\delta\boldsymbol{S}_{TR}^{q}\right) = \left(\delta\boldsymbol{S}_{TR}^{q}\right)^T \left[\overline{\nabla} W(\boldsymbol{X}_{OPT})\right] + \left(\delta\boldsymbol{S}_{TR}^{q}\right)^T \left[H(\boldsymbol{X}_{OPT})\right] \left(\delta\boldsymbol{S}_{TR}^{q}\right)/2 + \text{H.O.T.} \quad (6)$$

where $\delta S_{TR}^{q} = \left|\left|\delta\boldsymbol{S}_{TR}^{q}\right|\right|$.

For a skeletal structure to be designed with respect to sizing variables such as the cross sectional area of each member or the dimensions of different segments included in the transverse section, cost function is linear with respect to the optimization variables. Conversely, the cost function is non-linear with respect to configuration variables defining position of nodes. In the latter case, the effective reduction in cost achieved moving along the q^{th} descent direction by the δS_{TR}^{q} step may be less than that expected.

Since evaluating exactly $\Delta W\left(\delta S_{TR}^{q}\right)$ may be computationally too expensive, it is preferable to have a rather quick way to determine the change in cost, especially in optimization algorithms such as simulated annealing that require a large number of cost function evaluations. As is clear, the step size δS_{TR}^{q} on each descent direction must be such to ensure a good correspondence between the "linearized" change in cost $(\delta\boldsymbol{S}_{TR}^{q})^T[\nabla W(\boldsymbol{X}_{OPT})]$ and the actual ΔW. Therefore, MPISA implements the following trust region scheme ($q = 1,\ldots,2^N - 1$):

$$\frac{W(X_{OPT}) - W\left(X_{OPT} + \delta S_{TR}^{q}\right)}{\left(\delta S_{TR}^{q}\right)^T \overline{\nabla} W(X_{OPT})} \geq 0.75 \quad (7)$$

where δS_{TR}^{q} is the unknown step to be taken along the q^{th} descent direction in order to fall within the trust region.

As is known, trust region methods adaptively manage the amount of movement allowed in design space when the optimization is based on approximate models. A well established procedure consists in defining a reliability index γ which monitors how well the approximate model predicts the reduction in cost: the reliability index is the ratio of the actual change in the function to the change predicted by the approximation. After each optimization iteration, the trust region radius Δ is reduced or increased if the magnitude of the γ parameter is small or large, respectively; the Δ radius usually stays the same if $0.25 < \gamma < 0.75$. Therefore, 0.75 can be assumed as the γ threshold value at which movements may be accepted with no need for any modifications. More details on trust region methods are given in Alexandrov et al. (1998), and Wujek and Renaud (1998).

It appears that the rationale behind Eq. (7) is to reverse the traditional use of the trust region technique in order to adapt to the random design process entailed by simulated annealing. In fact, the reliability index, which is nothing but the ratio in the LHS of the trust region equation (7), is no longer computed to check the new intermediate design obtained at the end of an optimization cycle. Instead, Eq. (7) now includes an unknown step δS_{TR}^{q} to be determined for each descent direction in order to define a candidate design such that the reliability index is higher than the fixed threshold value. The kind of "approximate model" introduced in the MPISA algorithm derives from the fact that the linearized increase in cost is used for defining the possible search domain Ω. Therefore, the trust region relationship (7) has to be used for accepting or rejecting potential designs. In the former case, the step sizes along the descent di-

rection $\boldsymbol{S}_{TR}^{q}$ stays the same. In the latter case, the step size has to be reduced in order to make the current candidate design fall within the trust region.

Descent directions are hence re-defined as follows (8):

$$\boldsymbol{S}_{TR}^{q,final} = \max\left[1, \left(\delta S_{TR}^{q} / \left|\left|\boldsymbol{S}_{TR}^{q}\right|\right|\right)\right] \boldsymbol{S}_{TR}^{q} \quad (q = 1,\ldots,2^N - 1) \quad (8)$$

That is, the step size is always reduced and stays the same only if the movement $\left|\left|\boldsymbol{S}_{TR}^{q}\right|\right|$ originally defined in Step 3 falls within the trust region.

Let ω_q be the scaling factor (<1) eventually used for resizing the step size along $\boldsymbol{S}_{TR}^{q}$. As is clear, $\omega_q = 1$ for those directions whose step size has not been modified. Finally, say $\omega_{\min}$ the smallest scaling factor used in the step resizing process.

For each descent direction, variable movements can be redefined as:

$$\Delta x_j^{q,\text{final}} = \omega_q \cdot \Delta x_j^{q,\text{fin}} \quad (j = 1,\ldots,N) \quad (9)$$

The non-descent direction $\boldsymbol{S}_{ND}$ might play a role if the search along descent directions failed in finding a better design (see Step 6-C). As is known, the Metropolis' acceptance/rejection criterion is used in simulated annealing for estimating if trial designs at which the cost function increased can later lead to improvements in cost. Therefore, MPISA chooses the non-descent direction $\boldsymbol{S}_{ND}$ so to satisfy the Metropolis' criterion. Should this be the case, the points lying on $\boldsymbol{S}_{ND}$ could be utilized to continue the optimization process when the current annealing cycle did not result in cost improvements. An *ad hoc* probability function $P(\Delta W_{ND})$ is hence defined as:

$$P(\Delta W_{ND}) = e^{-\left[1 - \dfrac{(\delta S_{ND})^T \overline{\nabla} W(X_{OPT})}{\Delta W_{NL}(\delta S_{ND})}\right] \dfrac{1}{T_K}} \quad (10)$$

where $\Delta W_{ND} = W_{ND} - W_{OPT}$ is the difference in cost function values respectively evaluated at P_{ND} and P_{OPT}.

This probability function increases as the approximate value of ΔW_{ND} is close to the actual change in cost and accounts also for the effect of temperature T_K set in the current annealing cycle.

The $\boldsymbol{S}_{ND}$ direction is kept or changed in view of the following acceptance/rejection criterion:

$$\begin{aligned} &P(\Delta W_{ND}) > \max(\lambda_{ND}, \rho_{ND}) \Rightarrow \text{Keep } \boldsymbol{S}_{ND} \\ &P(\Delta W_{ND}) < \max(\lambda_{ND}, \rho_{ND}) \Rightarrow \text{Change } \boldsymbol{S}_{ND} \end{aligned} \quad (11)$$

where ρ_{ND} is a random number defined in the interval (0,1) while the λ_{ND} parameter is a scalar product defined as $\{[\nabla W(\boldsymbol{X}_{OPT}) / ||\nabla W(\boldsymbol{X}_{OPT})||] \cdot [\boldsymbol{S}_{ND} / ||\boldsymbol{S}_{ND}||]\}$. As is clear, λ_{ND} is always smaller than 1 except when the non-descent direction $\boldsymbol{S}_{ND}$ coincides with the direction of the cost function gradient.

The rationale behind criterion (11) is the following. If candidate designs defined by descent directions were not effective in reducing cost and improvement routines were unsuccessful, the optimizer has to pay some cost in order to improve design later. However, the penalty should be the smallest as possible. If λ_{ND} is small, the non-descent direction $\boldsymbol{S}_{ND}$ is far enough from the gradient of cost function and therefore we expect the penalty in cost not be too large. The random nature of the rejection/acceptance process of $\boldsymbol{S}_{ND}$ is ensured by the presence of the ρ_{ND} numbers.

If $\boldsymbol{S}_{ND}$ is to be changed, MPISA performs two operations. Firstly, the step size $||\boldsymbol{S}_{ND}||$ is reduced by $\omega_{\min}$. This will reduce the difference between the approximate cost function and the actual cost and hence increase the value of the probability function $P(\Delta W_{ND})$. Equations (10-11) are recomputed and the new step size is accepted if the acceptance criterion (11) is satisfied. The design $P_{ND}(x_{OPT,1}\omega_{\min}\Delta x_1^{ND}, x_{OPT,2} + \omega_{\min}\Delta x_2^{ND}, \ldots, x_{OPT,N} + \omega_{\min}\Delta x_N^{ND})$ is hence defined.

Conversely, if the acceptance criterion (11) is not satisfied by simply reducing the step size, MPISA perturbs randomly the components of $\boldsymbol{S}_{ND}$ in order to move away from the cost function gradient thus reducing the penalty in cost. The scalar product λ_{ND} between $\boldsymbol{S}_{ND}$ and $\nabla W(\boldsymbol{X}_{OPT})$ can hence decrease thus making it easier to satisfy the criterion (11). To this purpose, the limit direction

$\boldsymbol{S}_{LIM}^{closest}$ closest to $\boldsymbol{S}_{ND}$ must be considered. Therefore, $\boldsymbol{S}_{LIM}^{closest}$ will be such to maximize the internal product $\boldsymbol{S}_{ND}^{T}\boldsymbol{S}_{LIM}^{closest}$.

Figure 3 illustrates the case of a two-design variable problem. The new direction $\boldsymbol{S}_{ND}^{final}$ has to lie between $\boldsymbol{S}_{ND}$ and $\boldsymbol{S}_{LIM}$".

The new movements Δx_{ND}^{final} corresponding to $\boldsymbol{S}_{ND}^{final}$ are hence set as follows ($j=1,\ldots,N$):

$$\Delta x_j^{ND}\Delta x_j^{LIM,closest} > 0 \Rightarrow \Delta x_j^{ND,final} = \rho_{ND}^{j}\Delta x_j^{ND} \quad (12a)$$

$$\Delta x_j^{ND}\Delta x_j^{LIM,closest} < 0 \Rightarrow \Delta x_j^{ND,final} = \Delta x_j^{ND,\min} + \rho_{ND}^{j}\left[\Delta x_j^{ND,\max} - \Delta x_j^{ND,\min}\right] \quad (12b)$$

where ρ_{ND}^{j} is a random number defined in the interval (0,1) for each design variable; $\Delta x_{LIM}^{closest}$ are the movements corresponding to $\boldsymbol{S}_{LIM}^{closest}$; $\Delta x_j^{ND,\min}$ is defined as $\min\left(\Delta x_j^{ND}, \Delta x_j^{LIM,closest}\right)$; $\Delta x_j^{ND,\max}$ is defined as $\max\left(\Delta x_j^{ND}, \Delta x_j^{LIM,closest}\right)$.

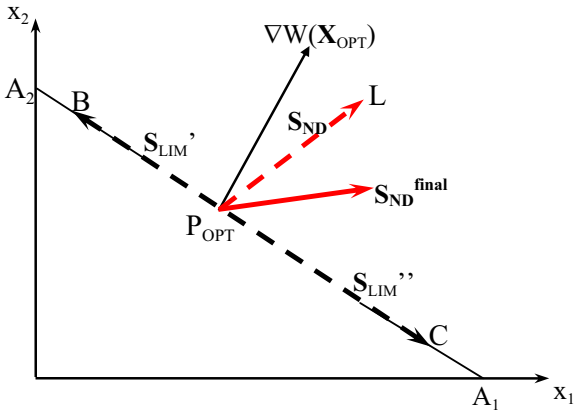


Figure 3: Definition of the non-descent direction included in Ω so to minimize cost penalty

Equations (10-11) are re-computed for the $\boldsymbol{S}_{ND}^{final}$ direction. If the probabilistic acceptance criterion is satisfied, the new non-descent design

$$P_{ND}\left(x_{OPT,1}+\Delta x_1^{ND,final}, x_{OPT,2}+\Delta x_2^{ND,final}, \ldots, x_{OPT,N}+\Delta x_N^{ND,final}\right)$$

is finally defined. Conversely, $\boldsymbol{S}_{ND}$ is reset as $\boldsymbol{S}_{ND}^{final}$ and MPISA searches a new direction $\boldsymbol{S}_{ND}^{final}$ yet closer to the limit direction $\boldsymbol{S}_{LIM}^{closest}$.

Say $P\left(\Delta W_{ND}^{final}\right)$ the probability function value computed for the non-descent direction finally accepted.

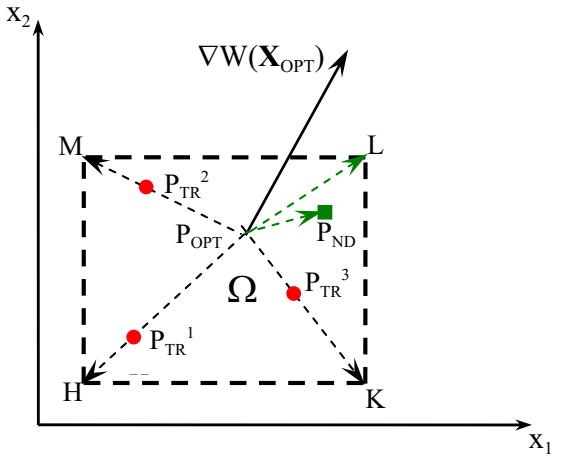


Figure 4: Replacement of domain Ω with a set of candidate designs (circle and square dots) independently defined

At the end of refinement process described in this Step 4, the domain Ω is no longer a hyper-prism. This is because the step size along each descent direction is adjusted independently from the other step sizes. Furthermore, the non-descent direction $\boldsymbol{S}_{ND}$ originally defined by some components of the descent directions $\boldsymbol{S}_{TR}^{q}$ is also changed independently (see Fig. 4). This strategy serves to increase the design freedom included in the optimization.

Step 5. Evaluation of trial designs. Choice of the best candidate design.

Each descent direction $\boldsymbol{S}_{TR}^{q,final}$ ($q=1,\ldots,2^N-1$) is limited by the current optimum design P_{OPT} and a trial design $P_{TR}^{q}\left(x_{OPT,1}+\Delta x_1^q, x_{OPT,2}+\Delta x_2^q, \ldots, x_{OPT,N}+\Delta x_N^q\right)$. The latter is identified by a circle dot in Fig. 4. Design variables are perturbed by the $\Delta x_j^{q,final}$ ($j=1,\ldots,N$) movements.

In the ISA algorithm previously proposed by the authors constraints are evaluated only at the trial point lying on direction $\boldsymbol{S}_{TR}^{1}$. Conversely, MPISA

now takes as candidate design the trial point lying on the descent direction for which it holds the condition ($q = 1,\ldots,2^N$-1):

$$\min\left\{\left[\boldsymbol{S}_{TR}^{q,final}\right]^T\left[-\overline{\nabla}W(\boldsymbol{X}_{OPT})\right]\right\} \tag{13}$$

Say s the sorting number denoting the descent direction $\boldsymbol{S}_{TR}^{trial}$ which meets condition (13). The corresponding point P_{TR}^s will hence be the point at which there is the largest reduction in weight. If there are two or more descent directions for which condition (13) is satisfied, all of the corresponding trial points are taken as candidate designs.

Let $\boldsymbol{X}_{TR}(x_{TR,1},x_{TR,2},\ldots,x_{TR,N})$ be the trial design vector containing the co-ordinates (i.e., the design variables) set for the P_{TR} point. Design variables are "temporary" updated as follows:

$$x_{TR,j} = x_{OPT,j} + \Delta x_j^s \quad (j = 1,\ldots,N) \tag{14}$$

where the Δx_j^s movements are the components of the descent direction defining the P_{TR} point.

It appears that the P_{TR} point is defined by MPISA without performing any structural analyses.

Step 6. Check if the candidate design actually reduces cost. Eventually, activate improvement routines.

Evaluate cost function and constraints at the trial point P_{TR} generated in Step 5. Let W_{TR} denote the corresponding value of the cost function. Calculate $\Delta W_{TR} = W_{TR} - W_{OPT}$. Remarkably, ΔW_{TR} never can be positive since this would be in conflict with the constraint posed by the trust region relationship (7). This fact allows us to avoid the N constraint evaluations required by the former program ISA when $\Delta W_{TR} > 0$. However, based on whether constraints are satisfied or not, there may be two different cases.

6-A. If it holds $\Delta W_{TR} < 0$ and the P_{TR} point is feasible, accept the design $\boldsymbol{X}_{TR}$ as the new optimum. Hence, set $\boldsymbol{X}_{OPT} \equiv \boldsymbol{X}_{TR}$ and $W_{OPT} = W_{TR}$. If $I_{GLOB} < I_{LIM}$, MPISA returns to Step 3. Otherwise, if $I_{GLOB} = I_{LIM}$, MPISA jumps to Step 9.

6-B. If it holds $\Delta W_{TR} < 0$ but the P_{TR} point is infeasible, MPISA builds a cubic approximation of the optimization problem about the current best record P_{OPT}. The original program - ISA - developed by the present authors considered a quadratic model fitted through three points: the current best record, the trial design and the point of intersection between $\boldsymbol{S}_{TR}^s$ and the boundaries of the linearized constraint domain. However, this strategy required N new constraint evaluations to linearize the optimization problem. For this reason, MPISA now generates only two new random designs Q and R lying on the segment limited by P_{OPT} and P_{TR}.
Coordinates of points Q and R are respectively defined as:

$$x_{Q,j} = x_{OPT,j} + \eta_Q(x_{TR,j} - x_{OPT,j}) \tag{15a}$$

and

$$x_{R,j} = x_{OPT,j} + \eta_R(x_{TR,j} - x_{OPT,j}) \tag{15b}$$

where $j = 1,\ldots,N$; η_Q and η_R are two random numbers in the (0,1) interval.
The cubic approximation is built by expanding the cost function and the NC_{act} active constraints in fashion of:

$$\begin{cases} W^{cubic}(\alpha) = W_{OPT} + b_w\alpha + c_w\alpha^2 \\ \qquad\qquad +d_w\alpha^3 \\ G_k^{cubic}(\alpha) = G_k(X_{OPT}) + b_k\alpha + c_k\alpha^2 \\ \qquad\qquad +d_k\alpha^3 \\ 0 \le \alpha \le 1 \end{cases} \tag{16}$$

where $k = 1,\ldots,NC_{act}$. A change of reference system is necessary in order to perform the series expansion of each non-linear function along the $\overline{P_{OPT}P_{TR}}$ segment. Remarkably, no sensitivity computations are required. The parameter α represents the magnitude of the step taken along $\overline{P_{OPT}P_{TR}}$. Coefficients c_w, d_w, and c_k, d_k, respectively defined for cost functions and constraints, account for local changes in curvature and can be determined by imposing that the cubic model fits the responses evaluated at Q and

R.
Once the cubic model is set, MPISA solves a set of NC_{act} cubic equations in order to find the steps $\Delta\alpha_k$ for which the constraint violation vanishes in the approximate model. The cubic equations derive from the equalities $G_k^{cubic}(\Delta\alpha_k) = 0$ $(k = 1,\ldots,NC_{act})$. The smallest step amongst steps $\Delta\alpha_k$ is taken as the trial step Δs_{TR}. The trial step Δs_{TR} serves to define the new trial design point F_{TR} on the $\boldsymbol{S}_{TR}^{s}$ direction at which the real cost function W is evaluated again.
If the cost function improves ($W_{TR} < W_{OPT}$), MPISA evaluates non-linear constraints at F_{TR}. If F_{TR} is feasible, accept it as the new current best record. Hence, MPISA executes Step 3 or Step 9 if it holds $I_{GLOB} < I_{LIM}$ or $I_{GLOB} = I_{LIM}$, respectively.
Conversely, if the improvement routine fails ($W_{TR} > W_{OPT}$), MPISA executes Step 7.

6-C. The case $\Delta W_{TR} > 0$ may occur if design space is locally non-convex. The trust region model (7) serves to resize step sizes along descent directions but does not deal directly with possible non-convexity. In order to overcome this problem, MPISA activates different improvement routines. Remarkably no linearization of constraints is performed by MPISA since the trust region model should have granted the accuracy of linear approximation but did not work well. Conversely, the original code ISA did linearize constraints and hence required a higher computational time.
A new trial point $P_{TR,NEW}$ is defined as the "symmetric" of the old P_{TR} point about the P_{OPT} point. This strategy is justified by the fact that the search direction limited by P_{OPT} and P_{TR} (identified as a descent direction in the approximate model) actually was found not to be a descent direction. Therefore, MPISA tries to perturb design by moving along the opposite direction. The cost function is re-evaluated at $P_{TR,NEW}$. If the new increment $\Delta W_{TR} = W_{TR,NEW} - W_{OPT}$ is negative, MPISA evaluates non-linear constraints at $P_{TR,NEW}$. If $P_{TR,NEW}$ is feasible, MPISA accepts it as the new current best record; the optimization algorithm is hence re-started from Step 3. Conversely, if $P_{TR,NEW}$ is infeasible, MPISA executes Step 6-B.
If the new increment $\Delta W_{TR} = W_{TR,NEW} - W_{OPT}$ is yet positive, the trial point $P_{TR,NEW}$ is compared to the P_{ND} point (square dot in Fig. 4) defined in Step 4. If $W_{TR,NEW} < W_{ND}$, MPISA chooses this as the new best record and executes Step 3. Conversely, if $W_{TR,NEW} > W_{ND}$, a probabilistic criterion is adopted.
The probability threshold $P(\Delta W_{ND})$ used for finding P_{ND} is compared to the acceptance probability $P(\Delta W_{TR}) = e^{-[1-\Delta W_{ND}/\Delta W_{TR}]/T_K}$.
If $P(\Delta W_{TR}) > P(\Delta W_{ND})$, the $P_{TR,NEW}$ point is taken as the new best record. Conversely, MPISA takes P_{ND} as the new best record. Optimization is restarted from Step 3.

Step 7. Local annealing. Perturb design variables one at a time.

Perturb the design variables one by one about the current best record P_{OPT} in order to move away from constraint boundaries and to escape from local minima. Linearize constraints about P_{OPT}. Use linearized constraints in order to reduce the number of exact analyses thus saving computation time.

MPISA checks if the design space is locally non-convex. Figure 5 illustrates the case of a two-design variable problem.

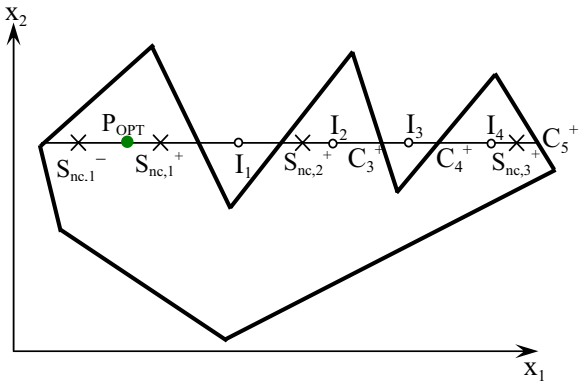


Figure 5: Line search strategy performed if constraint domain is non-convex

Let $\boldsymbol{S}_j^{1d}$ be the direction defined by perturbing only the j^{th} variable. MPISA solves the linear system formed by the linearized constraints and the $\boldsymbol{S}_j^{1d}$ direction. If there are NC_{act} active constraints, define hence the points $C_1^+, C_2^+, C_3^+, \ldots, C_{Nact}^+$ or $C_1^-, C_2^-, C_3^-, \ldots, C_{Nact}^-$ where superscripts + and - respectively indicate that the intersection occurs at a value of x_j larger or smaller than $x_{OPT,j}$.

For $x_j > x_{OPT,j}$, define the $\overline{P_{OPT}C_1^+}$ segment and the corresponding middle point I_o^+. Define other $NC_{act}^+ - 1$ segments in fashion of $\overline{C_{ma}^+ C_{ma+1}^+}$ ($m_a^+ = 1,\ldots,NC_{act}^+ - 1$): for each segment define the middle point I_{ma}^+ so that $\overline{C_{ma}^+ I_{ma}^+} = \overline{I_{ma}^+ C_{ma+1}^+}$. The same is done for $x_j < x_{OPT,j}$. Hence, the C_{ma}^- set of points for which there are NC_{act}^- active constraints can be defined.

Evaluate linearized constraints at each of the I_{ma}^+ and I_{ma}^- points in the negative ($m_a^+ = 0,\ldots,NC_{act}^+ - 1$) and positive verse ($m_a^- = 0,\ldots,NC_{act}^- - 1$) with respect to P_{OPT}. In general, there will be MF feasible segments and MI infeasible segments. It obviously holds $NC_{act} = MF + MI$, where $0 < MF < NC_{act}$. Let p be a counter for the MF feasible segments. MF and MI include both "negative" and "positive" point series.

Write the expression of cost function $W(\boldsymbol{X})$ in the one-dimensional reference system whose coordinate is the step $\Delta s_{nc,p}^+$ or $\Delta s_{nc,p}^-$ taken along the p^{th} feasible segment ($p = 1,\ldots,MF$) defined by increasing or decreasing x_j with respect to $x_{OPT,j}$. That is, rearrange the objective function as $W(\Delta s_{nc,p})$. Do this operation for each of the MF feasible segments.

Search for minimum cost along each of the MF segments. Since each $W(\Delta s_{nc,p})$ function is a single-variable function, find the $\Delta s_{nc,p}$ step by means of the straightforward condition $dW/d(\Delta s_{nc,p})$=0. Let be $S_{nc,p}$ the design point defined by the $\Delta s_{nc,p}$ step. Store each $S_{nc,p}$ point in the Γ database. Include in the Γ database also the current best record P_{OPT}.

Search for the S_{NCONV} point at which it holds $\min_{p=1,\ldots,MF} W(\Delta s_{nc,p})$. Transform the local coordinate into the new $x_{TR,j}$. Finally, rename the S_{NCONV} point as $P_{TR,j}$.

If there are only two intersections C_1^+ and C_1^- the constraint domain can be assumed locally convex. Figure 6 illustrates this situation for a two-design variable problem. MPISA generates another sequence of random numbers $N_{RND,j}$ ($j = 1,\ldots,N$) - one for each design variable - in the interval (0,1). Each trial point is generated as follows (17):

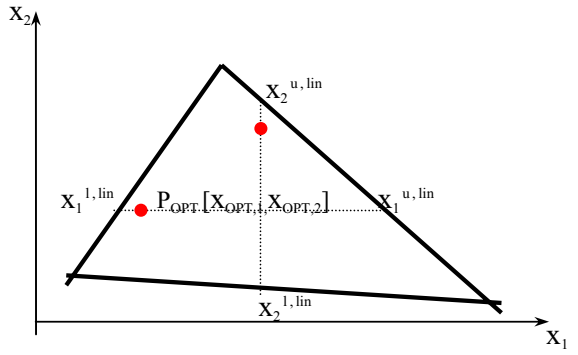


Figure 6: Definition of search domain for local annealing when design variables are perturbed one at a time

$$
\begin{aligned}
&N_{RND,j} > 0.5 \Rightarrow \\
&\qquad x_{TR,j} = x_{OPT,j} + (x_j^{u,lin} - x_j^{l,lin}) \cdot N_{RND,j} \cdot \gamma_K \\
&N_{RND,j} < 0.5 \Rightarrow \\
&\qquad x_{TR,j} = x_{OPT,j} - (x_j^{u,lin} - x_j^{l,lin}) \cdot N_{RND,j} \cdot \gamma_K
\end{aligned}
\tag{17}
$$

Similarly to Step 3, design variables are yet changed following their order sequentially. Say Δx_j^{1d} the movements $(x_{TR,j} - x_{OPT,j})$ or $(x_{OPT,j} - x_{TR,j})$ set with Eq. (17). It can be seen that, for each design variable, new designs are generated inside the segment limited by the intersections $x_j^{l,lin}$ and $x_j^{u,lin}$ with the boundaries of the linearized constraint domain. The two dots in Fig. 6 correspond to trial designs.

The knockdown factor γ_K is defined as:

$$\gamma_K = \min\left[T_K/T_o, \sqrt{\varepsilon_{LIN}}\right] \tag{18}$$

The purpose of γ_K is twofold: (i) to keep trial designs far enough from the boundaries of the constraint domain; (ii) to account for the fact that trial

points have been searched in the linearized design space.

As is clear, T_K/T_o decreases as the optimization progresses. Since the current best record becomes more and more closer to constraint boundaries, movement size should be reduced approaching the end of the optimization process.

Including parameter $\sqrt{\varepsilon_{LIN}}$ in the knockdown factor expression (18) is justified by the fact that the error introduced by linearization is about proportional to the square of the step size: that is, to the distance between trial designs and the current best record. In order to make a conservative estimate on ε_{LIN}, MPISA takes the point P_{FAR} - defined by the $\boldsymbol{X}_{FAR}$ design vector - lying on the boundary of linearized constraint domain and located at the largest distance from the current best record P_{OPT}. P_{FAR} is obviously one of the intersection points between directions $\boldsymbol{S}_j^{1d}$ and linearized domain boundaries. The error ε_{LIN} is hence evaluated at P_{FAR}:

$$\varepsilon_{LIN} = \max_{k=1,\ldots,NC} \left\{ \begin{array}{l} \left|1 - W_{LIN}(X_{FAR})/W(X_{FAR})\right|, \\ \left|1 - G_k^{LIN}(X_{FAR})/G_k(X_{FAR})\right| \end{array} \right\} \tag{19}$$

where the linearized functions W_{LIN} and G_k^{LIN} are defined as (20):

$$\begin{aligned} W(\boldsymbol{X}_{FAR}) &= W(\boldsymbol{X}_{OPT}) \\ &\quad + (\boldsymbol{X}_{FAR} - \boldsymbol{X}_{OPT})^T \overline{\nabla} W(\boldsymbol{X}_{OPT}) \\ G_k^{LIN}(\boldsymbol{X}_{FAR}) &= G_k(\boldsymbol{X}_{OPT}) \\ &\quad + (\boldsymbol{X}_{FAR} - \boldsymbol{X}_{OPT})^T \overline{\nabla} G_k(\boldsymbol{X}_{OPT}) \end{aligned} \tag{20}$$

Cost function gradient is not utilized for Eq. (17) since MPISA performs local annealing only if the step sizes taken on the descent directions $\boldsymbol{S}_{TR}^{q,final}$ in the current global annealing cycle led to violate constraints or pushed design search towards a non-convex region. MPISA attempts either to recover the constraint violation - eventually at the cost of some penalty in weight - or to get back to a convex region of design space. The former might imply $\boldsymbol{S}_{TR}^T \overline{\nabla} W(\boldsymbol{X}_{OPT}) > 0$. In the latter case, unlike the original code ISA, the new algorithm MPISA now includes a strategy for handling domain non-convexity.

Remarkably, MPISA is able to assess the different degree of non-convexity with respect to each design variable. For instance, Fig. 5 shows that multiple intersections with the constraint domain boundaries occur only for variable x_1. As is clear, the probability of reaching a globally optimum design may increase if one knows the role played by each design variable in the optimization process, especially if variable perturbations will push the design search towards a non-convex region. Therefore, the new capability included in MPISA turns to be extremely useful in optimization problems where the design space is comprised of different sub-set of optimization variables that are not directly connected: for instance, skeletal structures optimized with sizing and configuration variables.

While in the ISA case trial designs could end up infeasible with respect to linearized constraints, MPISA now generates trial points that always satisfy approximate constraints. This strategy is justified by the informal argument that if a trial design violates linearized constraints, non-linear constraints also will be very likely to be violated. Consequently, the trial design is not useful.

Now, let $P_{TR,j}$ denote the new trial design generated for the j^{th} variable. Let $\boldsymbol{X}_j(x_{OPT,1}, x_{OPT,2,\ldots}, x_{TR,j,\ldots}, x_{OPT,N})$ be the corresponding design vector. Compute cost function $W(\boldsymbol{X}_j)$ at each $P_{TR,j}$ and cost change $\Delta W_j = W(\boldsymbol{X}_j) - W_{OPT}$.

If it occurs $\Delta W_j < 0$, store the $\boldsymbol{X}_j$ vector and the W_j cost in the Π_1 database. However, as the W_j cost decreases the corresponding $\boldsymbol{X}_j$ design may get too close to the boundary of linearized constraint domain and finally end up infeasible when non-linear constrains are evaluated. For this reason, MPISA chooses from the Π_1 database the two designs $\boldsymbol{X}_{j,SMALL}$ and $\boldsymbol{X}_{j,LARGE}$, respectively, corresponding to the smallest and the largest amongst the weight changes $|\Delta W_j|$. Non-linear constraints are evaluated at these two points. If design $\boldsymbol{X}_{j,LARGE}$ is feasible, MPISA takes this as the new best record. Conversely, the $\boldsymbol{X}_{j,SMALL}$ design is taken as the new best record.

If $\Delta W_j > 0$ and linearized constraints are satisfied,

re-define the Metropolis' probability function as:

$$P(\Delta W_j) = e^{\frac{-\Delta W_j}{\left(\sum_{r=1}^{NDW} \Delta W_r / NDW\right) \cdot T_K}} \quad (21)$$

The NDW parameter in Eq. (21) is the number of trial points at which the cost function resulted larger than the current best records found throughout the optimization process. The ΔW_r terms are the corresponding weight penalties. The $\sum_{r=1,NDW} \Delta W_r / NDW$ ratio accounts for the general formation of all the previous candidate designs and serves to normalize the probability function with respect to the change in cost function.

Each design $\boldsymbol{X}_j$ is provisionally accepted or certainly rejected according to the Metropolis' criterion re-formulated as:

$$\begin{aligned} P(\Delta W_j) &> \max\left[NRD_j, P\left(\Delta W_{ND}^{final}\right)\right] \Rightarrow \text{Accept} \\ P(\Delta W_j) &< \max\left[NRD_j, P\left(\Delta W_{ND}^{final}\right)\right] \Rightarrow \text{Reject} \end{aligned} \quad (22)$$

The $\boldsymbol{X}_j$ designs provisionally accepted are included in the Π_2 database. If there are no trial designs yielding reductions in cost, MPISA extracts from Π_2 the design $\boldsymbol{X}_j^{BEST}$ for which the cost function value is the least and sets this as the current best record. In simple words, MPISA minimizes the increase in cost if the local annealing could not improve design.

It should be noted that the probability threshold set by MPISA in purpose to accept or not a design depends on both local and global information since the random perturbation of a single variable $(N_{RND,j})$ is compared to the level of probability $P(\Delta W_{ND}^{final})$ relative to the non-descent direction defined by perturbing simultaneously all optimization variables (see Step 4).

Once Step 7 is completed, MPISA returns to Step 2.

Step 8. Search for a feasible design or for a design violating constraints the least.

Set $I_{GLOB} = I_{GLOB} + 1$. This step is executed if the optimization process or a cooling cycle begins from an infeasible point. Any infeasible starting design is yet denoted as $\boldsymbol{X}_{OPT}$ adopting the notation used in Step 1. Let us assume that there are NCV violated constraints. MPISA tries to move back to a region where constraints get less violated than at $\boldsymbol{X}_{OPT}$ and cost function may eventually improve.

To this purpose, MPISA linearizes the optimization constraints and defines NCV directions each of which is the negative gradient of a violated constraint. That is, we have: $\delta_{G,V} = -\nabla G_V(\boldsymbol{X}_{OPT})$, where $v = 1,\ldots,NCV$.

MPISA checks if for some of the $\delta_{G,V}$ directions it holds $\delta_{G,V}^T \nabla W(\boldsymbol{X}_{OPT}) < 0$. Should this be the case, the linear system formed by the $\delta_{G,V}$ directions and the linearized constraints is solved.

Figure 7 illustrates this process for a two-variable problem with one violated constraint. Constraint numbers are indicated within circles. The direction $\delta_{G,1}$ intersects constraints 1, 2 and 3 in the points 1', 2' and 3', respectively. It can be seen that feasible designs may be generated by moving along the segment limited by points 1' and 3'.

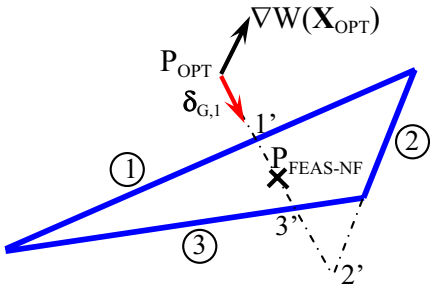


Figure 7: Design perturbation in case of infeasible points and one violated constraint

MPISA generates a random number η_{NF} in the interval (0,1) in order to define a feasible point $P_{FEAS-NF}$ lying between points 1' and 3'. These intersections are sorted based on the magnitude of their distances from the infeasible initial point P_{OPT}. The optimization variables corresponding to the coordinates of $P_{FEAS-NF}$ are defined as:

$$X_{FEAS-NF,j} = x_{1',j} + \eta_{NF} \cdot (x_{3',j} - x_{1',j}) \quad (j = 1,\ldots,N) \quad (23)$$

Figure 8 illustrates the strategy implemented in MPISA when there are more than one violated constraint. The constraint domain is also locally non-convex.

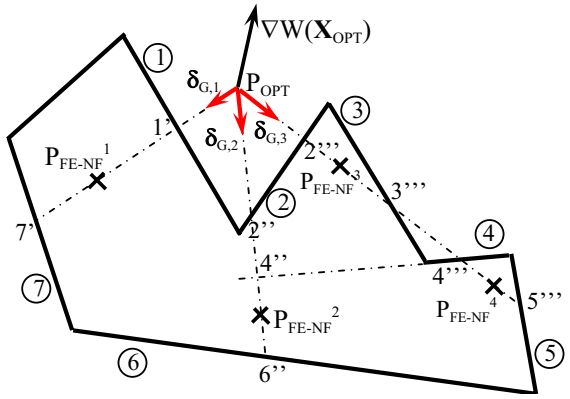


Figure 8: Design perturbation in case of infeasible points and multiple violated constraints

Since the initial point P_{OPT} violates constraints 1, 2 and 4, MPISA defines directions $\delta_{G,1}$, $\delta_{G,2}$ and $\delta_{G,3}$. For each direction, it holds $\delta_{G,V}^T \nabla W(\boldsymbol{X}_{OPT}) < 0$ $(v = 1,\ldots,3)$.

Direction $\delta_{G,1}$ intersects the linearized constraint domain at points 1' and 7'. Direction $\delta_{G,2}$ intersects the linearized constraint domain at points 2", 4" and 6". Finally, direction $\delta_{G,3}$ intersects the linearized constraint domain at points 2"' and 3"', 4"' and 5"'. The number of apices is relative to the sorting number of the constraint gradient currently examined.

Therefore, feasible designs P_{FE-NF}^r $(r = 1,\ldots,4)$ can be generated by moving along the four segments respectively limited by points 1' and 7', 2" and 6", 2"' and 3"', 4"' and 5"'. To this purpose, MPISA generates four new random numbers in the interval (0,1). That is, optimization variables (j=1,...,N) can be set as:

$$\begin{aligned} x_{FE-NF,j}^1 &= x_{1',j} + \eta_{NF,1} \cdot (x_{7',j} - x_{1',j}) \\ x_{FE-NF,j}^2 &= x_{2'',j} + \eta_{NF,2} \cdot (x_{6'',j} - x_{2'',j}) \\ x_{FE-NF,j}^3 &= x_{2''',j} + \eta_{NF,3} \cdot (x_{3''',j} - x_{2''',j}) \\ x_{FE-NF,j}^4 &= x_{4''',j} + \eta_{NF,1} \cdot (x_{5''',j} - x_{4''',j}) \end{aligned} \tag{24}$$

This procedure can be generalized as follows. Solve the linear system formed by each direction $\delta_{G,v}$ $(v = 1,\ldots,NCV)$ and the linearized constraints. Sort the $N_{INT,v}$ intersection points according to the magnitude of their distance from the initial point P_{OPT}. Define $N_{INT,v} - 1$ segments limited by these intersection points. Any segment is said "feasible" if it is limited by two points satisfying the linearized constraints. A random number in the interval (0,1) is hence generated each time a feasible segment is found in order to define a new trial design P_{FE-NF}^r where the r superscript ranges between 0 (no feasible segments) and the number of feasible segments $N_{SGFEA,v}$.

However, it may happen that there are no constraint gradients such that $\delta_{G,V}^T \nabla W(\boldsymbol{X}_{OPT}) < 0$. Figure 9 illustrates this scenario for a two-variable problem.

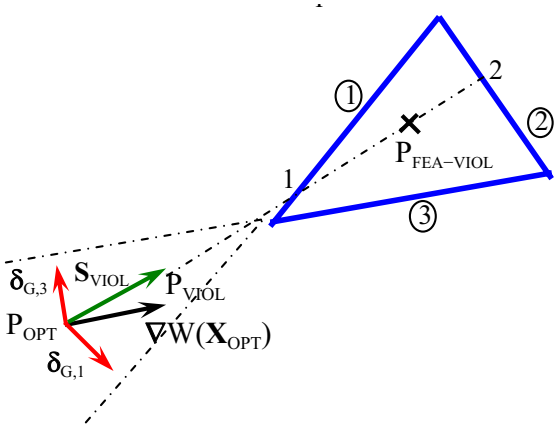


Figure 9: Design perturbation when there are no directions $\delta_{G,V}$ such that $\delta_{G,V}^T \nabla W(\boldsymbol{X}_{OPT}) < 0$

It appears that no feasible segments can be found by intersecting directions $\delta_{G,1}$ and $\delta_{G,3}$ - relative to violated constraints 1 and 3 - with the linearized constraint domain. Constraint domain boundary could be approached, and violation might be reduced, by moving along $\delta_{G,1}$ or $\delta_{G,3}$. However, constraint violation can be reduced more quickly if we move along a direction $\boldsymbol{S}_{VIOL}$ whose components are proportional to the gradients of violated constraints. Therefore, MPISA generates NCV random numbers η_v $(v = 1,\ldots NCV)$ in the interval (0,1). The $\boldsymbol{S}_{VIOL}$ direction is hence defined as:

$$\boldsymbol{S}_{VIOL} = \sum_{v=1}^{NCV} \eta_v \delta_{G,V} \tag{25}$$

Say P_{VIOL} the design point defined as $P_{VIOL}(x_{OPT,1} + \Delta S_{VIOL,1}, x_{OPT,2} + \Delta S_{VIOL,2}, \ldots, x_{OPT,N} + \Delta S_{VIOL,N})$ where $S_{VIOL,j}$ ($j = 1,\ldots,N$) are the components of the $\boldsymbol{S}_{VIOL}$ vector.

Since direction $\boldsymbol{S}_{VIOL}$ attempts to move back the design search towards a feasible region, MPISA checks if some feasible segment may lie on $\boldsymbol{S}_{VIOL}$. Therefore, the intersection points between $\boldsymbol{S}_{VIOL}$ and the linearized constraint domain are found by MPISA. In general, there will be only one feasible segment when the linearized constraint domain is convex while multiple feasible segments will lie on $\boldsymbol{S}_{VIOL}$ if this direction intersects a non-convex linearized constraint domain. Intersection points are again sorted based on the magnitude of their distance from P_{OPT}.

If there are no intersection points limiting feasible segments, constraints are re-linearized about P_{VIOL} and MPISA searches again for feasible segments. A new $\boldsymbol{S}_{VIOL}$ direction and a new P_{VIOL} point are defined. As is clear, directions $\delta_{G,V}$ are re-defined each time the linearization point (i.e., P_{VIOL}) changes. The process ends when it is possible to find feasible segments from the intersection between $\boldsymbol{S}_{VIOL}$ and the linearized constraint domain.

For instance, Fig. 9 shows that it is possible to find a feasible segment limited by points 1 and 2, that are the intersections between $\boldsymbol{S}_{VIOL}$ and linearized constraints 1 and 2. Feasible designs $P_{FEA-VIOL}$ (one intersection) or $P^{s}_{FE-VIOL}$ (multiple intersections) can be defined by generating random numbers in the interval (0,1) in a similar fashion as for points $P_{FEAS-NF}$ or P^{r}_{FE-NF} defined through Eqs. (23-24).

Each feasible design lying on a direction $\delta_{G,V}$ such that $\delta^{T}_{G,V}\nabla W(\boldsymbol{X}_{OPT}) < 0$ is stored in the database Π_{FEA-NF}. Conversely, each feasible design eventually lying on the $\boldsymbol{S}_{VIOL}$ direction is stored in another database $\Pi_{FEA-VIOL}$.

As is clear, trial points included in Π_{FEA-NF} are such that the double goal of reducing constraint violation and reducing cost function is achieved. Conversely, trial points included in $\Pi_{FEA-VIOL}$ satisfy linearized constraints but do not necessarily lie on a descent direction. Therefore, the optimizer may have to pay some weight penalty in order to get in a feasible region of the design space.

MPISA tries to maximize the improvement in cost or, alternatively, attempts to minimize the weight penalty. Therefore, MPISA extracts from the Π_{FEA-NF} database the $P^{r,max-red}_{FE-NF}$ point for which the cost reduction is largest with respect to the initial design. Non linear constraints are evaluated at $P^{r,max-red}_{FE-NF}$ and if this design is feasible, MPISA takes it as the new best record. Should the point $P^{r,max-red}_{FE-NF}$ not be feasible, MPISA checks the other points left in Π_{FEA-NF}. Again, the point at which cost improvement is largest is taken as trial design and non-linear constraints are evaluated. The process terminates when one of the points included in the Π_{FEA-NF} database satisfies non-linear constraints.

If the Π_{FEA-NF} database is empty, MPISA extracts from the $\Pi_{FEA-VIOL}$ database the $P^{s,least-pen}_{FE-VIOL}$ point for which the cost penalty is the least and sets this design as the new best record. The optimization process is re-started from Step 2.

It should be noted that linear approximation of optimization constraints used by MPISA allows to reduce significantly the number of exact analyses. In fact, classical simulated annealing algorithms usually deal with violated constraints by generating new random designs until non-linear constraints are satisfied. This process is completed by perturbing one design variable at a time. However, at least N exact analyses may be required each time one wants to see what happens in the neighborhood of a design point. Furthermore, this strategy may result in increasing the cost function too much. In order to overcome these problems, the ISA algorithm previously developed by the present authors already utilized a linearly approximated model and defined a search direction - whose target is the feasible design space - by combining the set of movements corresponding to the intersections between the $\delta_{G,V}$ directions and the linearized constraint domain boundary.

Remarkably, the new algorithm MPISA is able to find feasible segments for each constraint gradient or at least to minimize cost penalty yet find-

ing some feasible design. Therefore, one or more candidate designs can be defined for each $\delta_{G,V}$ rather than the single trial point originally defined in ISA. The stochastic nature of the optimization process is ensured by the fact that each trial design included in the Π_{FEA-NF} and $\Pi_{FEA-VIOL}$ databases is generated by MPISA using a random number.

As far as it concerns the accuracy of linear approximation, it should be noted that MPISA does not use any more the trust region scheme originally implemented in ISA to resize step along the search direction which pushes design back to a feasible region. Trust region served in ISA to ensure that the linear approximation portrayed accurately the actual non-linear problem. However, any point satisfying linearized constraints will certainly satisfy also non-linear constraints if it is far enough from constraint boundaries. Since random numbers defining trial points included in Π_{FEA-NF} and $\Pi_{FEA-VIOL}$ may not be equal to 0 or 1, trial points defined in MPISA are directly pushed away from linearized constraint boundaries. This fact increases the probability to have trial points that satisfy both linearized and non-linear constraints.

Step 9. Check for convergence of the optimization process.

If $K > 3$, MPISA checks if the optimization process converged. The following convergence criterion (26) is used:

$$\max\left\{\max\left[\frac{|W_{OPT,K}-W_{OPT,K-1}|}{W_{OPT,K}};\frac{|W_{OPT,K-1}-W_{OPT,K-2}|}{W_{OPT,K-1}};\frac{|W_{OPT,K-2}-W_{OPT,K-3}|}{W_{OPT,K-2}}\right];\right.$$
$$\left.\max\left[\frac{||X_{OPT,K}-X_{OPT,K-1}||}{||X_{OPT,K}||};\frac{||X_{OPT,K-1}-X_{OPT,K-2}||}{||X_{OPT,K-1}||};\frac{||X_{OPT,K-2}-X_{OPT,K-3}||}{||X_{OPT,K-2}||}\right]\right\}$$
$$\leq \varepsilon_{CONV} \quad (26)$$

where $W_{OPT,K}$ and $\boldsymbol{X}_{OPT,K,}$ respectively, denote the best record and the corresponding design vector found in the K^{th} cooling cycle.

The ε_{CONV} parameter is set to 10^{-5} in order to avoid premature convergence if the last four cooling cycles resulted in marginal improvements in design.

If criterion (26) is satisfied go to Step 11.

Step 10. Reset parameters for a new cooling cycle.

If $K < 3$ or the convergence criterion (26) is not satisfied:

- Reduce temperature in fashion of $T_{K+1} = \beta_K T_K$ where the parameter β_K is chosen as (27):

$$\beta_K = \left[\sum_{r=0}^{K-1}\beta_r/K\right] \cdot \max\left[\frac{0.95}{\left(1+\frac{N_{REJE}}{N_{TRIA}}\right)};\left(1-\frac{W_{FIN,K-1}}{W_{INIT,K-1}}\right)\right] \quad (27)$$

$W_{FIN,K-1}$ and $W_{INIT,K-1}$ are respectively the cost function values at the beginning and at the end of the current annealing cycle. N_{REJE} is the number of trial designs rejected by MPISA out of the total number of trial designs N_{TRIA} generated in the current cooling cycle. The N_{REJE} number includes each trial point which does not yield immediate improvement in design. For instance, if $I_{LIM} = 5$ global annealing cycles are performed within the current cooling cycle and MPISA executes Step 6-B in each global annealing cycle (that is, there are five infeasible trial points at which the cost function decreases), it yields $N_{REJE} = 5$ and $N_{TRIA} = I_{LIM} + N_{REJE} = 10$.

The simplest strategy suggested by optimization handbooks is to use a constant temperature reduction factor chosen between 0.9 and 0.99. Conversely, temperature reduction factor is adaptively updated by MPISA throughout the optimization process. In particular, Eq. (27) takes care of two important facts: the percentage of trial points that immediately yield improvements in design and the trend of the cost function. The former effect is captured by a term whose upper limit is the 0.95 value that is about the average between 0.9 and 0.99; this upper limit is eventually reduced by means of a knockdown factor λ_K defined as the inverse of $(1 + N_{REJE}/N_{TRIA})$: the λ_K factor is obviously equal to 1 if $N_{REJE} = 0$. The effect

of cost function is captured in (27) by computing the relative change in cost attained in the current cooling cycle.

The rationale behind Eq. (27) is the following. If the cost function decreased much in the current cooling cycle, the temperature can be kept high since the optimizer is still exploring a zone where cost function gradient is negative. Hence, new descent directions can be found easily and the risk to reject candidate designs is very low. In simple words, criterion (27) is driven by $(1 - W_{FIN,K-1}/W_{INIT,K-1})$. For instance, if Step 6-B is executed one time within each global annealing cycle ($N_{REJE} = 5$; $N_{TRIA} = 10$), one gets 0.633 for the $0.95\lambda_K$ product. This implies that if cost function improvement is larger than 63.4% (very likely to occur in the early cooling cycles if the initial design is far enough from constraint domain boundaries) makes the $(1 - W_{FIN,K-1}/W_{INIT,K-1})$ term predominate over the $0.95\lambda_K$ product.

Conversely, if the cost function improved marginally or even increased, the optimizer entered in a zone where there are few descent directions or a descent direction could not even be defined. Hence, the temperature should be reduced in order to reject many trial points because they certainly will not yield significant improvements in design. In simple words, since the $(1 - W_{FIN,K-1}/W_{INIT,K-1})$ term may get close to zero, criterion (27) is driven by the $0.95\lambda_K$ product: the larger the number of trial designs rejected in the current design cycle, the higher will be the reduction in temperature. This strategy is not too conservative. In fact, assuming that the 30% of total trials were rejected (that is, a very high rejection rate), one gets a value of about 0.75 for the $0.95\lambda_K$ product.

Eq. (27) is derived from the original formulation implemented in ISA. However, MPISA now introduces in Eq. (27) the $\Sigma_{r=0,\ldots,K-1}\beta_r/K$ correction term which is the average of all of the β_K factors defined in the optimization process until the current annealing cycle. As is clear, the correction term is equal to 1 in the first annealing cycle. The purpose of the $\Sigma_{r=0,\ldots,K-1}\beta_r/K$ correction term is to avoid too drastic reductions in temperature if a design cycle resulted in a very large number of rejected points or marginal improvements in design. This scenario may occur for instance near a local optimum. However, temperature can be reduced substantially only when the optimizer has already explored a sufficiently large portion of the design space and by-passed many local optima. The correction term averaging all β_K factors allows us not to increase too much the threshold value of acceptance probability $P(\Delta W)$ already in the early stages of optimization process.

- Reset K as K=K+1;
- Reset I_{GLOB} as $I_{GLOB} = 0$;
- Repeat from Step 2 onward.

Step 11. End optimization process and store results.

The optimum design and convergence history (cost function and constraint values) are written into output files made available to the user.

3 Test Cases

Numerical behavior of MPISA is tested in three optimization problems of skeletal structures. This kind of structure is widely used in optimization to demonstrate numerical efficiency of new design codes since complicated structures can be considered but no structural analysis is entailed by each new evaluation of the cost function.

In this work, two bar trusses and a frame structure are designed for minimum weight under multiple loading conditions and constraints on nodal displacements, element stress and buckling. Sizing and configuration optimization problems are considered.

Test cases are now described in detail.

3.1 Weight minimization of bar truss structures

The weight minimization problem for a bar truss structure comprised of NOD nodes and NEL ele-

ments may be stated as follows:

$$\begin{cases} \min W = \rho g \sum_{j=1}^{NEL} l_j x_j \\ u^l_{(x,y,z),k} \le u_{(x,y,z),k,ilc} \le u^u_{(x,y,z),k} \\ \sigma^l_j \le \sigma_{j,ilc} \le \sigma^u_j \\ x^l_i \le x_i \le x^u_i \end{cases} \tag{28}$$

where

- $i = 1,\ldots,N$; $j = 1,\ldots,NEL$; $k = 1,\ldots NOD$; $ilc = 1,\ldots,NLC$;
- x_j is the cross sectional area of the j^{th} element of the structure included as sizing variable in the optimization process;
- l_j is the length of the j^{th} element of the structure;
- g is the gravity acceleration value (9.81 m/s^2); ρ is the material density;
- NLC is the number of independent loading conditions acting on the structure;
- $u_{(x,y,z),k,ilc}$ are the displacements of the k^{th} node in the directions x, y, z, with the lower and upper bounds $u^l_{(x,y,z),k}$ and $u^u_{(x,y,z),k}$;
- $\sigma_{j,ilc}$ is the stress in the j^{th} element, with the lower and upper bounds σ^l_j (compression stress limit includes critical bucking load) and σ^u_j (tensile);
- The ilc subscript indicates that displacement and stress constraints are relative to the ilc^{th} loading condition. The constraints on stresses and displacements are put in a dimensionless form.

If the optimization includes also configuration variables in fashion of nodal coordinates, Eq. (28) becomes (29):

$$\min W = \rho g \sum_{j=1}^{NEL} x_j \cdot \sqrt{(x_{j1}-x_{j2})^2 + (y_{j1}-y_{j2})^2 + (z_{j1}-z_{j2})^2} \tag{29}$$

where $x_{j1,2}$, $y_{j1,2}$, $z_{j1,2}$ are the co-ordinates of the nodes limiting the generic j^{th} element of the structure.

In test case 1, the planar two-hundred bar truss structure shown in Fig. 10 is designed to carry five independent loading conditions:

a) 453.592 kgf (1000 lbf) acting in the positive x-direction at node points 1, 6, 15, 20, 29, 43, 48, 57, 62 and 71;

b) 453.592 kgf (1000 lbf) acting in the negative x-direction at node points 5, 14, 19, 28, 42, 47, 56, 61, 70 and 75;

c) 4535.924 kgf (10000 lbf) acting in the negative y-direction at node points 1, 2, 3, 4, 5, 6, 8, 10, 12, 14, 15, 16, 17, 18, 19, 20, 22, 24, 26, 28, 29, 30, 31, 32, 33, 34, 36, 38, 40, 42, 43, 44, 45, 46, 47, 48, 50, 52, 54, 56, 57, 58, 59, 60, 61, 62, 64, 66, 68, 70, 71, 72, 73, 74, 75;

d) Loading conditions a) and c) acting together;

e) Loading conditions b) and c) acting together.

Constraints are imposed on nodal displacements and member stresses.

This design problem is often presented as an example of fairly large-scale optimization. The structure has 77 nodes. The optimization includes 200 design variables (cross sectional area of truss members) and 3500 non-linear constraints. The Young's modulus of the material is $2.069 \cdot 10^{11}$ N/m^2 while the density is 7833.413 kg/m^3. The lower bound of the cross sectional area is set to 0.00064516 m^2 (0.1 in^2). The displacements of the free nodes must be less than 0.0127 m (0.5 in). The allowable stress (the same in tension and compression) is set to 21.092 kgf/mm^2 (30,000 psi). More details on geometry and loading conditions are given in the work of Venkayya (1978).

Interestingly, since the cost function of the truss problem is linear, the $\overline{\nabla} W$ gradient vector required by MPISA in global annealing cycles can be immediately determined and does not change through the entire optimization process.

In test case 2, the cantilevered truss structure shown in Fig. 11 is to be designed with constraints on nodal displacements, member stresses

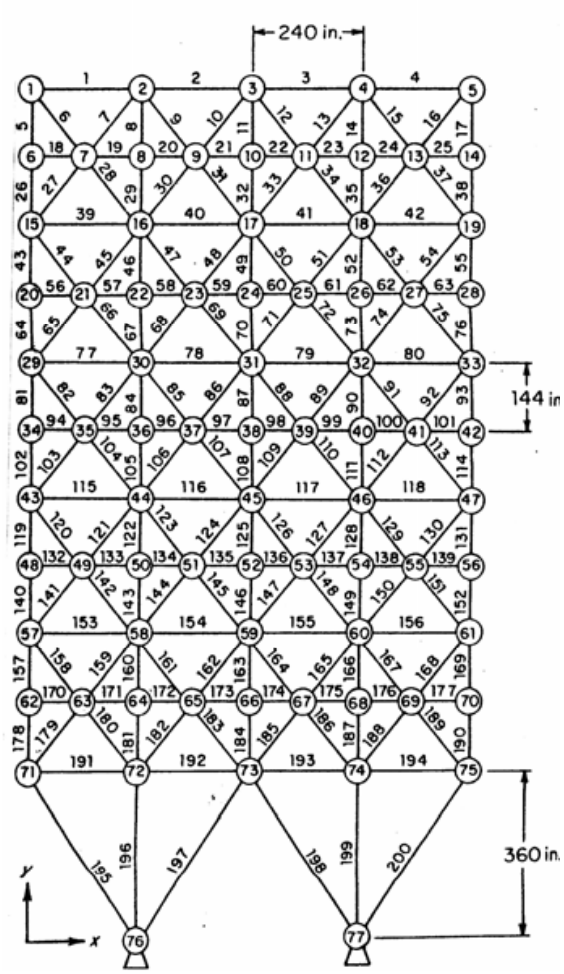


Figure 10: Schematic of planar 200-bar truss structure

and critical buckling loads. The structure has 45 elements and 20 nodes. Element numbering progresses as we move towards the tip of the structure (i.e., nodes 9-12). The truss is comprised of nine repeating modules each of which contains five elements. In addition, the nodal co-ordinates of the eighteen free nodes are included as configuration variables. Therefore, 81 design variables are considered. The Young's modulus of the material is $6.897 \cdot 10^{10}$ N/m^2 while the density is 2767.990 kg/m^3. Vertical forces are applied to the structure: respectively, 68038.856 kgf (150,000 lbf) at nodes 9 and 10 acting downward, and 22679.619 kgf (50,000 lbf) at nodes 11 and 12 acting upward. The lower bound of the cross sectional areas is again 0.00064516 m^2 (0.1 in^2).

A total of 162 non-linear constraints are considered. The allowable tensile stress is 17.577 kgf/mm^2 (25,000 psi), the stress limit in compression accounts also for buckling loads (Dhingra and Lee, 1994). The displacements of the free nodes must be less than 0.0508 m (2 in).

As far as it concerns the computation of the $\overline{\nabla} W$ gradient vector, it should be noticed that cost function sensitivities with respect to configuration variables can be easily obtained in fashion of closed form expressions while sizing variable sensitivities are obtained in the same way as in the two-hundred bar truss case.

Optimization runs started from both lower and upper bounds of cross sectional areas where the former is set as 0.00064516 m^2 (i.e., 0.1 in^2) while the latter is set as 0.64516 m^2 (i.e., 100 in^2).

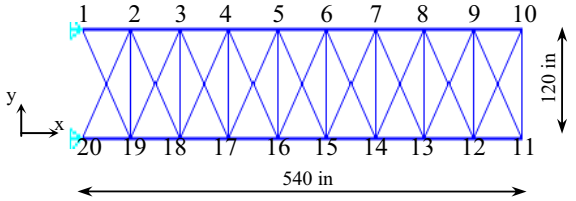


Figure 11: Schematicof 45-member cantilevered bar truss

MPISA is compared to the ISA algorithm recently developed by the present authors. Although ISA is based on the same idea of alternating global and local annealing cycles, MPISA now includes a substantially improved formulation.

The two truss structures have been optimized also with the powerful Sequential Quadratic Programming (SQP) routine available in the commercial software Matlab, Version 6.5. The SQP method [Haftka, 1992; Rao, 1996] is a globally convergent technique which formulates and solves a series of approximate sub-problems. Quadratic approximation is used for the cost function while constraints are linearized.

3.2 Weight minimization of frame structures

The frame shown in Fig. 12 is to be designed for minimum weight. The frame has 30 nodes and 45 elements (nodes are numbered, elements not).

The elements are grouped into twelve groups: the first three groups include ten horizontal elements each (those located at the same y) while the other nine groups include two vertical elements each (those located at the same x). The frame is modeled by using I-beam elements: the schematic and the local reference system of the element cross-sections are also shown in Fig. 12. The geometric dimensions b, h, t_w and t_f of the segments of the cross-section of the elements of each group are included as sizing variables. Since four design variables are considered for each group, forty-eight sizing variables are considered. In addition, the co-ordinates of nodes 5, 8, 11, 14, 17, 20, 23, 26, 29, 6, 9, 12, 15, 18, 21, 24, 27 and 30 are included as configuration variables. Therefore, the frame structure is optimized with 84 design variables.

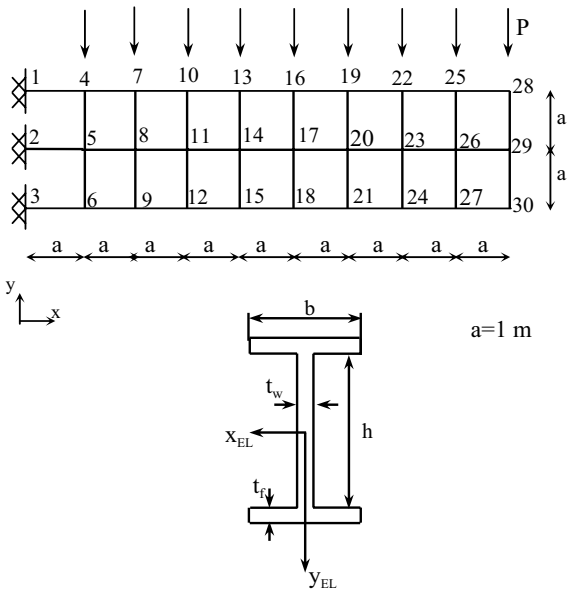


Figure 12: Schematic of frame structure and element cross-section nomenclature

The cost function is of the same type as that written for truss structures in fashion of Eq. (29). However, the cross sectional area x_j included as design variable for the j^{th} group of elements now depends on the cross-section dimensions b, h, t_w and t_f. That is: $x_j = 2 \cdot b \cdot t_f + h \cdot t_w$. Therefore, the cost function is yet linear with respect to each sizing variable but non-linear with respect to configuration variables. Cost function gradient can be easily obtained in fashion of closed form expressions.

The material density is 8303.971 kg/m^3 and the Young's modulus is $2.069 \cdot 10^{11}$ N/m^2. The frame is designed to carry 9071.8474 kgf (20,000 lbf) vertical forces P acting downward: the loads are applied at nodes 4, 7, 10, 13, 16, 19, 22, 25 and 28. A total of 324 constraints are imposed on nodal displacements (less than 0.0254 m; i.e., 1 in), Von Mises equivalent stresses at nodes (the stress limit is 17.577 kgf/mm^2; i.e., 25,000 psi) and Euler buckling. Geometric constraints are also imposed on the design variables: $0.005 \leq t_w/b \leq 0.1$, $0.005 \leq t_f/h \leq 0.1$, and $0.05 \leq b/h \leq 1$. The lower bound of the b and h dimensions is set to 0.01 m while the lower bound of the t_w and t_f thicknesses is set to 0.001 m. The initial configuration of the structure is shown in Fig. 12. Two different initial designs are considered by changing sizing variables: (i) the infeasible design is such that for all frame elements b=0.05 m, h=0.1 m, $t_w = t_f$=0.01 m; (ii) the feasible design is such that for all frame elements $b = h$=0.3 m, $t_w = t_f$=0.1 m.

MPISA is again compared to the SQP routine implemented in Matlab 6.5. Since the frame problem was proposed and solved for the first time by the present authors (see most recent results in Lamberti and Pappalettere, 2004), MPISA is now compared also to the LSTRLP optimization algorithm used in the aforementioned study. LSTRLP combined Sequential Linear Programming, line search and trust region techniques; approximate sub-problems were solved using the simplex routine. However, LSTRLP optimizations were run only for the infeasible starting design b=0.05 m, h=0.1 m, $t_w = t_f$=0.01 m. In order to reduce the computation cost entailed in the optimization process by the use of the simplex linear solver, the original LSTRLP algorithm is now combined with the well known commercial optimizer DOT® - developed by the Vanderplaats R&D - into an optimization code denoted as TRLP-DOT in the rest of the paper. In TRLP-DOT, move limits imposed on optimization variables are computed (notice that different lower and upper move limits are set for each variable) and each interme-

diate design is checked and eventually improved as prescribed by LSTRLP while the approximate sub-problem defined in the current iteration is solved using the feasible-directions based routine available in DOT®. Numerical tests reported in Genovese et al. (2005) demonstrate the good numerical efficiency of TRLP-DOT in optimizing complicated skeletal structures with many design variables.

4 Results and Discussion

The MPISA optimization algorithm has been implemented by a Fortran 90 code. Optimizations carried out with MPISA, Matlab 6.5 and TRLP-DOT have been run on a Dell Optiplex GX-260 Windows Workstation equipped by a single 2.66 Ghz Pentium IV CPU and 1 Gb of RAM memory. Some results relative to ISA and LSTRLP were obtained with different hardware configurations. Relative computation speeds are however indicated in Tab. 1 which summarizes results obtained in the different optimization problems.

The set of design problems solved in this research is certainly indicative since it included some examples of large-scale sizing optimization. Furthermore, combined sizing-configuration optimization is complicated by the fact that sizing and configuration variables belong to two well distinct design spaces. Several *ad hoc* schemes have been proposed in literature to solve this class of problems: tailored approximations of constraints (Hansen and Vanderplaats, 1990; Vanderplaats, 1998); evolutionary node shifting to simultaneously minimize displacement and/or stress values and weight penalty (Wang et al., 2002); setting of individual move limits for each design variable (Lamberti and Pappalettere, 2003 and 2004).

Non-convexity of design space in the test problems considered in this study comes from the wide variety of interactions which may be established between hundreds of design variables in the large-scale problem or between different types of variables in the sizing/configuration problems. Therefore the use of simulated annealing is very logical.

The real challenge in optimization is to identify which variable or group of variables really drives the overall design process in the different optimization cycles. It appears that this ability is intrinsically possessed by MPISA which alternates global and local search based on the nature of the current best record (i.e., far or close to constraint boundaries, feasible or infeasible, near to local minima or to non-convex regions, etc.) and explores simultaneously several different regions of the design space.

Although Matlab is not a dedicated optimization code and a considerable fraction of CPU time is spent in the communication between the optimizer and the routines written by the user for structural analysis, cost function and constraint definition, its built-in SQP routine proved to be very efficient in optimization of skeletal structures (Lamberti and Pappalettere, 2004). Indeed, comparison between MPISA and Matlab 6.5 is indicative in terms of ability of the different optimizers to find the global optimum regardless of the CPU time required in the design process. In structural optimization problems, weight or stress minimization is certainly the main goal of the entire design process and it is even more important than computational speed that may be too sensitive to programming skills and/or coding options.

Table 1 reports the optimized structural weight, number of optimization iterations (annealing cycles or approximate sub-problems), number of structural analyses and total CPU time required in the optimization process. The number of global annealing cycles performed by MPISA and ISA is also shown in the table within round brackets.

It appears that MPISA converged to the lowest structural weight either when the optimization process started from feasible points - the most active constraints are about 7.5%, 74% and 24% of allowable limits for the two-hundred bar truss, cantilevered bar truss and frame structure, respectively - or the initial design violated very much the optimization constraints (for the two hundred bar truss, 7600% violation on nodal displacements and 4850% violation on element stresses; for the cantilevered bar truss, 73900% violation on nodal displacements, 32000% violation on element stresses and 73332.5% violation on critical buckling load; for the frame, 1705% violation on

Table 1: Comparison of optimization results obtained with different algorithms

Test case	Initial design	Optimization Algorithm	Structural weight (kg)	Number of cooling (global) cycles / optimization iterations	Structural analyses	CPU time (s)
Two-hundred bar truss	Feasible	MPISA	12495.922	34 (25)	11232	1602
		ISA	12767.294	43 (21)	15411	3133*
		SQP-Matlab	12501.000	54	11914	3729
	Infeasible	MPISA	12494.949	39 (22)	12811	1921
		ISA	12764.179	40 (15)	18508	3613*
		SQP-Matlab	12503.300	127	24532	9102
Cantilevered bar truss	Feasible	MPISA	3318.273	25 (17)	5657	38.3
		ISA	3551.007	33 (17)	7785	60.6*
		SQP-Matlab	3357.520	56	4937	117
	Infeasible	MPISA	3320.378	38 (22)	7601	49.2
		ISA	3581.897	42 (19)	11350	83.7*
		SQP-Matlab	3397.120	60	5100	248
Frame	Feasible	MPISA	2489.143	36 (26)	3110	108
		SQP-Matlab	2513.756	55	4843	280
		TRLP-DOT	2648.545	13	1214	26
	Infeasible	MPISA	2501.851	37 (26)	3230	156
		SQP-Matlab	2532.104	43	3365	246
		TRLP-DOT	2692.923	26	2342	47
		LSTRLP+	2827.475	21	1815	98.3+

+ Lamberti and Pappalettere, 2004. This optimization was run on a CPU about 70% slower than that used in the present study.

* These optimization runs were performed on a CPU about 50% slower than that used in the present study.

nodal displacements, 1950% violation on element stresses and 396% violation on critical buckling load). Remarkably, designs optimized by MPISA are practically insensitive to starting point: in fact, the largest percent difference in weight is only 0.48% and occurred in the frame case.

Table 1 shows that reductions in structural weight achieved by MPISA with respect to optimized designs reported in literature are very significant: about 270 kg (2.2% weight saving) for the two-hundred bar truss structure; between 230 and 260 kg (7.5% weight saving) for the cantilevered bar truss structure; about 325 kg (13% weight saving) in the frame structure case.

In the two-hundred bar truss problem, the SQP routine implemented in Matlab could reach practically the same structural weight found by MPISA. However, in the sizing-configuration problems, the weight penalty seen for Matlab with respect to MPISA ranges between 1 and 2% and increases when the optimization runs started from infeasible designs.

In the frame problem, Sequential Linear Programming based optimizers (LSTRLP and TRLP-DOT codes) resulted considerably less efficient than MPISA and SQP-Matlab. In particular, TRLP-DOT performed better than literature but designed a structure about 190 kg heavier than MPISA.

The multi-point annealing search implemented in MPISA allowed to reduce the number of required optimization cycles with respect to the original code ISA. This improvement is more significant in case of feasible starting points. For all test problems, MPISA has run a considerably higher

number of global annealing cycles out of the total number of optimization iterations than in the ISA case: between 56 and 73% vs. 37-51%.

Among the different optimization algorithms considered in this study, the SQP routine implemented in Matlab required the largest number of design cycles overall. MPISA is the fastest code in the two truss problems. Finally, TRLP-DOT required the smallest number of optimization cycles in the frame problem but converged to sub-optimal designs.

It can be seen from Tab. 1 that the improved formulation implemented in MPISA required less structural analyses per optimization cycle than ISA: from 5-10% in case of feasible starting designs to about 30% in case of infeasible starting designs. This results is very significant if we consider that ISA itself was considerably less expensive than classical simulated annealing in terms of structural analyses.

MPISA seems very efficient in terms of computation time. In fact, the average CPU time required to complete one optimization cycle in MPISA is considerably lower than for SQP-Matlab which must solve a quadratic sub-problem in each optimization iteration. However, MPISA is slightly slower than the original code ISA developed by the present authors. Although this is not a limitation since MPISA found better designs than ISA, we have to remind that the considerably large number of global annealing cycles performed by MPISA using the multi-point search strategy led to repeat many times the task of constructing the Ω domain that includes the descent directions where candidate designs lie. Such operation has been demonstrated in Section 2 to be not expensive in terms of required structural analyses. However, it may entail solving many linear systems - up to 2^N-1 - in fashion of Eq. (3) each of which is formed by N equations.

In order to gather more evidence of the relative performance of different optimization algorithms considered in this study, convergence curves recorded in the optimization process and the corresponding constraint margins are respectively presented in Figs. 13-15 and Figs. 16-18.

Figures 13 and 14 prove that in the truss problems MPISA converged to the optimum designs more quickly than ISA. As is clear, the multi-point search strategy implemented in MPISA allows to explore a larger fraction of design space and hence results in a higher probability of reducing the cost function already in the first optimization cycles. Consequently, it is much easier to approach the region containing the optimum design.

Furthermore, MPISA convergence curves did not exhibit steps where the cost function improves marginally. At least, cost steps cover a much smaller number of annealing cycles than in the ISA case. This behavior, more evident if the optimization started from a feasible design, can be explained in view of the fact that MPISA now combines the multi-point search strategy (Step 3) with a local annealing search accounting for constraint domain non-convexity (Step 7).

The reason why the convergence behavior of MPISA is considerably better than that exhibited by SQP-Matlab when the optimization started from a feasible design is that the present code perturbs the design variables by moving along descent directions. However, while Matlab can explore only the descent direction which results from the solution of the current approximate sub-problem, MPISA is able to generate and analyze simultaneously a much larger number of descent directions (at most 2^N-1).

Data reported in Tab. 1 suggest the following interesting question. The difference in optimized weights between MPISA and SQP-Matlab seen in the two-hundred bar truss problem is indeed very marginal (less than 0.1%). It is generally acknowledged that minor improvements in weight in large-scale problems including hundreds of design variables may result from the interaction between constraint tolerances and permitted constraint violations rather than by differences in algorithm formulations. However, the present authors want to point out that no constraint violation was permitted in this study. Therefore, any penalty in structural weight indicates that the corresponding algorithm may have missed the global optimum. This statement is supported by the analysis of the constraint margins plotted in Fig. 16.

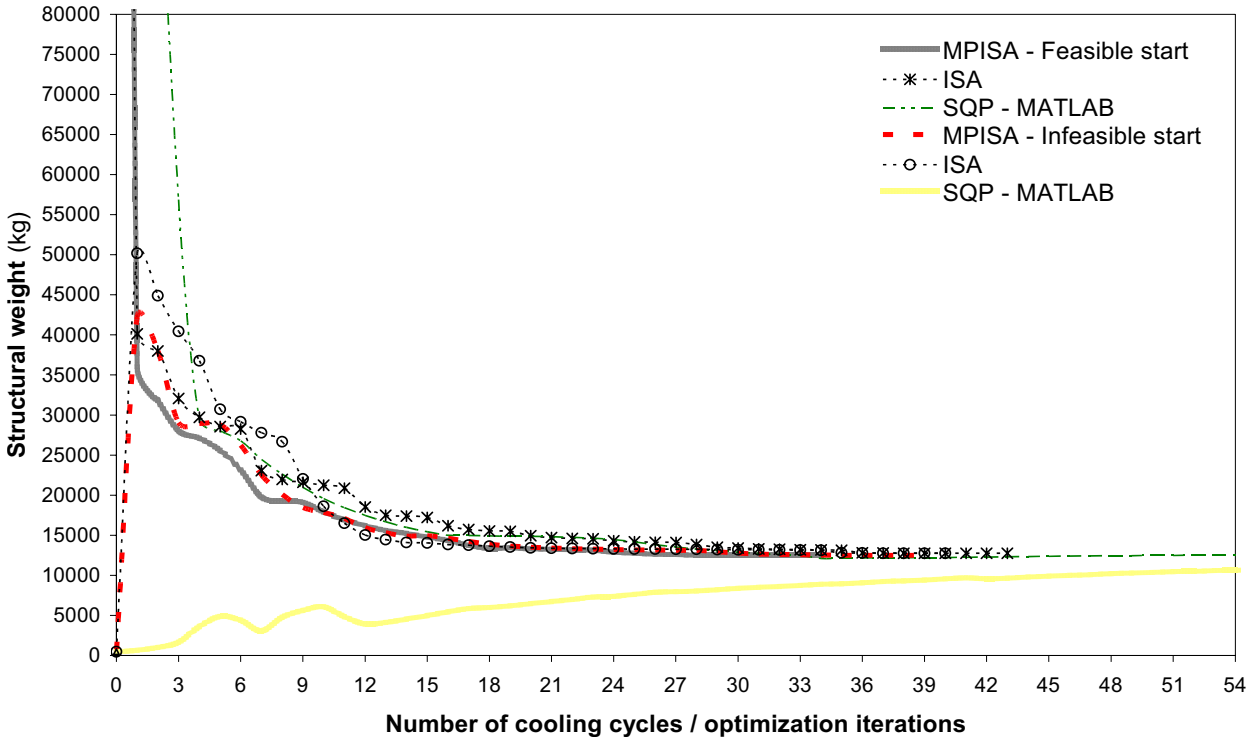


Figure 13: Comparison of convergence curves obtained in the case of the two-hundred bar truss structure

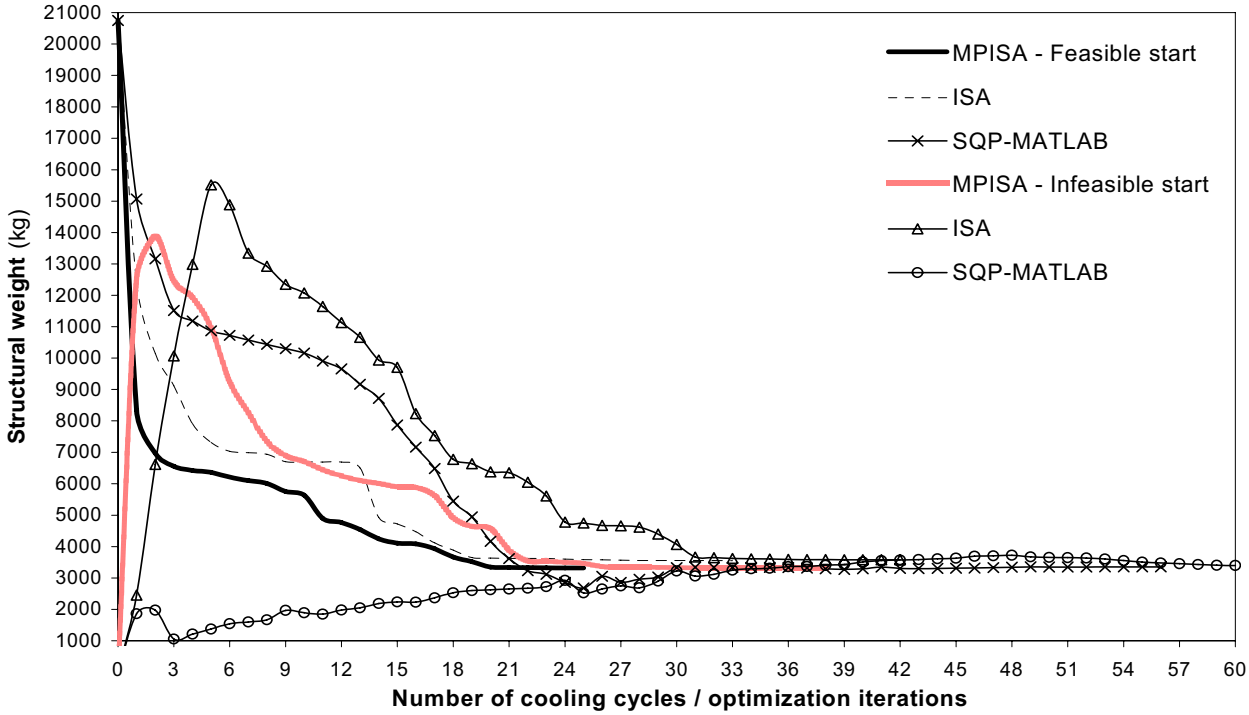


Figure 14: Comparison of convergence curves obtained in the case of the forty-five bar structure

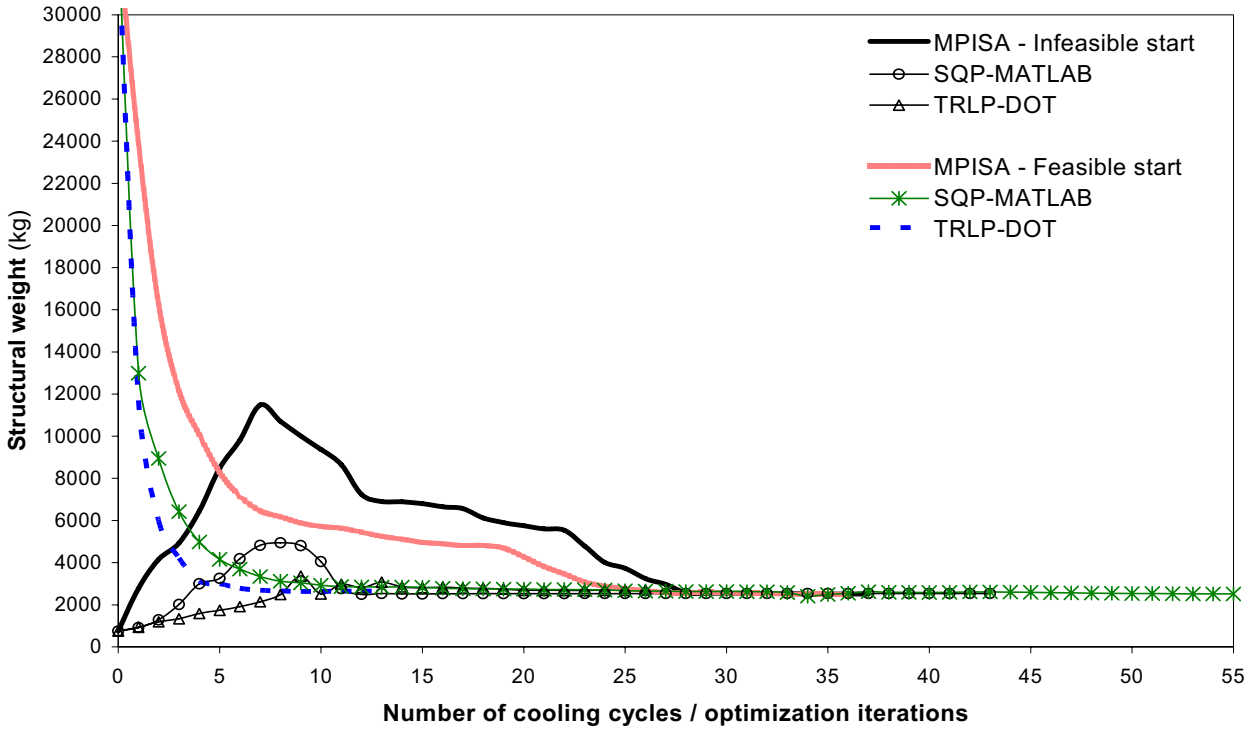


Figure 15: Comparison of convergence curves obtained in the case of the frame structure

It can be seen that MPISA was able either to recover immediately any constraint violations and to keep the design search process inside a feasible region throughout the optimization process. Conversely, SQP-Matlab yet exhibited 9% constraint violation after 33 design cycles when the optimization started from a feasible design and required more than 60 iterations to find a feasible intermediate design when the optimization process started from an infeasible point. Therefore, the SQP-Matlab optimizer had repeatedly to pay some weight penalty in order to reduce constraint violation in the subsequent iterations.

In the frame problem, SQP-Matlab and TRLP-DOT seem much faster than MPISA in reducing structural weight when the initial design is feasible (see Fig. 15). However, SQP-Matlab and TRLP-DOT perturbed the design by moving along feasible directions which are too steep. Consequently, some infeasible intermediate designs have been generated very soon (see constraint margins plotted in Fig. 18). Such a fast reduction in cost involved very large step sizes thus deteriorating the quality of the approximation. Hence, TRLP-DOT had to repeatedly shrink the move limits while SQP-Matlab had to reduce step sizes along the descent direction obtained solving the approximate sub-problem in order to improve the quality of approximation in the subsequent sub-problems. Nevertheless, TRLP-DOT and SQP-Matlab, respectively, generated infeasible intermediate designs yet after about 15 and 35 optimization iterations even though the initial design satisfied largely the optimization constraints. Oscillatory behavior of constraint margins is certainly due to local non-convexity of design space where the set of active constraints driving the optimization process changes sharply. Since MPISA is the only algorithm explicitly formulated for dealing with this problem as it checks for local non-convexity caused by each single design variable and attempts to find anyhow the best feasible design regardless of the local shape of constraint domain (see Step 7), the present code could generate feasible intermediate designs in most of the cooling cycles performed in the optimization process.

Insensitivity of MPISA to sharp changes in the set

of active constraints derives from its inherent ability to generate intermediate designs close or far enough to constraint domain boundaries based on the degree of local non-convexity of design space (see descriptions of Steps 3 and 7). This operation is performed for each optimization variable and greatly increases the design freedom. Hence, there is a much higher probability of capturing the type and set of constraints which will be effectively active at the optimum. For instance, Figure 17 clearly proves how MPISA could shift continuously from one set of active constraints to another during the optimization process. In the cantilevered bar truss problem, constraints on nodal displacements became very soon critical in the first design cycles. However, when buckling constraints started to turn active, displacement and stress constraints have been somehow relaxed and vice versa. Some intermediate designs generated by MPISA even resulted uncritical with respect to each different type of constraints.

Table 1 shows that optimum designs obtained in the sizing-configuration problems for infeasible starting points are generally heavier than those obtained in case of feasible starting points. In the cantilevered bar truss problem, weight penalty is marginal only for MPISA (about 0.06%) while is much larger for ISA and SQP-Matlab: respectively, about 0.9 and 1.2%. In the frame problem, the weight penalty raised from 0.48% seen in the MPISA case to 0.75% and 1.7% respectively for SQP-Matlab and TRLP-DOT. These results can be explained in view of the large oscillations in constraint margins that occur for SQP-Matlab (Fig. 17) and TRLP-DOT (Fig. 18). In particular, the largest constraint violation exhibited by SQP-Matlab at an intermediate design occurred in the cantilevered bar truss problem: 80% and 1000%, respectively, for feasible and infeasible initial designs (see Fig. 17). In the frame case, the largest constraint violation at an intermediate design generated by TRLP-DOT is about 60%. Buckling constraints continuously shifted from critical to uncritical and vice versa when the optimization started from an infeasible point (see Fig. 18).

As has been already remarked, oscillations of constraint margins seen for SQP-Matlab and TRLP-DOT are caused by poor approximations that fail in capturing the local non-convexity of design space and cannot follow sharp changes in the active constraint set. Therefore, step sizes or move limits had to be reduced by SQP-Matlab and TRLP-DOT in order to improve the quality of approximation in the subsequent iterations. However, this strategy forced the gradient based optimizers to narrow the portion of design space explored in each optimization iteration. This finally led to weight penalty because the initial design is very far from the optimum as it strongly violated constraints. Conversely, the random search process utilized in simulated annealing is intrinsically more able than gradient based optimizers to explore a sufficiently large fraction of design space regardless of the position of the starting point with respect to the optimum point. This statement is well supported by the fact that even the original simulated annealing code ISA which did not include any multi-point search strategy is less sensitive to initial design than SQP-Matlab although ISA itself found just sub-optimal designs. As is clear, the multi-point search implemented in MPISA made the present code even more insensitive to initial design since extended further the fraction of design space explored by the optimizer.

Generally speaking, when the initial design violates strongly the optimization constraints as it happens in the present study, weight history may follow two well distinct trends: (i) the optimizer reduces slightly the constraint violation and increases cost function a little in each new design cycle; (ii) the optimizer increases much the cost function already in the first design cycle in order to move back quickly to the feasible space. Indeed, both strategies (i) and (ii) could not be successful in reaching the global optimum. In the former case, the optimizer can get stuck in a local minimum for which some active constraint dominates much the design process. Furthermore, a very large number of iterations may be required in order to complete the optimization process. In the latter case, if the cost function increased much, the optimizer may jump into a region very far

from the optimum and a considerable number of extra cycles where no constraints are active could be required in order to reduce structural weight. Oscillatory behavior could be encountered when the type of the most violated constraint changes as the optimizer approaches the constraint domain boundary. This may be the case of a locally non-convex constraint domain.

Figures 13-15 show that the gradient-based optimizers such as SQP-Matlab and TRLP-DOT slowly increased the cost function while MPISA and ISA increased suddenly the structural weight already in the first design cycles. In particular, analysis of constraint margins plotted in Figs. 16-18 reveals that SQP-Matlab could find the first feasible intermediate design after about 60, 45 and 20 optimization cycles, respectively, for the two-hundred bar truss, cantilevered bar truss and frame structure problems. Similarly, TRLP-DOT required more than 20 iterations in the frame problem and exhibited yet 30% constraint violation at the iteration 19 (see Fig. 18). Conversely, MPISA already satisfied optimization constraints after only 2 annealing cycles in the truss problems and 7 annealing cycles in the frame problem.

Remarkably, MPISA was able to contain the increase in weight with respect to the initial design: about 8000 kg (i.e., 65% of the optimum weight) less than ISA in the two-hundred bar truss problem and about 1650 kg (i.e., 50% of the optimum weight) less than ISA in the cantilevered bar truss problem (see again Figs. 13-14). Furthermore, MPISA performed less annealing cycles than ISA in which cost function increases. The improvement is very evident in the cantilever bar truss problem (see Fig. 14). This behavior was somehow expected since MPISA approaches the boundaries of constrain domain more quickly than ISA but, at the same time, now searches for feasible segments limited by linearized constraints yet accounting for constraint domain non-convexity, and explores then these segments in order to improve the current best record.

A careful analysis of MPISA optimization histories reveals that convergence curves corresponding to feasible or infeasible starting designs coincided after only 9 annealing cycles in the two hundred bar truss case while much more cycles (25-30) had to be performed in the sizing-configuration problems. The first argument that can be made in order to explain this difference is that while for the large scale problem the ratio $\kappa_{INIT-OPT}$ between the initial and optimum structural weights does not change much in case of feasible or infeasible initial designs (36 vs. 27), in the sizing-configuration problems the same ratio changed significantly (6.3 vs. 160 - cantilevered bar truss - or 3.4 vs. 13.5 - frame). Another argument is that the large scale problem included much more design variables than the sizing-configuration problems. However, both these arguments do not seem completely exhaustive. In fact, in the ISA case, convergence curves coincided after 32-33 annealing cycles for both truss problems regardless of the number of design variables and values of the $\kappa_{INIT-OPT}$ ratio.

In reality, one should consider the larger design freedom introduced by having two well distinct design spaces - sizing and configuration variables - and the way in which MPISA performed the random search. Since MPISA carried out a very large number of global cycles in the optimization runs (see Tab. 1), cost function gradients have been repeatedly evaluated in the optimization process. The perturbation given to each design variable is initially "weighted" (see Eq. (2) of Step 3) by the ratio μ_j between the corresponding cost function sensitivity and the magnitude of the cost function gradient. The μ_j coefficients do not change for sizing variables while do obviously depend on the current value taken by each configuration variable. Although the same argument could be made in principle also for ISA, this algorithm did not include the multi-point search strategy implemented in MPISA which uses cost function gradient information also for accepting/modifying trial design points based on trust region or linearization error models. This strategy allowed us to reduce the number of annealing cycles until optimization histories coincide (in the worst case, MPISA was however 30% faster than ISA) but made MPISA optimization history more sensitive to the number of configuration variables.

Another important information that can be de-

rived from constraint margins plotted in Figs. 16-18 is which types and set of constraints are active at the optimum design. For instance, Fig. 16 shows that in the two-hundred bar truss case displacement constraints are always active during the entire optimization process. Conversely, stress constraints became soon inactive or never turned active. However, safety margin computed for MPISA resulted considerably smaller than that relative to ISA: from about 63% (ISA) to less than 25% (MPISA).

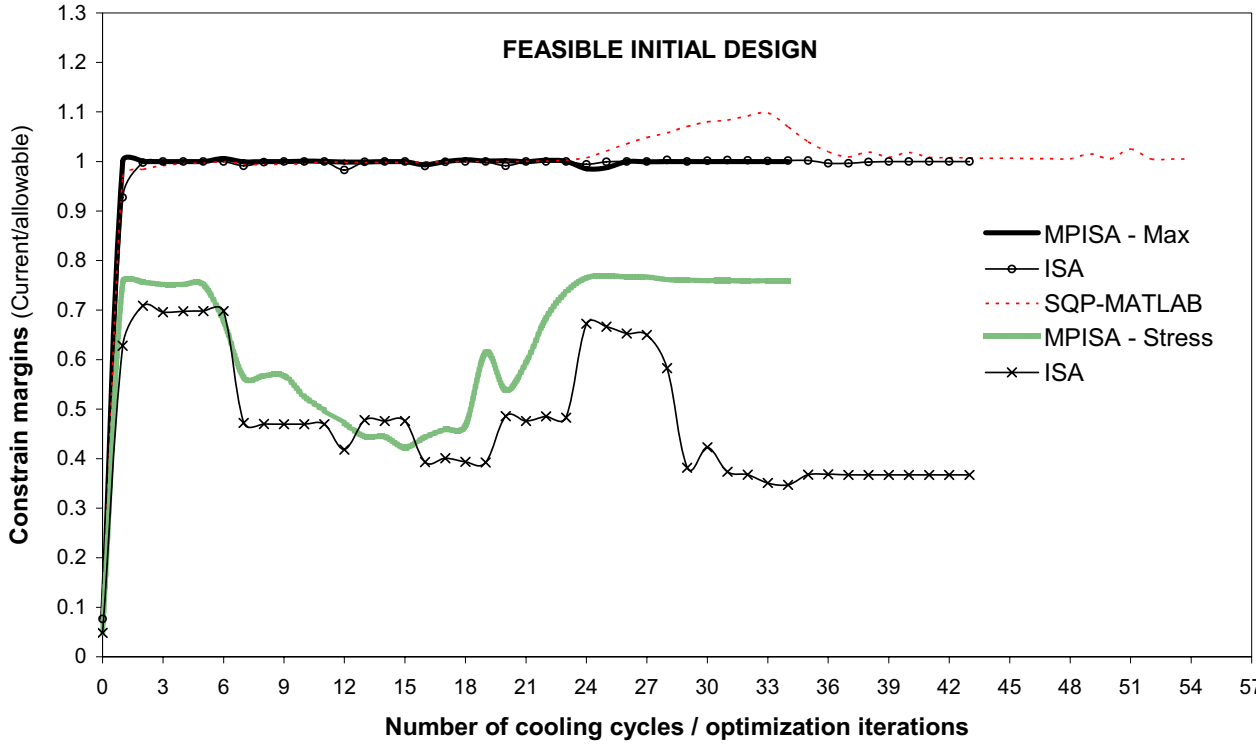
In the two sizing-configuration problems (cantilevered bar truss and frame structures), displacement, stress and buckling constraints are all active for the designs optimized by MPISA and SQP-Matlab (see Figs. 17-18). However, Fig. 18 shows that the optimum design found by TRLP-DOT for the infeasible starting point is not critical in buckling (about 67% safety margin). Therefore, such design is clearly sub-optimal. This statement is supported by the fact that in the early design cycles, both MPISA and TRLP-DOT found intermediate designs for which buckling constraints are not critical (about 80% safety margin). However, while TRLP-DOT exhibited large oscillations in buckling constraint margins and finally converged to the uncritical safety margin of 67%, MPISA gradually turned the buckling constraint margin critical.

Figure 19 presents some of the designs optimized in the truss problems. Similar trends have been observed in the frame structure case - and therefore are not reported in the paper in order to save space - although element grouping strongly reduced the design freedom. The different loads applied to the two planar trusses generate a sort of resultant bending state. Consequently, the optimizer distributed stiffness in different elements trying to design structures with a rather uniform resistance. Stiffer elements are designed near the nodes where displacement constraints are applied. Stiffness progressively decreases as we move towards the tip of the structure (i.e., elements 1-4 and 40-45, respectively, for the two-hundred and cantilevered bar truss).

Distributions of optimized variables show that "thick" (i.e., with a large cross-sectional area) elements are surrounded by thinner elements (some cross-sectional areas are even at their minimum gage) in order to distribute stresses in the structure more or less uniformly. It can be seen that MPISA has been able to reduce the area size for most of the "thick" elements. This explains why MPISA could save weight with respect to ISA. Interestingly, some very stiff elements have been designed by SQP-Matlab even though the optimized weight is very close to that found by MPISA. This is because the commercial optimizer reduced constraint violation for some critical zone of the structure by increasing the corresponding design variables but did not have enough design freedom to recover the extra penalty in weight yield by constraint relaxation. Such a behavior is perfectly consistent with constraint margin oscillations seen in the Matlab optimization histories.

More indications on sensitivity of optimized designs to initial point are provided by Figs. 20 and 21, respectively for the truss structures and the frame. The top of Fig. 20 shows the distribution of the ζ_{SIZ} ratio between the optimized cross section values obtained when the optimization is started from a feasible point and an infeasible point, respectively. For a given element, $\zeta_{SIZ} = 1$ indicates that the optimized design is insensitive to the starting point. It is apparent from the figure that MPISA achieved the smallest dispersion (about 1.01%) of the ζ_{SIZ} parameter around 1 vs. the 2.5% dispersion seen in the ISA case. As expected, SQP-Matlab is the optimization code most sensitive to starting design since it is based on gradient evaluations.

In the cantilevered bar truss case, plots in the bottom of Fig. 20 show also the distribution of the ζ_{LEN} ratio between the optimized lengths of each truss element for the two initial designs (i.e., feasible or infeasible). It appears that MPISA is again the most robust algorithm overall. In fact, the dispersion on element cross sectional area is just 1% vs. 3.9% seen in the ISA case. The dispersion on element length is practically the same as in the ISA case: elements in the optimized configuration become about 1% longer if the initial design is infeasible. Therefore, the multi-point simulated annealing optimizer is also able to deal



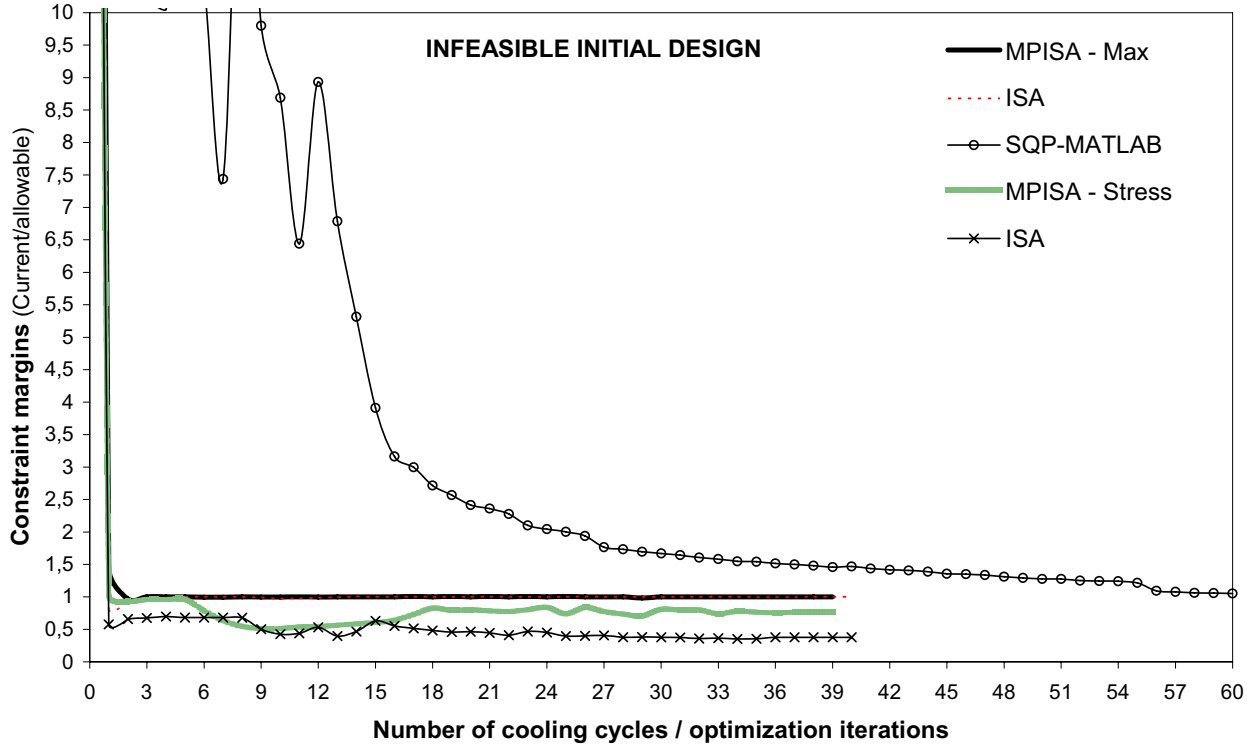


Figure 16: Evolution of constraint margins in the optimization of the two-hundred bar truss structure

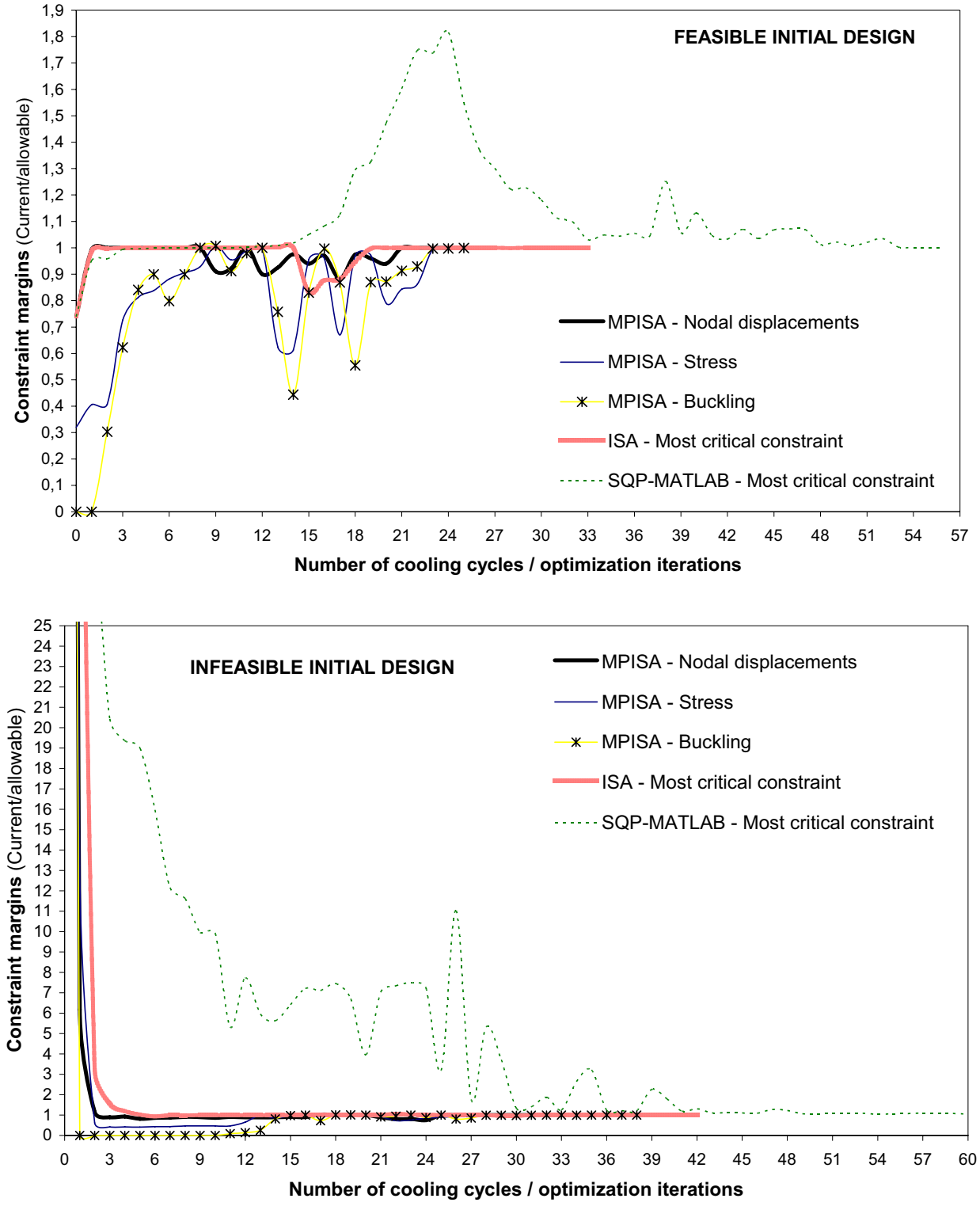


Figure 17: Evolution of constraint margins in the optimization of the forty-five bar truss structure

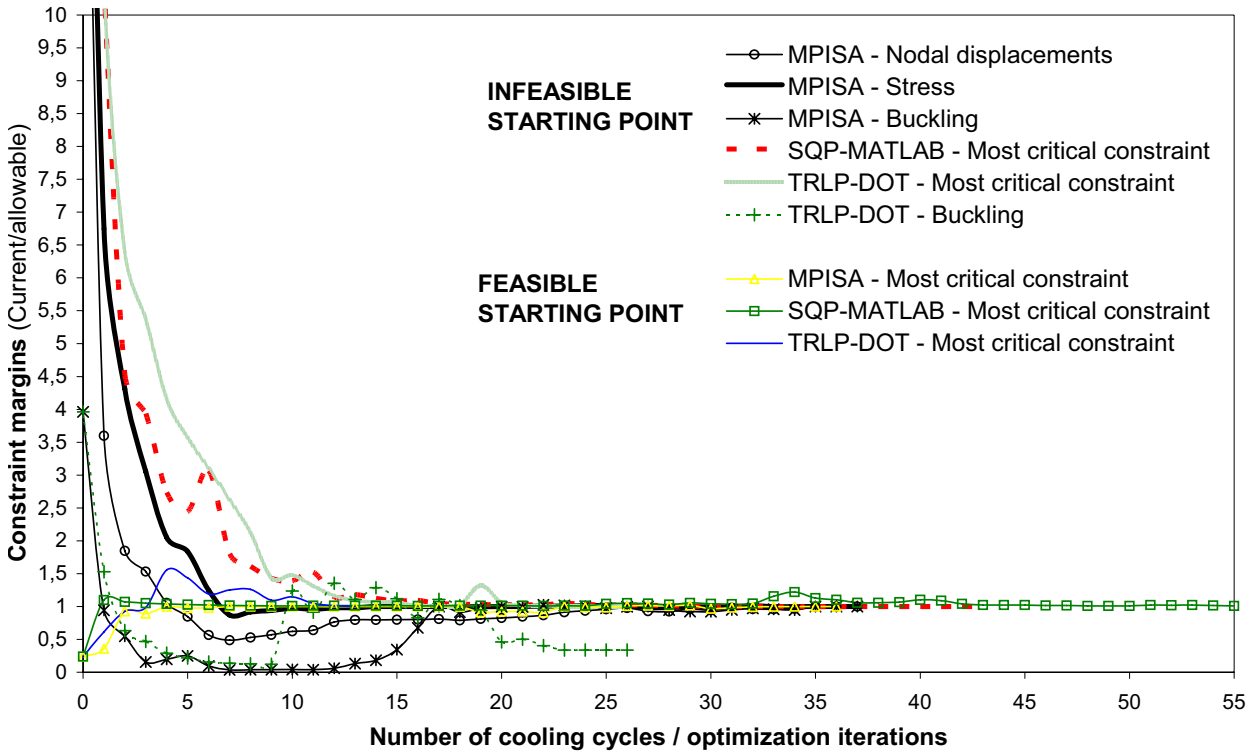


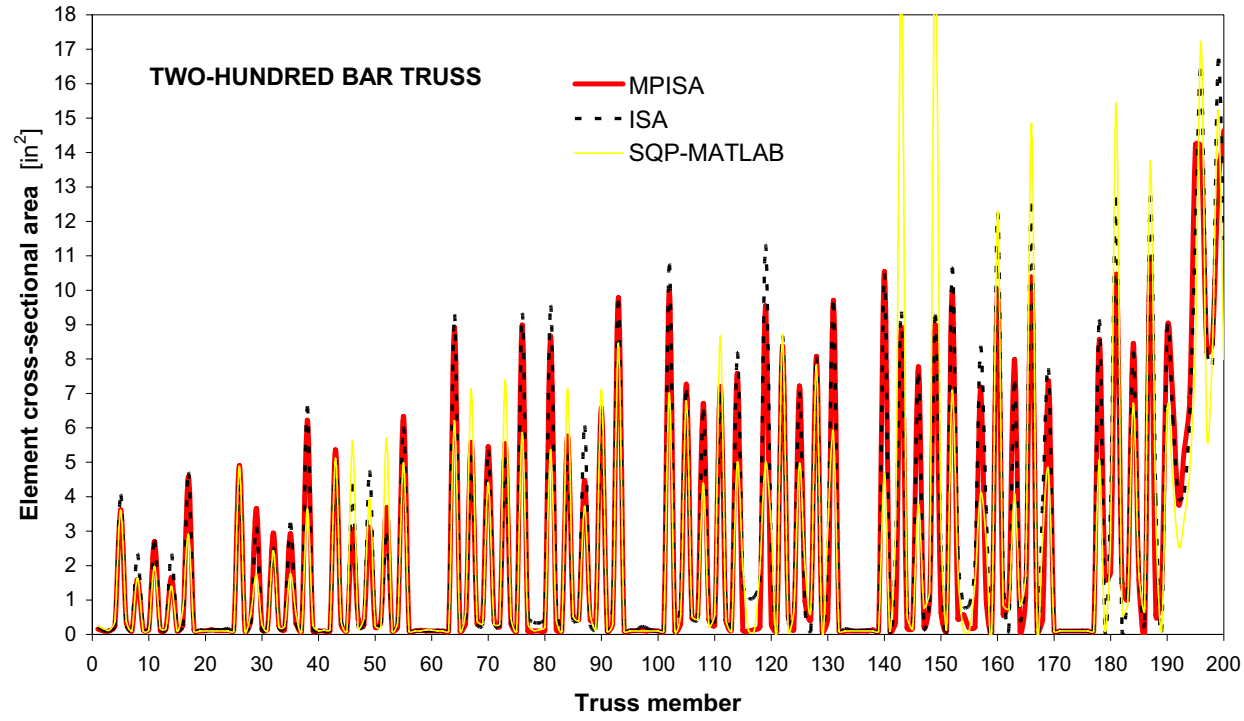
Figure 18: Evolution of constraint margins in the optimization of the frame structure

efficiently with the buckling constraints in spite of the large initial violation of 73332.5%. The SQP-Matlab optimizer again exhibited the largest statistical dispersion: the element length is more sensitive to initial design than element cross section.

Finally, Fig. 21 shows for the frame case the trends of the ζ_{LEN} ratio evaluated for the element length and the ζ_{INE} ratio evaluated for the element moment of inertia. The moment of inertia can be obviously computed only for each of the 12 groups into which frame elements have been divided. It appears that MPISA is the most robust optimization algorithm: 1% dispersion on element length vs. 4-5% seen for SQP-Matlab and TRLP-DOT; 4.3% dispersion on element moment of inertia vs. 24-38% seen for SQP-Matlab and TRLP-DOT. The considerably lower dispersion on element length and moment of inertia exhibited by MPISA with respect to the gradient based optimizers is indicative of the inherent ability of the proposed optimization algorithm to "capture" how sensitive may be the design process to each optimization variable. This fact also explains why MPISA however designed the lightest frame structures yet having all types of constraints active at the optimum.

5 Summary and Conclusions

This paper described a novel optimization algorithm implementing an advanced formulation of Simulated Annealing. The algorithm - denoted as MPISA (Multi Point Improved Simulated Annealing) - has a multi-level architecture combining global and local annealing search mechanisms. In order to speed up convergence, MPISA includes gradient information and generates a random set of descent directions. This allows to analyze simultaneously different candidate designs rather than a single trial point. Local annealing where optimization variables are perturbed one by one is performed each time global annealing did not yield significant improvements in design. Trial designs close or far from constraint boundaries are generated based on the degree of local non-convexity of constraint domain. Trust region



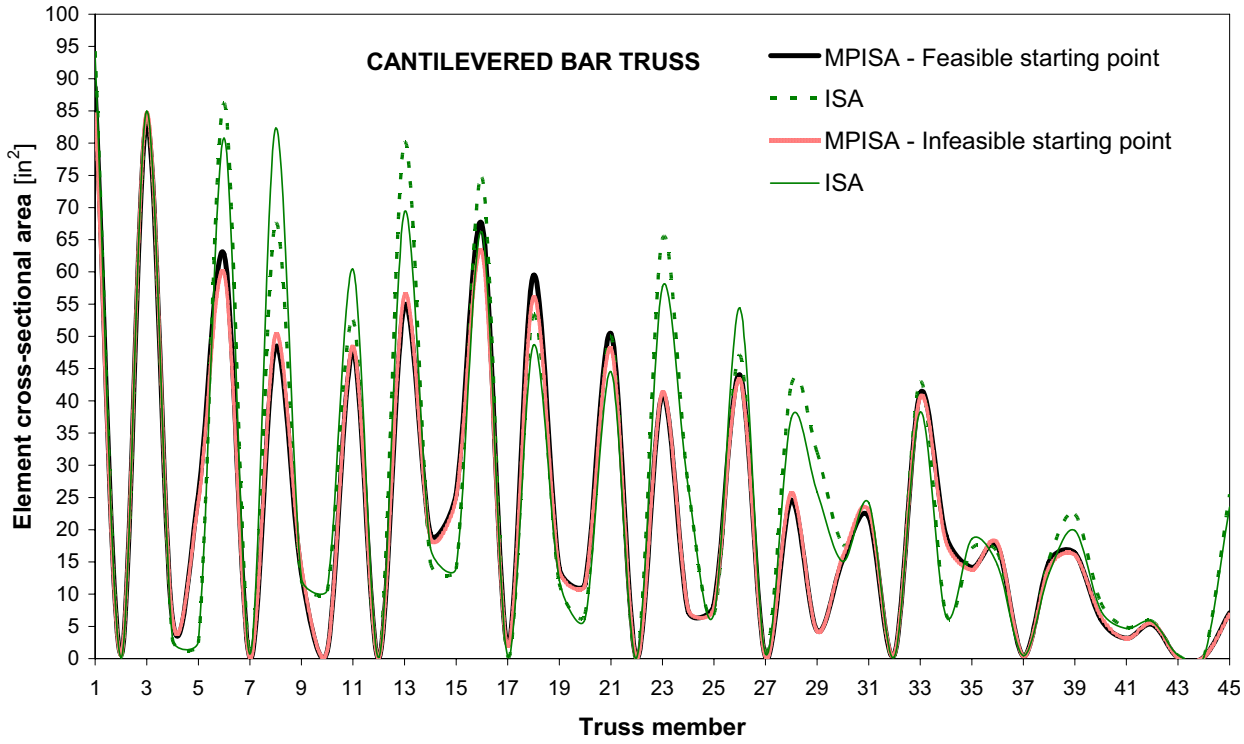
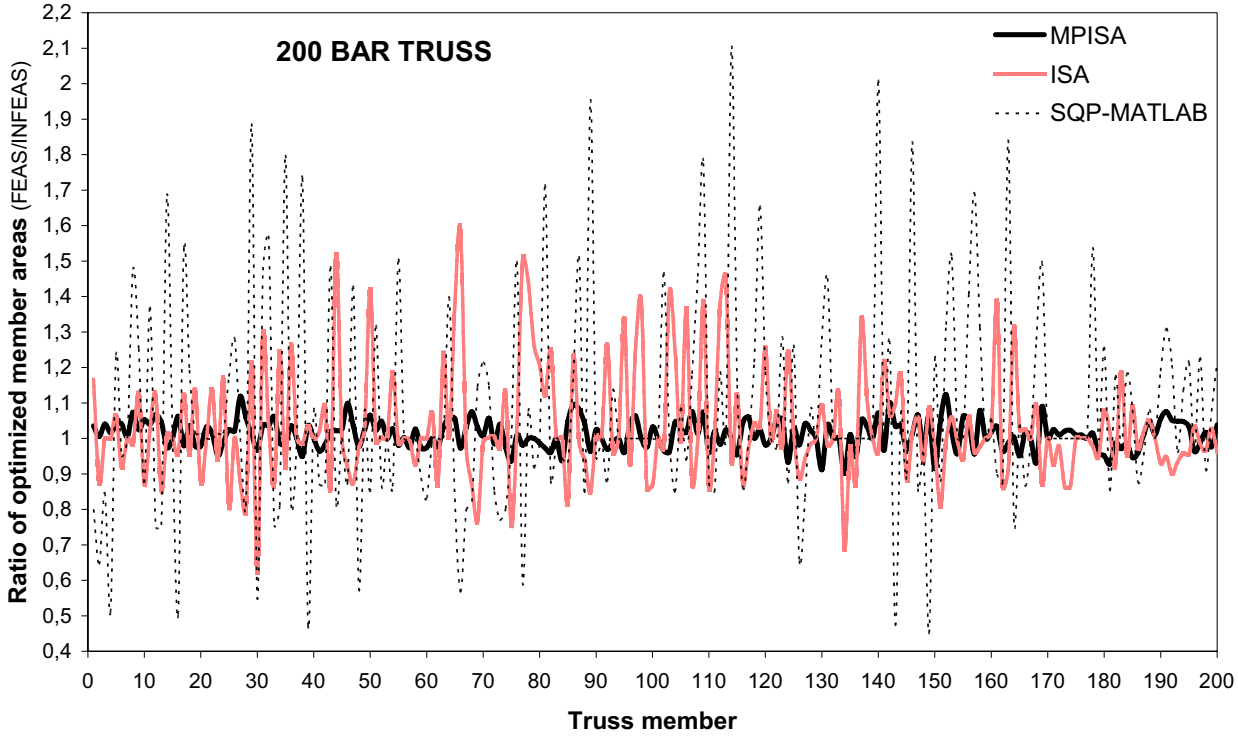


Figure 19: Comparison of optimized designs obtained in the truss problems



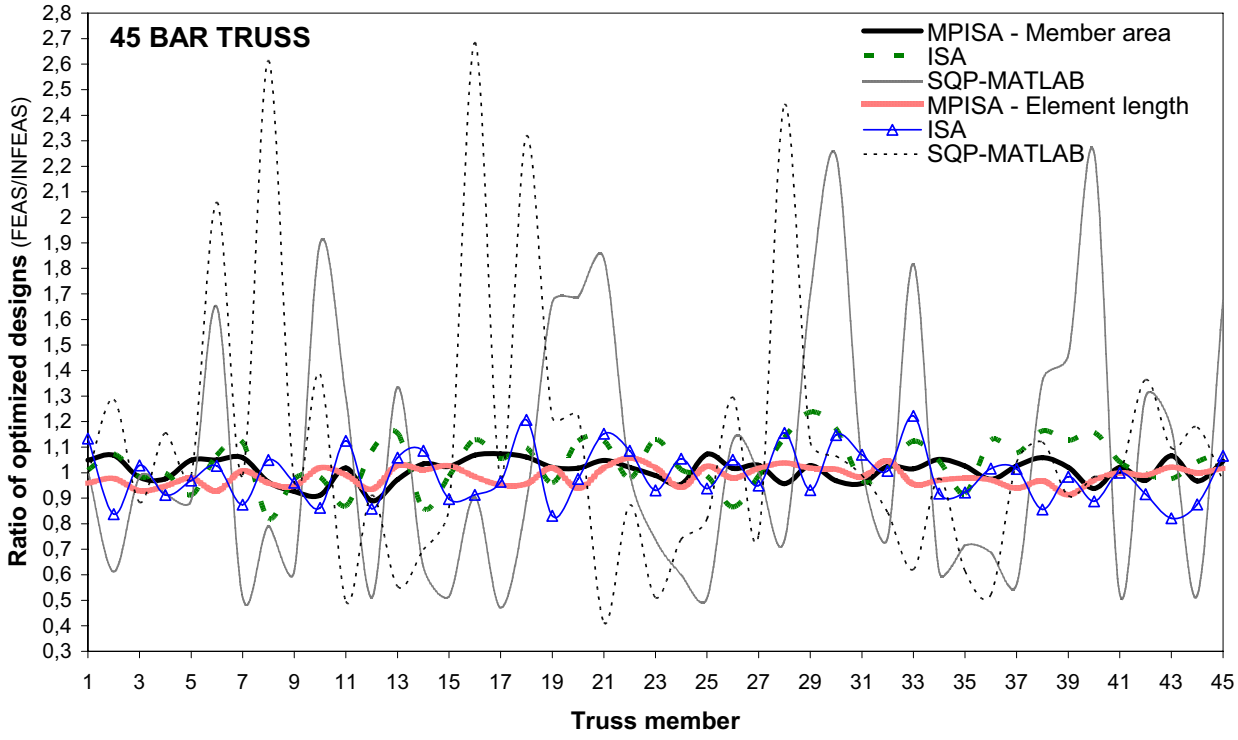


Figure 20: Sensitivity of optimized designs to starting point in truss problems

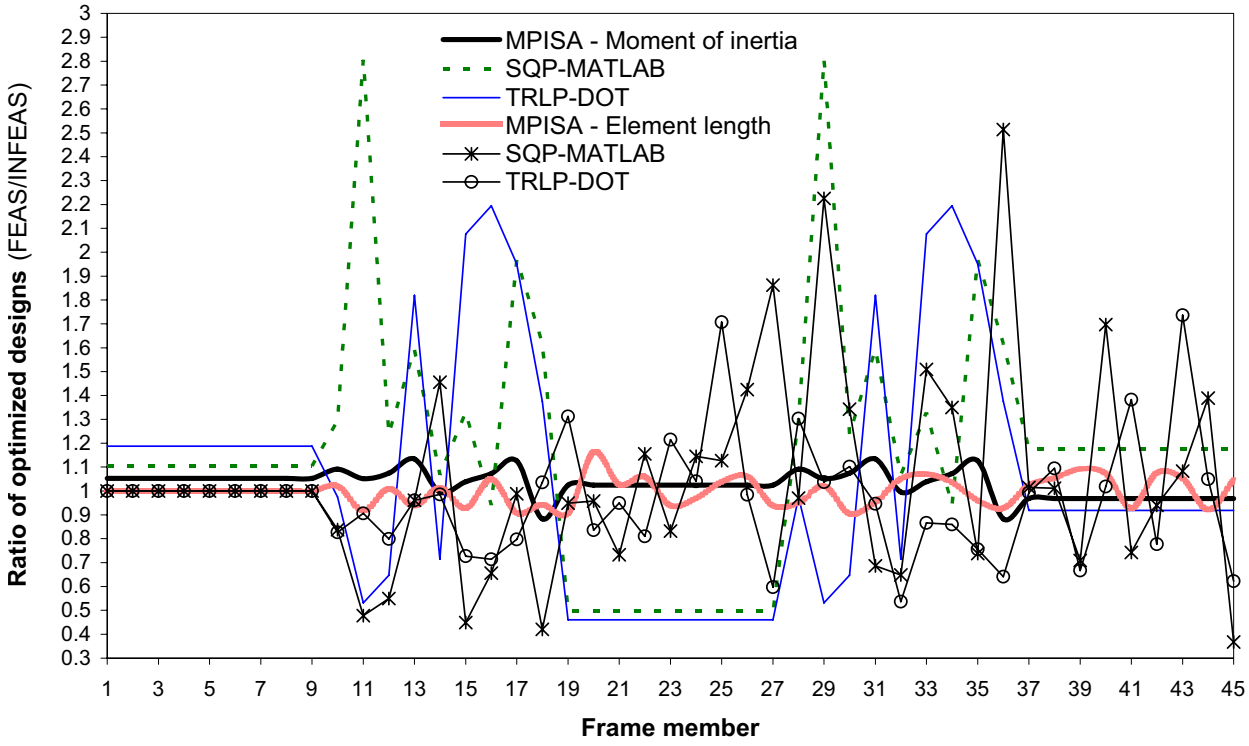


Figure 21: Sensitivity of optimized design to starting point for the frame structure

and linearization error based schemes are utilized to accept/modify candidate designs. Multi-point search strategy is used also when intermediate designs end up infeasible. Finally, the cooling schedule implemented in MPISA reduces adaptively the temperature based on the optimization history.

MPISA was tested in three complicated structural optimization problems exhibiting non-convex and/or non-smooth behavior: (i) weight minimization of a large-scale truss structure with 200 design variables and 3500 non-linear constraints; (ii) sizing-configuration optimization of a cantilevered bar truss with 81 design variables; (iii) sizing-configuration optimization of a frame structure with 84 design variables. This set of problems can be certainly considered indicative also in view of the fact that in all test problems gradients are not available explicitly.

MPISA was compared to another state-of-art simulated annealing code - ISA -, gradient based optimizers recently presented in literature and commercial software. The ISA code from which MPISA has been derived adopted the same multi-level architecture of MPISA but without including the multi-search strategy and the local non-convexity detection strategy implemented in MPISA.

Results indicate that MPISA was the most efficient optimization algorithm and achieved considerable reductions in structural weight with respect to designs reported in literature. Furthermore, MPISA resulted very competitive or performed even better than commercial gradient based optimizers.

The complexity and variety of the optimization problems successfully solved in this study certainly support the conclusion that MPISA is a very powerful code for design optimization of structures. However, the present authors point out that MPISA should be tested further in other structural optimization problems in order to draw more general conclusions.

References

Alexandrov, N.; Dennis, J.E.; Lewis, R.M.; Torczon, V. (1998): A trust region framework for

managing the use of approximate models in optimization. *Structural Optimization*, vol. 15, pp. 16-23.

Aymerich F; Serra, M. (2006): An Ant Colony Optimization algorithm for stacking sequence design of composite laminates. *CMES: Computer Modeling in Engineering and Sciences*, vol. 13, pp. 49-65.

Baumann, B.; Kost, B. (2005): Structure assembly by stochastic topology optimization. *Computers and Structures*, vol. 83, pp. 2175-2184.

Blachut, J. (2003): Optimal barreling of steel shells via simulated annealing algorithm. *Computers and Structures*, vol. 81, pp. 1941-1956.

Bushnell, D. (1996): Recent enhancements to PANDA2 (A code for minimum weight structural design). *Proceedings of the 37th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, AIAA Paper No. 96-1337.

Clerc, M. (2006): *Particle Swarm Optimization*. ISTE Publishing Company, London (UK).

Dhingra, A.K., Lee, B.H. (1994): A genetic algorithm approach to single and multiobjective structural optimization with discrete-continuous variables. *International Journal for Numerical Methods in Engineering*, vol. 37, pp. 4059-4080.

Dorigo, M.; Stutzle, T. (2004): *Ant Colony Optimization*, MIT Press, Cambridge (USA).

Engelbrecht, A.P. (2005): *Fundamentals of Computational Swarm Intelligence*, John Wiley & Sons, Chichester (UK).

Erdal, O.; Sonmez, F.O. (2005): Optimum design of composite laminates for maximum buckling load capacity using simulated annealing. *Composite Structures*, vol. 71, pp. 45-52.

Fedelinski, P.; Gorski, R. (2006): Analysis and optimization of dynamically loaded reinforced plates by the coupled boundary and finite element method. *CMES: Computer Modeling in Engineering and Sciences*, vol. 15, pp. 31-40.

Genovese, K.; Lamberti, L.; Pappalettere, C. (2005): Improved global-local simulated annealing formulation for solving non-smooth engineering optimization problems. *International Journal of Solids and Structures*, vol. 42, pp. 203-237.

Goldberg, D.E. (1989): *Genetic Algorithms in Search, Operation and Machine Learning*. Addison-Wesley, Reading (USA).

Haftka, R.T.; Gurdal, Z. (1992): *Elements of Structural Optimization*, 3rd Edn, Kluver Academic Publishers, Dordrecht (Netherlands).

Hansen, S.R.; **Vanderplaats, G.N.** (1990): Approximation method for configuration optimization of trusses. *AIAA Journal*, vol. 28, pp. 161-168.

Higginson, J.S.; Neptune, R.R.; Anderson, F.C. (2005): Simulated parallel annealing within a neighborhood for optimization of biomechanical systems. *Journal of Biomechanics*, vol. 38, pp. 1938-1942.

Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. (1983): Optimization by simulated annealing. *Science*, vol. 220, pp. 671-680.

Lamberti, L.; Venkataraman, S.; Haftka, R.T.; Johnson T.F. (2003): Preliminary design optimization of stiffened panels using approximate analysis models. *International Journal for Numerical Methods in Engineering*, vol. 57, pp. 1351-1380.

Lamberti, L.; Pappalettere, C. (2003): A numerical code for lay-out optimization of skeletal structures with sequential linear programming. *Engineering with Computers*, vol. 19, pp. 101-129.

Lamberti, L.; Pappalettere, C. (2004): Improved sequential linear programming formulation for structural weight minimization. *Computer Methods in Applied Mechanics and Engineering*, vol. 193, pp. 3493-3521.

Luo, Y.Z.; Tang, G.J. (2005): Spacecraft optimal rendezvous controller design using simulated annealing. *Aerospace Science and Technology*, vol. 9, pp. 732-737.

Pantelides, C.P.; Tzan, S.R. (2000): Modified iterated simulated annealing algorithm for structural synthesis. *Advances in Engineering Software*, vol. 31, pp. 391-400.

Rao, S.S. (1996): *Engineering Optimization*. Wiley Interscience, New York (USA).

Schutte, J.F.; Koh, B.I.; Reinbolt, J.A.; Haftka, R.T.; George, A.D.; Fregly, B.J. (2005): Evaluation of a particle swarm algorithm for biomechanical optimization. *Journal of Biomechanical Engineering - Transactions of the ASME*, vol. 127, pp. 465-474.

Shim, P.Y.; Manoochehri, S. (1997): Generating optimal configurations in structural design using simulated annealing. *International Journal for Numerical Methods in Engineering*, vol. 40, pp. 1053-1069.

Tapp, C.; Hansel, W; Mittelstedt, C.; Becker, W. (2004): Weight minimization of sandwich structures by a heuristic topology optimization algorithm. *CMES: Computer Modeling in Engineering and Sciences*, vol. 5, pp. 563-574.

Vanderplaats, G.N. (1995): *DOT® Users Manual, Version 4.20*. VR&D Inc., Colorado Springs (USA).

Vanderplaats, G.N. (1998): *Numerical Optimization Techniques for Engineering Design*. VR&D Inc., Colorado Springs (USA).

Venkayya, V.B. (1978): Structural optimization: a review and some recommendations. *International Journal for Numerical Methods in Engineering*, vol 13, pp. 203-228.

Yu Chen, T. ; Su, J.J. (2002) : Efficiency improvement of simulated annealing in optimal structural designs. *Advances in Engineering Software*, vol. 33, pp. 675-680.

Wang, D.; **Zhang, W.H.**; **Jiang, J.S.** (2002): Combined shape and sizing optimization of truss structures. *Computational Mechanics*, vol. 29, pp. 307-312.

Wujek, B.A.; Renaud, J.E. (1998): New adaptive move-limit management strategy for approximate optimization. *AIAA Journal*, vol. 36, pp. 1911-1934.

Xie, Y.M; Steven, G.P. (1997): *Evolutionary Structural Optimization*, Springer-Verlag, London (UK).