Parallel Direct Search in Structural Optimization

J.B. Cardoso[1], P.G. Coelho[1] and A.L. Custódio[2],
[1] Department of Mechanical and Industrial Engineering, New University of Lisbon, Portugal,
[2] CMA and Department of Mathematics, New University of Lisbon, Portugal

Abstract

Since its early heuristic development, Direct Search Methods (DSM) have proved to be robust and reliable, both from theoretical and practical perspectives. Some of these algorithms are now able to solve noisy, nonsmooth or nonconvex problems and its algorithmic structure favours parallelization, drawing the attention of the structural optimization community as a promising alternative to the use of Meta-Heuristics.

In the present work, the performance of DSM is compared to Genetic Algorithms, when solving hard or expensive structural optimization problems. Parallel implementations are considered when large computational times are involved in function evaluation.

Keywords: Structural Optimization, Derivative-Free Optimization, Direct Search Methods, Pattern Search Methods, Mesh Adaptive Direct Search, Genetic Algorithms, Parallel Computing.

1 Introduction

Direct Search Methods (DSM) is a class of algorithms suited for Derivative-Free Optimization. The members of this class do not use, neither approximate, derivatives of the objective function. Progress is made by only comparing objective function values, computed at sets of points which satisfy some geometrical requirements [1, 2]. The type of sets which need to be considered will, in particular, depend on the problem smoothness.

Since its early heuristic development, in the 50's and 60's (see, for example, [3]), the practical application of these algorithms has proved to be robust and reliable in locating at least local optimums. This empirical conclusion was confirmed by several works related to convergence analysis, which have drawn the attention of the mathematical community since the 90's. Examples are [4] and [5], where the convergence of two particular instances of DSM is analyzed (Pattern Search Methods and Mesh Adaptive Direct Search, respectively).

Only recently the structural optimization community was attracted by the features of this class of algorithms [6]. In fact, structural optimization typically uses Gradient Based Methods (GBM), which are extremely efficient but require appropriate smoothness of the problem functions and, at least, the availability of first order derivatives. When discontinuities are present, if there are discrete variables, or when gradient information is unavailable or unreliable for use, a set of methodologies named as Meta-Heuristics has been considered. Genetic Algorithms (GA) [7], one instance of this class of Meta-Heuristics, are robust methods, from an empirical point of view, suited for application to a broad range of problems, but requiring a high number of function evaluations. This last point could be a major drawback when expensive finite elements analysis must be performed in order to compute function values.

As previously mentioned, DSM are designed to solve general problems with unknown derivatives, often working well in practice with noisy, nonsmooth or nonconvex functions. Also, the number of function evaluations required by a DSM is commonly considered to be higher than the one required by a GBM, but lower than the total number of function values computed by a GA. Recent work, not only allowed to improve the numerical performance of Pattern Search Methods [8, 9], but also extended the convergence analysis of this class of algorithms to nonsmooth and even discontinuous functions [5, 10].

The continuous increase of computational power and, in particular, the wide spreading of parallel computing had a major impact in structural analysis and optimization, widening the spectrum of problems which we are now able to tackle. Thus, parallelization is a valuable feature, when choosing an algorithm to solve a structural optimization problem. GA are intrinsically suitable for parallelization, which means that the high computational times related to serial implementations, resulting from the high number of function evaluations performed, can be overcome by considering high level performance computing in clusters.

The same applies to DSM, and several algorithmic instances were proposed which consider parallelization techniques [11, 12, 13]. Specifically, already released packages, like NOMAD [14] or PSWARM [15], provide parallel implementations of DSM that can be used for structural optimization.

The present work accesses the numerical performance of some recent implementations of DSM and its competitiveness against GA, both when considering serial and parallel codes. In Section 2, we briefly describe DSM, focusing in the potential available for parallelization. In Section 3, two structural examples are used to compare DSM and GA. The first problem consists in the optimization of a press brake to produce an uniform plate bending angle along the bending line. The problem is formulated as the unconstrained minimization of the bending error, with design variables associated to the bed and ram dimensions. Three public domain codes, NOMAD [14], PSWARM [15] and SID-PSM [16], are used to access the capability of DSM to solve this problem. A second problem, concerning the expensive structural optimization of a semi-trailer chassis, is considered. The problem is formulated as a weigh minimization, with size and shape variables related to the dimensions of the chassis components. Parallel versions of NOMAD [14] and PSWARM [15] are used to compute a solution. In both cases comparisons are made against GA.

2 Direct Search Methods (DSM)

The distinguishing feature of a DSM from other Derivative-Free Optimization algorithms is that no models are built for the objective function. The iterations proceed by only comparing objective function values. Typically the objective function is sampled at sets of points with good geometrical properties. Different algorithms make use of different sets. For instance, positive bases or positive spanning sets are used by DSM of directional type, like Pattern Search [17], Generating Set Search [1] or Mesh Adaptive Direct Search [5], while in the case of the simplex of Nelder-Mead [18] sampling is performed in sets of $n+1$ affinely independent points, named as *simplex*.

In this work we will focus on DSM of directional type. An iteration of these algorithms is organized around a *search step* and a *poll step*. The search step is optional and unnecessary for the convergence analysis, once that only a finite number of function evaluations is performed. The main purpose of this step is to improve the algorithmic efficiency. Quadratic interpolation models [9], particle swarm heuristics [19], surrogate based optimization [20] are some of the strategies possible to consider at this phase. If a better point is found then the iteration is declared as successful and the poll step is omitted. Generating Set Search [1] requires a *sufficient improvement* of the objective function value in order to declare that a better point was found, while Pattern Search [17] and Mesh Adaptive Direct Search [5] only require *simple improvement*.

If no better point was found, in order to retain the convergence properties of the method, which confer robustness to the algorithm, the poll step must be considered. At this step, a local search is performed around the current iterate by testing points corresponding to directions belonging to a positive basis or a positive spanning set, scaled by a step size parameter.

A positive spanning set is a set of vectors that spans $\mathbb{R}^n$ through nonnegative combinations of its elements. A positive basis is a positive spanning set such that no proper subset of it retains the same property. For any vector in the space, regardless of its position, it can be stated that at least one of the elements of a positive spanning set or a positive basis makes an acute angle with it. This is the main motivation for considering these types of sets in Derivative-Free Optimization. Figure 1 shows two different choices for positive bases in $\mathbb{R}^2$.

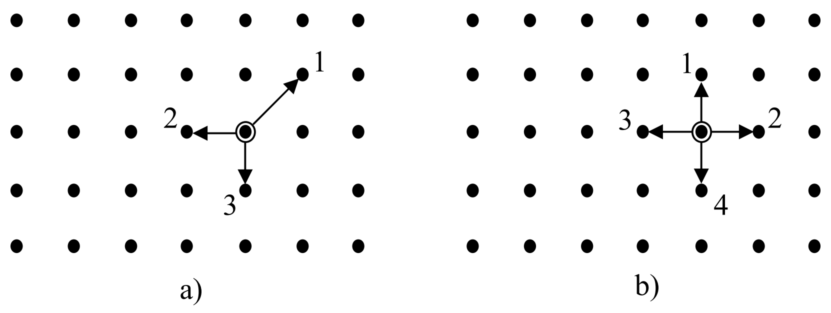


Figure 1: For $\mathbb{R}^2$, a minimal and a maximal positive basis, respectively.

If a better point is found, then the iteration is declared as successful, the point is accepted as the new current iterate and the step size is kept constant (or can even be increased). Otherwise, the current iterate is maintained and the step size parameter is obligatory decreased. Any unsuccessful poll step of a DSM will correspond to a total number of function evaluations which equals the number of elements in the positive spanning set or in the positive basis considered.

Opportunistic strategies can be implemented for the poll step of the algorithms, meaning that the testing procedure will stop once that a better point is found. In this case, the order imposed to the poll directions is crucial to improve the algorithmic efficiency. Custódio and Vicente [8] proposed ordering strategies based in the computation of simplex descent indicators, at no additional cost in terms of total number of function evaluations.

When requiring simple improvement of the objective function value, in order to accept the corresponding point as the new current iterate, some care must be taken in the computation of the positive bases or positive spanning sets to be used and in the update of the step size parameter. Namely, to preserve the convergence properties of the algorithms, some integer/rational requirements must be satisfied. In this case, all the points generated by the DSM must lie on an implicit integer/rational lattice or mesh, which will be coarsened for successful iterations and refined for unsuccessful ones. If there is enough smoothness in the objective function, and the number of positive spanning sets or positive bases considered during the course of the optimization process remains finite, convergence to some form of stationarity (depending on the objective function smoothness) can be established for at least a subsequence of the sequence of iterates generated by the DSM [4, 17].

The restriction to use a finite number of positive spanning sets or positive bases imposes a directional dependence, which could affect convergence if the objective function is nonsmooth, or in the presence of general constraints. This was the main motivation for the development of Mesh Adaptive Direct Search (MADS) [5], another algorithmic instance of the DSM class. MADS generates a set of poll directions asymptotically dense in $\mathbb{R}^n$. From a first probabilistic implementation (LTMADS [5]), the algorithm evolved to a deterministic strategy for computing the poll directions (ORTHOMADS [21]), which additionally ensures orthogonality of the vectors.

Audet and Dennis [22] and Abramson et al. [23] have extended DSM to mixed variable Derivative-Free Optimization. In this case the problem contains both continuous and discrete variables, where the latter ones can even be categorical, meaning that no order relation is defined between the set of admissible values. In this case, the user needs to define a local neighbourhood for the discrete variables. Similarly to what has been described, the poll step performs a local search with respect to the continuous variables, keeping the discrete variables at the fixed value reached in the previous iteration. Positive spanning sets or positive bases are used in this local search. Additionally, the poll step evaluates the points belonging to the local neighbourhood of the discrete variables (keeping constant the continuous, with value equal to the one reached at the previous iteration).

In case of failure, an *extended poll step* is executed. Promising poll centres are selected from the points evaluated in the local neighbourhood of the discrete variables. For each one of the selected poll centres, a local search is performed with respect to the continuous variables, considering the previously used positive spanning set or positive basis. Success means that a better point was found by comparison with the current iterate (and not with the poll centre extracted from the local neighbourhood).

The high complexity of the objective functions present in practical applications, which traduces in expensive computational times, and the need to treat problems with a high number of variables, motivate parallel implementations of DSM. Asynchronous Parallel Pattern Search [11] is one of the first algorithms belonging to the DSM class, which considers parallel strategies. In fact, taking into account the structure of the poll step, it is very natural to parallelize it by evaluating different points at different processors. The use of asynchronous strategies can lead to an improvement in efficiency, when the computational times related to function evaluation differ considerably between points or when the processors present different performance characteristics or are subject to varying loads.

Recently, Audet et al. [13] proposed to use Parallel Space Decomposition in MADS algorithm. The resulting algorithm (PSD-MADS) splits the variables of the original problem in a fixed number of subsets (where some variables can belong to more than one subset). Each processor will solve a subproblem of the original one, respecting to one of the variables subsets and fixing the remaining variables to the values corresponding to the best point found so far. All subproblems are solved using the MADS algorithm. One particular process will act on the original problem, considering all variables, but testing at each poll step a particular single direction. The set of all directions considered by MADS in this particular process should be asymptotically dense in $\mathbb{R}^n$. Computing a solution for a given subproblem will cause the update of the process performing the optimization on the whole set of variables.

In the present work we consider three computational implementations of DSM. The first, SID-PSM [16], is a serial Pattern Search Method, which uses simplex derivatives to improve efficiency. The search step is based on the minimization of quadratic polynomial interpolation models [9], which are computed by using previous function evaluations (minimum Frobenius norm models, at the beginning of the optimization process, and regression models, by the end). Previous evaluated points are also used to build simplex gradients which allow the reordering of the poll directions, before starting to poll [8].

The second implementation considered is PSWARM [15]. This computational code is also a Pattern Search Method, but with a search step defined by a particle swarm heuristic, like described in [19]. The code is available in serial and parallel versions, in the latter case using a synchronous strategy.

NOMAD is a numerical implementation of MADS [5]. Serial and parallel versions are available, in the latter case with and without synchrony. In the present work we considered the Parallel Space Decomposition of MADS, implemented in PSD-MADS [13], and also P-MADS implementation. This last code corresponds to an unmodified implementation of MADS, where different points are evaluated in parallel by different processors.

3 Examples

In order to access the performance of DSM to solve structural optimization problems, two examples were analysed. The first problem consists in the optimization of a press brake, where an analytical formulation is available for the objective function. The objective function is not too expensive to evaluate but the optimum point is located in a steepest valley, where the objective function is nondifferentiable. The second problem consists in the structural optimization of a semi-trailer chassis design. In this case, each function evaluation requires an expensive numerical simulation using a finite elements model.

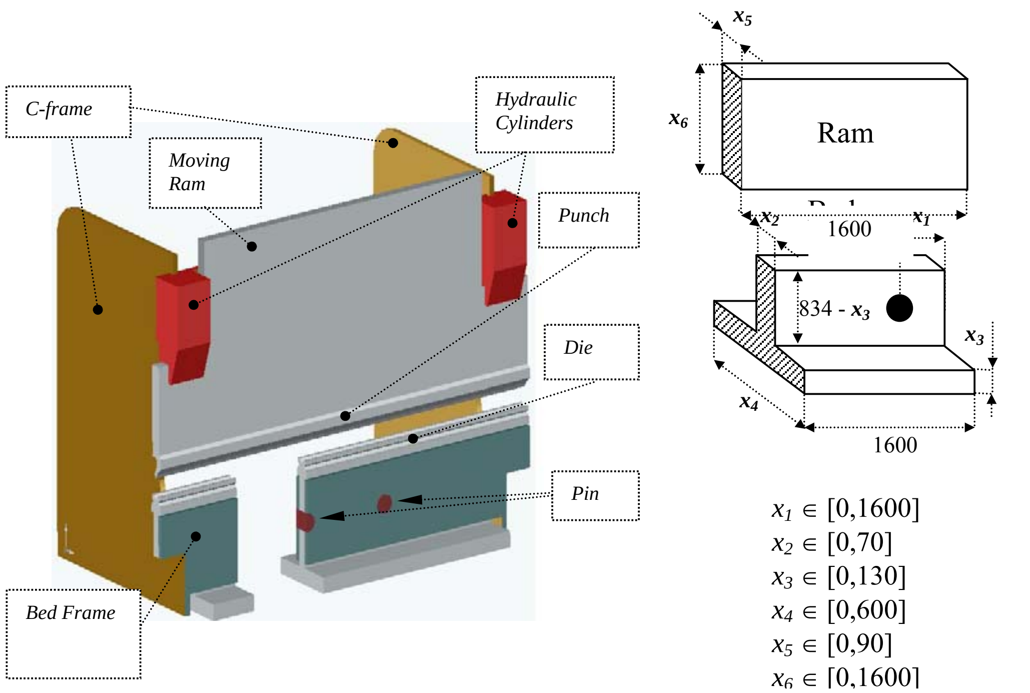
3.1 Press Brake Design Optimization

A press brake is a machine tool designed to bend flat metal plates, along a straight line, to a certain angle. A typical press brake presents a C-frame design, with a moving ram, which holds a punch, and a die located on a bed frame. Upon inserting the workpiece between the bed and the ram, a pair of hydraulic actuators forces the punch inside the die, bending the flat plate to the desired angle.

The bending angle is very sensitive to the penetration, i.e. to the relative displacement of the punch and the die. For example, a variation of 0.05mm in the penetration will cause a variation of 1º in the bending angle for a 1mm thick plate, bent in a 10mm die. The angular precision of the workpiece depends on the uniformity of the bending angle along the bending line. This uniformity is achieved with a constant penetration of the punch and the die, obtained through parallel deflections of the ram and the bed. The ram and the bed are long, narrow beams, but their finite stiffness causes non-constant penetration and non-uniform bending angles.

Using structural analysis methodologies, this problem is formulated as the unconstrained minimization of the bending error with the design variables associated to the dimensions of the bed and ram. For a detailed description and analysis of the formulation see [24].

In the present work, we choose one of the particular layouts proposed in [24], which is characterized by a rectangular cross-section for the ram and a T-shape for the bed. This layout, together with the six design variables considered, (x_1, x_2, …, x_6), and the corresponding bounds, is represented in Figure 2.



$x_1 \in [0,1600]$
$x_2 \in [0,70]$
$x_3 \in [0,130]$
$x_4 \in [0,600]$
$x_5 \in [0,90]$
$x_6 \in [0,1600]$

Figure 2: A press brake.

In [24], the authors proved the nondifferentiability of the objective function at the optimum, which motivated the use of GA to solve the problem. In the present work, three numerical implementations of DSM, namely SID-PSM, NOMAD and PSWARM, are used with the same purpose.

Since function evaluation is not too expensive (approximately 0.0014 seconds for each function value), serial versions were considered for the three codes, using the default options. The initial point provided to the optimizers was (1600, 70, 130, 600, 90, 1600), corresponding to the upper bound on the problem variables. The stopping criteria for the DSM was reaching a step size of 10^{-5} or performing 9600 function evaluations. This last computational budget corresponds to the total number of function evaluations allowed for GA (150 generations, each one with 64 individuals and a chromosome length of 96 genes, 16 for each variable). The variables were scaled to the [0, 1] multi-interval before running SID-PSM and PSWARM.

For stochastic algorithms, like GA and PSWARM, a total of 10 different runs were performed. Results for the best, worst and average number of function evaluations required and final objective function value can be found in Table 1. Only the best point computed is reported.

Table 1: Results for the optimization of the press brake design.

Algorithm		**GA**	**NOMAD**	**SID-PSM**	**PSWARM**
x_1 (mm)		434.21	435.60	545.52	431.83
x_2 (mm)		69.99	70.00	70.00	69.78
x_3 (mm)		129.98	130.00	129.96	129.97
x_4 (mm)		599.95	599.99	599.80	600.00
x_5 (mm)		89.82	74.17	89.79	90.00
x_6 (mm)		1377.32	1496.79	1599.92	1371.19
Obj.Func. Value	Minimum	0.025604	0.025650	0.02664	0.02567
	Average	0.026418			0.02629
	Maximum	0.027619			0.02859
Number of Obj.Func. Evaluations	Minimum	9600	1897	171	769
	Average	9600			881
	Maximum	9600			1018

DSM computed good quality solutions, with a considerable reduction in the total number of function evaluations required, when compared to GA. SID-PSM is remarkably faster, requiring only 1.8% of the number of function evaluations used by GA. Nevertheless, the final function value computed by this solver exceeds in 4% the best result obtained with GA.

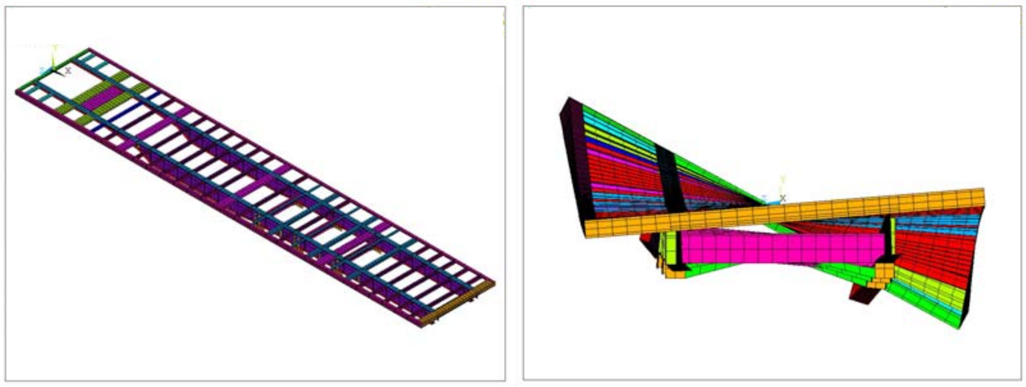
Although NOMAD and PSWARM require a higher number of function evaluations than SID-PSM (SID-PSM budget represents 9% and 19.4% of the total number of function evaluations computed by NOMAD and PSWARM, respectively), these two computational codes are still more efficient than GA. The total number of functions evaluations performed by each solver represents 19.8% and 9.2% of the one for GA, respectively. Also, the best final computed solutions are only 0.18% and 0.26% higher than the best solution computed by a GA.

3.2 Semi-Trailer Design Optimization

A semi-trailer, as shown in Figure 3, is formed by a chassis, where axis and wheels are attached, and a load carrying platform. This platform is built with beams and plates and designed to withstand the several operating loads. The design scenario considered corresponds to the torsion occurring in the platform, when performing a curve. The stiffness of the platform plays an important part in the semi-trailer performance, and the structural optimization with a minimum stiffness constraint is here carried out.

Figure 3: A semi-trailer.

A finite elements model, comprising 12747 nodes and 11864 elements, (see Figure 4a) was built in ANSYS [25] with shell elements. The rear suspension linking points are fixed and a 5000 Nm torque is applied to the *king pin*, i.e., the pin that couples the semi-trailer with the tractor. Figure 4b represents the resulting deformed chassis.



a) b)

Figure 4: a) Finite elements model of the semi-trailer chassis;
b) Chassis deformation.

A set of 12 design variables, containing 7 discrete, 1 logical and 4 continuous, was chosen. The discrete variables consist in the thickness of the components represented in Figure 5 and listed in Table 2. The continuous variables represent component dimensions as shown in Figure 6. Table 2 shows the variables allowable ranges.

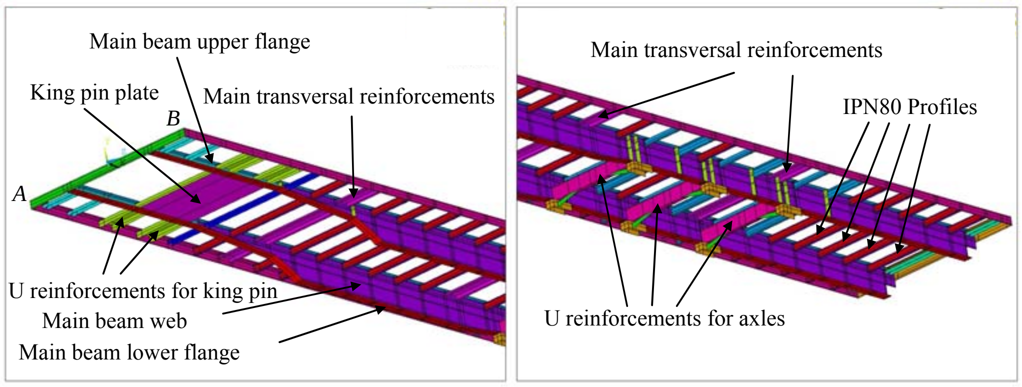

Main beam upper flange
King pin plate
B
Main transversal reinforcements
A
U reinforcements for king pin
Main beam web
Main beam lower flange
Main transversal reinforcements
IPN80 Profiles
U reinforcements for axles

Figure 5: Details of the structure of the semi-trailer chassis.

Table 2: Design variables considered in the optimization of the semi-trailer chassis structure.

Variable Number	**Description**	**Variable Range (mm)**
1	Main beam web	5, 6, 8, 10
2	Main beam lower flange	12, 15
3	Main beam upper flange	12, 15
4	Main transversal reinforcements	3, 3.5, 4, 4.5
5	U reinforcements for king pin	4, 5, 6, 8
6	U reinforcements for axles	2, 4, 5, 6
7	King pin plate	5, 6, 8, 10
8	IPN80 profiles	Used/Not used
9	Height for U reinforcements for axles	$180 \leq X_9 \leq 240$
10	Width for U reinforcements for king pin	$200 \leq X_{10} \leq 350$
11	Width for main transversal reinforcements	$142 \leq X_{11} \leq 186$
12	Flange width for main transversal reinforcements	$24 \leq X_{12} \leq 44$

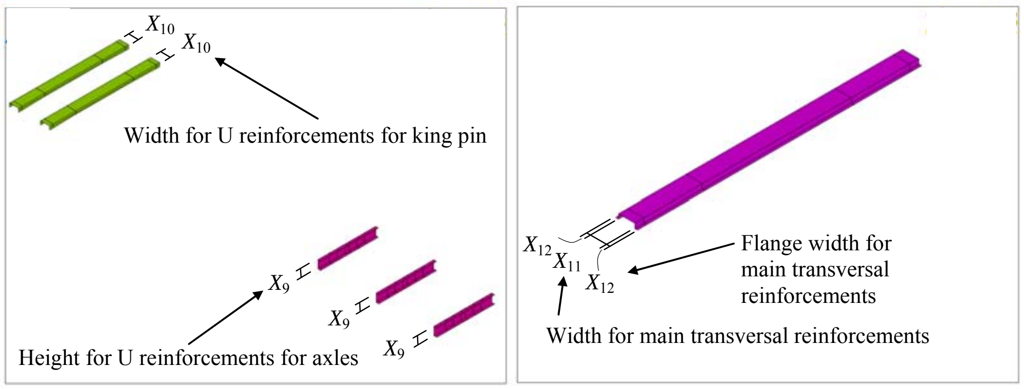

X_{10}
X_{10}
Width for U reinforcements for king pin
X_9
X_9
X_9
Height for U reinforcements for axles
X_{12}
X_{11}
X_{12}
Flange width for main transversal reinforcements
Width for main transversal reinforcements

Figure 6: Continuous design variables for the semi-trailer chassis.

This structural optimization problem is formulated as the weight minimization of the semi-trailer chassis structure, subject to variable bounds and a constraint on the frontal section rotation angle, which can not exceed 4.15º. This last constraint is modelled by imposing a maximum value of 180 mm to the sum of the vertical displacements of points *A* and *B* (see Figure 5) and treated with a penalty function. Each function evaluation takes approximately 400 seconds of computational time.

From the three DSM implementations considered, NOMAD is the only that provides both parallel versions (PSD-MADS and P-MADS) and the capability to solve mixed variable problems. PSWARM provides a parallel implementation but it is restricted to continuous variables, which motivated a first approach to the problem by relaxing all the discrete variables to continuous ones.

In this case the variable bounds were set equal to the lowest and highest values in each variable range (see Table 2). An initial point was selected corresponding to the upper bound on the design variables. The codes were run on a distributed memory Beowulf type cluster with 10 processors, imposing a limit of 960 for the total number of functions evaluations computed. Additionally, a minimum step size of 10^{-5} was allowed. Numerical results are shown in Table 3.

NOMAD was also used to solve the initial mixed variable problem, where the discrete variables were treated as integers, using the simple strategy implemented in the code which rounds variables and allows a minimum step size of one. An alternative approach was to consider these variables as categorical, defining the corresponding neighbourhoods. The computational results obtained with this second approach were worst than the ones reported.

To validate the computed results, the problem was also solved using a parallel implementation of GA, under a simple master-server model. A total of 32 generations, each one with 30 individuals was considered. At each generation, the points to evaluate were spread among 10 processors. Each point was represented by a chromosome with 29 genes, as reported in Table 4.

Table 3: Results for the optimization of the semi-trailer chassis design, considering continuous design variables.

Variable Number	Description	PSD-MADS	P-MADS	PSWARM
1	Main beam web	5.00	5.00	5.00
2	Main beam lower flange	13.57	13.76	12.19
3	Main beam upper flange	12.00	12.00	12.00
4	Main transversal reinforcements	3.00	3.00	3.73
5	U reinforcements for king pin	4.00	4.00	6.48
6	U reinforcements for axles	2.00	2.00	2.00
7	King pin plate	5.00	5.00	5.76
8	IPN80 profiles	0.00	0.00	0.00
9	Height for U reinforcements for axles	180.00	180.00	180.00
10	Width for U reinforcements for king pin	350.00	350.00	350.00
11	Width for main transversal reinforcements	186.00	142.00	151.31
12	Flange width for main transversal reinforcements	24.00	24.00	28.10
Objective Function Value (N)		15258.47	15247.09	15512.65
Number of Function Evaluations		589	900	962
Computational Time		474m 57.290s	671m 36.597s	711m 5.058s

Table 4: Chromosome structure considered in GA.

Variable Number	Description	Genes	Admissible values (mm)
1	Description	1,2	5, 6, 8, 10
2	Main beam web	3	12, 15
3	Main beam lower flange	4	12, 15
4	Main beam upper flange	5,6	3, 3.5, 4, 4.5
5	Main transversal reinforcements	7,8	4, 5, 6, 8
6	U reinforcements for king pin	9,10	2, 4, 5, 6
7	U reinforcements for axles	11,12	5, 6, 8, 10
8	King pin plate	13	Used/Not used
9	IPN80 profiles	14,15,16,17	$180 \leq X_9 \leq 240$
10	Height for U reinforcements for axles	18,19,20,21	$200 \leq X_{10} \leq 350$
11	Width for U reinforcements for King Pin	22,23,24,25	$142 \leq X_{11} \leq 186$
12	Width for main transversal reinforcements	26,27,28,29	$24 \leq X_{12} \leq 44$

The initial point and the additional stopping criteria consider for DSM were identical to the ones used in the continuous approach. Numerical results for all the algorithms can be found in Table 5.

Table 5: Results for the optimization of the semi-trailer chassis design, considering mixed variables.

Variable Number	Description	PSD-MADS	P-MADS	GA
1	Main beam web	5	5	5
2	Main beam lower flange	15	12	15
3	Main beam upper flange	12	12	12
4	Main transversal reinforcements	3	3	3
5	U reinforcements for king pin	4	8	4
6	U reinforcements for axles	2	2	2
7	King pin plate	5	5	5
8	IPN80 profiles	Not used	Not used	Not used
9	Height for U reinforcements for axles	180.00	180.00	183.75
10	Width for U reinforcements for king pin	275.00	347.13	275.00
11	Width for main transversal reinforcements	177.75	142.00	169.50
12	Flange width for main transversal reinforcements	24.00	24.00	32.75
Objective Function Value (N)		15484.79	15543.45	15503.43
Number of Function Evaluations		468	960	960
Computational Time		380m 34.251s	641m 44.264s	714m 0.069s

When only continuous variables were considered, both NOMAD and PSWARM were able to obtain very good solutions to the problem. GA were not used, since the number of chromosomes required to accurately represent a continuous variable would cause the algorithms to be extremely inefficient. Thus, for the continuous approach DSM results are compared against the ones of GA when solving the mixed variable version of the problem.

PSD-MADS was very fast in reaching the final solution, computing a number of function evaluations that only represents 61% of the one considered by GA. This percentage decreases to 49% for the mixed variable approach to the problem. Similar gains are obtained in terms of computational time. PSD-MADS uses 66% and 53% of the total time required by GA for the continuous and mixed variable approaches, respectively. PSWARM performance is similar to the one of GA. Both for the continuous and mixed variables versions of the problem, the best solutions were found by DSM implementations.

Since the considered GA makes an efficient use of the available computer power (each processor computes 3 functions per generation), the performance ratios obtained for NOMAD and PSWARM demonstrate the ability of these algorithms to fully exploit the available processors, by adequately spreading all the function evaluations.

4 Conclusions

DSM seem to be a promising alternative to the use of Meta-Heuristics (such as GA) in structural optimization problems.

Three different solvers (SID-PSM, NOMAD and PSWARM), belonging to this algorithmic class, were tested in two different structural optimization problems. By comparison with GA, each one of these solvers provided good quality solutions, under low computational budgets of function evaluations. The total number of objective function values computations is remarkably low in SID-PSM, thus a slight decrease was noticed in the quality of the final solution.

Efficient parallel implementations for some solvers of this class, like NOMAD and PSWARM, are already available, the former being also suited for mixed variable optimization problems.

References

[1] T. G. Kolda, R. M. Lewis, and V. Torczon, "Optimization by direct search: New perspectives on some classical and modern methods", SIAM Rev., 45, pp. 385–482, 2003.

[2] A. R. Conn, K. Scheinberg, and L. N. Vicente, "Introduction to Derivative-Free Optimization", MPS-SIAM Series on Optimization, SIAM, Philadelphia, 2009.

[3] R. Hooke and T. A. Jeeves, “'Direct Search' solution of numerical and statistical problems", J. ACM, 8, pp. 212–229, 1961.

[4] V. Torczon, "On the convergence of pattern search algorithms", SIAM J. Optim., 7, pp. 1–25, 1997.

[5] C. Audet and J. E. Dennis Jr., "Mesh adaptive direct search algorithms for constrained optimization", SIAM J. Optim., 17, pp. 188–217, 2006.

[6] S. Karakaya and O. Soykasap, "Buckling optimization of laminated composite plates using genetic algorithm and generalized pattern search algorithm", Struct. Multidisc. Optim., 39, pp. 477–486, 2009.

[7] D. E. Goldberg, "Genetic algorithms in search, optimization, and machine learning", Addison-Wesley Longman Publishing Co., Inc., Boston, 1989.

[8] A. L. Custódio and L. N. Vicente, "Using sampling and simplex derivatives in pattern search methods", SIAM J. Optim., 18, pp. 537–555, 2007.

[9] A. L. Custódio, H. Rocha, and L. N. Vicente, "Incorporating minimum Frobenius norm models in direct search", Comput. Optim. and Appl., 46, pp. 265–278, 2010.

[10] L. N. Vicente and A. L. Custódio, "Analysis of direct searches for discontinuous functions", to appear in Math. Program.

[11] P. D. Hough, T. G. Kolda, and V. J. Torczon, "Asynchronous parallel pattern search for nonlinear optimization", SIAM J. Sci. Comput., 23, pp. 134–156, 2001.

[12] T. G. Kolda, "Revisiting asynchronous parallel pattern search for nonlinear optimization", SIAM J. Optim., 16, pp. 563–586, 2005.

[13] C. Audet, J. E. Dennis Jr., and S. Le Digabel, "Parallel space decomposition of the mesh adaptive direct search algorithm", SIAM J. Optim., 19, pp. 1150–1170, 2008.

[14] http://www.gerad.ca/NOMAD/Project/Home.html

[15] http://www.norg.uminho.pt/aivaz/pswarm/

[16] http://www.mat.uc.pt/sid-psm/

[17] C. Audet and J. E. Dennis Jr., "Analysis of generalized pattern searches", SIAM J. Optim., 13, pp. 889–903, 2003.

[18] J. A. Nelder and R. Mead, "A simplex method for function minimization", Comput. J., 7, pp. 308–313, 1965.

[19] A. I. F. Vaz and L. N. Vicente, "A particle swarm pattern search method for bound constrained global optimization", J. Global Optim., 39, pp. 197–219, 2007.

[20] A. J. Booker, J. E. Dennis Jr., P. D. Frank, D. B. Serafini, V. Torczon, and M. Trosset, "A rigorous framework for optimization of expensive functions by surrogates", Struct. Multidiscip. Optim., 17, pp. 1–13, 1998.

[21] M. A. Abramson, C. Audet, J. E. Dennis Jr., and S. Le Digabel, OrthoMADS: "A deterministic MADS instance with orthogonal directions", SIAM J. Optim., 20, pp. 948–966, 2009.

[22] C. Audet and J. E. Dennis Jr., "Pattern search algorithms for mixed variable programming", SIAM J. Optim., 11, pp. 573–594, 2000.

[23] M. A. Abramson, C. Audet, J. W. Chrissis, and J. G. Walston, "Mesh adaptive direct search algorithms for mixed variable optimization", Optim. Lett., 3, pp. 35–47, 2009.

[24] P. G. Coelho, L. O. Faria, and J. B. Cardoso, "Structural analysis and optimisation of press brakes", Int. J. of Machine Tools Manuf., 45, pp. 1451–1460, 2005.

[25] ANSYS®, ANSYS Inc™.\\