Nonlinear programming by mesh adaptive direct searches [1]

Mark A. Abramson

Air Force Institute of Technology, Department of Mathematics and Statistics, 2950 Hobson Way, Building 640, Wright Patterson AFB, Ohio, 45433 USA, (http://www.afit.edu/en/ENC/Faculty/MAbramson/abramson.html, Mark.Abramson@afit.edu)

Charles Audet

GERAD and Département de Mathématiques et de Génie Industriel, École Polytechnique de Montréal, C.P. 6079, Succ. Centre-ville, Montréal (Québec), H3C 3A7 Canada (www.gerad.ca/Charles.Audet, Charles.Audet@gerad.ca)

J.E. Dennis Jr.

Computational and Applied Mathematics Department, Rice University - MS 134, 6100 South Main Street, Houston, Texas, 77005-1892 (www.caam.rice.edu/∼dennis, dennis@rice.edu).

Abstract This paper is intended not as a survey, but as an introduction to some ideas behind the class of mesh adaptive direct search (MADS) methods. Space limitations dictate a brief description of various key topics to be provided along with several references, which themselves provide further references.

The convergence theory for the methods presented here make a case for closing the gap between nonlinear optimizers and nonsmooth analysts. However these methods are certainly not of purely theoretical interest; they are successful on difficult practical problems. To encourage further use, we give references to available implementations. MADS is implemented in the direct search portion of the MathWorks MATLAB Genetic Algorithm and Direct Search (GADS) Toolbox.

Keywords : Mesh adaptive direct search algorithm, filter algorithm, barrier approach, constrained optimization, nonlinear programming.

[1] Audet's work was supported by FCAR grant NC72792 and NSERC grant 239436-05. Dennis was supported by LANL/DOE 94895-001-04 34, LBNL/DOE 671590, and both were supported by AFOSR F49620-01-1-0013, the Boeing Company, and ExxonMobil Upstream Research Company.

1 Introduction - the problem and its properties

For us, derivative-free optimization excludes methods that use standard finite difference approximations to derivatives in a Newton or SQP algorithmic framework. Those are well established and valuable methods. Indeed, there are many reasons to use them in place of really derivative-free methods like the ones treated here if one can. However, our target class of problems are not amenable to such an approach.

Since this is to be only one of several papers devoted to derivative-free optimization, we will concentrate on summarizing our work without making an attempt to give a survey of the topic. This is a relief because derivative-free optimization is already a diverse area of optimization, and it is growing fast, due in part to the importance of these algorithms for applications.

Our interest in the topic of direct search methods came directly from users, and our interaction with users continues to be our strongest influence. However, it would be incorrect to assume that these methods are not interesting for their own sake. We have found all the theoretical challenge we would wish for in this area. This leads to another point we will try to make: derivative-free optimization will and should form closer ties between computational optimization and nonsmooth analysis. We believe that nonsmooth analysis should be included in any curriculum meant to train nonlinear optimization researchers.

In this paper, we consider the general nonlinear optimization problem:

$$\min_{x \in \Omega} \quad f(x) \tag{1}$$

where $\Omega = \{x \in X : c_j(x) \leq 0, j \in J\} \subset \mathbb{R}^n$ and $f, c_j : X \rightarrow \mathbb{R}^n \cup \{\infty\}$ for all $j \in J = \{1, 2, \ldots, m\}$, and where X is a subset of $\mathbb{R}^n$. No differentiability assumptions on the objective and constraints are required for these algorithms. However, the strength of the optimality results at a limit point are closely tied to the local smoothness of the functions there and to properties of the tangent cone to Ω at a limit point produced by the algorithm.

We treat X and $C(x) = (c_1(x), c_2(x), \ldots, c_m(x))^T \leq 0$ differently because they are intended to model different classes of constraints. The set X includes the set of points to which x must belong in order that the functions $f(x)$ and $C(x)$ can be evaluated. We only require that the user provides a routine that says whether or not x is in X. We refer to these constraints as "yes/no" or *oracular* constraints.

Another interesting aspect of these problems is that even when $x \in \Omega$, we may not get a value for $f(x)$ or $C(x)$, though it may take as long to find that out as it would have if we had been able to get a value. We model this situation by setting $f(x) = +\infty$. This happens, for example, in some multi-disciplinary optimization (MDO) problems, in which getting a value depends on runtime linking of legacy PDE solvers [17].

Every trial point, as well as the initial point, must satisfy the X constraints, but $C(x) \leq 0$ is only required to hold at the solution. In fact, the user is often interested in how much optimality is possible with a slight relaxation of these constraints. We call these *open* constraints, and we treat them by a modification given in [14] of the filter method.

Filter algorithms were introduced by Fletcher and Leyffer [31] as a way to globalize sequential linear and quadratic programming (SLP and SQP) without using any merit function for weighting the relative merits of improving feasibility and optimality. A filter algorithm introduces a function that aggregates constraint violations and then treats the resulting biobjective problem. A trial point is accepted if it either reduces the value of the objective function or that of the constraint violation; otherwise it is said to be filtered. Although this clearly is less parameter-dependent than a penalty function, specifying a constraint violation function still implies an assignment of relative weights to reducing the violation of each constraint. The algorithm maintains feasibility with respect to X by modifying the aggregate constraint violation for Ω to be $+\infty$ outside of X.

A key feature of the optimization problems we most often meet in practice is that they involve running an expensive simulation to get ancillary variables needed to evaluate the blackbox function codes that define f and C. This means that we need to be parsimonious with function and constraint values, and it also implies that there are likely to be few correct digits in the output. As a result, derivatives are unlikely to exist everywhere, and if they do exist, difference quotients are not likely to give derivative approximations suited to use in derivative-based algorithms.

Often in practice, users express a desire to obtain the "global optimizer" of $f(x)$ on Ω. As we have described the problems, this is not something any algorithm can guarantee in practice. Still, global optimization algorithms generally find useful solutions when they can be applied to real problems. Indeed, with some attention to globality, the algorithms we outline here give equally useful solutions. We believe that this is because of synergy between this user request and another important user desire - robustness. In this context, one can think of a robust optimizer as one occurring in a broad valley. Such"global optimizers" are rather easier to find than those belonging to a narrow, but deeper basin.

2 What are mesh adaptive direct search (MADS) methods?

The methods we consider here are direct search methods. As the name *mesh adaptive direct search* (MADS) implies, these methods generate iterates on a tower of underlying meshes on the domain space. However, also as the name implies, they perform an adaptive search on the meshes including controlling the refinement of the meshes. The reader interested in the rather technical details should read [12, 13]. Here we ask the reader to imagine an underlying mesh and an algorithm for generating trial points on the mesh and adapting the fineness of the mesh to approach a local optimizer. We stress that the full mesh is never explicitly generated.

It is possible to dispense with the mesh as in [43, 44], which seems a simplification on the face of it. The argument against doing away with the mesh is that one must then use a sufficient decrease condition rather than accepting any point that provides simple decrease. Sufficient decrease conditions in the derivative-free situation are not as simple as a backtracking Goldstein-Armijo strategy in the quasi-Newton case [30]. Our suspicion is that whether or not to use the mesh is a matter of taste, not of algorithmic effectiveness, though we have no actual experience without the mesh on real problems.

Above, we mentioned the utility of nonsmooth analysis in derivative-free optimization. MADS is a case in point. We discovered MADS as a direct result of weaknesses in the generalized pattern search (GPS) class of algorithms [47], when applied to nonsmooth problems, which were exposed when we used nonsmooth analysis to analyze GPS [12, 14].

We could also call the methods considered SEARCH – POLL methods because each iteration consists of two steps, SEARCH and POLL . The goal of an iteration is to find unfiltered points in X. If SEARCH fails to find an unfiltered point, then POLL is executed, and if POLL does not succeed, then the mesh is refined.

The SEARCH step is crucial in practice because it is so flexible, but it is a difficulty for the theory for the same reason. SEARCH can return any point on the underlying mesh, but of course, it is trying to identify an unfiltered point. Any aspirations to find a local minimizer in a deeper basin than the one we start in is concentrated in the SEARCH step. When we discuss some SEARCH strategies, we will justify this point.

The POLL step is more rigidly defined, though there is still some flexibility in how this is implemented. Since the POLL step is the basis of the convergence analysis, it is the part of the algorithm where most research has been concentrated.

Lewis and Torczon [39] recognized that POLL should consider points on the mesh neighboring the incumbent solution in a set of directions whose nonnegative linear combinations span the space. This may seem simple, but it is a crucial observation. Coope and Price [21, 22, 23] extended this notion to the idea of frames, which can be thought of as doing away with the requirement that the POLL points be mesh neighbors. Audet and Dennis [13] suggested MADS as a way to implement frames so that the directions used in infinitely many POLL steps generate a dense set in the tangent cone at a MADS limit point $\hat{x} \in X$. This allows strong convergence results [13, 5] and excellent computational results for the MADS algorithms [15, 16, 41].

2.1 Some SEARCH strategies

The SEARCH step can be empty. By this we mean that the algorithm can be implemented as a sequence of POLL steps only. This is a reasonable choice when a local minimizer in the same basin as the initial guess is sufficient. Another reasonable strategy is to try a step in the same direction as a previously successful POLL step. It must be said that although this seems reasonable, we understand that some researchers have found this approach of limited value at best.

We have experimented with random search as a SEARCH strategy. This has some success on the initial iteration, but it seems to be a waste of function values after that.

In our experience, the best SEARCH strategies involve the use of surrogates for f and C. We use *surrogate* to mean an inexpensive function that the user can employ to look extensively on the current mesh for points that the surrogate predicts will improve the current incumbent solution. Surrogates generally are of two types, simplified physics simulations and surfaces fit to a set of points in X usually chosen by some space filling design. We use the term *surrogate* rather than *approximation* because we do not want to imply that anything is required with respect to how well the surrogates approximate the problem functions [18].

Boeing uses DACE surrogates [46] in their Design Explorer filter implementation [10]. They generate data sites by an orthogonal array, and then fit a DACE model to the data. The SEARCH consists of a global Newton SQP method applied to the surrogate problem to try to generate several good local optimizers for that problem. Then they use the expensive "true" problem functions at those points to decide whether the SEARCH has been successful. Whenever new values of the true problem functions have been computed, they are used to recalibrate the surrogates. This surrogate management framework leads to very successful methods. Details are given in [10].

Alison Marsden has solved trailing edge shape design problems using both types of surrogates in an insightful way. She generates trial points using the MATLAB DACE surrogate package [40] and then uses a less expensive turbulence model to check whether a trial point is in X. If it is, then she runs the more accurate simulation. Her SEARCH consists of applying an evolutionary algorithm to the DACE surrogates. See [41] for details.

Another interesting application of surrogates is in [15], where a framework to identify good algorithmic parameter values is given. To illustrate this framework, MADS was applied to an objective function that measured the CPU time required by a trust-region algorithm [32] to solve a set of difficult problems. A natural surrogate function was constructed by having the trust-region method to solve a different list of easy problems.

2.2 The POLL step

The POLL step is more rigidly defined than the SEARCH step. It is necessarily called when the SEARCH fails to produce an unfiltered point. The POLL step consists of a local exploration around the current incumbent solution. The trial points are generated in some directions scaled by a mesh size parameter. When either the SEARCH or the POLL step is successful, then the mesh size parameter is either kept constant or increased. Otherwise, when both steps fail to generate an unfiltered point, the incumbent is declared to be a *mesh local optimizer* [21] and the mesh size parameter is decreased.

In GPS, the POLL directions were restricted to belong to some finite set. The GPS convergence results [12, 4] are closely tied to these fixed directions. Furthermore, there are some known examples [9] for which GPS falls short of converging to a satisfactory solution because of this restriction.

MADS overcomes this limitation by allowing a larger set of POLL directions. In fact, as k (the iteration counter) goes to infinity, the union of the normalized POLL directions over all k becomes dense in the unit sphere. This algorithmic construction allows stronger convergence results [13].

In some cases, incomplete derivative information may be available. For example, in some MDO problems, derivatives for some disciplines may be available, but not for others, and derivatives across multiple disciplines are not available. If the full gradient is available, directions can be chosen so that all but one are ascent directions, which can be ignored, thus reducing the required number of function evaluations to one per iteration [6]. In this case, MADS reduces to an approximation of steepest descent. Even if only some partial derivatives are known, MADS can exploit this information to reduce the number of function evaluations in each POLL step [6] without sacrificing theoretical convergence properties. In lieu of derivatives, Custódio and Vicente [26] compute a simplex gradient from a subset of previously evaluated points having certain geometrical properties and use it as an approximation to the true gradient. However, convergence requires local proximity to the limit point.

Since MADS is opportunistic, in that it moves immediately to a new improved mesh point as soon as it is found, the order in which POLL points are evaluated can impact performance. One approach in which we have witnessed such improvement is what we call *dynamic polling*, in which the most recent successful direction is moved to the front of the queue after each successful POLL step. Dynamic polling was shown useful in [13] on a chemical engineering parameter fit problem [33]. If we were to use a surrogate in the SEARCH step, then evaluating the surrogate at each POLL point and then ordering them by surrogate function value would also be a wise choice. Custódio and Vicente [26] have also seen a reduction in function evaluations by computing a simplex gradient and ordering POLL points according to how small an angle the corresponding poll directions make with the negative of the simplex gradient. One must keep in mind, however, that these strategies (dynamic polling, surrogate and simplex gradient ordering) do not necessarily lead to improved computational times in all cases.

3 Why study these methods

In previous sections, we have mentioned some applications of MADS. In this section, we will make some general remarks about applications, but since the interested reader can find all the details we can furnish in the referenced papers, we save space here.

Also in this section we will discuss the theoretical support for MADS. We hope that other derivative-free optimization researchers will consider using nonsmooth analysis to analyze their methods. The discovery of MADS was a direct result of our nonsmooth analysis of GPS, and that has made us enthusiastic about building a bridge to this advanced theoretical part of our discipline.

3.1 Importance in practice

It is likely that every paper in these special issues will make a case for the practical importance of derivative-free optimization methods. We second everything the other authors say, but we will use our space here to try to make a couple of points that other authors may not make.

The first point is that, in spite of the formidable aspects of our target class of problems, we are often able to solve them quite efficiently. The main reason we were the first to solve them is that there are barriers to applying traditional derivative-based methods, and heuristic searches use too many function evaluations for these relatively expensive problem functions.

These problems typically take minutes to weeks for each function value, and many of them have what Tim Kelley [19], who also sees such problems, likes to call “hidden constraints”. This second point means that one calls the simulation codes that must be run to evaluate the functions for perfectly innocuous arguments, and they fail. Furthermore, they fail after running for about the same length of time as when they succeed. In [18], this sort of failure happens to us roughly twice in every three function calls.

The main reason we have seen for these evaluation failures is that the function evaluations depend on runtime linking of single discipline solvers; e.g., separate structures and fluids codes. This is characteristic of multidisciplinary design optimization (MDO) problems [25, 8]. The interested reader will find a vast amount of MDO literature on the web.

Thus, to get the ancillary variables needed to evaluate the objective and constraints, one must do a multidisciplinary analysis, meaning the runtime linking of the codes. In our experience, an MDA can be thought of as solving a large system of nonlinear equations for which no Jacobian information is practical. In such a circumstance, there is little one can try except simple successive substitution or nonlinear Gauss-Seidel. This is sensitive to the order in which the blocks are processed, and it is apt to fail.

Another difficulty inherent to some of the target problems is that the functions are often contaminated with noise. It is not infrequent that evaluating a function twice at the same value of x returns slightly different values.

3.2 Theoretical support

These algorithms are intended to be applied to nonsmooth problems, or to any problems for which derivatives are impractical, even by finite differences. Typically, both the objective function and the constraints are evaluated by running a black box computer code. There is no way one can measure the smoothness of these functions.

The convergence results state that if the MADS algorithm is applied to such problems, then some optimality conditions are guaranteed. In [12] and [13] we give a hierarchy of convergence results based on various degrees of smoothness of the objective and constraints.

At the bottom of the hierarchy, we have a result that if the iterates produced by the algorithm are bounded, then there is an $\hat{x}$, which is the limit of mesh local optimizers on meshes that get infinitely fine. Assuming a bounded sequence of iterates is a standard assumption in nonlinear optimization, and it holds for our algorithms if the initial level set is bounded.

Then, by adding more smoothness, the local optimality results become successively stronger for a limit point $\hat{x}$. At the smoothest end of the hierarchy, we have that if f is strictly differentiable near $\hat{x}$, and if the constraint qualification that the tangent cone $T_\Omega(\hat{x})$ to the feasible region Ω at $\hat{x} \in \Omega$ is non-empty and full-dimensional, then the directional derivative f' satisfies

$$f'(\hat{x};d) \geq 0 \quad \text{for every } d \in T_\Omega(\hat{x}).$$

This is the KKT first-order optimality condition: there are no feasible strict descent directions. In the unconstrained case, the tangent cone is the entire space, and this last condition becomes $\nabla f(\hat{x}) = 0$.

The intermediate results of the convergence analysis are based on different degrees of smoothness. The directional derivatives f' are not appropriate to deal with non-smooth functions, as they are undefined. We turned to the nonsmooth community and found exactly the analytical tool that we needed to analyze the convergence of our methods: the Clarke Calculus [20].

Clarke proposes a generalization $f^\circ(\hat{x};d)$ of the directional derivative for locally Lipschitz functions, and generalizations [45, 20, 35] of the tangent cone; namely, the hypertangent cone $T_\Omega^H(\hat{x})$, the Clarke tangent cone $T_\Omega^{Cl}(\hat{x})$, and the contingent cone $T_\Omega^{Co}(\hat{x})$. Armed with these definitions, we can show that depending on the smoothness, the limit point $\hat{x}$ generated by MADS satisfies

$$\begin{array}{lll} f^\circ(\hat{x};d) \geq 0 & \text{for every} & d \in T_\Omega^H(\hat{x}),\ T_\Omega^{Cl}(\hat{x}) \\ & & \quad \text{or in } \ T_\Omega^{Co}(\hat{x}). \end{array}$$

Furthermore, in [5], we discover that with more smoothness (namely, that f is twice strictly differentiable near $\hat{x}$), $\hat{x}$ satisfies a second-order Clarke-KKT necessary condition for optimality that depends on a generalization of the Hessian matrix [34]. In fact, with additional assumptions, $\hat{x}$ satisfies a second-order Clarke-KKT sufficient condition for optimality, thus ensuring convergence of MADS to a local minimizer [5].

In stating these results, we make the assumption that the set of directions used infinitely often is dense in the hypertangent cone at $\hat{x}$. As stated earlier, MADS is designed specifically so that this can be accomplished, but in order to do it in practice, our selection of positive spanning directions is done randomly. Consequently, most of our convergence results are with probability one.

4 What is still needed

There are practical issues we still need to deal with for the class of problems discussed above. Anyone who has worked with users has had the experience of being told that the problem has a certain property, e.g., ten design variables, only to be told after solving the problem that it would be nice to be able to deal with one hundred design variables. This is a sure sign of progress in the project. In this section, we will give brief descriptions of some of the main issues raised by users after an initial success with the first formulation.

4.1 Categorical variables

Nonlinear mixed integer problems are hard enough, but many engineering design problems involve categorical variables. These are discrete variables constrained to a discrete set as a part of X. The problem functions cannot be evaluated unless all categorical variables take on feasible discrete values. For example, simulating an oil field with 25.3 oil wells is out of the question unless one interpolates and thereby increases the number of expensive simulations required.

We use the term *mixed variable programming* (MVP) to denote mathematical programming problems with both continuous and categorical variables [11]. An example is found in the design of a fixed-length thermal insulation system [36] in which the objective is to minimize the power required subject to some reasonable linear constraints.

In this problem, the system consists of a series of insulators of various material types and thicknesses, each pair of which is separated by a metal plate, called a *heat intercept*, to which power is applied to maintain it at a specified temperature. The material thicknesses and intercept temperatures are the continuous variables, while the number and types of insulators are categorical. In fact, the insulator types are not even numeric, although each material type can be mapped to the numeric value of its index into a list of seven possible material types that may be selected. A further interesting complication is that the number of insulators, which defines the problem dimension, is itself a design variable. This problem was solved numerically in [36] using the algorithm introduced in [11]. Realistic nonlinear constraints on system mass, tensile yield stress, and thermal contraction were added to the problem in [3], and the resulting problem was solved numerically using a pattern search filter method [7].

Because of the general lack of ordinality with categorical variables, MVP problems present some unique challenges. For example, there is no general notion of local optimality. To overcome this challenge, the user must provide a set-valued neighborhood function that defines the set of discrete neighbors at every point. Local optimality is then defined with respect to this function at the limit point. In the example above, given a design of the system, discrete neighbors were formed in 3 ways: swapping a single insulator for another of a different material, adding an insulator and heat intercept at any location (and adjusting the continuous variables appropriately), or deleting any insulator with its adjacent intercept. Once the algorithm is appropriately modified, we guarantee that the resulting solution could not be improved by moving to a neighboring point, as defined by these three classes of neighbors.

The main modification to the algorithm consists of augmenting the POLL step to include points in the set of discrete neighbors, along with other promising points [11]. Convergence properties of GPS for MVP problems with a smooth objective function and bound constraints on the continuous variables were established in [11] and extended to general linear constraints and nonsmooth functions in [2]. Convergence results for the GPS filter method for MVP problems with nonlinear constraints was introduced in [2, 7].

The class of MVP problems is actually quite common in practice, even though the field is very new, and there are some important algorithmic and structural considerations that merit further research.

4.2 Multiple objectives

It is almost always true that real optimization problems have multiple objectives. They may not appear in this form, but scratch the surface and they will. For example, a client might suggest a bound constraint on some function $\gamma(x)$. But, when asked about the value of the bound, the client will say it should be as small as possible. In other words, the constraint is really another objective.

Another way multiple objectives show themselves is in documenting the problem solution for the client. Presented with a solution to an optimization problem, the client (or his/her boss) will want to know how much better/worse the objective would be if a certain constraint were to be relaxed/tightened.

The reader will see in both cases the standard objective synthesis approach of minimizing a weighted sum of the individual objectives is not helpful. In both cases, the decision maker wants to trade off one objective against the others. What we need is to give the client a notion of the Pareto surface. To see a simple case of the deficiencies in the the weighted sum approach, see [29]. We do not recommend this approach; however, [28] is an interesting way to find a single important Pareto point.

Since the filter approach is based on multi-objective ideas, we hope that our filter approach can be adapted to provide helpful tools. However, this is not as straight forward as one might hope.

4.3 Ability to handle more decision variables

It would be useful to extend MADS to handle hundreds of decision variables, on problems where parallelism [37] alone would not be sufficient to solve the problem. As with all direct search methods, we expect to see the number of function values required to solve an arbitrary n dimensional problem increase much faster than n. Our goal is to find alternative direct search methods that slow the growth.

5 Conclusions

Direct search methods are here to stay as a valuable subarea of optimization. They are interesting theoretically, and they are indispensable in practice. These special issues will document many of the advances that have been made in the area, but much remains to be done.

We have sketched some useful properties and limitations of MADS algorithms. A researcher willing to build a strong theoretical background in nonsmooth analysis and learn to work with users will find this a satisfying and fruitful area in which to work. The experience of helping a user formulate and solve a problem thought to be intractable is the ultimate validation for an applied mathematician. Come on in, the water is fine.

A reader interested in obtaining software should visit [1, 24, 27, 38, 40, 42].

References

[1] M. A. Abramson. NOMADm optimization software. http://en.afit.edu/-ENC/Faculty/MAbramson/NOMADm.html.

[2] M. A. Abramson. *Pattern Search Algorithms for Mixed Variable General Constrained Optimization Problems*. PhD thesis, Department of Computational and Applied Mathematics, Rice University, August 2002.

[3] M. A. Abramson. Mixed variable optimization of a load-bearing thermal insulation system using a filter pattern search algorithm. *Optim. Engng.*, 5(2):157–177, 2004.

[4] M. A. Abramson. Second-order behavior of pattern search. *SIAM J. Optim.*, 16(2):315–330, 2005.

[5] M. A. Abramson and C. Audet. Second-order convergence of mesh adaptive direct search. Technical Report TR05-09, Department of Computational and Applied Mathematics, Rice University, Houston Texas, 2005.

[6] M. A. Abramson, C. Audet, and J. E. Dennis, Jr. Generalized pattern searches with derivative information. *Math. Programming*, Series B, 100:3–25, 2004.

[7] M. A. Abramson, C. Audet, and J. E. Dennis, Jr. On the convergence of filter pattern search algorithms for mixed variable constrained optimization problems. Technical Report TR04-09, Department of Computational and Applied Mathematics, Rice University, Houston Texas, 2004.

[8] N. M. Alexandrov and M. Y. Hussaini, editors. *Multidisciplinary Design Optimization: State of the Art*. SIAM, Philadelphia, 1997.

[9] C. Audet. Convergence results for pattern search algorithms are tight. *Optim. Engng.*, 5(2):101–122, 2004.

[10] C. Audet, A. J. Booker, J. E. Dennis, Jr., P. D. Frank, and D. W. Moore. A surrogate-model-based method for constrained optimization. AIAA Paper 2000–4891, Presented at the 8th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Long Beach, California, 2000.

[11] C. Audet and J. E. Dennis, Jr. Pattern search algorithms for mixed variable programming. *SIAM J. Optim.*, 11(3):573–594, 2000.

[12] C. Audet and J. E. Dennis, Jr. Analysis of generalized pattern searches. *SIAM J. Optim.*, 13(3):889–903, 2003.

[13] C. Audet and J. E. Dennis, Jr. Mesh adaptive direct search algorithms for constrained optimization. *To appear in SIAM J. Optim.*, 2004.

[14] C. Audet and J. E. Dennis, Jr. A pattern search filter method for nonlinear programming without derivatives. *SIAM J. Optim.*, 14(4):980–1010, 2004.

[15] C. Audet and D. Orban. Finding optimal algorithmic parameters using the mesh adaptive direct search algorithm. Technical Report G-2004-96, Les Cahiers du GERAD, 2004. submitted to SIOPT.

[16] V. Béchard, C. Audet, and J. Chaouki. Robust optimization of chemical processes using a MADS algorithm. Technical Report G-2005-16, Les Cahiers du Gerad, 2005.

[17] A. J. Booker, A. R. Conn, J. E. Dennis, Jr., P. D. Frank, M.W. Trosset, and V. Torczon. Global modeling for optimization: Boeing/IBM/Rice collaborative project 1995 final report. Technical Report ISSTECH–95–032, Boeing Information & Support Services, Research and Technology, M/S 7L–68, Seattle, Washington 98124, 1995.

[18] A. J. Booker, J. E. Dennis, Jr., P. D. Frank, D. B. Serafini, V. Torczon, and M. W. Trosset. A rigorous framework for optimization of expensive functions by surrogates. *Structural Optimization*, 17(1):1–13, February 1999.

[19] T. D. Choi, O. J. Eslinger, C. T. Kelley, J. W. David, and M. Etheridge. Optimization of automotive valve train components with implicit filtering. *Optim. and Engng.*, 1:9–28, 2000.

[20] F. H. Clarke. *Optimization and Nonsmooth Analysis*. Wiley, New York, 1983. Reissued in 1990 by SIAM Publications, Philadelphia, as Vol. 5 in the series Classics in Applied Mathematics.

[21] I. D. Coope and C. J. Price. Frame-based methods for unconstrained optimization. *J. Optim. Theory Appl.*, 107(2):261–274, 2000.

[22] I. D. Coope and C. J. Price. On the convergence of grid-based methods for unconstrained optimization. *SIAM J. Optim.*, 11(4):859–869, 2001.

[23] I. D. Coope and C. J. Price. Positive bases in numerical optimization. *Computational Optimization and Applications*, 21(2):169–175, February 2002.

[24] G. Couture, C. Audet, J. E. Dennis, Jr., and M. A. Abramson. The NOMAD project. http://www.gerad.ca/NOMAD/.

[25] E. Cramer, J. E. Dennis, Jr., P. D. Frank, R. M. Lewis, and G. R. Shubin. On the convergence of grid-based methods for unconstrained optimization. *SIAM J. Optim.*, 11(4):859–869, 2001.

[26] A. L. Custódio and L. N. Vicente. Using sampling and simplex derivatives in pattern search methods. Technical report, Department of Mathematics, University of Coimbra, December 2004.

[27] A. L. Custódio and L. N. Vicente. SID-PSM: A pattern search method guided by simplex derivatives for use in derivative-free optimization. http://www.mat.uc.pt/sid-psm, 2005.

[28] Indraneel Das. On characterizing the 'knee' of the pareto curve based on normal-boundary intersection. *Structural Optimization*, 18:107–115, October 1999.

[29] Indraneel Das and John E Dennis, Jr. A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization programs. *Structural Optimization*, 14:63–69, 1997.

[30] J. E. Dennis, Jr. and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, NJ, 1983. Reissued in 1996 by SIAM Publications, Philadelphia, as Vol. 16 in the series Classics in Applied Mathematics.

[31] R. Fletcher and S. Leyffer. Nonlinear programming without a penalty function. *Mathematical Programming*, Series A, 91:239–269, 2002.

[32] N. I. M. Gould, D. Orban, and Ph. L. Toint. GALAHAD—a library of thread-safe FORTRAN 90 packages for large-scale nonlinear optimization. *ACM Transactions on Mathematical Software*, 29(4):353–372, December 2003.

[33] R. E. Hayes, F. H. Bertrand, C. Audet, and S. T. Kolaczkowski. Catalytic combustion kinetics: Using a direct search algorithm to evaluate kinetic parameters from light-off curves. *The Canadian Journal of Chemical Engineering*, 81(6):1192–1199, 2003.

[34] J.-B. Hiriart-Urruty, J.-J. Strodiot, and V. H. Nguyen. Generalized Hessian matrix and second-order optimality conditions for problems with $C^{1,1}$ data. *Appl. Math. Optim.*, 11(1):43–56, 1984.

[35] J. Jahn. *Introduction to the Theory of Nonlinear Optimization*. Springer, Berlin, 1994.

[36] M. Kokkolaras, C. Audet, and J. E. Dennis, Jr. Mixed variable optimization of the number and composition of heat intercepts in a thermal insulation system. *Optim. and Engng.*, 2(1):5–29, 2001.

[37] T.G. Kolda and V.J. Torczon. On the convergence of asynchronous parallel pattern search. *SIAM J. Optim.*, 14(4):939–964, 2004.

[38] Sandia National Laboratories. Asynchronous parallel pattern search. http://software.sandia.gov/appspack, 2005.

[39] R. M. Lewis and V. Torczon. Rank ordering and positive bases in pattern search algorithms. Technical Report 96–71, Institute for Computer Applications in Science and Engineering, Mail Stop 132C, NASA Langley Research Center, Hampton, Virginia 23681–2199, 1996.

[40] S. N. Lophaven, H. B. Nielsen, and J. Søndergaard. DACE: A MATLAB kriging toolbox version 2.0. Technical Report IMM-TR-2002-12, Technical University of Denmark, Copenhagen, 2002.

[41] A. L. Marsden. *Aerodynamic Noise Control by Optimal Shape Design*. PhD thesis, Stanford University, 2004.

[42] Mathworks. MATLAB genetic algorithm and direct search (GADS) toolbox. http://www.mathworks.com/products/gads/, 2005.

[43] C. J. Price and I. D. Coope. Frame based ray search algorithms in unconstrained optimization. *J. Optim. Theory Appl.*, 116:259–377, 2003.

[44] C. J. Price and I. D. Coope. Frames and grids in unconstrained and linearly constrained optimization: A nonsmooth approach. *SIAM J. Optim.*, 14(2):415–438, 2003.

[45] R. T. Rockafellar. Generalized directional derivatives and subgradients of nonconvex functions. *Canad. J. Math.*, 32(2):257–280, 1980.

[46] T. J. Santner, B. J. Williams, and W. I. Notz. *The Design and Analysis of Computer Experiments*. Springer Verlag, 2003.

[47] V. Torczon. On the convergence of pattern search algorithms. *SIAM J. Optim.*, 7(1):1–25, February 1997.