J Glob Optim (2008) 41:299–318
DOI 10.1007/s10898-007-9234-1

Nonsmooth optimization through Mesh Adaptive Direct Search and Variable Neighborhood Search

Charles Audet · Vincent Béchard · Sébastien Le Digabel

Received: 26 October 2006 / Accepted: 29 August 2007 / Published online: 11 October 2007
© Springer Science+Business Media, LLC. 2007

Abstract This paper proposes a way to combine the Mesh Adaptive Direct Search (MADS) algorithm, which extends the Generalized Pattern Search (GPS) algorithm, with the Variable Neighborhood Search (VNS) metaheuristic, for nonsmooth constrained optimization. The resulting algorithm retains the convergence properties of MADS, and allows the far reaching exploration features of VNS to move away from local solutions. The paper also proposes a generic way to use surrogate functions in the VNS search. Numerical results illustrate advantages and limitations of this method.

Keywords Nonsmooth optimization · Mesh Adaptive Direct Search · Generalized Pattern Search · Variable Neighborhood Search

1 Introduction

The paper considers optimization problems of the form

$$\min_{x\in\Omega\subseteq\mathbb{R}^n} f(x) \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$, $\Omega = \left\{x \in X : c_j(x) \le 0,\ j = 1, 2, \ldots, m\right\}$ and $X \subseteq \mathbb{R}^n$ represents closed constraints, i.e. constraints that necessarily need to be satisfied in order for the functions to evaluate. The closed constraints often include bounds constraints $L \le x \le U$ with L and U in $(\mathbb{R} \cup \{\pm\infty\})^n$, and possibly boolean constraints that indicate if they are satisfied or not, and in the latter case, there is no quantification by which they are violated.

C. Audet · V. Béchard · S. Le Digabel (✉)
Département de Mathématiques et de Génie Industriel, Ecole Polytechnique de Montréal and GERAD, C.P. 6079, Succ. Centre-ville, Montreal, QC, Canada H3C 3A7
e-mail: Charles.Audet@gerad.ca
www.gerad.ca/Charles.Audet

V. Béchard
e-mail: vincent.bechard@polymtl.ca

S. Le Digabel
e-mail: Sebastien.Le.Digabel@gerad.ca

The functions $c_j : \mathbb{R}^n \to \mathbb{R}$ for $j = 1, 2, \ldots, m$ represent the other constraints and are referred to as the open constraints.

The objective function f and the different functions defining the set Ω are typically provided as black-boxes in the sense that the way to obtain a function value from a given $x \in \mathbb{R}^n$ is not provided in an analytical way, or may be time consuming or expensive to evaluate. The black-boxes may also fail to return a value at some points. This is modeled by setting the function value to infinity and is called the *barrier approach*. The case where none of their eventual property, derivatives for example, can be exploited is considered. Such black-box functions are widely used in different engineering disciplines [1,3,10,12,13,23,28,31,35]. Black-box functions are typically evaluated by running computer code. Approximations of the black-box functions can also be made through easier to evaluate surrogate functions (see [14]), and this paper proposes a different way to exploit such surrogates.

Different derivative-free direct search methods are designed for problem (1), such as GPS [14,45], MADS [2,9] and DIRECT [21,30]. The reader may consult [32,33] for surveys of direct search methods. Under appropriate conditions, these methods ensure convergence to a point satisfying necessary optimality conditions based on the Clarke calculus [18].

In the present paper, we exploit the flexibility of the MADS algorithm so that it includes the far reaching searches of the Variable Neighborhood Search (VNS [25,36]). The fundamental structures of MADS and VNS are complementary: on the one hand, in case of failure to identify improved points, VNS explores increasingly larger regions, and on the other hand, MADS explores smaller and smaller neighborhoods. The purpose of this paper is to present a generic coupling of MADS and VNS that may be applied to the class of problems for which MADS was designed.

The main reason why we chose to combine VNS with the MADS algorithm instead of another optimization method is that the convergence analysis of the resulting method follows directly. Each MADS iteration is partitioned into a “search” and a “poll” step. The search step is intended to be flexible (but still must satisfy some minimal requirements), and the poll step must follow strict rules. The MADS algorithm was conceived in such a modular way precisely to allow the user to create and use his own search strategies. In the present paper work we take advantage of the flexibility of the search step by proposing a generic VNS search step. The way in which VNS generates trial points makes it easy to verify that the search requirements are satisfied. The MADS poll step and the update rules are the same as in [9], thus the MADS convergence analysis holds.

The paper is divided as follows. Section 2 proposes an overview of the MADS and VNS methods. Section 3 presents a generic algorithm that couples MADS and VNS and allows the use of surrogate functions. Section 4 describes a practical implementation. Finally some numerical results, including the detailed description of an engineering problem, are presented in Sect. 5. The proposed algorithm is compared to the classic MADS algorithm and to MADS with a classic search strategy.

2 Descriptions of the MADS and VNS algorithms

2.1 MADS

The MADS algorithm [9] for problem (1) extends the Generalized Pattern Search (GPS) algorithm for linearly constrained optimization [14,45]. Both GPS and MADS are iterative algorithms where the black-box functions are evaluated at some trial points, which are either accepted as new iterates or rejected. At any iteration (denoted by the integer k), all trial points generated by these algorithms are constructed to lie on the mesh

$$M(k, \Delta_k) = \bigcup_{x \in V_k} \left\{ x + \Delta_k D z : z \in \mathbb{N}^{n_D} \right\} \subset \mathbb{R}^n$$

where V_k is the set of points evaluated by the start of iteration k, $\Delta_k \in \mathbb{R}^+$ is the mesh size parameter, and D a fixed matrix whose columns are in $\mathbb{R}^n$. In most cases, D is chosen to be the $n \times 2n$ matrix $[-I \; I]$ where I is the $n \times n$ identity matrix. We make the standard assumption that all the trial points are in a compact set C. This assumption, together with the fact that the mesh is constructed using integer combinations of $\Delta_k D$ ensures that $C \cap M(k, \Delta_k)$ contains a finite number of points.

In order to simplify the notations of the present paper, the parameter Δ_k is the equivalent of Δ_k^m in [9].

Each iteration is divided into two main steps, the search and the poll, followed by an update step that determines the success of the iteration. The new mesh size, the new current iterate and the stopping criteria are updated or verified at the end of the iteration.

The closed constraints defining the set X are handled by the barrier approach, as in MADS, consisting in rejecting trial points outside X. For the other constraints (c_j, $j = 1, 2, \ldots, m$), a filter approach is used [8,22]: if the points are in X, they are stored and classified, using their objective function value and a measure of the constraints violation, which permits to accept the promising points and to reject the others.

The poll evaluates the functions f and c_j's at mesh points near the current iterate x_k. The convergence analysis of [9] relies on the rigid rules that the poll step need to follow. The poll step remains unchanged in the present paper. The way of choosing the directions used to generate the poll points makes the difference between GPS and MADS: in GPS, the normalized set of poll directions is finite, whereas it may be asymptotically dense in the unit sphere with MADS, allowing a better exploration of the space of variables. At iteration k, the set containing the trial poll points is called the frame P_k, given by $P_k = \{x_k + \Delta_k d : d \in D_k\}$, with D_k the set of directions used to construct P_k. D_k is a set formed by taking positive integer combinations of the columns of D. We will not say more about the poll.

The search step is very flexible and allows the algorithm the opportunity to generate trial points anywhere on the mesh: the way of generating these points is free of any rules, as long as they remain on the current mesh $M(k, \Delta_k)$ and that the search terminates in finite time. This partition of an iteration into a search and a poll steps is the key feature of MADS which is exploited in the present paper

Some search strategies are tailored for a specific application: for example if the problem is to optimize a wing shape, then some known wings models or configurations may be used. Other search strategies are generic, as the use of Latin Hypercube sampling [42,43]. Furthermore, different types of searches may be combined. This paper introduces a generic search inspired by the VNS metaheuristic.

A high level description of the algorithm is summarized in Fig. 1. The MADS parameters taken for the tests of Sect. 5 are the default parameters used in our NOMAD software [5]. The values of critical parameters (such as the initial mesh size parameter Δ_0) will be given in that section. We encourage the reader to consult [9] for a complete description of the algorithm.

A hierarchical convergence analysis is available for MADS, based on the black-boxes differentiability: the main convergence result is that under local Lipschitz assumptions, the algorithm produces a Clarke stationary point, i.e. a point $\hat{x} \in \Omega$ at which the generalized Clarke derivative of f is non negative for all the directions in the Clarke tangent cone at $\hat{x}$

Fig. 1 MADS algorithm

```
[0] Initializations
      $x_0 \in X$, $\Delta_0 \in \mathbb{R}^+$
      $k \leftarrow 0$

[1] Poll and search steps
      Search step
            evaluate the functions on a finite number
                  of points of $M(k, \Delta_k)$
      Poll step
            compute $p$ MADS directions $D_k \in \mathbb{R}^{n \times p}$
            construct the frame $P_k \subseteq M(k, \Delta_k)$
                  with $x_k$, $D_k$, and $\Delta_k$
            evaluate the functions on the $p$ points of $P_k$

[2] Updates
      determine the type of success of iteration $k$
      solution update ($x_{k+1}$)
      mesh update ($\Delta_{k+1}$)
      $k \leftarrow k + 1$
      check the stopping conditions, goto [1]
```

(see [18]). A corollary of this result is that without constraints and if f is strictly differentiable, then $\nabla f(\hat{x}) = 0$.

2.2 VNS

The VNS is a metaheuristic proposed by Hansen and Mladenović [25,36], and has been proved efficient on a large range of problems. More often than not, it is applied to combinatorial problems [17,24,26,27], but it is possible to use it with continuous variables as in [4,16,20] and in the present work.

Two fundamental elements are required to define a VNS method: a descent method and a neighborhood structure. The descent is a method executing moves with respect to the neighborhood structure, which defines all the different possible trial points reachable from the current solution. The objective of these moves is to improve the current solution, and are repeated until no improvement is possible. The last point of the descent is a local optimum with respect to the neighborhood structure used.

Local searches often terminate in the vicinity of a nearby local optimum. VNS uses a random perturbation method to attempt to move away from a local optimal solution, far enough so that a new descent from the perturbed point leads to an improved local optimum, localized in a new and hopefully deeper valley. The perturbation method relies on the neighborhood structure, and is parametrized by a non negative scalar ξ_k, the VNS amplitude at iteration k, which gives the order of the perturbation (it is not necessary small, as the term "perturbation" might suggests, and "shaking" will be used for the routine executing it). The implementation details of the perturbation method has to be defined specifically for each type of problems, as long as the idea of amplitude is defined and dependent of ξ_k. For example ξ_k could be a minimal desired distance between the two points before and after the perturbation, or the number of random elementary moves leading to the perturbed point. The most efficient perturbation methods are often linked to the problem properties. In the present paper, a generic perturbation method is described.

A description of the VNS metaheuristic is given in Fig. 2. The algorithm essentially consists of two loops. Each iteration of the inner loop is decomposed into two steps: first the

Fig. 2 VNS metaheuristic for minimizing $f : \mathbb{R}^n \to \mathbb{R}$

```
[0] Initializations
        $it_{max}$, $\xi_{max}$, $\xi_0$, $\delta \in \mathbb{N}^+$
        $x_0 \in X$
        $k \leftarrow 0$, $it \leftarrow 0$
[1] while $(it \le it_{max})$
        $\xi_k \leftarrow \xi_0$
        while $(\xi_k \le \xi_{max})$
            $x' \leftarrow shaking(x_k, \xi_k)$
            $x'' \leftarrow descent(x')$
            if $(f(x'') < f(x_k))$
                $x_{k+1} \leftarrow x''$
                $\xi_{k+1} \leftarrow \xi_0$
            else
                $x_{k+1} \leftarrow x_k$
                $\xi_{k+1} \leftarrow \xi_k + \delta$
            $k \leftarrow k + 1$
        $it \leftarrow it + 1$
```

current solution (typically a local optimum) is perturbed with an amplitude factor ξ_k, and then a descent is performed from the perturbed point. If a better solution is obtained, it becomes the new iterate, and the amplitude is reset to its initial value. Otherwise the amplitude is increased by a value $\delta > 0$ (called the VNS increment) so that the next perturbation will lead to a point more distant than the previous one. Finally, the inner loop terminates after a maximum amplitude ξ_{max} is reached.

The outer loop consists in repeating this process it_{max} times. The it_{max} parameter of the first level loop is crucial for the efficiency of most VNS implementations. However, in our context, this loop will implicitly be made by the MADS algorithm, and therefore we fix $it_{max} = 1$.

3 Coupling the MADS and VNS algorithms

The VNS algorithm and the MADS poll step have a complementary behavior: when no success has been made during an iteration, the next poll step generates trial points closer to the poll center, while the VNS explores a more distant region with a larger perturbation amplitude. This paper proposes to incorporate the VNS method in the MADS algorithm, as a search step (called the VNS search). The poll step remains unchanged so that the convergence analysis of MADS still holds.

3.1 General description

The MADS mesh provides a natural neighborhood structure to be used by the two VNS components (descent and perturbation) and only the update of the perturbation amplitude ξ_k has to be made outside of the VNS search step.

The entire convergence analysis of MADS is preserved when the two following conditions are met: first, at iteration k, all the VNS search trial points must lie on the mesh $M(k, \Delta_k)$, and second, their number must be finite. The general way to define the perturbation and the descent is now given, and in the next section, a practical implementation will be described and proved to be a valid search, by satisfying those two conditions.

Adding a VNS exploration in the search step of a MADS algorithm can be done by introducing only two new parameters. One parameter is $\Delta_V > 0$ and relates to the VNS shaking method, as described in the next paragraph. The other parameter is $\rho > 0$ and defines a stopping criteria for the descent. It is introduced at the end of this subsection.

The shaking at iteration k generates a point x' belonging to the current mesh $M(k, \Delta_k)$. The amplitude of the perturbation is relative to a coarser mesh, whose coarseness is independent of Δ_k. To do so, we introduce the VNS mesh size parameter

$$\Delta_V > 0 \text{ and VNS mesh } M(k, \Delta_V). \tag{2}$$

This parameter is constant and independent of the iteration number k. The reason why the VNS perturbation needs to be independent of the current mesh $M(k, \Delta_k)$ is that it should not be influenced by the specific MADS behavior (which is in fact the contrary of the VNS behavior, as said in the introduction of this section). Only the fact that an iteration succeed, or the number of successive failed iterations, can rule the VNS amplitude, as in the original VNS algorithm. Another way of viewing that fact is that for a given value of ξ_k, the perturbation has to be the same regardless of the mesh fineness or coarseness.

The shaking may be viewed as the function

$$\begin{aligned} \mathit{shaking} : \ \big(M(k, \Delta_k), \mathbb{N}\big) &\to M(k, \Delta_V) \subseteq M(k, \Delta_k) \\ (x, \xi_k) &\mapsto x' = \mathit{shaking}(x, \xi_k) \end{aligned}$$

where $\xi_k \in \mathbb{N}$ is the perturbation amplitude.

As it will be illustrated in the next section, the VNS mesh size parameter Δ_V can also be used as a criteria to decide if a VNS search should be performed at a given iteration.

The VNS descent is viewed as a function

$$\begin{aligned} \mathit{descent} : \ M(k, \Delta_V) &\to M(k, \Delta_k) \\ x' &\mapsto x'' = \mathit{descent}(x') \end{aligned}$$

and has to generate a finite number of mesh points. While the shaking randomly changes a point in hopes of moving away from a local optimum, the idea of the descent is to obtain an improved point x'' from x' (in the filter sense of [8], i.e. in terms of objective value and constraints violation). Ideally, a descent step must lead toward a local optimum. The descent step in VNS is important because x', as a randomly perturbed point, has a weak probability of being an interesting point.

In the MADS context, local optimality is defined with respect to the mesh. The descent step ideally leads to a mesh local optimum, with respect to the current step size Δ_k and the directions used. The descent method described here is generic, but it is easy to see that a specialized descent method could be used for each type of problem that could exploit some inner properties. In order to reduce the number of functions evaluations, the descent step may terminate as soon as it generates a trial point y close to another point x previously considered by the algorithm, i.e. when $\|x - y\|_\infty \leq \rho$,

$$\text{where } \ \rho > 0 \ \text{ is called the Descent Stop (DS) parameter.} \tag{3}$$

The idea of this Descent Stop criteria is to avoid exploring a region previously visited. We believe that this strategy of terminating prematurely a descent could be exported to the general VNS metaheuristic. Figure 3 gives a description of the algorithm. In the update step,

Fig. 3 General algorithm of the coupling of MADS and VNS

```
[0] Initializations
    $x_0 \in X$, $\Delta_0 \in \mathbb{R}^+$
    $\xi_0$ , $\xi_{max}$, $\delta$, $\Delta_V$
    $k \leftarrow 0$

[1] Poll and search step
    Search step (optional)
        $x' \leftarrow shaking\ (x_k, \xi_k)$      (perturb. of ampl. $\xi_k$
                                       on $M(k, \Delta_V)$)
        $x'' \leftarrow descent\ (x')$            (descent on $M(k, \Delta_k)$)
        $S_k \leftarrow$ finite number of
                points of $M(k, \Delta_k)$   (possibly empty)
        evaluate the functions on $S_k \cup \{x''\}$
    Poll step
        compute $p$ MADS directions $D_k \in \mathbb{R}^{n \times p}$
        construct the frame $P_k \subseteq M(k, \Delta_k)$
            with $x_k$, $D_k$, and $\Delta_k$
        evaluate the functions on the $p$ points of $P_k$

[2] Updates
    update of VNS amplitude ($\xi_{k+1} \leftarrow \xi_0$ or $\xi_{k+1} \leftarrow \xi_k + \delta$)
    updates of solution and mesh
    $k \leftarrow k + 1$
    check the stopping conditions, goto [1]
```

ξ_{k+1} is set to ξ_0 if no success has been made or if ξ_{max} has been reached. This last point allows to mimic the outer loop of the original VNS algorithm, the loop on it in Fig. 2.

3.2 Use of a static surrogate

Surrogate functions may be used in several ways in the context of GPS algorithms (see [14] for a generic framework using surrogates). They are useful when the functions defining the problem are costly to evaluate, because they are less complicated and give an approximation of the true functions. Surrogates do not need to be good approximations of the true functions: in [10], the surrogate function differed from f by a factor of roughly 200, but both f and the surrogate shared some similarities.

Two types of surrogates can be defined: static surrogates, tailored for a specific problem (see [13] for example), and dynamic surrogates, constructed dynamically during the execution of the algorithm through previous evaluations and possibly with some interpolation technique (kriging for example, see [34,39]).

In [10], three strategies are proposed for the use of surrogates with the MADS algorithm: first an entire run of an optimization algorithm on the surrogate may lead to a good starting point for another run with the true function. A surrogate may be used to order the poll and search trial points. The more promising points are evaluated first, and if an improvement is made, the others are not considered. Finally a surrogate may be used to determine if a search point is valuable enough so that the true function is to be evaluated.

The use of surrogates in this paper can be seen as a fourth way to use them. The descent step can be entirely performed on the surrogate function. The true function can then be evaluated only once, at the final point produced by the descent. This strategy reduces the cost of a descent to a single expensive evaluation of the true function, and some less expensive surrogate functions.

4 Practical implementation

The algorithm presented in the previous section is generic and flexible. We now present a specific implementation, using the LTMADS implementation of [9], by specifying some parameter values and defining the perturbation and descent methods

$$shaking : \big(M(k, \Delta_k), \mathbb{N}\big) \to M(k, \Delta_V) \subseteq M(k, \Delta_k)$$

and

$$descent : M(k, \Delta_k) \to M(k, \Delta_k).$$

For the perturbation $x' \leftarrow shaking(x_k, \xi_k)$, at iteration k, two conditions are imposed: first, the point $x' \in M(k, \Delta_V)$ is chosen so that the distance in ℓ_∞ norm between x_k and x' is $\xi_k \Delta_V$. This distance is not based on the current mesh size Δ_k because the perturbation amplitude should only be linked to the value of ξ_k, as in the original VNS. This is the case as Δ_V is a parameter fixed by the user at the beginning of the algorithm. Second, in order to ensure that x' belongs to the current mesh $M(k, \Delta_k)$, the *shaking* procedure is triggered at iteration k when the VNS mesh size parameter is an integer multiple of the mesh size parameter Δ_k, i.e., there exists a non negative integer ℓ such that $\Delta_V = \ell \Delta_k$. Under the LTMADS mesh size update rule and the basic $2n$ directions $D = [-I\ I]$, this condition is met as soon as $\Delta_k \leq \Delta_V$, and x' necessarily belongs to $M(k, \Delta_k)$ since $M(k, \Delta_V) = M(k, \ell\Delta_k) \subseteq M(k, \Delta_k)$. The choice of the VNS mesh size parameter Δ_V directly influences at which iterations a VNS search is performed.

Figure 4 shows two examples of meshes of sizes Δ_V and Δ_k with possible choices for a perturbation, with $n = 2$. The perturbation algorithm used in Sect. 5 is given in Fig. 5.

A final remark concerning the choices of the amplitudes ξ_0 and ξ_{max} and of the VNS increment δ is that they are chosen so that a perturbation of order 20 from a point at its lower bound generates a perturbed point on its upper bound. This value has been chosen as it is empirically good for VNS codes.

Since the poll step is efficient in identifying mesh local optima, it is natural to use it for the VNS descent step. However, the current mesh size cannot be reduced so that all the points evaluated during the descent step belong to the current mesh. The VNS descent terminates

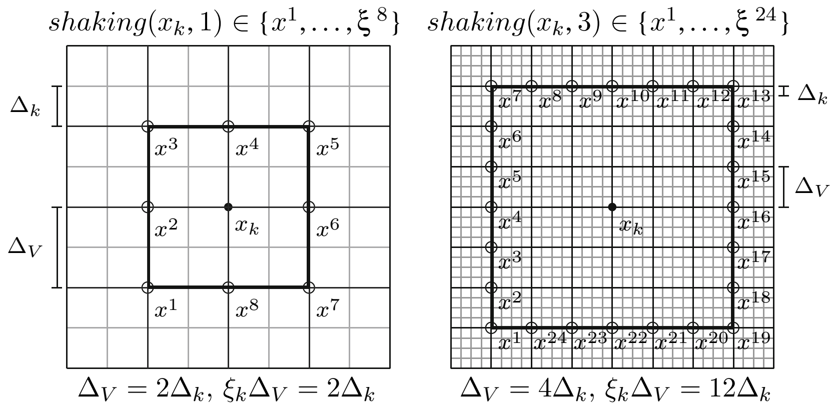
$shaking(x_k, 1) \in \{x^1, \ldots, \xi^8\}$
$shaking(x_k, 3) \in \{x^1, \ldots, \xi^{24}\}$
Δ_k
Δ_V
x^1 x^2 x^3 x^4 x^5 x^6 x^7 x^8 x_k
x^1 … x^{24} x_k
$\Delta_V = 2\Delta_k,\ \xi_k\Delta_V = 2\Delta_k$
$\Delta_V = 4\Delta_k,\ \xi_k\Delta_V = 12\Delta_k$

Fig. 4 Two examples of meshes $M(k, \Delta_k)$ (gray), $M(k, \Delta_V)$ (black) and possible perturbation choices (points x^i on the bold frame at distance $\xi_k \Delta_V$ from x_k)

```
$shaking\ (x_k, \xi_k)$
    generate randomly $w \in \mathbb{Z}^n$ such that $\|w\|_\infty = \xi_k$
    for each variable index $i \in [1; n]$
        if lower bound $L^i$ is finite and $\left(x_k^i + \Delta_V w^i < L^i\right)$
            $w^i \leftarrow \lceil (L^i - x_k^i) / \Delta_V \rceil$
        if upper bound $U^i$ is finite and $\left(x_k^i + \Delta_V w^i > U^i\right)$
            $w^i \leftarrow \lfloor (U^i - x_k^i) / \Delta_V \rfloor$
    $x' \leftarrow x_k + \Delta_V w$
```

Fig. 5 Practical implementation for the perturbation method; $L^i \in \mathbb{R} \cup \{-\infty\}$ and $U^i \in \mathbb{R} \cup \{+\infty\}$ respectively refer to the lower and upper bounds of variable i, $i \in [1; n]$

when the poll fails on the current mesh size (this is similar to the extended poll of [6]). It uses the MADS directions of the LTMADS implementation and its own mesh size parameter, called the descent mesh size. The initial value of the descent mesh size is taken to be the current mesh size.

The descent works on its own filter for the constraints management (the descent filter), reseted each time a new descent is performed. Each descent may be opportunist (the evaluations are stopped at the first success) or complete (they are completed regardless of the successes made). The mesh update is made as in LTMADS, for the dedicated descent mesh size.

The LTMADS optimist strategy is also used: if, during a poll iteration of the descent, an improvement point in the direction d is found, the next descent iteration will evaluate the functions at a point further along the direction d.

With this practical implementation for the perturbation and descent methods, the following pair of propositions ensures that the VNS search is valid as a search step of the MADS algorithm.

Proposition 4.1 *At iteration k, if the VNS search occurs, it generates trial points lying on the current mesh $M(k, \Delta_k)$.*

Proof At iteration k, it has already been shown that the perturbed point x' lies on the current mesh, since $\exists \ell \in \mathbb{N}^+$ such that $\Delta_V = \ell \Delta_k$ (condition for the VNS search to occur). By applying the rules of the MADS poll for the descent, all the trial points will also belong to the mesh. Even if surrogates are used for the descent, the unique final evaluation will be made for a point on the mesh. □

Proposition 4.2 *At iteration k, if the VNS search occurs, it generates a finite number of trial points.*

Proof Proposition 4.1 ensures that all the VNS search trial points are on the current mesh. Combining this with the assumption that all the trial points are in a compact set implies that their number is finite (see Proposition 3.4 in [7] for a more detailed proof). □

In practice, one may simply limit the number of different VNS trial points. For example, in the numerical tests of Sect. 5, a limit of 60 trial points is imposed.

5 Numerical tests

This section describes numerical results for three different problems on which the algorithm was tested. The first one is a bound constrained analytic two variables problem, the second is a multidisciplinary optimization problem with 10 variables and 10 constraints, and the last one is an engineering problem from the chemical industry with eight variables and eleven black-box constraints. Surrogate functions are used in the last two problems. All source codes for these three problems are available on the web site www.gerad.ca/Charles.Audet.

5.1 Algorithm parameters and testing protocols

The MADS implementation used is the research version of the code NOMAD [5], with the following parameters: the poll step uses the MADS $2n$ directions and is complete (i.e., black-box functions are evaluated for all poll trial points). The MADS dynamic ordering of the poll directions is performed after each successful iteration (see [9] for details). Scaling of the variables is done in a way that the mesh size parameters Δ_0, Δ_{min}, and Δ_V are always presented as proportions of the variable ranges. For example, for a $n = 2$ problem with $L = [-1\ 0]$ and $U = [1\ 10]$, an indicated value of $\Delta_0 = 0.1$ corresponds in fact to the scaled vector $\Delta_0^{\text{scl}} = [0.2\ 1]$ (the same for Δ_{min} and Δ_V). Scaling can also be done directly into the black-box code.

The VNS mesh size parameter Δ_V is rounded so that the condition $\Delta_k = \ell\Delta_V$ is verified when $\Delta_k \leq \Delta_V$ so that the user does not need to compute the exact value of Δ_V compatible with some Δ_k). For example, without scaling, if the user chooses $\Delta_V = 0.001$ and $\Delta_0 = 1$, Δ_V will be rounded to $1/1024$, the closest integer power of 4.

The filter [8,22] is used for open constraints with the squared ℓ_2 norm to define the constraint violation. The algorithm terminates when Δ_k drops below a parameter Δ_{min} or when a limit on the number of true evaluations is reached (n_{eval}^{max}). In the present work, we use n_{eval}^{max}=10,000. These MADS parameters remains unchanged throughout the numerical tests, as it is not the point here to analyze the MADS-alone behavior.

The Latin Hypercube (LH [42,43]) search strategy is also used with two parameters $n_{init} = 100$ and $n_{iter} = 10$: n_{init} is the number of LH trial points generated at the first iteration of MADS, and n_{iter} the number of LH trial points generated at each subsequent iteration $k \geq 1$. The initial LH step is complete while the other steps are opportunist (i.e., the iteration terminates immediately after a first success).

An upper limit of 60 trial points is imposed for every VNS search. For the descent step of VNS, the evaluations are opportunist and the directions used are the standard MADS $2n$ directions $\{\pm e_i : i = 1, 2, \ldots, n\}$ where e_i is the ith column of the identity matrix. When the premature descent stop technique (DS) is used, the parameter ρ of Eq. 3 is given. If available, surrogates are used in the descent step of VNS as described in Sect. 3.2, and not in the other MADS components.

Six algorithmic variants defined by different combinations of parameters are tested. The different algorithms configurations are detailed in Fig. 6 and Table 1. For comparison purposes, the two first algorithms do not use the new features proposed in this paper: algorithm A uses only MADS without a search strategy and algorithm B combines MADS and Latin Hypercube (LH) search. All other algorithms use MADS and VNS, and differ in their use of LH search, surrogates (Sgte = “yes” or “no”) or DS strategy (ρ value or “no”). Latin Hypercube (LH) search always uses parameters $n_{init} = 100$ and $n_{iter} = 10$.

Because MADS directions are randomly chosen and two runs with the same parameters can give different results, 30 runs are made for each algorithm. Throughout the tests, algorithms D, E, and F, which use VNS and one additional feature (DS, LH or surrogates) can be mixed into other variants. These variants are denoted by D+E, D+F, and E+F (algorithm D+E uses MADS, VNS, surrogates, and LH, the same logic applies to D+F and E+F).

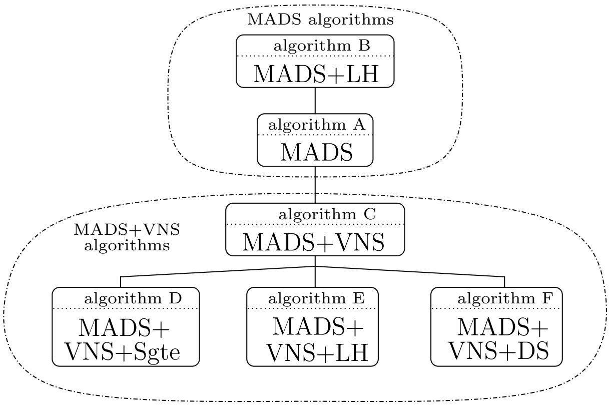

Fig. 6 Schematic description of the six main types of algorithms. Algorithms C, D, E, and F are variations of the new algorithm proposed in this paper

Table 1 Detailed parameters description of the six main algorithms

Algorithm	MADS	LH	VNS	Sgte	DS
A	Yes	No	No	No	No
B	Yes	$n_{init} = 100$ $n_{iter} = 10$	No	No	No
C	Yes	No	$\Delta_V = 0.01$ or 0.05 or 0.1	No	No
D	Yes	No	$\Delta_V = 0.01$ or 0.05 or 0.1	Yes	No
E	Yes	$n_{init} = 100$ $n_{iter} = 10$	$\Delta_V = 0.01$ or 0.05 or 0.1	No	No
F	Yes	No	$\Delta_V = 0.01$ or 0.05 or 0.1	No	$\rho = 0.005$ or 0.01 or 0.1

MADS parameters are default [5] parameters plus $\Delta_0 = 0.001$ or 0.05 and $\Delta_{min} = 10^{-12}$ or 10^{-7}

Results are summarized in Figs. 8, 9, and 11 and presented through 7 subgraphs. Each subgraph represents the objective function value (f) versus the number of black-box evaluations ($neval$) for the 30 runs of each series. One black-box evaluation is counted for the call of all the black-boxes defining the problem (objective and constraints). Note that when no surrogate is used, $neval$ denotes the number of true evaluations. With surrogates, another statistic (specific for each problem) must be used, taking also into account the surrogate cost. The last subgraph gives a summary of the six tests (one for each main algorithm) with average values. This larger subgraph allows to compare directly the algorithms, in terms of quality of the solution and evaluations cost. Results for variants D+E, D+F, and E+F are not shown in the first 6 subgraphs but appear in the summary subgraph.

Finally, Tables 2, 4, and 7 present the numerical values of all the tests. Each row of these tables is dedicated to one algorithm, the first columns give the parameters used, and the other columns give average, best and worst values for f and $neval$ over the 30 runs. The runs above the horizontal line do not use the algorithmic features proposed in this document.

5.2 An analytic problem with many local optima

This problem is taken from [46, problem 4]. It has two variables and the function to minimize is

Table 2 Analytic problem: testing parameters and numerical results

Algorithm	Parameters			Average		Objective (f)		$Neval$	
	LH	VNS (Δ_V)	DS (ρ)	Obj. (f)	$Neval$	Best	Worst	Best	Worst
A	No	No	No	−1.865	321	−3.063	0.453	266	436
B	Yes	No	No	−2.248	1,283	−3.307	−1.596	916	1,823
C	No	0.01	No	−3.009	5,182	−3.307	−2.575	3,399	9,016
E	Yes	0.01	No	−3.055	6,146	−3.307	−2.705	3,708	10,000
F	No	0.01	0.01	−2.837	2,809	−3.307	−2.413	2,202	4,088
E+F	Yes	0.01	0.01	−2.778	3,171	−3.307	−1.964	2,401	4,258

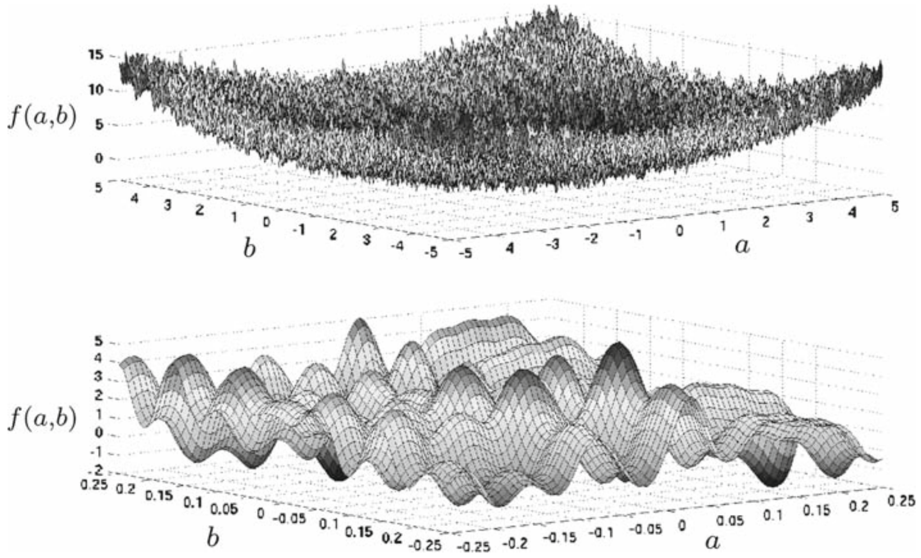

Fig. 7 Graph of the analytic problem function with bounds [−5; 5] and [−0.25; 0.25]

$$f(a,b) = e^{\sin 50a} + \sin(60e^b) + \sin(70\sin a) + \sin\big(\sin(80b)\big) - \sin\big(10(a+b)\big) + \frac{1}{4}(a^2+b^2).$$

The closed bounds constraints $-5 \le a, b \le 5$ are added since it can easily be shown that $f(a,b) \ge f(0,0)$ whenever (a,b) lies outside these bounds.

The graph of the objective function is shown in Fig. 7. One may observe the numerous local optimal solutions. The plot on the top part of the figure shows the function on the entire domain, and the one on the bottom zooms in on the rectangle $-\frac{1}{4} \le a, b \le \frac{1}{4}$.

Since the function is analytic and not costly, no surrogate function is used. Our first set of results used $x_0 = [0\,0]^T$ as starting point. Unfortunately, this starting point is very close to the global optimal solution ($x^* \simeq [-0.024\ 0.211]^T$ with $f \simeq -3.307$), and all runs converged trivially to the global solution. Therefore, a new starting point far from the optimum was chosen: $x_0 = [3\ 3]^T$, for an objective function value of $f \simeq 4.721$. The initial mesh size parameter Δ_0 is set to be 0.05 times the variable ranges and Δ_{min} to 10^{-12} times the ranges.

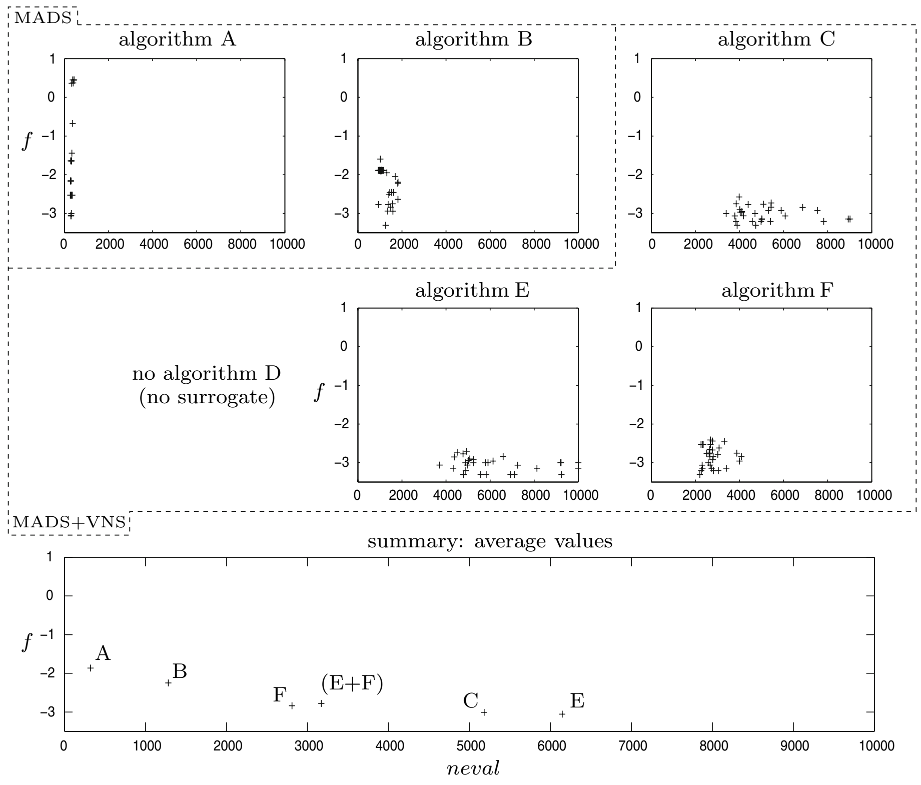

Fig. 8 Analytic problem: graphs f (objective function value) versus $neval$ (number of black-box evaluations)

Two different values for Δ_V are used in the different runs: 0.01 and 0.05 times the variable range.

Results are shown in Table 2 and Fig. 8. The MADS alone and MADS with LH (algorithms A and B) performed well in terms of number of function evaluations, but did not often terminate near the global optimum. The VNS algorithms C and E (VNS and VNS+LH) seem very appropriate in terms of global quality of the 30 runs (best average values for f). This quality came at the cost of additional evaluations. Tests using the DS strategy have fewer evaluations, but a lower quality of solution as well.

On a problem with a large number of local solutions, it is not surprising to see that the VNS search strategy is useful, since VNS is a metaheuristic designed to move away from local solutions.

We have also observed that the larger value reached by the ξ_k parameter is almost always below 10, for an allowed maximum of $\xi_{max} = 20$. It means that the outer loop on it in the original VNS algorithm (Fig. 2), which is implicitly made in the update step of the new algorithm (Fig. 3), is not useful for this problem. This remark holds also for the other two problems.

5.3 A MDO problem: aircraft range optimization

This problem is a Multidisciplinary Design Optimization (MDO) problem taken from [37,41] from the mechanical engineering literature. Three coupled disciplines (structure, aerodynam-

Table 3 MDO problem: starting point (x_0), best known point (x^*), and bounds ($L, U \in \mathbb{R}^n$)

	x_0	x^*	L	U
	0.25	0.40	0.10	0.40
	1	0.75	0.75	1.25
	1	0.75	0.75	1.25
	0.5	0.156244	0.1	1.0
	0.05	0.06	0.01	0.09
	45,000	60,000	30,000	60,000
	1.6	1.4	1.4	1.8
	5.5	2.5	2.5	8.5
	55	70	40	70
	1,000	1,500	500	1,500
f	−535.82 (infeasible)	−3964.20		

ics, and propulsion) are used to represent a simplified aircraft model, with 10 variables. The objective function is to maximize the aircraft range under bounds constraints and 10 open constraints. The black-box performs an iterative fixed points method through the different disciplines in order to compute the aircraft range. A natural surrogate consists in using a greater relative error as stopping criteria, and a smaller limit on the maximal number of fixed points iterations. The total number of fixed points iterations (from surrogates or true functions evaluations) is used instead of the number of black-boxes evaluations in the results ($fpit$).

The starting point x_0, the best known point x^* and the bounds are given in Table 3. The parameters Δ_0, Δ_{min}, and Δ_V are taken respectively to 0.001, 10^{-7}, and 0.1 times the variables ranges. Note that this problem can be solved more efficiently by other MDO-specialized optimization methods, as in [41], but the purpose of the results shown here is to make a comparison between the basic MADS algorithm and the coupling of MADS and VNS.

Results on Fig. 9 and Table 4 show the same trend as the ones of the analytic problem: the use of the VNS improves the global quality of the solutions, at the cost of additional evaluations. Here, the runs that stand out as the best ones are for algorithms D and D+F (VNS with surrogates). The use of the DS strategy and surrogates in algorithm D+F gave excellent results by lowering the number of evaluations for almost the same quality in terms of objective function value. For this problem the LH search seems not very appropriate, even when combined with the VNS search.

5.4 An engineering problem: styrene process optimization

The problem to optimize is a styrene production process simulation. Optimization of chemical processes simulation using an outside optimizer has previously been studied in [11,15,29]. Styrene production process is divided into four steps: reactants preparation (pressure rise and evaporation), catalytic reactions (as in [40]), styrene recovery (first distillation), and benzene recovery (second distillation). There is also an important recycling of unreacted ethylbenzene. All these steps appear in Fig. 10.

A chemical process simulator was developed based on the Sequential Modular Simulation (SMS) paradigm. SMS is a widely used approach [19,38,44], mostly because it lies on fast iterative methods. For some given operating parameters, each block can compute its output only when its input has been calculated. The main deficiency is recycling loops: the first blocks of the loop require the evaluation of the lasts ones.

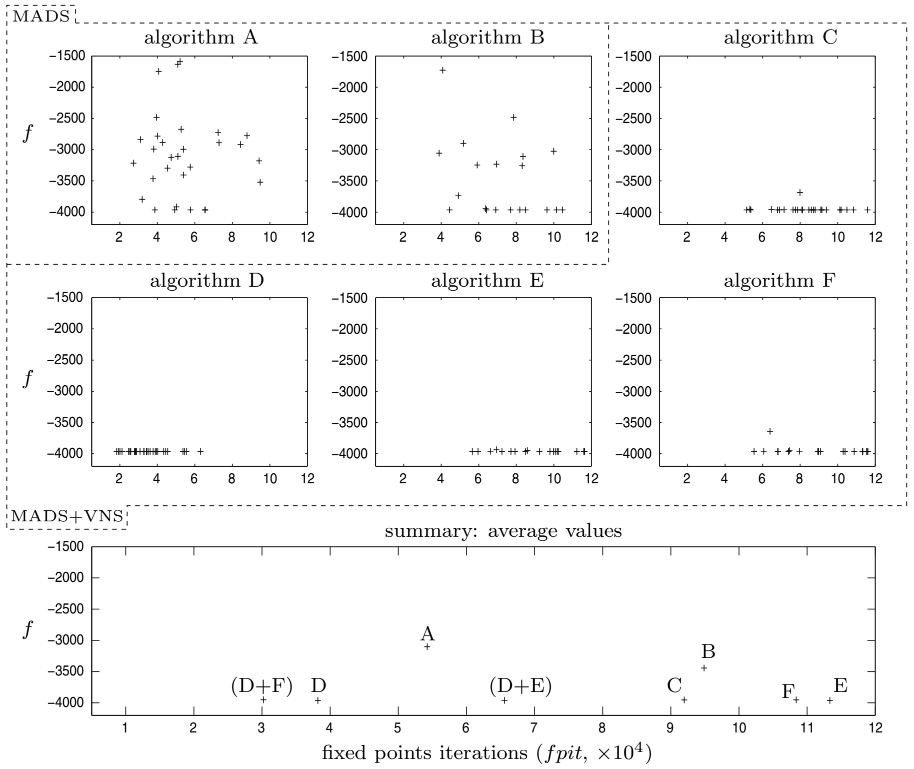

Fig. 9 MDO problem: graphs f (objective function value) versus $fpit$ (number of fixed points iterations)

Table 4 MDO problem: testing parameters and numerical results

Algorithm	Parameters				Average		Objective (f)		$fpit$	
	LH	VNS (Δ_V)	DS (ρ)	Sgte	Obj. (f)	$fpit$	Best	Worst	Best	Worst
A	No	No	No	No	−3101.39	54,253	−3964.20	−1588.35	27,273	94,815
B	Yes	No	No	No	−3443.09	94,881	−3964.20	−1355.66	18,328	201,797
C	No	0.1	No	No	−3954.30	91,963	−3964.20	−3685.85	51,513	152,713
D	No	0.1	No	Yes	−3964.16	38,193	−3964.20	−3964.03	18,339	125,261
E	Yes	0.1	No	No	−3962.66	113,342	−3964.20	−3937.25	56,589	165,063
F	No	0.1	0.25	No	−3952.61	108,393	−3964.20	−3640.72	55,429	175,015
D+E	Yes	0.1	No	Yes	−3961.38	65,586	−3964.20	−3881.93	26,220	148,164
D+F	No	0.1	0.1	Yes	−3950.60	30,200	−3964.20	−3657.49	9,984	69,839

The simulator black-box uses some common methods such as Runge-Kutta, Newton, Wegstein (fixed points), secant, bisection, and many other chemical engineering related solvers. The objective is to maximize the Net Present Value (NPV) of the styrene production process project, while satisfying industrial and environmental regulations. This is given by

$$NPV = \sum_{i=0}^{n} \frac{(S_i - C_i)(1 - T_a) - I_i + D_i}{(1 + T_r)^i}$$

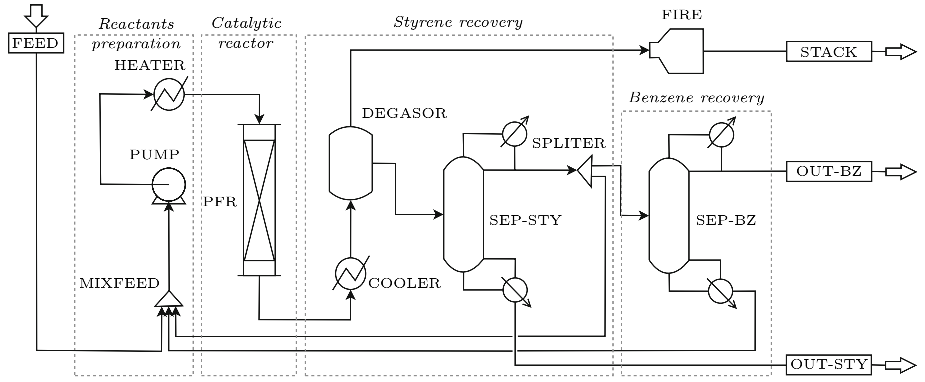

Fig. 10 Flowsheet of the styrene production process

Table 5 Styrene problem: variables and objective description, starting point (x_0), best point found (x^*), and bounds ($L, U \in \mathbb{R}^n$)

Description	units	x_0	x^*	L	U
Outlet temperature in block HEATER	K	870	1,100	600	1,100
Length of reactor (block PFR)	m	13.88	16.9836	2	20
Light key fraction in block SEP-STY	–	0.086014	0.0968282	10^{-4}	0.1
Light key fraction in block SEP-BZ	–	0.008092	0.0001	10^{-4}	0.1
Outlet pressure of block PUMP	atm	7.22	2	2	20
Split fraction in block SPLITER	–	0.2599	0.224742	0.01	0.5
Air excess fraction in block FIRE	–	1.668	1.96261	0.1	5
Cooling temperature of block COOLER	K	330	403.025	300	500
$f = -NPV$	$-\$$	-10942600	-33539100		

Table 6 Styrene problem: constraints description, with partition into three groups

Group	j	Description of constraint c_j
Simulator	1	True if the simulation has succeed
Process	2	True if column SEP-STY is structurally acceptable
	3	True if column SEP-BZ is structurally acceptable
	4	True if mixture in FIRE can burn and if environmental regulations on CO and NO_x are met
	5	Minimal purity of produced styrene
	6	Minimal purity of produced benzene
	7	Minimal overall ethylbenzene conversion into styrene
Economics	8	Maximal payout time
	9	Minimal discounted cashflow rate of return
	10	Maximal total investment
	11	Maximal annual equivalent cost

where the index i denotes a year between 0 and n, S_i the sales, C_i the operating costs, T_a the income tax rate, I_i the investment, D_i is the depreciation, and T_r is the actualization rate. The source of difficulty in that optimization problem is that S_i, C_i, I_i, and D_i are functions of the simulator output. This output contains information such as equipment sizing, flow rates and compositions, units efficiencies, power needs, and so on. The simulation of the process must be successfully performed before the constraint and objective functions are evaluated. Tables 5 and 6 give additional information on the problem characteristics.

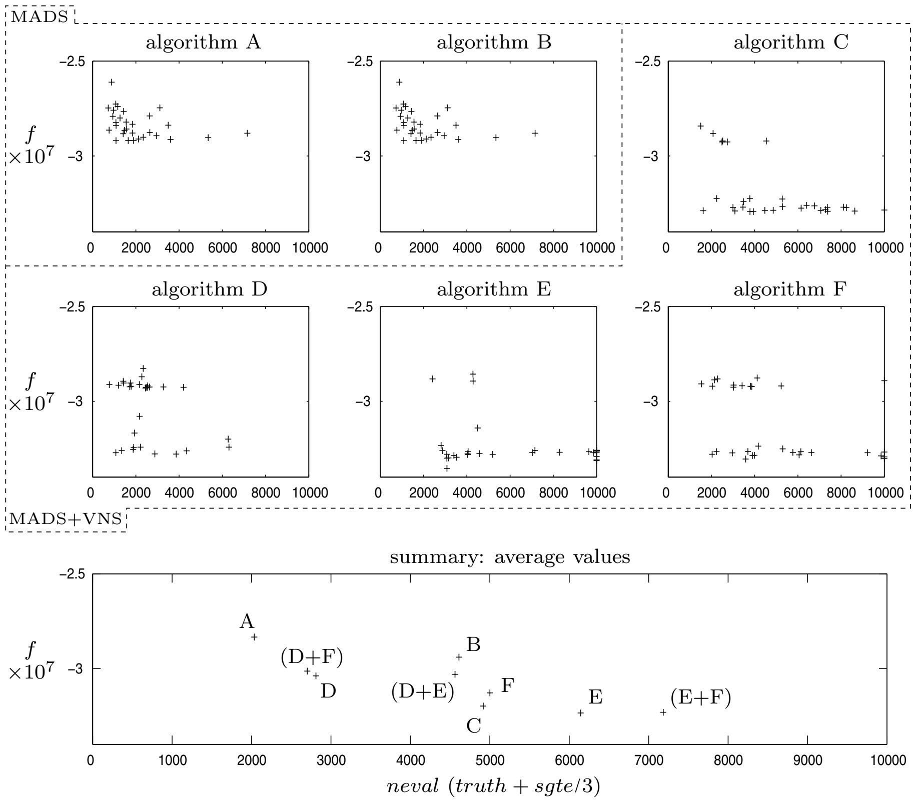

Fig. 11 Styrene problem: graphs f (objective function value) versus *neval* = true evaluations + surrogate evaluations/3

The surrogate is obtained by using greater tolerance values and smaller maximum numbers of iterations in the various numerical methods. As opposed to the MDO problem of the previous subsection, computational comparison between true and surrogate functions is not trivial. Intensive tests suggest that one true evaluation has approximatively the same cpu-time of three surrogate evaluations. For the results in Fig. 11 and Table 7, the "*neval*" value is then an approximation of the evaluations cost, obtained by *neval* defined to be the sum of number of true evaluations, with a third of the number of surrogate evaluations. Because of this, *neval* exceeds in some cases the upper bound of 10,000. The scaling is this time done directly into the black-box code with new unified bounds of [0 1]. The fixed parameters used for these series of 30 tests are $[\Delta_0\ \Delta_{min}\ \Delta_V]^T = [0.001\ 10^{-7}\ 0.05]^T$.

Figure 11 and Table 7 do not suggest a clear domination of one VNS strategy over all others. However, one may observe that algorithms C and E using VNS are the more stable runs in the sense that they very often produce a good solution with only a few outliers (this is mostly apparent in the plots of Fig. 11). Runs using MADS and MADS with LH search could not reach the best objective function values. VNS systematically generated the best solutions.

Table 7 Styrene problem: testing parameters and numerical results

Algorithm	Parameters				Average		Objective (f)		$Neval$	
	LH	VNS (Δ_V)	DS (ρ)	Sgte	Obj. (f)	$Neval$	Best	Worst	Best	Worst
A	No	No	No	No	−28334450	2,036	−29189900	−26104800	722	7,159
B	Yes	No	No	No	−29397777	4,612	−32650700	−24822000	1,236	10,000
C	No	0.05	No	No	−31974893	4,919	−32932200	−28417900	1,503	10,000
D	No	0.05	No	Yes	−30382607	2,811	−32783500	−28266000	775	11,193
E	Yes	0.05	No	No	−32339227	6,145	−33539100	−28566200	2,410	10,000
F	No	0.05	0.005	No	−31278540	5,001	−33046500	−28765300	1,538	10,000
D+E	Yes	0.05	No	Yes	−30302470	4,562	−32881500	−26903100	911	10,681
D+F	No	0.05	0.005	Yes	−30132383	2,703	−32793100	−26222400	783	11,280
E+F	Yes	0.05	0.005	No	−32304503	7,184	−33063200	−29173800	1,898	10,000

6 Discussion

This paper proposes a generic way to incorporate the VNS metaheuristic into the search step of the MADS algorithm. Notice that a similar combination was recently detailed in [47] where a generic particle swarm GPS search strategy is defined.

Our proposed algorithm belongs to the general MADS framework, and thus preserves all of its convergence properties. The algorithm remains simple, with only two additional parameters: the VNS mesh size parameter Δ_V used in the shaking, and the optional descent stopping parameter ρ. In the numerical results presented here we either used Δ_V to be one tenth, one twentieth or one hundredth of the range of the variables. In our software packages we will use one tenth as the default value. We observed that the algorithm was more sensitive to the descent parameter ρ, and will not adventure in suggesting default values.

The algorithm was applied to three problems and compared to classic MADS with or without the classic Latin Hypercube (LH) search strategy. Good results were obtained in terms of quality: the random aspect of the MADS directions is attenuated, leading to more stable solutions. This improvement in terms of quality generally comes to the price of a higher number of black-box evaluations, but VNS seems to use these additional evaluations more efficiently, compared to other methods such as LH search.

Acknowledgements Work of the first author was supported by FCAR grant NC72792 and NSERC grant 239436-05, AFOSR F49620-01-1-0013, the Boeing Company and ExxonMobil Upstream Research Company. Work of the first and third author was supported by the Consortium for Research and Innovation in Aerospace in Quebec.

References

1. Abramson, M.A.: Mixed variable optimization of a load-bearing thermal insulation system using a filter pattern search algorithm. Optim. Eng. **5**(2), 157–177 (2004)
2. Abramson, M.A., Audet, C.: Second-order convergence of mesh-adaptive direct search. SIAM J. Optim. **17**(2), 606–619 (2006)
3. Alberto, P., Nogueira, F., Rocha, U., Vicente, L.N.: Pattern search methods for user-provided points: application to molecular geometry problems. SIAM J. Optim. **14**(4), 1216–1236 (2004)
4. Audet, C., Brimberg, J., Hansen, P., Le Digabel, S., Mladenović, N.: Pooling problem: alternate formulations and solution methods. Manage. Sci. **50**(6), 761–776 (2004)

5. Audet, C., Couture, G., Dennis, J.E. Jr.: NOMAD Project (LTMADS Package). Software available at www.gerad.ca/nomad
6. Audet, C., Dennis, J.E. Jr.: Pattern search algorithms for mixed variable programming. SIAM J. Optim. **11**(3), 573–594 (2000)
7. Audet, C., Dennis, J.E. Jr.: Analysis of generalized pattern searches. SIAM J. Optim. **13**(3), 889–903 (2003)
8. Audet, C., Dennis, J.E. Jr.: A pattern search filter method for nonlinear programming without derivatives. SIAM J. Optim. **14**(4), 980–1010 (2004)
9. Audet, C., Dennis, J.E. Jr.: Mesh adaptive direct search algorithms for constrained optimization. SIAM J. Optim. **17**(1), 188–217 (2006)
10. Audet, C., Orban, D.: Finding optimal algorithmic parameters using the mesh adaptive direct search algorithm. SIAM J. Optim. **17**(3), 642–664 (2006)
11. Béchard, V., Audet, C., Chaouki, J.: Robust optimization of chemical processes using a MADS algorithm. Technical report G–2005–16, Les Cahiers du GERAD, Montréal (2005)
12. Booker, A.J., Dennis, J.E. Jr., Frank, P.D., Moore, D.W., Serafini, D.B.: Managing surrogate objectives to optimize a helicopter rotor design—further experiments. AIAA Paper 1998–4717, Presented at the 8th AIAA/ISSMO symposium on multidisciplinary analysis and optimization, St. Louis (1998)
13. Booker, A.J., Dennis, J.E. Jr., Frank, P.D., Serafini, D.B., Torczon, V. : Optimization using surrogate objectives on a helicopter test example. In: Borggaard, J., Burns, J., Cliff, E., Schreck, S. (eds.) Optimal Des. Control, Progress in Systems and Control Theory, pp. 49–58. Birkhäuser, Cambridge (1998)
14. Booker, A.J., Dennis, J.E. Jr., Frank, P.D., Serafini, D.B., Torczon, V., Trosset, M.W.: A rigorous framework for optimization of expensive functions by surrogates. Struct. Optim. **17**(1), 1–13 (1999)
15. Bowden, R.O., Hall, J.D.: Simulation optimization research and development. Simulation conference, 1693–1698, Presented at the 1998 winter simulation conference (1998)
16. Brimberg, J., Mladenović, N.: A variable neighbourhood algorithm for solving the continuous location-allocation problem. In: Hamacher, D. (ed.) Stud. Location Anal. vol. 10, pp. 1–12, Athens, Greece (1996)
17. Caporossi, G., Hansen, P.: Variable neighborhood search for extremal graphs 1: the AutoGraphiX system. Discrete Math. **212**, 29–44 (2000)
18. Clarke F.H.: 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
19. Douglas, J.M.: Conceptual Design of Chemical Processes. McGraw-Hill, New York (1988)
20. Drazić, M., Lavor, C., Maculan, N., Mladenović, N.: A Continuous VNS Heuristic for Finding the Tridimensional Structure of a Molecule. Technical Report G–2004–22, Les Cahiers du GERAD, Montréal (2004)
21. Finkel, D.E., Kelley, C.T.: Convergence analysis of the direct algorithm. SIAM J. Optim. 2004 (to appear)
22. Fletcher, R., Leyffer, S.: Nonlinear programming without a penalty function. Math. Program. Series A **91**, 239–269 (2002)
23. Fowler, K.R., Kelley, C.T., Miller, C.T., Kees, C.E., Darwin, R.W., Reese, J.P., Farthing, M.W., Reed, M.S.C.: Solution of a well-field design problem with implicit filtering. Optim. Eng. **5**(2), 207–234 (2004)
24. Hansen, P., Mladenović, N.: J-MEANS: a new local search heuristic for minimum sum of squares clustering. Pattern Recogn. **34**(2), 405–413 (2001)
25. Hansen, P., Mladenović, N.: Variable neighborhood search: principles and applications. Eur. J. Oper. Res. **130**(3), 449–467 (2001)
26. Hansen, P., Mladenović, N., Perez-Britos, D.: Variable neighborhood decomposition search. J. Heuristics **7**(4), 335–350 (2001)
27. Hansen, P., Mladenović, N., Urosevic, D.: Variable neighborhood search for the maximum clique. Discrete Appl. Math. **145**(1), 117–125 (2004)
28. Hayes, R.E., Bertrand, F.H., Audet, C., Kolaczkowski, S.T.: Catalytic combustion kinetics: Using a direct search algorithm to evaluate kinetic parameters from light-off curves. Can. J. Chem. Eng. **81**(6), 1192–1199 (2003)
29. Himmelblau, D.M., Edgar, T.F., Lasdon, L.S.: Optimization of Chemical Processes, 2nd edn. McGraw-Hill, New York (2003)
30. Jones, D.R., Perttunen, C.D., Stuckman, B.E.: Lipschitzian optimization without the Lipschitz constant. J. Optim. Theory Appl. **79**(1), 157–181 (1993)
31. Kokkolaras, M., Audet, C., Dennis, J.E. Jr.: Mixed variable optimization of the number and composition of heat intercepts in a thermal insulation system. Optim. Eng. **2**(1), 5–29 (2001)
32. Kolda, T.G., Lewis, R.M., Torczon, V.: Optimization by direct search: new perspectives on some classical and modern methods. SIAM Rev. **45**(3):385–482 (electronic) (2003)
33. Lewis, R.M., Torczon, V., Trosset, M.W.: Direct search methods: Then and now. J. Comput. Appl. Math. **124**(1–2), 191–207 (2000)

34. Lophaven, S., Nielsen, H., Søondergaard, J.: DACE—A Matlab Kriging toolbox, version 2.0. Technical report IMM-REP-2002-12, Informatics and mathematical modelling, Technical University of Denmark (2002)
35. Marsden, A.L., Wang, M., Dennis, J.E. Jr., Moin, P.: Optimal aeroacoustic shape design using the surrogate management framework. Optim. Eng. **5**(2), 235–262 (2004)
36. Mladenović, N., Hansen, P.: Variable neighborhood search. Comput. Oper. Res. **24**(11), 1097–1100 (1997)
37. Perez, R., Liu, H.H.T., Behdinan, K.: Evaluation of multidisciplinary optimization approaches for aircraft conceptual design. In: AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, NY, September (2004)
38. Seader, J.D., Sieder, W.D., Lewin, D.R.: Process Design Principles: Synthesis, Analysis and Evaluation. John Wiley and Sons Inc., New York (1999)
39. Serafini, D.B.: A framework for managing models in nonlinear optimization of computationally expensive functions. Ph.D. thesis, Department of Mathematical Sciences, Rice University, Houston, Texas (1998)
40. Snyder, J.D., Subramaniam, B.: A novel reverse flow strategy for ethylbenzene dehydrogenation in a packed-bed reactor. Chem. Eng. Sci. **49**, 5585–5601 (1994)
41. Sobieszczanski-Sobieski, J., Agte, J.S., Sandusky, R.R. Jr.: Bi-level integrated system synthesis (BLISS). Technical Report NASA/TM-1998-208715, NASA, Langley Research Center, August 1998
42. Stein, M.: Large sample properties of simulations using latin hypercube sampling. Technometrics **29**(2), 143–151 (1987)
43. Tang, B.: Orthogonal array-based latin hypercubes. J. Am. Stat. Assoc. **88**(424), 1392–1397 (1993)
44. Timmerhaus, K.D., Peters, M.S., West, R.E.: Plant Design and Economics for Chemical Engineers, 5th edn. McGraw-Hill, New York (2003)
45. Torczon, V.: On the convergence of pattern search algorithms. SIAM J. Optim. **7**(1), 1–25 (1997)
46. Trefethen, N.: The hundred dollar hundred digit challenge. SIAM News **35**(1), January–February 2002. www.siam.org/news/news.php?id=388
47. Vaz, A.I.F., Vicente, L.N.: A particle swarm pattern search method for bound constrained global optimization. J. Global Optim. 39(2), 197–219 (2007)