Exploiting Sparsity in Unconstrained Optimization Without Derivatives

Benoît Colson and Philippe L. Toint
Department of Mathematics, FUNDP, Rempart de la Vierge, 8, B-5000 Namur (Belgium)

12 February 2001

Abstract. This paper is concerned with derivative-free unconstrained optimization. We first discuss a method combining the use of interpolation polynomials and trust-region techniques to minimize a function whose derivatives are not available. We then show how the resulting algorithm may be adapted in a suitable way to consider problems for which the Hessian matrix is known to be sparse. Numerical experiments confirm the favourable behaviour of the method and in particular the advantages in terms of storage, function evaluations and speed.

Keywords: derivative-free optimization, interpolation models, trust-region methods, sparsity

1. Introduction

We consider the unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} f(x),$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a nonlinear, smooth, real-valued function. We further assume that the derivatives of f are not available, for instance because their evaluation is very difficult or time-consuming, or because $f(x)$ is the output of physical, chemical or econometric measurements. Models involving such complex functions are frequently encountered in the industrial world, more particularly in engineering and design optimization problems, which results in a high demand from practitioners for efficient algorithmic tools.

This subject has been extensively researched by the optimization community, and various approaches have been proposed. In the class of "pattern search" techniques, where a predefined class of geometrical patterns is used to explore the variable space, early contributions include those of Box [3], Campey and Nickols [7], Hooke and Jeeves [18], Spendley, Hext and Himsworth [32], Nelder and Mead [21], Dixon [16]. This approach was revisited more recently by Dennis and Torczon [14] [PDS], Buckley [5], Wright [37], Torczon [34], Dennis and Torczon [15], Alexandrov et al. [1], Audet et al. [2]. Linesearch methods have also been investigated for the problem, including proposals by Powell [23] [VA04,VA24], Brent [4] [PRAXIS], Callier and Toint [6], Lucidi and Sciandrone [19], while finite-difference techniques have been coupled with quasi-Newton algorithms in a very natural way, and are discussed in particular by Powell [24], Stewart [33], Powell [26, 25], Gill, Murray and Wright [17] or Dennis and Schnabel [13], Nocedal and Wright [22] or Conn, Gould and Toint [8]. In this paper we consider an algorithm based on a fourth possible approach, namely the use of interpolation methods to model the objective function and their combination with trust-region techniques. This class of methods has been pioneered by Winfield [35, 36], who used full quadratic models but didn't study convergence. This technique was later reinvestigated by Powell [27, 28, 29, 30], Marazzi and Nocedal [20], Conn and Toint [12], Conn, Scheinberg and Toint [9, 10, 11]. The main ideas of these latter contributions may be summarized as follows. Assuming that $f(x)$ is computable for all $x \in \mathbb{R}^n$, the basic idea of these methods is to use all the available function values to build a polynomial model interpolating the objective function f at the points at which its value is known. The model is then minimized within a trust region, yielding a new – potentially good – point. To check this possible improvement, the objective f is evaluated at the new point – thus possibly enlarging the interpolation set – and the whole process may then be repeated until convergence is achieved.

The aim of this paper is to specify these methods for the particular case where we know that the Hessian of $f(x)$ is sparse and to do this in a way that takes advantage of this particular structure so as to minimize the computational costs and in particular the number of function evaluations. Such a situation arises when it is known that the objective function can be viewed as the sum of "blocks" that do not individually involve all the variables, for instance.

In Section 2 we present the main ingredients of the abovementioned algorithm, including some preliminary considerations as regards the geometry of the interpolation set. The latter topics are extended in Section 3, where we consider the use of Newton fundamental polynomials as a basis for building the function model at each iteration and show how this technique may be specified for the case where the Hessian matrix $\nabla^2_{xx} f(x)$ is known to be sparse. The resulting savings in terms of computational effort are assessed through a series of numerical results reported in Section 4.

2. Description of the algorithm

At iteration k of the process, starting from iterate x^k, the algorithm computes the step $s \in \mathbb{R}^n$ minimizing the quadratic model

$$m_k(x_k + s) = f(x_k) + \langle g_k, s\rangle + \frac{1}{2}\langle s, H_k s\rangle, \tag{1}$$

for some $g_k \in \mathbb{R}^n$ and some symmetric $n \times n$ matrix H_k, where $\langle \cdot, \cdot \rangle$ denotes the dot product. It must be emphasized here that g_k and H_k do not necessarily correspond to the first and second derivatives of $f(\cdot)$ respectively since we assume they are not available. Rather, the latter vector and matrix are built by requiring that the model (1) satisfy

$$m_k(y) = f(y) \quad \forall y \in Y, \tag{2}$$

where Y denotes the set of points at which the value of f is known, including the current iterate x^k. Building (1) requires to determine $f(x_k)$, the components of g_k and the entries of H_k, that is

$$p = 1 + n + \frac{1}{2}n(n+1) = \frac{1}{2}(n+1)(n+2) \tag{3}$$

parameters. Since the model (1) is entirely determined by the conditions (2), this is equivalent to say that we need to know at least p function values, that is

$$|Y| = p. \tag{4}$$

However this is not sufficient to guarantee the good quality of the model, and we need further geometric conditions – known as *poisedness* – on the points in Y to ensure both the existence and uniqueness of an interpolant. If we denote by $\{\phi_i(\cdot)\}_{i=1}^p$ a basis of the linear space of n-dimensional quadratics, then the model we are looking for may be written as

$$m_k(x) = \sum_{i=1}^{p} \alpha_i \phi_i(x)$$

for some scalars α_i $(i = 1, \ldots, p)$ to be computed. Particularizing conditions (2) to this expression for m_k yields

$$\sum_{i=1}^{p} \alpha_i \phi_i(y) = f(y) \quad \text{for all } y \text{ in } Y,$$

which is a system of $|Y| = p$ linear equations whose unknowns are the α_i's. It follows that $Y = \{y_1, \ldots, y_p\}$ is poised if the determinant

$$\delta(Y) = \det \begin{pmatrix} \phi_1(y_1) & \cdots & \phi_p(y_1) \\ \vdots & & \vdots \\ \phi_1(y_p) & \cdots & \phi_p(y_p) \end{pmatrix} \tag{5}$$

is nonzero. In practice, we consider Y to be poised whenever $|\delta(Y)| \geq \epsilon$ provided the y_j's span the neighbourhood of the current iterate x_k in a "good way". But this measure of poisedness is not the only possible one, and we will use another technique in what follows, that is more consistent with our use of the Newton fundamental polynomials as a basis for constructing the interpolating quadratics.

Before giving the complete formulation of the algorithm, we must mention some more of its ingredients related to the management of the interpolation set and to the fact that the computation of the interpolant is embedded in a trust-region framework.

The first issue deals with the acceptance of a new iterate. Assuming that after a successful iteration we get a point x_k^+ of associated function value $f(x_k^+)$ lower than $f(x_k)$, one may now consider the problem of finding the best way to make x_k^+ play a role in the next iterations when building the interpolating quadratic. Indeed, including x_k^+ in Y means that we need to remove another point y from Y (except maybe in the course of the first few iterations, where Y might be incomplete) in which case we must proceed with care so as not to deteriorate the geometry of Y, making the latter set "as poised as possible". Moreover, since x_k^+ is obtained through a minimization process which does not take the geometry of Y into account, there is no guarantee that the quality of the geometry of the interpolation set remains acceptable once x_k^+ is included in Y. This means that the quality of the geometry of Y as well as its poisedness might be deteriorated as and when new iterates are computed and accepted by the algorithm.

A second feature of trust-region algorithms is the management of the trust-region radius. At iteration k, the trust region is defined by $\mathcal{B}_k = \{x_k + s \mid \|s\| \leq \Delta_k\}$. In classical setups, the radius Δ_k is decreased when no significant progress can be made as regards the reduction of the objective function. In our framework, however, we must first verify that the interpolation set is poised before reducing Δ_k since a bad geometry might be the major reason for the algorithm to stall. If Y is not poised, we have to improve its geometry before possibly reducing Δ_k. It is particularly important not to modify Δ_k too early in the process since the geometry improvement is precisely achieved by introducing a new point y^+ in Y such that $\|y^+ - x_k\| \leq \Delta_k$ and using a suitable improvement measure to evaluate the advantage of replacing some past point $y^- \in Y \backslash \{x_k\}$ by y^+.

We will see in Section 3 how the use of the Newton fundamental polynomials provides a way to assess the abovementioned geometry improvements.

This concludes the description of the main ingredients of the algorithm, whose complete formulation may be stated as follows:

Algorithm UDFO

Step 0: Initialization. An initial point x_0 is given, as well as a trust-region radius $\Delta_0 > 0$, an initial model based on a vector g_0 and a matrix H_0, as well as constants $\gamma_1 \leq 1 \leq \gamma_2$ and $0 < \eta_0 \leq \eta_1 < 1$.

Step 1: Criticality test. If g_k is small, then improve the geometry until Y is poised in a ball of radius $\delta \leq \mu \|g_k\|$ centered at x_k.

Step 2: Subproblem. Compute s_k from

$$\min_{s \in \mathcal{B}_k} m_k(x_k + s).$$

Step 3: Evaluation. Compute $f(x_k + s_k)$ and

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)}.$$

If $\rho_k \geq \eta_1$, define $\mathcal{X}_k = \{x_k + s_k\}$.
Otherwise $\mathcal{X}_k = \{x_k\}$.

Step 4: Model management.
If $\rho_k \geq \eta_1$, include $x_k + s_k$ in Y.
Otherwise, if Y is not poised, improve the geometry (possibly enlarging $\mathcal{X}_k$).

Step 5: Next iterate. Compute

$$\hat{x}_k = \arg \min_{x \in \mathcal{X}_k} f(x)$$

and

$$\hat{\rho}_k = \frac{f(x_k) - f(\hat{x}_k)}{m_k(x_k) - m_k(x_k + s_k)}.$$

If $\hat{\rho}_k \geq \eta_0$, accept $\hat{x}_k$, and set $x_{k+1} = \hat{x}_k$.
Otherwise let the iterate unchanged, *i.e.* set $x_{k+1} = x_k$.

Step 6: Radius update.
If $\hat{\rho}_k \geq \eta_0$ or Y was poised, set $\hat{\Delta} = \Delta_k$.
If $\rho_k \geq \eta_1$, choose $\Delta_{k+1} \geq \hat{\Delta}$.
Otherwise choose $\Delta_{k+1} \in [\gamma_1 \hat{\Delta}, \gamma_2 \hat{\Delta}]$.
Go to Step 1.

3. Newton fundamental polynomials, geometry ans sparsity

As indicated earlier, the interpolating quadratics (1) are built using Newton fundamental polynomials. The scope of this section is to present the basic concepts used when constructing these polynomials as well as to derive from this technique procedures for improving the geometry of Y and for including a new iterate x_k^+ in the interpolation set. Further, we show how to adapt the whole method so as to consider the case where the Hessian $\nabla^2_{xx} f(x)$ is sparse. For more details on Newton fundamental polynomials – and more generally on multivariate interpolation – we refer the reader to Sauer and Xu [31].

Newton fundamental polynomials are built with an increasing degree up to the degree d of the desired interpolation polynomial and are arranged by blocks. This is why the set Y is first partitioned into $d+1$ blocks $Y^{[\ell]}$ ($\ell = 0, \ldots, d$). The ℓ-th block contains $|Y^{[\ell]}| = \binom{\ell+n-1}{\ell}$ points and to each point $y_i^{[\ell]} \in Y^{[\ell]}$ corresponds a single Newton fundamental polynomial of degree ℓ satisfying the conditions

$$N_i^{[l]}(y_j^{[m]}) = \delta_{ij}\delta_{lm} \text{ for all } y_j^{[m]} \in Y^{[m]} \text{ with } m \leq l. \tag{6}$$

The details of the procedure for constructing the Newton fundamental polynomials for a given set Y have been omitted in this presentation (the interested reader is refered to *e.g.* Conn, Scheinberg and Toint [9]) since we prefer to focus on other algorithmic issues. It may be sufficient for now to mention that this procedure may be considered as a particular case in which the well-known Gram-Schmidt orthogonalization procedure is applied to an initial polynomial basis with respect to the inner product

$$\langle P, Q \rangle = \sum_{y \in Y} P(y) Q(y).$$

As an example, a suitable initial basis in the framework of quadratic interpolation (*i.e.* for $d = 2$) might be to start with the following polynomials:

$$1,\ \{x_i\}_{1 \leq i \leq n},\ \{x_i^2\}_{1 \leq i \leq n}, \{x_i x_j\}_{1 \leq i < j \leq n}. \tag{7}$$

Once the Newton fundamental polynomials are built, the interpolating polynomial $m_k(x)$ is given by

$$m_k(x) = \sum_{y_i^{[\ell]} \in Y} \lambda_i^{[\ell]}(Y, f) N_i^{[\ell]}(x), \tag{8}$$

where the coefficients $\lambda_i^{[\ell]}(Y,f)$ are generalized finite differences defined by the following formulae (see Sauer and Xu [31])

$$\lambda_0(x) = f(x) \text{ and } \lambda_{\ell+1}(x) = \lambda_\ell(x) - \sum_{i=1}^{|Y^{[\ell]}|} \lambda_\ell(y_i^{[\ell]}) N_i^{[\ell]}(x) \;\; (\ell = 0, \ldots, d-1).$$

We now particularize the method to the case where the Hessian matrix $\nabla^2_{xx} f(x)$ is known to be sparse independently of x. This implies that there exists a symmetric index set

$$\mathcal{S} = \{ \; (i,j) \mid 1 \leq i,j \leq n \text{ and } \langle e_i, \nabla^2_{xx} f(x) e_j \rangle = 0 \; \forall x \in \mathbb{R}^n \}. \quad (9)$$

Intuitively, the sparse nature of the Hessian should be reflected in the model (1) approximating $f(x)$ within the trust-region. However, as we said before, working in the framework of derivativre-free optimization implies that the matrix H_k appearing in (1) may be different from the Hessian of $f(\cdot)$ and thus the particular structure of the Hessian is expected to reverberate in a less direct way in the algorithm.
A simple idea for the process for building the interpolating quadratics to take advantage of the simplified structure of the Hessian is to reduce the approximation space to the set of quadratic polynomials satisfying (9). Implementing this strategy in the framework of multivariate interpolation amounts to compute the Newton fundamental polynomials $N_i^{[\ell]}$ by orthogonalizing a *partial* basis of polynomials. For instance, if we consider an initial basis of the type described by (7), we will exclude the polynomials taking the form $x_i x_j$ for $(i,j) \in \mathcal{S}$.
As a result, the number of polynomials to generate is decreased and the algorithm needs fewer interpolation points. In other words, the interpolation set Y may become significantly smaller than for the case for which the Hessian matrix is dense.
This way to proceed and to deal with sparsity has a triple impact on the method and the computations performed at each iteration of the algorithm: first, less data points are needed for complete interpolation since $|Y|$ is now smaller than p, therefore allowing the algorithm to revise its Hessian approximation more quickly; second, less polynomials need being built and stored and, third, each polynomial is smaller. As one could expect, these are advantageously reflected in the numerical tests reported in Section 4.

We now return to the question of measuring poisedness of the interpolation set in conjunction with Newton fundamental polynomials. When constructing these polynomials, the procedure has to normalize them and to this end performs a division by $|N_i^{[\ell]}(y_i^{[\ell]})|$. While from a theoretical point of view it may be sufficient to require this term to be nonzero, in practice we must verify that

$$|N_i^{[\ell]}(y_i^{[\ell]})| \geq \theta \tag{10}$$

for some parameter $\theta > 0$. The values $|N_i^{[\ell]}(y_i^{[\ell]})|$ are known as *pivots* and θ is called the *pivoting threshold*. If condition (10) is satisfied, then the Newton fundamental polynomials are said to be *well poised*.

This connection between geometry, poisedness and Newton fundamental polynomials allows us to derive suitable mechanisms to control and improve the quality of the geometry of Y. When having to *remove a point* y^- from Y, we choose y^- to be the point associated to the smallest orthogonalization pivot, that is the point $y_i^{[\ell]}$ for which $|N_i^{[\ell]}(y_i^{[\ell]})|$ is minimal, thus making the Newton fundamental polynomials well poised for the subsequent computations. On the other side, a reasonable strategy for *improving the geometry* of the interplation set Y might be to replace a point $y^- = y_i^{[\ell]} \neq x_k$ by another point y^+ such that $|N_i^{[\ell]}(y^+)|$ is larger, for instance

$$y^+ = \arg\max_{y \in \mathcal{B}_k} |N_i^{[\ell]}(y)|$$

provided $|N_i^{[\ell]}(y^+)| \geq 1 + \theta$ since otherwise the geometry was already fine.

Of course these are not the only possible strategies for the management of Y and the interested reader may find a more complete treatment of these issues in Conn, Scheinberg and Toint [10] and Conn, Gould and Toint [8].

For the sake of completeness, we would like to conclude this section by mentioning the two main convergence results allowing us to justify algorithm UDFO. The first result is a bound on the interpolation error; it says that, under appropriate assumptions (see Sauer and Xu [31] or Theorem 9.4.4 in Conn, Gould and Toint [8] for the details), one has

$$|f(x) - m_k(x)| \leq \kappa \Delta_k^3 \max_{Y,x} |N_i^{[\ell]}(y_j^{[\ell+1]})|$$

for all $x \in \mathcal{B}_k$ and some constant κ independent of k. This result is of crucial importance since it guarantees that the model $m_k(x)$ is sufficiently close to $f(x)$ within the trust region. When particularizing this to the case of quadratic models ($d = 2$), we may deduce that algorithm UDFO converges towards critical points and, more precisely,

– if the model $m_k(x)$ is at least *fully linear* (($|Y| \geq n + 1$) and the interpolation set Y is well poised, the algorithm produces a sequence of iterates converging towards a *first-order* critical point,

– if the model is at least *fully quadratic* ($|Y| \geq p$) and Y is well poised, then convergence to *second-order* critical points is ensured;

(see Conn, Scheinberg and Toint [9] for details).

4. Numerical experience

The software we designed for implementing the algorithm presented above is called UDFO. It is coded in Fortran.
We tested UDFO by selecting a number of problems having a banded Hessian in the CUTE collection (see Table I).

Table I. Problem specifications for numerical tests.

Problem name	Semi-bandwidth	Problem name	Semi-bandwidth
`POWER`	1	`MOREBV`	3
`TRIDIA`	2	`SCHMVETT`	3
`BRYDN3LS`	2	`CRAGGLVY`	4
`EXTROSNB`	2	`POWELLSG`	4
		`BDQRTIC`	5

All tests were performed for three possible dimensions, namely $n = 10$, 15 and 20, except for problems `CRAGGLVY` and `POWELLSG`[1].
The storage required for solving these problems involves two arrays: the first one contains the coefficients of the Newton fundamental polynomials defined by (6) while the second one is dedicated to the components of the model (8). As might be expected, a first major advantage of exploiting sparsity is a considerable reduction in the dimension of both these arrays, due to the smaller number of polynomials to be generated. Figure 1 shows how the total length of the arrays dramatically increases with the dimension n and the value of the semi-bandwidth.
We also measured the impact in terms of the number of iterations, CPU time and most of all the number of function evaluations, the latter criterion being especially important in the framework of derivative-free methods. Actually, we compared the computations performed by

[1] Problems `CRAGGLVY` and `POWELLSG` require n to be even and to be a multiple of 4 respectively so we had to take $n = 10$, 16 and 20 for the former and $n = 12$, 16 and 20 for the latter.

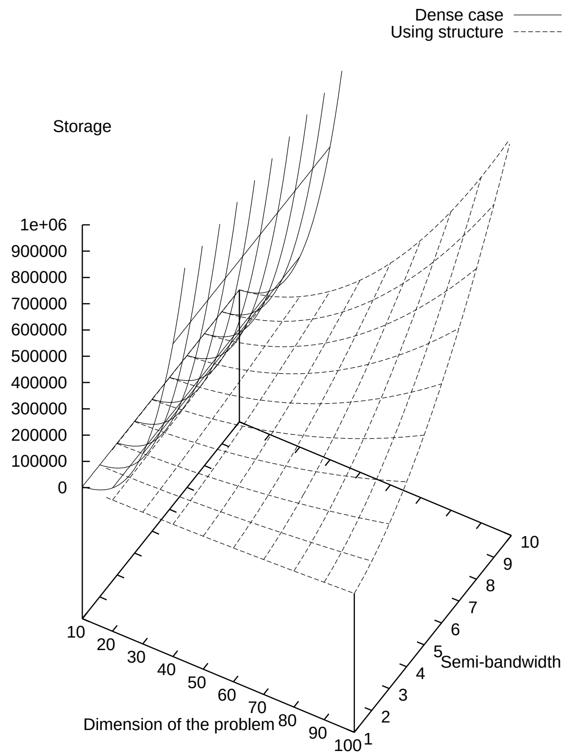

Figure 1. Storage required for both approaches.

UDFO when giving as input a semi-bandwidth value between 1 and 10 independently of the actual one, so as to measure the effect of specifying accurately the sparse structure of the Hessian. The results for $n = 10$ are reported for five problems (having an increasing actual semi-bandwidth value b ranging from 1 to 5) on the barcharts in Figure 2. They show that the three abovementioned measurements globally increase as and when the assumed bandwidth value (given as input to the program) exceeds the real one while underestimating the bandwidth sometimes largely deteriorates the performance of the algorithm.
A second series of barcharts is displayed on Figure 3, showing the evolution of the impact of using structure when the dimension of the

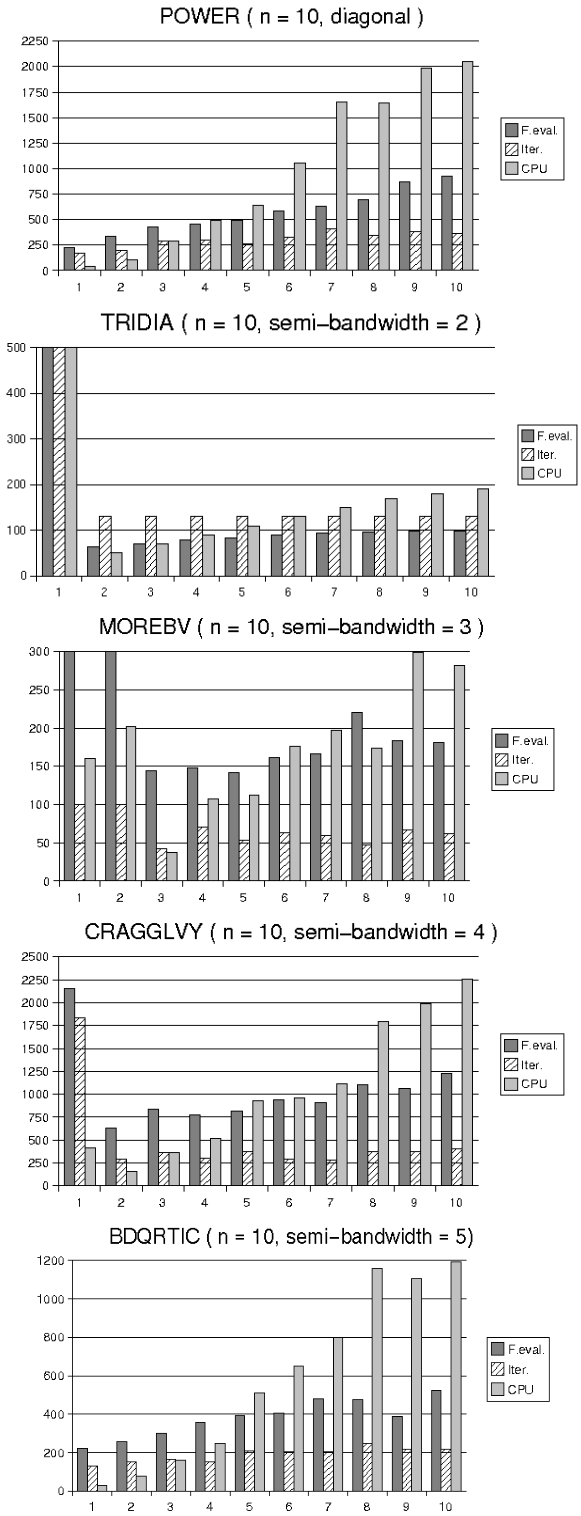

Figure 2. Results for $n = 10$ and problems with an actual semi-bandwidth b comprised between 1 and 5.

problem increases, which is based on the following three ratios:

$$\rho_{\text{eval}} = \frac{\text{Function evaluations using structure}}{\text{Function evaluations in the dense case}},$$

$$\rho_{\text{iter}} = \frac{\text{Number of iterations using structure}}{\text{Number of iterations in the dense case}},$$

$$\rho_{\text{CPU}} = \frac{\text{CPU time using structure}}{\text{CPU time in the dense case}}.$$

The barcharts of Figure 3 show that the main savings are in terms

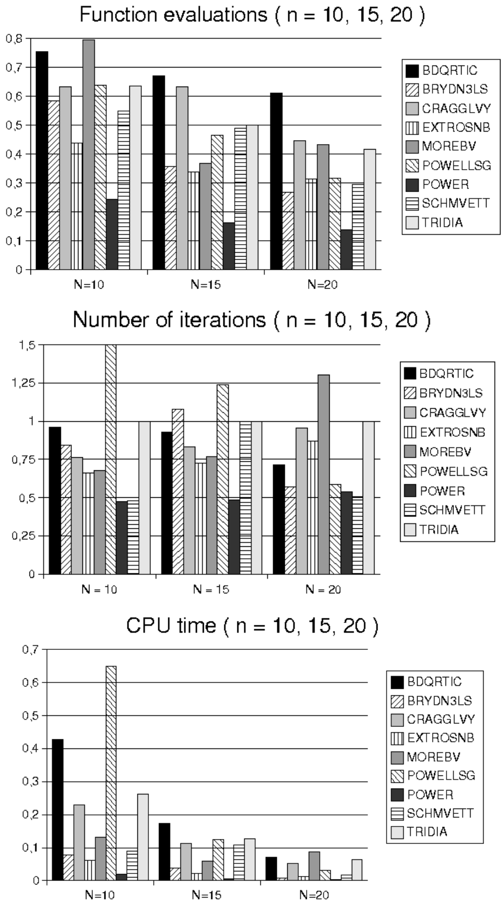

Figure 3. Evolution of the impact of using structure when the dimension of the problem increases ($n = 10, 15$ and 20).

of CPU time and function evaluations and that these savings are more important for larger problems.

5. Conclusion and perspectives

The method discussed in this paper proves to be a very useful technique when sparsity is known. Its main advantages are the much smaller dimension of the working space and the increased speed, as was confirmed by a series of numerical experiments.
We believe that the method can be extended so as to use dominant parts of the Hessian matrix only, *i.e.* introducing "artificial sparsity" and ignoring some of its entries that may be considered as being neglectible. Finding appropriate criteria for selecting such entries and implementing them within the framework described in this paper should yield a further progress from the point of view of storage and speed. Explicit consideration of the partially separable structure of the objective function should also result in important practical savings. Both these extensions are subject of ongoing research.

References

1. N. M. Alexandrov, J. E. Dennis, R. M. Lewis, and V. Torczon. A trust region framework for managing the use of approximation models. *Structural Optimization*, 15(1):16–23, 1998.
2. C. Audet, A. Booker, J. E. Dennis, P. Frank, and D. W. Moore. A surrogate-model-based method for constrained optimization. AIAA paper 2000-4891, AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, September 2000.
3. G. E. P. Box. Evolutionary operation: a method for increasing industrial productivity. *Applied statistics*, 6:81–101, 1957.
4. R. P. Brent. *Algorithms for minimization without derivatives.* Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1973.
5. A. G. Buckley. A derivative-free algorithm for parallel and sequential optimization. presentation at the NATO ASI on Algorithms for Continuous Optimization, Il Ciocco, 1994.
6. F. M. Callier and Ph. L. Toint. Recent results on the accelerating property of an algorithm for function minimization without calculating derivatives. In A. Prekopa, editor, *Survey of Mathematical Programming*, pages 369–376. Publishing House of the Hungarian Academy of Sciences, 1977.
7. I. G. Campey and D. G. Nickols. Simplex minimization. Program specification, Imperial Chemical Industries Ltd, UK, 1961.
8. A. R. Conn, N. I. M. Gould, and Ph. L. Toint. *Trust-Region Methods.* Number 01 in MPS-SIAM Series on Optimization. SIAM, Philadelphia, USA, 2000.
9. A. R. Conn, K. Scheinberg, and Ph. L. Toint. On the convergence of derivative-free methods for unconstrained optimization. In A. Iserles and M. Buhmann, editors, *Approximation Theory and Optimization: Tributes to M. J. D. Powell*, pages 83–108, Cambridge, England, 1997. Cambridge University Press.

10. A. R. Conn, K. Scheinberg, and Ph. L. Toint. Recent progress in unconstrained nonlinear optimization without derivatives. *Mathematical Programming, Series B*, 79(3):397–414, 1997.
11. A. R. Conn, K. Scheinberg, and Ph. L. Toint. A derivative free optimization algorithm in practice. Technical Report TR98/11, Department of Mathematics, University of Namur, Namur, Belgium, 1998.
12. A. R. Conn and Ph. L. Toint. An algorithm using quadratic interpolation for unconstrained derivative free optimization. In G. Di Pillo and F. Gianessi, editors, *Nonlinear Optimization and Applications*, pages 27–47, New York, 1996. Plenum Publishing. Also available as Report 95/6, Dept of Mathematics, FUNDP, Namur, Belgium.
13. J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1983. Reprinted as *Classics in Applied Mathematics 16*, SIAM, Philadelphia, USA, 1996.
14. J. E. Dennis and V. Torczon. Direct search methods on parallel machines. *SIAM Journal on Optimization*, 1(4):448–474, 1991.
15. J. E. Dennis and V. Torczon. Managing approximation models in optimization. In N. M. Alexandrov and M. Y. Hussaini, editors, *Multidisciplinary Design Optimization*, pages 330–347, Philadelphia, USA, 1997. SIAM.
16. L. C. W. Dixon. *Nonlinear Optimisation.* The English Universities Press Ltd, London, 1972.
17. P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization.* Academic Press, London, 1981.
18. R. Hooke and T. A. Jeeves. Direct search solution of numerical and statistical problems. *Journal of the ACM*, 8:212–229, 1961.
19. S. Lucidi and M. Sciandrone. A coordinate descent method without derivatives. Technical Report 10-95 (in preparation), University of Rome "La Sapienza", Rome, 1995.
20. M. Marazzi and J. Nocedal. Wedge trust region methods for derivative free optimization. Technical Report OTC 2000/10, Optimization Technology Center, Argonnne National Laboratory, Argonne, Illinois, USA, 2000.
21. J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.
22. J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer Verlag, Heidelberg, Berlin, New York, 1999.
23. M. J. D. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Computer Journal*, 17:155–162, 1964.
24. M. J. D. Powell. A method for minimizing a sum of squares of nonlinear functions without calculating derivatives. *Computer Journal*, 7:303–307, 1965.
25. M. J. D. Powell. A Fortran subroutine for unconstrained minimization requiring first derivatives of the objective function. Technical Report R-6469, AERE Harwell Laboratory, Harwell, Oxfordshire, England, 1970.
26. M. J. D. Powell. A new algorithm for unconstrained optimization. In J. B. Rosen, O. L. Mangasarian, and K. Ritter, editors, *Nonlinear Programming*, pages 31–65, London, 1970. Academic Press.
27. M. J. D. Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation. In S. Gomez and J. P. Hennart, editors, *Advances in Optimization and Numerical Analysis, Proceedings of the Sixth Workshop on Optimization and Numerical Analysis, Oaxaca, Mexico*,

volume 275, pages 51–67, Dordrecht, The Netherlands, 1994. Kluwer Academic Publishers.
28. M. J. D. Powell. Trust region methods that employ quadratic interpolation to the objective function. Presentation at the 5th SIAM Conference on Optimization, Victoria, 1996.
29. M. J. D. Powell. On the Lagrange functions of quadratic models defined by interpolation. Technical Report NA10, Department of Applied Mathematics and Theoretical Physics, Cambridge University, Cambridge, England, 2000.
30. M. J. D. Powell. UOBYQA: unconstrained optimization by quadratic interpolation. Technical Report NA14, Department of Applied Mathematics and Theoretical Physics, Cambridge University, Cambridge, England, 2000.
31. Th. Sauer and X. Yuan. On multivariate Lagrange interpolation. *Mathematics of Computation*, 64:1147–1170, 1995.
32. W. Spendley, G. R. Hext, and F. R. Himsworth. Sequential application of simplex designs in optimization and evolutionary operation. *Technometrics*, 4, 1962.
33. G. W. Stewart. A modification of Davidon's minimization method to accept difference approximations of derivatives. *Journal of the ACM*, 14, 1967.
34. V. Torczon. On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, 7(1):1–25, 1997.
35. D. Winfield. *Function and functional optimization by interpolation in data tables.* PhD thesis, Harvard University, Cambridge, USA, 1969.
36. D. Winfield. Function minimization by interpolation in a data table. *Journal of the Institute of Mathematics and its Applications*, 12:339–347, 1973.
37. M. H. Wright. Direct search methods: once scorned, now respectable. In D. F. Griffiths and G. A. Watson, editors, *Proceedings of the 1995 Dundee Biennal Conference in Numerical Analysis*, Reading, Massachusetts, USA, 1996. Addison-Wesley Publishing Company.