Exploiting problem structure in pattern-search methods for unconstrained optimization

by C. Price[1] and Ph. L. Toint[2]

Report 04/01 12th January 2004

[1] Department of Mathematics & Statistics,
University of Canterbury,
Private Bag 4800, Christchurch, New Zealand.
Email: c.price@maths.canterbury.ac.nz

[2] Department of Mathematics,
University of Namur,
61, rue de Bruxelles, B-5000 Namur, Belgium,
Email: philippe.toint@fundp.ac.be

Exploiting problem structure in pattern-search methods for unconstrained optimization

C. J. Price and Ph. L. Toint

12th January 2004

Abstract

A direct search method for unconstrained optimization is described. The method makes use of any partial separability structure that the objective function may have. The method uses successively finer nested grids, and minimizes the objective function over each grid in turn. All grids are aligned with the coordinate directions which allows the partial separability structure of the objective function to be exploited. This has two advantages: it reduces the work needed to calculate function values at the points required; and it provides function values at other points as a free by-product. Numerical results show that using partial separability can dramatically reduce the number of function evaluations needed to minimize a function, in some cases allowing problems with thousands of variables to be solved.

Keywords: partial separability, pattern search, derivative-free, numerical results.

1 Introduction

In this paper we look at the use of partial separability in pattern search methods for unconstrained optimization. Pattern search methods have been used for many years to solve unconstrained optimization problems where derivative information was neither available, nor able to be reliably estimated. Perhaps the most famous of the older pattern search methods are those by Hooke and Jeeves [8], and by Nelder and Mead [10]. More recent works in unconstrained pattern search algorithms include the multi-directional search of Dennis and Torczon [6], and Torczon's work on generalized pattern search methods [11]. Other similar recent works include the grid search framework of Coope and Price [4]. Partial separability was first looked at by Griewank and Toint in [7], and more recently applied to derivative free methods by Colson and Toint [1, 2, 3]. Herein we wish to apply the ideas behind partial separability to the grid based pattern search methods of [4].

The basic form of the unconstrained optimization problem is to find a local minimum of the function $f(x)$ where $x \in R^n$. We make the assumption that f is continuously differentiable over the region of interest. This means that second derivatives may not exist, and so stationary points are accepted as solutions.

We are interested in functions which are partially separable. That is to say f is of the form

$$f(x) = \sum_{i=1}^{q} f_i(x). \tag{1}$$

We are mainly interested in the case when each element f_i of the objective function depends only on a small proportion of the elements of the vector of decision variables x. It is assumed that each f_i is available individually, along with a list of the decision variables on which f_i depends. These assumptions allow us to exploit partial separability to significantly reduce the number of function evaluations needed to minimize f.

The extreme case of partial separability is when $q = n$, and each f_i depends only on the variable x_i. In this case one can clearly minimize over each dimension independently because there are no interactions between changes in different variables. Partial separability is used in a similar way, however there are interactions between some pairs of variables. Two variables are said to interact if at least one element f_i depends on both variables. Sets of non-interacting variables are useful because the change in f from altering all variables in the set is just the sum of the changes in f arising from altering each variable in the set individually.

Exploiting partial separability is advantageous in two separate ways. It can reduce the computational cost of each function value sought, and it can provide function values at related points as a by-product. For example, consider a totally separable function

$$f(x) = \sum_{i=1}^{n} f_i(x_i)$$

with the value of each element f_i known at a point x. We wish to explore f in the vicinity of x by calculating f at each point of the form $x + hv$, where h is a positive constant and v ranges over the set $\{e_1, \ldots, e_n, -e_1, \ldots, -e_n\}$. Here e_i is the i^{th} column of the identity matrix. Calculating f at each point requires the evaluation of one f_i, and so the total cost of the $2n$ function values at the points $x \pm he_i$, $i = 1, \ldots, n$ is *two* function evaluations of f. However, any set of changes to f from steps of the form $\pm he_i$ are independent of one another provided no two such steps alter the same variable. Hence as a by-product we obtain the function values at all other points of the form $x + h\sum_{i=1}^{n} \eta_i e_i$ where each η_i is either -1, 0, or 1. Thus we obtain f at $3^n - 1$ new points for the total cost of *two* function evaluations.

When f is only partially separable the gains are not as dramatic as the totally separable case, but they still can be very substantial. For example, let f be of the form

$$f(x) = f_1(x_n, x_1, x_2) + \sum_{i=2}^{n-1} f_i(x_{i-1}, x_i, x_{i+1}) + f_n(x_{n-1}, x_n, x_1). \tag{2}$$

Then, given each $f_i(x)$, calculating f at $x \pm he_i$, $i = 1, \ldots, n$ costs six function evaluations. When n is divisible by 3, an inductive argument shows that the number of other function values obtained as by-products is $7^{(n/3)} - 2n - 1$.

A final example shows that the two advantages of partial separability are distinct from one another. Let $f = f_3(x_1, x_2) + f_2(x_1, x_3) + f_1(x_2, x_3)$. Calculating f at all points of the form $x \pm he_i$, $i = 1, \ldots, 3$ costs four function evaluations, given that each $f_i(x)$, $i = 1, \ldots, 3$ is known in advance. No further function values are generated as by-products.

2 A positive basis algorithm for partially separable problems

A pattern search method conforming to framework A in [4] is constructed. The overall form of the algorithm is one of searching over a sequence of successively finer grids. The sequence of grids is nested. That is to say each grid is a subset of its predecessor. Each grid is aligned with the coordinate directions. This allows each decision variable to be altered individually. This property is needed to exploit partial separability. The m^{th} grid $\mathcal{G}^{(m)}$ is of the form

$$\mathcal{G}^{(m)} = \left\{ x^{(1)} + 2^{1-m} h^{(1)} \sum_{i=1}^{n} \eta_i e_i : \eta_i \in Z \right\}.$$

Here $x^{(1)}$ is the initial point, $h^{(1)}$ is the initial step size, Z is the set of integers, and e_i is the i^{th} column of the identity matrix. Loosely speaking, each grid is obtained by taking its predecessor and 'filling in' all of the points half way between each pair of existing grid points.

For each grid a finite subsequence of iterates is generated, where each iterate lies on that grid. The subsequence of function values corresponding to this subsequence of iterates is required to be strictly decreasing. The finite subsequence terminates when no lower grid point can be found around the current iterate. Before terminating, the algorithm must examine all grid points which differ from the current iterate by some member of a positive basis [5]. This final iterate is termed a grid local minimizer. Once a grid local minimizer is located algorithm proceeds onto the next grid in the sequence.

The same positive basis $\mathcal{V}_+ = \{e_1, \ldots, e_{n+r}\}$ (up to sign changes and scaling) is used for all iterations. Here $e_1, \ldots, e_n$ are the columns of the identity matrix, and $e_{n+1}, \ldots, e_{n+r}$ are defined as follows so that the structure of $\mathcal{V}_+$ matches the partial separability structure of f. Before the first iteration the algorithm groups the decision variables x_j into subsets specified by the sets of indices $S_1, \ldots, S_r$. For each such subset, all variables indexed by S_p appear in exactly the same elements f_i. This means we can speak of sets of non interacting subspaces because all variables in the same subspace are equivalent in terms of which variables they interact with. The r members $e_{n+1}, \ldots, e_{n+r}$ of $\mathcal{V}_+$ are defined as

$$e_{n+p} = -\sum_{j \in S_p} e_j \quad \text{for} \quad p = 1, \ldots, r.$$

This positive basis $\mathcal{V}_+$ is the union of r component positive bases, each of which is a positive basis for the subspace $\text{span}\{x_j : j \in S_p\}$ for some $p \in \{1, \ldots, r\}$. For convenience each $e_1, \ldots, e_n$ is given a separate step size $h_j^{(k)}$, $j = 1, \ldots, n$, where k is the iteration number. The ratio between the smallest and largest magnitudes of the step sizes at any given iteration is bounded. The first n members $v_1, \ldots, v_n$ of a scaled positive basis are then chosen as $v_j = h_j^{(k)} e_j$, $j = 1, \ldots, n$. ¿From time to time some or all of these first n directions may be reversed. This is done by changing the signs of the relevant step sizes. The remaining v_j, $j > n$ are defined automatically via

$$v_{n+p}^{(k)} = - \sum_{j \in S_p} h_j^{(k)} e_j \quad \text{for} \quad p = 1, \ldots, r.$$

The main purpose of the algorithm is to find a grid local minimizer on each of an infinite sequence of progressively finer grids. The convergence theory [4] applies to the subsequence of grid local minimizers. Provided the sequence of iterates is bounded it is shown that the sequence of grid local minimizers is infinite, and hence has cluster points. It is shown that the cluster points of the grid local minimizers are stationary points of the objective function f.

THE ALGORITHM

1. Initialize and analyse the subspace structure: Set $k = m = 1$. Set $h_j^{(1)} = 1$ for all j. Choose the initial point $x^{(1)}$ and the stopping tolerance $\tau_{\text{stop}} = 10^{-5}$.

2. **repeat**

   (a) For each subspace index set S_p, calculate the best value of the increment

   $$\sum_{i=1}^{q} f_i(x^{(k)} + v_j^{(k)}) - f(x^{(k)}) \quad \text{over } j \in S_p \cup \{n+p\}.$$

   Ignore all non-negative increments.

   (b) choose the best increments from a set of non-interacting subspaces.

   (c) Increase the step sizes $h_j^{(k)}$, and reverse the step directions (if relevant).

   (d) Choose $x^{(k+1)}$ as the lowest known point on the grid $\mathcal{G}^{(m)}$, and increment k.

   until a grid local minimum is located.

3. If $|h_j^{(k)}| < \tau_{\text{stop}}$ for all $j = 1, \ldots, n$ then stop.

4. Halve all step sizes which are maximal in magnitude. Increment m and go to step 2.

The algorithm halts at the first grid local minimizer for which all step sizes have magnitudes less than the positive stopping tolerance τ_{stop}.

2.1 Analysing the Subspace Structure

This subsection describes how the partially separable structure of the problem is used. The algorithm operates under the assumption that it is provided with a list of which variables occur in each element f_i.

The structure analysis in the first step of the algorithm proceeds broadly as follows. We first invert the user supplied lists of variables appearing in each element to obtain lists of elements in which each variable appears. These new lists are then sorted into order of increasing length, and grouped into blocks of lists having the same length. Each such block is then sorted into lexicographical order. Identical lists are identified as they indicate variables which appear in exactly the same set of elements. Each set of variables which appears in the same elements spans a subspace in which a component of the positive basis is formed. Each of these components is a positive basis over its associated subspace. The union of these positive bases is a positive basis over R^n.

Lists of variables in each subspace are constructed, and finally, these lists are inverted to produce lists of subspaces in which each variable appears.

For example, if there are five elements which depend on 13 decision variables as follows:

f_1	depends on	x_1	x_2	x_3			
f_2	..	x_2	x_3	x_4	x_5	x_6	x_7
f_3	..	x_7	x_8	x_9	x_{11}		
f_4	..	x_{11}	x_{12}	x_{13}			
f_5	..	x_5	x_6	x_{10}			

Then (after sorting) the list of elements for each variable is as follows:

x_1	appears in	f_1	
x_4	..	f_2	
x_8	..	f_3	
x_9	..	f_3	
x_{12}	..	f_4	
x_{13}	..	f_4	
x_{10}	..	f_5	
x_2	..	f_1	f_2
x_3	..	f_1	f_2
x_7	..	f_2	f_3
x_5	..	f_2	f_5
x_6	..	f_2	f_5
x_{11}	..	f_3	f_4

There are nine subspaces, respectively spanned the following sets of variables: $\{x_1\}$; $\{x_4\}$; $\{x_8, x_9\}$; $\{x_{12}, x_{13}\}$; $\{x_{10}\}$; $\{x_2, x_3\}$; $\{x_7\}$; $\{x_5, x_6\}$; and $\{x_{11}\}$. Each of the one dimensional subspaces has a positive basis component of the form $\{+e_i, -e_i\}$. Each of the two dimensional subspaces has a positive basis component of the form $\{+e_i, +e_j, -(e_i + e_j)\}$.

2.2 Selecting the Best Increments

Step 2(b) is implemented by examining the subspaces sequentially. A list of subspaces whose best increments *reduce* the function value is formed. These subspaces are candidates for the set of non interacting subspaces, where two subspaces are non interacting if and only if no element f_i contains variables from both subspaces. For instance, in the previous example the subspaces $\{x_2, x_3\}$ and $\{x_5, x_6\}$ are interacting because they both alter f_2. In contrast $\{x_2, x_3\}$ and $\{x_{11}\}$ are not interacting. Subspaces are added to the non interacting set one at a time. After each such subspace is added all subspaces which interact with it are purged from the list of candidates. The process continues until the list of candidates is empty.

The standard form of the algorithm always selected the first subspace on the list of candidates whose best increment reduced f. A variation (called the greedy variation) was also considered. This variation looked at all subspaces on the list of candidates, and selected the subspace with the increment giving the greatest reduction in f.

2.3 Adjusting the Step Sizes

The step sizes h_j corresponding to the first n members of the positive basis $\mathcal{V}_+$ are altered as follows. After each grid local minimizer all h_j which are maximal in magnitude are reduced by a factor of 2.

The algorithm also increases the step sizes from time to time. Step sizes are increased on every third iteration, beginning again on the third iteration after each grid local minimum is found. In an iteration in which step sizes are increased, only the h_j corresponding v_j's along which steps have been taken *that iteration* are increased. These h_j are doubled. An upper bound on the magnitude of each h_j is imposed, but the sign of each h_j is not altered. Specifically

$$h_j \longleftarrow \operatorname{sign}(h_j) \min\left\{ |h_j|, 128 \min_{i=1,\ldots,n} |h_i| \right\}.$$

This bound is a requirement of the convergence theory.

A variation considered in this papers is to reverse the search steps at each iteration. This is most easily done by changing the signs of all h_j, except those h_j which are increased in the same iteration. Component positive bases for one dimensional subspaces are of the form $\{h_j e_j, -h_j e_j\}$. Changing the sign of h_j simply reverses the order of the directions, leaving the component positive basis unchanged. Hence reversing is only applied to directions which belong to a component positive basis of dimension greater than one.

3 Numerical Results

A number of test problems were chosen from the CUTE test set, some of which also appear in [9]. The function Nzf1 was a made-up function with the subspace structure given in the example in Section 2.1. Its precise form is:

$$
\begin{aligned}
f \;=\; & \left(3x_1 - 60 + \frac{(x_2 - x_3)^2}{10}\right)^2 \\
& + \left(x_2^2 + x_3^2 + x_4^2(1 + x_4^2) + x_7 + \frac{x_6}{1 + x_5^2 + \sin(x_5/1000)}\right)^2 \\
& + \left(x_7 + x_8 - x_9^2 + x_{11}\right)^2 \\
& + \left(\ln(1 + x_{11}^2) + x_{12} - 5x_{13} + 20\right)^2 \\
& + (x_5 + x_6 + x_5 x_6 + 10x_{10} - 50)^2
\end{aligned}
$$

Other test functions have a variety of subspace structures. The Boundary Value and Broyden 3D functions have a tridiagonal structure, which is the same as the function in equation (2) except the first and last elements do not depend on x_n and x_1 respectively. The Freudenstein-Roth, Tridiagonal, and Modified Beale functions have a similar bi-diagonal structure. The Broyden banded function has a banded structure with a bandwidth of seven. The Arrowhead function is somewhat different. It has nine elements of the form $f_i(x_i, x_{10})$, for $i = 1, \ldots, 9$.

The extended Woods and extended Rosenbrock functions are formed by taking several copies of the standard Woods and Rosenbrocks functions and making the sets of variables which appear in each copy disjoint from one another. There is further structure inside each copy of the Woods function. The extended Variably Dimensioned, Trigonometric, Brown almost linear, and Linear full rank functions (numbers 25, 26, 27 and 32 in [9]) were formed by overlapping five six dimensional copies of the relevant function in [9], where the last two variables of each copy are also the first two variables in the following copy. The initial point of the Linear full rank function was divided by π to prevent the algorithm from stepping *exactly* to the solution in a finite number of steps.

The Lminsurf and Nlmsurf functions are formed by taking a square two dimensional mesh which is 5 squares by 5 squares. Each of the 36 intersections of the mesh corresponds to a variable, with the variables corresponding to the internal 16 intersections being the unknowns. The mesh has 25 small squares. For each of these squares there is one element which is a function of the four variables corresponding to the four corners of that square.

On all test problems the computational effort needed to calculate f for all steps in the positive basis is reduced by partial separability. There are also extra function values available as by-products on all of these problems, although for the Arrowhead, Tridiagonal, and Broyden banded functions this is dependent on which variable(s) are altered. On all other problems partial separability always yields a large number of by-product function values.

In order to assess the gains made by exploiting partial separability, one must also solve the test problems without using partial separability. This can be done easily in the existing framework by setting $q = 1$ and $f_1 \equiv f$. When partial separability is used the number of times each element f_i is calculated is known,

Function	n	q	r	final f
Arrowhead	10	9	10	5.9 e–16
Boundary value	10	10	10	2.2e–7
Broyden 3D	10	10	10	2.1e–9
Broyden banded	10	10	10	1.2e–9
Ext. Rosenbrock	10	5	5	4.073e–6
Freudenstein-Roth	10	9	10	1014
Lminsurf	16	25	16	9
Modified Beale	6	5	6	1.589e–5
Nlmsurf	16	25	16	35.4
Nzf1	13	5	9	4.4e–12
Tridiagonal	3	3	3	2.4e–10
Ext. Woods	16	20	16	3.8e–8
Ext. Trig.	22	5	9	2.562e–3
Ext. Var. Dim.	22	5	9	8.3e–8
Ext. Brown	22	5	9	3.0e–8
Ext. Linear	22	5	9	1.7e–10

Table 1: Basic information about the test functions. The final function value is listed for the algorithm with both the greedy and reversing options together.

and some elements may be calculated more than others. For comparison purposes this information must be expressed as an equivalent number of function evaluations. This is done by assuming each element f_i is equally expensive. Therefore q evaluations of an element f_i require the same computational work as *one* function evaluation.

The numerical results are presented in Tables 1 to 6. Table 1 lists the dimension (n), number of elements (q), and number of subspaces (r) of each test problem. In addition the final column of this table lists the estimate of the optimal function value found by the algorithm using the greedy and reversing variations.

Table 2 lists the number of function evaluations required to minimize each test function for a variety of algorithm variations. Results generated with the reverse and greedy variations are identified with the letters 'R' and 'G' respectively in the second row of the table. Results generated without exploiting partial separability are identified by the letter 'O' (for 'one element'). These results show that exploiting the partially separable nature of these functions can greatly reduce the amount of work required to locate a minimizer. The greedy variation yields a small gain when partial separability is used. It is not relevant in the one element case. The reversing variation is very effective in the one element case. It increases the number of directions the algorithm can look in over two iterations, thereby giving a greater chance of finding a good step.

In many cases identical results were obtained with and without reversing when partial separability was used. In each such case the function had all subspaces of dimension one, and each component positive basis was of the form $\{h_j e_j, -h_j e_j\}$ for some $j \in \{1, \ldots, n\}$. Reversing any such pair of directions

Function	n	Number of function evaluations					
		G	GR	—	R	O	OR
Arrowhead	10	105	105	129	129	584	386
Boundary value	10	16994	16994	16553	16553	70423	84965
Broyden 3D	10	349	349	378	378	5270	1706
Broyden banded	10	1463	1463	2338	2338	21187	2091
Ext. Rosenbrock	10	16939	3268	16939	3268	175572	69653
Freudenstein-Roth	10	299	299	330	330	3290	4555
Lminsurf	16	483	483	536	536	22033	4253
Modified Beale	6	15502	15502	14219	14219	52137	42414
Nlmsurf	16	1928	1928	1752	1752	57716	17970
Nzf1	13	257	163	233	373	112001	1247
Tridiagonal	3	532	532	620	620	1017	1441
Ext. Woods	16	448	448	275	275	60181	10065
Ext. Trig.	22	3843	885	4005	941	25715	6441
Ext. Var. Dim.	22	23206	4566	23899	4984	>6e+5	28107
Ext. Brown	22	11974	8035	23699	7072	438	6096
Ext. Linear	22	1233	1168	1249	1183	53936	8350

Table 2: A comparison of several variants of the algorithm. The letters 'G' 'R' and 'O' respectively indicate when the greedy, reversing, and one element options were used.

by changing the sign of h_j generates no new search directions, and so achieves nothing.

Six problems have subspaces of dimension greater than one: extended Rosenbrock, Nzf1, extended Trigonometric, extended Variably Dimensioned, extended Brown almost linear, and extended Linear full rank. The extended Rosenbrock problem has only two dimensional subspaces. Reversing increases the number of possible search directions in each subspace from three to eight. The reversing strategy reduced the computational effort needed to minimize the extended Rosenbrock function by more than a factor of five. The Nzf1 function has five one dimensional and four two dimensional subspaces. Reversing gave a significant improvement when the greedy variation was used, but was counterproductive when the greedy variation was not used. The remaining four functions each have two four dimensional subspaces and seven two dimensional subspaces. Reversing produced some improvement on the extended Linear full rank, and dramatic improvements on the other three. For the extended Linear full rank problem, the direction of the vector from the initial point to the solution is $-(e_1 + \cdots + e_{22})$. The function is a convex quadratic, so the initial set of search directions contain all possible search directions which most nearly point from the initial point to the solution. Hence reversing was comparatively ineffective (although still advantageous) on this problem.

Table 3 lists the number of function evaluations needed to minimize the functions Lminsurf and Broyden 3D for various dimensions. Problem Lminsurf stems from a two dimensional surface fitting problem, which means that the

n	Lminsurf		Broyden 3D	
	GR	OR	GR	OR
9	215	501	343	1241
16	483	3724	334	3605
25	484	10557	364	8087
36	890	19796	379	16503
49	1002	—	363	24531
64	1149	—	362	—
81	1413	—	389	—
100	1634	—	362	—
121	2045	—	362	—
144	2120	—	392	—
169	2689	—	361	—
196	3233	—	361	—
5625	79511	—	535	—

Table 3: Number of function evaluations required to minimize the Lminsurf and Broyden 3D functions in various dimensions.

problem dimension n must be a square. Results for two versions of the algorithm are listed for each problem. These two versions are respectively the best version listed in Table 2 which exploits partial separability, and the best version which does not. The results show that the version using partial separability is clearly superior, and the margin of superiority increases with dimension. With partial separability the number of function evaluations needed to minimize Lminsurf is approximately linear in dimension, and with Broyden 3D it is almost constant. Without partial separability the number of function evaluations required is approximately quadratic in dimension for Lminsurf, and it is slightly less so for Broyden 3D.

The final row of Table 3 lists results for $n = 5625$, which is the largest dimension in which Lminsurf could be solved in a reasonable amount of time (about a week) on a four precessor 900 MHz Daktari with, on average, about two other users present. The computations were performed in MATLAB without multi-threading (i.e. only one processor was used). Broyden 3D was solved for $n = 5625$ in an afternoon. Four other problems were also solved in either 5625 or 5626 dimensions. Details of these solutions are listed in Table 4.

Tables 5 and 6 examine the rate of convergence of the algorithm for a small number of test functions. These results were generated using partial separability, and the reverse and greedy variations. Each table lists the iteration number ('itn'), number of function evaluations ('nf'), and current function value (f) when the first grid local minimizer is found at which the maximum ('max($|h|$)') of the absolute values of the step sizes is less than the value listed in the left most column of each table. The third row (labelled 'start') lists function value at the initial point. The two functions in Table 5 are moderately ill-conditioned, whereas those in Table 6 are well conditioned. The results for the extended Rosenbrock function show that the algorithm made good progress initially, then

Problem	n	itn	nf	f	time
Broyden 3D	5625	19737	535	3.4e–8	4.4
Broyden banded	5625	2054	2077	1.4e–7	3.0
Lminsurf	5625	87911	79511	9	212
Ext. Variably Dim.	5626	3252	5130	4.8e–6	4.7
Ext. Trigonometric	5626	4337	949	0.8874	1.1
Freudenstein-Roth	5625	240	221	6.8e+5	1.9

Table 4: Solution details for several high dimensional problems with the greedy and reversing options selected. The solution times are elapsed time in hours, with other users present.

$\max(\|h\|)$	Ext. Rosenbrock			Ext. Woods		
	itn	nf	f	itn	nf	f
start	0	1	2319	0	1	7.7e+4
1	7	19	1.02	8	23	168
1e–1	11	31	0.21	27	72	7.42
1e–2	15	43	5e–2	62	165	0.41
1e–3	31	91	4e–2	122	325	6e–3
1e–4	581	1741	5e–4	146	395	2e–6
1e–5	1090	3268	4e–6	164	448	4e–8

Table 5: Progress rates for the extended Rosenbrocks and Woods functions using the reversing and greedy options.

$\max(\|h\|)$	Lminsurf			Nzf1		
	itn	nf	f	itn	nf	f
start	0	1	21.38	0	1	4931
1	38	152	9.017	16	45	1.15
1e–1	61	229	9.000	26	69	5e–3
1e–2	78	289	9.000	38	102	4e–5
1e–3	96	356	9.000	43	120	5e–8
1e–4	115	432	9.000	49	141	1e–9
1e–5	127	483	9.000	55	163	4e–12

Table 6: Progress rates for the Lminsurf and Nzf1 functions using the reversing and greedy options.

almost no progress until $\max(|h|)$ fell below 10^{-3}, after which slow but steady progress was made. Progress on the extended Woods function was much steadier, with very good progress initially, and after $\max(|h|)$ first fell below 10^{-3}. In contrast, for Lminsurf the algorithm had obtained the optimal function value by the time $\max(|h|)$ first fell below 0.1. On Nzf1 the algorithm made good steady progress at all times. Collectively these results show that the rate of reduction of $\max(|h|)$ is strongly problem dependent, with ill-conditioning being a significant factor.

4 Conclusion

A grid based pattern search algorithm which exploits partial separability has been described and implemented. Numerical results show that the use of partial separability dramatically reduces the computational effort required to minimize partially separable functions. This allowed problems with thousands of variables to be solved in a reasonable amount of time. Two variations of the algorithm were examined: use of a greedy approach in choosing which steps to take in an iteration, and reversing the step directions. The former is only applicable to partially separable functions, and was shown to marginally improve the algorithm. It is possible to improve on the greedy strategy by finding the combination of non-interacting steps which yields the greatest reduction in the objective function, but this requires solving a combinatorial optimization problem every iteration. The relatively modest gains obtained by using the greedy strategy suggest that the gains obtained by such an approach would be marginal. The strategy of reversing the step directions was very effective except when all subspaces were of dimension one. On these problems the reversing strategy made no difference as it did not alter the set of search directions.

Pattern search algorithms have limited efficiency, but are very robust and adaptable. Using partial separability makes them applicable to large problems, including problems arising from discretization.

Acknowledgements Ph. L. Toint would like to express his gratitude to the University of Canterbury Erskine Fellowship for its support during a visit to New Zealand in the spring 2003, which created the opportunity for the research presented here. Thanks are also due to Ian Coope and Bob Broughton for their interest and friendly discussions.

References

[1] B. Colson and Ph. L. Toint, *Exploiting Band Structure in Unconstrained Optimization Without Derivatives*, Optimization and Engineering, vol. 2, pp. 349–412, 2001.

[2] B. Colson and Ph. L. Toint, A Derivative-free Algorithm for Sparse Unconstrained Optimization Problems, in Siddiqi, A. H. and M. Kočvara (Eds), Trends in Industrial and Applied Mathematics, Kluwer, pp. 131–149, 2002.

[3] B. Colson and Ph. L. Toint, *Optimizing Partially Separable Functions Without Derivatives*, Technical report 03/20, Department of Mathematics, University of Namur, 2003.

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

[5] Davis, C., *Theory of positive linear dependence*, American Journal of Mathematics, Vol. 76 (1954), pp 733-746.

[6] Dennis, J. E., and V. Torczon, *Direct search methods on parallel machines*, SIAM J. Opt vol. 1 (1991), pp 448–474.

[7] Griewank, A. and Ph. L. Toint, *On the unconstrained optimization of partially separable functions*, in M. J. D. Powell (Ed), Nonlinear Optimization 1981, pp. 301–312, 1982.

[8] Hooke, R., and T. A. Jeeves, *Direct search solution of numerical and statistical problems*, J. Assoc. Comput. Mach. vol. 8 (1961), pp 212–219.

[9] Moré, J. J., B. S. Garbow, and K. E. Hillstrom, *Testing unconstrained optimization software*, ACM Trans. Math. Software, Vol. 7 (1981), pp 17–41.

[10] Nelder, J. A. and R. Mead, *A simplex method for function minimization*, Computer J. vol. 7 (1965), pp 308–313.

[11] Torczon, V., *On the convergence of pattern search algorithms*, SIAM J. Opt. vol. 7 (1997), pp 1–25.