Solution of Large Scale Partially Separable Unconstrained Optimization Problems Using Element-by-Element Preconditioners

M. J. Daydé[i], J. P. Décamps[i], J.-Y. L'Excellent[i], N. I. M. Gould[ii]

1 INTRODUCTION

We study the numerical solution of large scale unconstrained optimization problems. We consider the minimization of a partially separable objective function $f(\boldsymbol{x})$. The function f is said to be *partially separable* (see [1]) if

$$f(\boldsymbol{x}) = \sum_{i=1}^{p} f_i(\boldsymbol{x}), \tag{1}$$

where each *element* function f_i has a large invariant subspace. Typically, this occurs when $f_i(\boldsymbol{x})$ is only a function of a small subset of the variables $\boldsymbol{x}$, but may, of course, happen for other reasons (see, for example, [2]). The decomposition (1) is extremely general. Indeed, any sufficiently differentiable function with a sparse Hessian matrix may be expressed in this form [1]. In this paper, we shall be concerned with those partially separable functions for which

$$f(\boldsymbol{x}) = \sum_{i=1}^{p} f_i(\boldsymbol{x}^i), \tag{2}$$

where each set of *local* variables, $\boldsymbol{x}^i \in \Re_i^n$, is a subset of the *global* variables, $\boldsymbol{x} \in \Re^n$, and $n_i \ll n$. In unconstrained optimization, one of the key computation is to obtain an (approximate) solution, d, to the Newton equations

$$\nabla_{xx} f(\boldsymbol{x}) d = -\nabla_x f(\boldsymbol{x}). \tag{3}$$

[i]ENSEEIHT-IRIT, 2 rue Camichel, 31071 Toulouse CEDEX, France
[ii]Rutherford Appleton Laboratory, Oxfordshire, OX11 0QX, England

If f has the form (2), these equations become

$$\left(\sum_{i=1}^{p} \nabla_{xx} f_i(\boldsymbol{x}^i)\right) d = -\sum_{i=1}^{p} \nabla_x f_i(\boldsymbol{x}^i). \tag{4}$$

where the element Hessians, $\nabla_{xx} f_i(\boldsymbol{x}^i)$ are extremely sparse matrices. Note that similar linear systems arise when solving constrained optimization problems using penalty functions or augmented Lagrangian methods (see, for example, [2], [3] and [4]). It has been demonstrated [5] that Element-by-Element preconditioners are extremely effective for the solution of such systems of linear equations, particularly if a preprocessing step in which element Hessians are amalgamed is performed. This has the effect of decreasing overlap between elements, and of improving vectorization and parallelization. PAREBE [6] is a software package which provides a set of routines to solve unassembled symmetric positive linear systems by exploiting the structure — arising from partial separability in our case — of a matrix. The package uses the conjugate gradient algorithm [7] with several preconditioning techniques. It also includes a preprocessing technique, known as element merging or amalgamation, to improve the unassembled structure of the matrix, and colouring algorithms for the parallelization of the matrix-vector products and the preconditioning step. The unconstrained optimization algorithm we describe in this paper is based on the truncated Newton method (see, for instance, [8] and [9]) and makes use of PAREBE to determine the search direction. We report on experiments on a range of test problems from the CUTE collection (see [10]) expressed in SIF format. Results are provided on two computers, the Alliant FX/80, a parallel vector machine, and the HP 715/64 a RISC workstation.

In Section 2, we describe the truncated Newton algorithm used, then in Sections 3 and 4 we describe briefly the preconditioners used and our preprocessing procedure. In Section 5, we compare the efficiency of the preconditioners considered on a range of test problems.

2 THE TRUNCATED NEWTON ALGORITHM

The algorithm can be described as follows:

0. Select $\boldsymbol{x}_0,\ k = 0,\ f(\boldsymbol{x}_0)$
1. Compute $g(\boldsymbol{x}_k) = \nabla_x f(\boldsymbol{x}_k)$. If $\|\boldsymbol{g}(\boldsymbol{x}_k)\|_2 < \epsilon$, stop
2. Compute $\boldsymbol{H}(\boldsymbol{x}_k) = \nabla_{xx} f(\boldsymbol{x}_k)$
   Find a search direction $\boldsymbol{p}_k$ such that $\|\boldsymbol{H}(\boldsymbol{x}_k)\ \boldsymbol{p}_k + \boldsymbol{g}_k\|_2 \leq \eta_k$
   where $\eta_k = \min(0.1, \|\boldsymbol{g}(\boldsymbol{x}_k)\|_2^{1/2})\ \|\boldsymbol{g}(\boldsymbol{x}_k)\|_2$,
   by solving the system $\boldsymbol{H}(\boldsymbol{x}_k)\ \boldsymbol{p}_k = -\boldsymbol{g}(\boldsymbol{x}_k)$

using Preconditioned Conjugate Gradient as supplied in PAREBE
If $\boldsymbol{H}(\boldsymbol{x}_k)$ is found not to be positive definite then
use a modification of the conjugate gradient (see [11])

3. $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \, \boldsymbol{p}_k$
where α_k is the largest scalar of the form 2^{-l}, $l = 0,\ 1,\ 2,\ ...$
such that $f(\boldsymbol{x}_k + 2^{-l} \, \boldsymbol{p}_k) \leq f(\boldsymbol{x}_k) + \beta_k \, 2^{-l} \, \boldsymbol{p}_k^T \, \boldsymbol{g}(\boldsymbol{x}_k)$
and $\beta_k = 10^{-4}$
4. $k \leftarrow k + 1$, goto 1

An interface between this algorithm and the CUTE test set has been written. Constrained problems have been included by minimizing the artificial penalty function $\hat{f}(\boldsymbol{x}) = f(\boldsymbol{x}) + \rho \, \|\boldsymbol{c}(\boldsymbol{x})\|^2$ where f is the cost function, $\boldsymbol{c}$ the vector of constraints and ρ the penalty parameter set to 1.

3 PRECONDITIONERS USED

For a more detailed description of the preconditioners mentioned here see [5]. We use the following preconditioners in our tests: no preconditioning (NONE), the diagonal preconditioner (DIAG), the EBE Element-by-Element preconditioner (EBE) and the Gauss-Seidel Element-by-Element preconditioner (GSEBE).

More precisely, we consider the solution of the system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, where $\boldsymbol{A} = \sum_{i=1}^{n_e} \boldsymbol{A}_i$. Let $\Delta(\boldsymbol{M})$ be the diagonal of $\boldsymbol{M}$, $\boldsymbol{W} = \Delta(\boldsymbol{A})$, $\boldsymbol{W}_i = \Delta(\boldsymbol{A}_i)$.
The EBE preconditioner is

$$P_{EBE} = \sqrt{\boldsymbol{W}} \prod_{i=1}^{n_e} \boldsymbol{L}_i \prod_{i=1}^{n_e} \boldsymbol{D}_i \prod_{i=n_e}^{1} \boldsymbol{L}_i^T \sqrt{\boldsymbol{W}}, \tag{5}$$

where $\boldsymbol{L}_i$ and $\boldsymbol{D}_i$ are the $\boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^T$ factors of the Winget decomposition of $\overline{\boldsymbol{A}_i} = \boldsymbol{I} + \sqrt{\boldsymbol{W}}^{-1}(\boldsymbol{A}_i - \boldsymbol{W}_i)\sqrt{\boldsymbol{W}}^{-1} = \boldsymbol{L}_i \boldsymbol{D}_i \boldsymbol{L}_i^T$.
The GSEBE preconditioner is

$$P_{GSEBE} = \sqrt{\boldsymbol{W}} \prod_{i=1}^{n_e} (\boldsymbol{I} + \boldsymbol{L}_i) \prod_{i=n_e}^{1} (\boldsymbol{I} + \boldsymbol{L}_i^T) \sqrt{\boldsymbol{W}}, \tag{6}$$

where $\sqrt{\boldsymbol{W}}^{-1}(\boldsymbol{A}_i - \boldsymbol{W}_i)\sqrt{\boldsymbol{W}}^{-1} = \boldsymbol{L}_i + \boldsymbol{L}_i^T$.

4 PREPROCESSING

Let $f(\boldsymbol{x})$ be of the form (2). Clearly, this decomposition may not be unique, and different decompositions may significantly affect the performance of the preconditioners considered in this paper since their efficiency depends on the partitioning of the initial matrix and on the

magnitude of the off-diagonal elements of the elementary matrices. It has been observed [5] that, in some of the experiments, the performance of the conjugate gradient method, both for vectorization and parallelization, is limited because of a non-optimal choice of the decomposition. Frequently, the construction of an initial decomposition of f into elements by a user depends more on considerations on the easiness of expressing the function and its derivatives, rather than considerations of computational performance. It is our experience with many of the test examples in the CUTE package, for example, that the overlap between elements is high (and even that some elements are entirely subsumed by other elements). It pays to merge the two elements into a single super-element or *group* if the local variable set for one element is completely contained within another. More generally, if the local variable sets for two elements significantly overlap it is advantageous to merge the elements into a single group. Therefore, we regard it to be crucial for performance to determine an heuristic to partition the original set of elements into computationally attractive disjoint groups.

It is clear that determining an optimal partitioning of the elements into groups may be costly, or even impossible. The main difficulty is to define a suitable heuristic to control this amalgamation process. Several parameters are used to control this merging process in PAREBE. This problem has been considered [12] from the point of view of improving the matrix-vector product at the heart of many iterative methods for minimizing partially separable functions. Two elements are amalgamated if it leads to a decrease in the number of floating point operations in the matrix-vector product. In PAREBE, the same idea is applied for decreasing the cost of a preconditioned conjugate gradient iteration (time spent in triangular solves and in matrix-vector products). Several strategies for amalgamating elements are available in PAREBE. We only describe here the one we have used in our experiments. Let $\mathcal{G}_i$ denote an element, $\mathcal{V}_i$ denote the set of indices of variables used by the element $\mathcal{G}_i$, and $|\mathcal{V}_i|$ denote the cardinal of $\mathcal{V}_i$. Finally, let $tim(i)$ be the estimated time spent in a matrix-vector product of order i for the preconditioners NONE and DIAG (strategy **amalg1**) and in a matrix-vector product and in two triangular solves of order i for the other preconditioners (strategy **amalg2**). The amalgamation process we have used computes the *benefit*,

$$b(\mathcal{G}_i, \mathcal{G}_j) = \frac{tim(|\mathcal{V}_i|) + tim(|\mathcal{V}_j|) - tim(|\mathcal{V}_i \cup \mathcal{V}_j|)}{tim(|\mathcal{V}_i|) + tim(|\mathcal{V}_j|)}, \tag{7}$$

for all pairs of elements, and amalgamates the pair with the largest benefit as long as it is larger than a threshold value (equal to zero in our experiments). Note that $tim(i)$ only depends on i and can be computed once and for all on the target computer, and subsequently retrieved as

needed.

5 NUMERICAL RESULTS

We report on experiments on a set of unconstrained and constrained optimization problems (denoted as U and C) from the CUTE collection. In order to isolate the effect of the preconditioners, we have only selected convex problems so as to avoid issues relating to the modification of the conjugate gradient method when the Hessian is not positive definite. In Table 1 we report some statistics on each of our test problems : the name of the problem, whether it is constrained (c) or unconstrained (u), the number of variables (#v.), number of elements (#elts), minimum size of elements (Min.), maximum size of elements (Max.), average size of elements (Avrge) with the following merging strategies (#A.) : 0. no merging (**amalg0**); 1. **amalg1**; 2. **amalg2**, the *degree of overlap*, defined as $Ov. = \frac{\#elts\ Avrge}{\#v.}$ (that is the average number of elements containing each variable) and κ the condition number of the Hessian at the solution.

We show in Tables 2 and 3 the number of Newton iterations (#it.op), the total execution time (t.op), the total number of iterations in the linear system solution (#it.sl) and the time spent in the solution of the linear systems (t.sl) both with and without amalgamation. The total execution time does not include the time for performing the symbolic part of the amalgamation process which is reported in the last column of the tables (t.amal).

The amalgamation time can become prohibitive on some problems. On problem `BDQRTIC` , it is large because one variable is included in all the groups. This occurs in some academic problems from the CUTE collection, but is less common on real problems. The problem `POWER` (see Table 2) also has a particular structure. One group contains all the variables, thus the numerical amalgamation assembles the whole matrix and EBE behaves as a direct method. As a consequence, the execution time becomes prohibitive because the methods we use are not optimal for dense problems, where blocked factorizations would be much more efficient. With these exceptions, our amalgamation procedure is efficient. Indeed, amalgamation reduces the optimization time and the linear solution time in nearly all the cases, even when the numerical amalgamation time is included. If we also include the time of symbolic amalgamation, we still have benefits on difficult problems (see for instance `DIXON3DQ`, `FLETCHBV2`, `GENHS28`). On parallel computers, the amalgamation procedure can be parallelized. We have also experimented using graph partitioning techniques (see [13]) that give similar results with a preprocessing

Problems	C/U	#v.	#A.	#elts	Min.	Max.	Avrge	Ov.	κ
BDQRTIC	U	1000	0	1992	1	5	3.0	6.0	5×10^3
			1	249	8	8	8.0	2.0	
			2	125	8	12	12.0	1.5	
CRAGGLVY	U	1000	0	2495	1	2	1.6	4.0	1×10^2
			1	167	4	7	7.0	1.2	
			2	125	8	9	9.0	1.1	
DIXON3DQ	U	1000	0	1000	1	2	2.0	2.0	6×10^4
			1	500	1	3	3.0	1.5	
			2	300	1	7	5.0	1.5	
DIXON3DQ	U	2000	0	2000	1	2	2.0	2.0	2×10^4
			1	1000	1	3	3.0	1.5	
			2	500	1	7	5.0	1.3	
DIXON3DQ	U	3000	0	3000	1	2	2.0	2.0	1×10^4
			1	1500	1	3	3.0	1.5	
			2	750	1	7	5.0	1.3	
ENGVAL1	U	1000	0	999	2	2	2.0	2.0	9×10^0
			1	499	3	4	3.0	1.5	
			2	249	5	8	5.0	1.3	
FLETCBV2	U	1000	0	2001	1	2	1.5	3.0	2×10^4
			1	167	4	7	7.0	1.2	
			2	125	8	9	9.0	1.1	
FMINSURF	U	1024	0	962	4	1024	5.0	4.8	8×10^4
			1	1	1024	1024	1024	1.0	
			2	1	1024	1024	1024	1.0	
MOREBV	U	1000	0	1000	2	3	3.0	3.0	2×10^4
			1	249	6	8	6.0	1.5	
			2	125	8	10	10.0	1.3	
POWELLSG	U	1000	0	1000	2	2	2.0	2.0	1×10^4
			1	250	4	4	4.0	1.0	
			2	250	4	4	4.0	1.0	
POWER	U	1000	0	1	1000	1000	1000	1.0	5×10^2
			1	1	1000	1000	1000	1.0	
			2	1	1000	1000	1000	1.0	
SCHMVETT	U	1000	0	998	3	3	3.0	3.0	5×10^1
			1	249	6	8	6.0	1.5	
			2	125	8	10	10.0	1.3	
TRIDIA	U	1000	0	1000	1	2	2.0	2.0	9×10^3
			1	499	3	4	3.0	1.5	
			2	249	5	8	5.0	1.3	
GENHS28	C	1000	0	999	3	1000	4.0	4.0	1×10^2
			1	1	1000	1000	1000	1.0	
			2	1	1000	1000	1000	1.0	
GRIDNETA	C	924	0	485	2	924	5.7	3.0	8×10^2
			1	1	924	924	924	1.0	
			2	1	924	924	924	1.0	
GRIDNETD	C	924	0	485	2	924	5.7	3.0	6×10^2
			1	1	924	924	924	1.0	
			2	1	924	924	924	1.0	
HAGER1	C	2001	0	2001	1	3	2.0	2.0	3×10^4
			1	334	3	7	7.0	1.2	
			2	251	5	9	9.0	1.1	

Table 1: Statistics on the range of CUTE problems solved before and after preprocessing (on a HP 715/64 workstation).

Amalg		No/Yes	No/Yes	No/Yes	No/Yes	Yes
Problem	Prec.	#it.op	t.op	#it.sl	t.sl	t.am
BDQRTIC	NONE	14/14	2.3/1.9	79/79	0.9/0.5	2.9
1000	**DIAG**	9/9	1.1/1.2	17/ 17	0.2/ 0.3	2.9
	EBE	10/10	2.1/1.6	12/11	1.1/0.6	145.5
	GSEBE	10/10	1.8/1.5	13/13	0.8/0.5	138.2
CRAGGLVY	**NONE**	15/15	2.7/2.0	115/115	1.1/0.5	1.1
1000	DIAG	18/18	2.7/2.3	82/82	0.9/0.4	1.1
	EBE	14/14	3.1/2.1	29/30	1.7/0.7	1.1
	GSEBE	14/14	3.0/2.0	31/35	1.5/0.6	1.1
DIXON3DQ	NONE	4/4	8.9/5.0	1748/1748	8.8/4.9	0.3
1000	DIAG	5/5	9.7/5.5	1881/1881	9.6/5.3	0.2
	EBE	5/5	16.2/3.6	673/440	16.1/3.5	0.2
	GSEBE	5/5	16.5/5.5	696/704	16.4/5.4	0.3
DIXON3DQ	NONE	4/4	27.6/16.0	2748/2748	27.4/15.7	1.0
2000	DIAG	5/5	29.7/17.1	2846/2846	29.4/16.9	1.0
	EBE	5/5	67.0/11.1	1030/628	66.7/10.9	1.0
	GSEBE	5/5	67.4/27.8	1070/1107	67.1/27.5	1.0
DIXON3DQ	NONE	5/5	59.5/36.3	3754/3754	59.1/35.9	2.1
3000	DIAG	5/5	63.8/40.3	3878/3878	63.3/39.9	2.1
	EBE	5/5	122.1/23.3	1350/833	121.6/22.8	2.1
	GSEBE	5/5	122.6/54.1	1389/1444	122.1/53.7	2.1
ENGVAL1	**NONE**	10/10	0.7/0.7	29/29	0.1/0.2	0.2
1000	**DIAG**	10/10	0.7/0.7	20/20	0.1/0.1	0.2
	EBE	8/8	0.8/0.8	9/9	0.3/0.3	0.2
	GSEBE	8/8	0.7/0.7	9/9	0.2/0.3	0.2
FLETCBV2	NONE	1/1	6.9/2.7	969/969	6.7/2.6	0.7
1000	DIAG	1/1	6.9/2.8	969/969	6.8/2.7	0.7
	EBE	1/1	10.8/1.7	324/200	10.7/1.6	0.7
	GSEBE	1/1	10.8/2.7	326/347	10.7/2.5	0.7
FMINSURF	NONE	32/39	1285/1250	10094/10094	1257/1218	0.1
1024	DIAG	46/34	298.4/255.7	2079/1720	259.8/225.8	0.1
	EBE	36/31	1637/1724	787/34	1607/1697	0.1
	GSEBE	49/37	373.0/249.3	969/611	330.6/216.2	0.1
MOREBV	NONE	2/2	3.3/1.8	549/549	3.2/1.7	0.3
1000	DIAG	2/2	3.7/2.0	594/594	3.6/1.9	0.3
	EBE	2/2	4.4/1.6	186/164	4.3/1.5	0.3
	GSEBE	2/2	5.1/2.0	217/221	5.0/1.9	0.3
POWELLSG	NONE	18/18	0.8/0.8	67/67	0.3/0.3	0.2
1000	**DIAG**	17/17	0.8/0.7	61/61	0.3/0.3	0.2
	EBE	17/17	1.7/1.0	44/17	1.3/0.5	0.2
	GSEBE	17/17	1.5/1.0	44/43	1.0/0.6	0.3
POWER	NONE	31/31	68.3/82.8	423/423	47.2/60.7	0.0
1000	**DIAG**	30/30	23.9/37.1	30/30	3.4/16.5	0.0
	EBE	30/30	1082/1096	30/30	1062/1075	0.0
	GSEBE	30/30	46.0/59.9	62/62	25.3/38.7	0.0
SCHMVETT	NONE	6/6	1.3/1.3	47/47	0.3/0.2	0.3
1000	DIAG	6/6	1.3/1.2	38/38	0.2/0.2	0.3
	EBE	6/5	1.5/1.2	15/12	0.5/0.3	0.3
	GSEBE	5/5	1.1/1.2	11/14	0.3/0.3	0.3

Table 2: Results on a HP 715/64 workstation.

Amalg		No/Yes	No/Yes	No/Yes	No/Yes	Yes
Problem	Prec.	#it.op	t.op	#it.sl	t.sl	t.am
TRIDIA	NONE	9/9	3.3/2.0	576/576	3.1/1.7	0.3
1000	**DIAG**	8/8	0.4/0.4	46/46	0.2/0.2	0.2
	EBE	7/5	0.7/0.4	18/11	0.5/0.3	0.3
	GSEBE	7/7	0.6/0.5	18/18	0.4/0.3	0.3
GENHS28	NONE	7/7	9.8/13.5	75/75	8.7/12.4	0.1
1000	DIAG	6/6	6.4/9.6	46/46	5.4/8.6	0.1
	EBE	6/1	9.9/1.8	19/1	8.9/1.6	0.1
	GSEBE	6/6	8.1/11.2	18/18	7.1/10.2	0.1
GRIDNETA	NONE	6/6	20.9/23.7	207/207	20.2/23.0	0.0
924	DIAG	6/6	19.6/22.4	194/194	18.9/21.7	0.0
	EBE	6/1	40.1/4.3	153/1	39.4/4.1	0.0
	GSEBE	6/6	89.0/28.5	354/99	88.3/27.8	0.0
GRIDNETD	NONE	6/6	18.5/21.3	181/181	17.6/20.5	0.0
924	DIAG	6/6	17.7/20.6	173/173	16.9/19.8	0.0
	EBE	6/3	33.4/12.4	117/3	32.5/11.9	0.0
	GSEBE	6/6	68.1/24.7	269/82	67.2/23.9	0.0
HAGER1	NONE	11/11	60.5/35.7	5843/5851	59.8/35.1	0.7
2001	DIAG	10/10	125.3/73.5	11927/11876	124.7/72.9	0.6
	EBE	12/1	635.4/0.2	15132/1	634.6/0.2	0.7
	GSEBE	10/1	243.5/0.2	6055/1	243.0/0.1	0.7

Table 3: Results on a HP 715/64 workstation.

cost even more attractive (often divided by a factor of 10). We also notice that the number of Newton iterations in the optimization process does not differ a lot from preconditioner to preconditioner. This is to be expected since the inner and outer iteration convergence tolerances are the same for all the cases. Only the total number of iterations in the linear system solution is significantly influenced by the choice of the preconditioner. The cost of one iteration of EBE has been observed to be roughly 3 times the cost of an iteration of DIAG. Thus, EBE becomes competitive when the number of iteration is reduced by at least 3 in comparison with DIAG. This typically happens for ill-conditioned problems when using amalgamation. A large number of problems from the CUTE collection (including our problems: BDQRTIC, ENGVAL1, POWELLSG, POWER, SCHMVETT) give rise to well-conditioned linear systems. Indeed, in these cases, preconditioning does not help in reducing the number of iterations and, in general, the number of iterations of the linear solver per truncated Newton iteration is very low. However, as the size of the problem increases, the effectiveness of more sophisticated preconditioners is more noticeable. Additionally, these preconditioners can lead to a more accurate solution. For instance, on DIXON3DQ, which is numerically one of the harder problems to solve, there is no convergence without preconditioning nor with diagonal preconditioner when n = 3000 and with the convergence criterion $\|\boldsymbol{g}(\boldsymbol{x}_k)\| < \sqrt{\epsilon_m}$, where ϵ_m is the machine precision,

but fortunately convergence is achieved using EBE and GSEBE. GSEBE appears to be the most efficient preconditioner on problems CRAGGLVY, FMINSURF, SCHMVETT and HAGER1. This is because the construction of the preconditioner is very cheap, it only requires a scaling of the elements. On these problems, the construction time is large for EBE while the convergence is slow with no or diagonal preconditioning. EBE is the best preconditioner on problems such as DIXON3DQ, FLETCBV2, MOREBV, TRIDIA using amalgamation. The gain is impressive —close to 10— in some cases.

We compare in Table 4 the serial versus the parallel execution on 8 processors of the Alliant FX/80. The three first column are similar to the previous tables. Additionally, we report in column one the number of colours obtained with the three amalgamation strategies. These colours correspond to groups of elements that do not share any variable. Thus, we can parallelize the computations over elements within each colour (see [14]). Column #it.op gives the number of optimization iterations for both serial and parallel execution, #it.cg the number of cg iterations and t.op the total execution time without the time for symbolic amalgamation. The last column shows the speed-up achieved (sequential t.op divided by parallel t.op). The experiments demonstrate that amalgamation increases substantially the performance of the code when the element size is large enough to take advantage of vectorization. The speedup achieved is good as soon as the problem structure is adequate (see problems DIXON3DQ and MOREBV for example).

6 CONCLUSION

We have demonstrated in this study that the use of Element-by-Element preconditioners for solving large scale unconstrained optimization problems is efficient when the linear systems arising from the Newton equation are difficult to solve. The structure of the problems arising from partial separability is also fundamental because the degree of overlap has an impact on the conjugate gradient convergence when using preconditioners such as EBE, and because the sizes of the elementary Hessians influence the performance, especially on vector processors, where large elements are required for efficient vectorization. Our experiments suggest that — except when the structure of the problem is not suitable (for exemple when an element includes all variables) — Element-by-Element preconditioners are competitive with diagonal preconditioning on easy problems, and can provide large gains on hard problems. The use of our preprocessing technique — element amalgamation — improves the structure of the problems (because elements are often small with a large degree of overlap) and the convergence of Element-by-Element precon-

Problem	Prec	Amalg	#it.op seq/par	#it.cg seq/par	t.op seq/par	Speedup 8 procs
DIXON3DQ	NONE	No	4/4	1748/1748	117.8/ 22.6	5.2
1000		Yes	4/4	1748/1748	44.8/ 10.8	4.2
	DIAG	No	5/5	1945/1945	131.7/ 25.4	5.2
#colours		Yes	5/5	1881/1881	48.9/ 11.9	4.1
3	**EBE**	No	5/5	673/ 826	152.1/ 37.0	4.1
3		Yes	5/5	209/ 219	17.1/ 6.0	2.9
3	GSEBE	No	5/5	686/ 871	153.5/ 35.7	4.3
		Yes	5/5	724/ 694	46.4/ 11.4	4.1
ENGVAL1	NONE	No	10/10	29/29	10.9/ 9.9	1.1
1000		Yes	10/10	29/29	11.6/ 10.3	1.1
	DIAG	No	10/10	20/20	10.5/ 10.1	1.0
#colours		Yes	10/10	20/20	11.4/ 10.3	1.1
3	EBE	No	8/8	9/9	11.1/ 8.8	1.3
3		Yes	8/8	9/9	11.5/ 9.2	1.2
3	**GSEBE**	No	8/8	9/9	10.0/ 8.4	1.2
		Yes	8/8	9/9	10.4/ 8.9	1.2
MOREBV	NONE	No	2/2	549/549	44.1/ 9.3	4.7
1000		Yes	2/2	549/549	16.3/ 5.1	3.2
	DIAG	No	2/2	594/594	47.7/ 9.9	4.8
#colours		Yes	2/2	594/594	17.5/ 5.2	3.3
5	**EBE**	No	2/2	186/228	49.9/ 12.0	4.2
3		Yes	2/2	31/29	5.4/ 3.1	1.7
3	GSEBE	No	2/2	217/256	57.1/ 13.1	4.4
		Yes	2/2	177/177	14.1/ 4.8	2.9
POWELLSG	NONE	No	18/18	67/67	12.2/ 9.1	1.3
1000		Yes	18/18	67/67	12.3/ 10.6	1.2
	DIAG	No	17/17	61/61	11.7/ 9.0	1.3
#colours		Yes	17/17	61/61	11.7/ 10.2	1.2
2	EBE	No	17/17	44/44	21.1/ 11.0	1.9
1		Yes	17/17	17/17	14.1/ 11.1	1.3
1	GSEBE	No	17/17	44/68	18.6/ 11.0	1.7
		Yes	17/17	43/43	14.6/ 10.9	1.3
SCHMVETT	NONE	No	6/6	47/47	13.9/ 11.5	1.2
1000		Yes	6/6	47/47	13.4/ 11.6	1.2
	DIAG	No	6/6	38/38	13.3/ 11.5	1.2
#colours		Yes	6/6	38/38	13.1/ 11.5	1.1
5	EBE	No	6/6	15/17	16.3/ 12.3	1.4
3		Yes	5/5	11/12	12.4/ 10.4	1.2
3	**GSEBE**	No	5/6	11/17	11.2/ 11.8	1.1
		Yes	5/5	12/13	11.7/ 10.2	1.2
TRIDIA	NONE	No	9/9	576/576	42.1/ 11.1	3.8
1000		Yes	9/9	576/576	19.5/ 7.5	2.6
	DIAG	No	8/8	46/46	6.7/ 4.6	1.5
#colours		Yes	8/8	46/46	6.3/ 4.6	1.4
3	EBE	No	7/8	18/26	8.7/ 5.5	1.6
3		Yes	6/6	12/13	6.3/ 4.3	1.5
3	GSEBE	No	7/7	18/21	7.7/ 4.5	1.7
		Yes	7/7	17/17	6.5/ 4.5	1.4

Table 4: Number of iterations, execution time, and performance of the optimization algorithm on CUTE problems (on an ALLIANT FX/80). Under the name of Problem, appears successively the number of variables, the number of colours before amalgamation, and the number of colours with **amalg0** to **amalg2**.

ditioners as well as their uniprocessor execution rate by increasing the size of elements. At present, the cost of the symbolic part of this procedure is excessive in some cases and we are currently working to improve this. The range of preconditioners and the preprocessing technique used in that paper are implemented in the PAREBE package [6]. Both this package and the truncated Newton optimization technique described in this paper will shortly be available by anonymous ftp.

References

[1] A. GRIEWANK and Ph. L. TOINT. On the unconstrained optimization of partially separable functions. In M. J. D. Powell, editor, *Nonlinear Optimization 1981*, pages 301–312. Academic Press, London and New York, 1982.

[2] A. R. CONN, N. I. M. GOULD, and Ph. L. TOINT. An introduction to the structure of large scale nonlinear optimization problems and the lancelot project. In R. Glowinski and A. Lichnewsky, editors, *Computing Methods in Applied Sciences and Engineering*, pages 42–54. SIAM, Philadelphia, USA, 1990.

[3] M. R. HESTENES. Multiplier and gradient methods. *J. Optim. Theory Appl.*, 4:303–320, 1969.

[4] M. J. D. POWELL. A method for nonlinear constraints in minimization problems. In R. Fletcher, editor, *Optimization*, pages 283–298. Academic Press, London and New York, 1969.

[5] M. J. DAYDÉ, J.-Y. L'EXCELLENT, and N. I. M. GOULD. On the use of element-by-element preconditioners to solve large scale partially separable optimization problems. Technical Report RT/APO/94/4, ENSEEIHT-IRIT, Toulouse, France, 1994. to appear in the SIAM J. Sci. Comput.

[6] M. J. DAYDÉ, J.-Y. L'EXCELLENT, and N. I. M. GOULD. PAREBE : Parallel element-by-element preconditioners for the conjugate gradient algorithm. Technical report, ENSEEIHT-IRIT, Toulouse, France, 1996. to appear.

[7] M. R. HESTENES and E. L. STIEFEL. Methods of conjugate gradients for solving linear systems. *Natl. Bur. Stand. J. Res.*, 49:409–436, 1952.

[8] R. S. DEMBO and T. STEIHAUG. Truncated-newton algorithms for large-scale unconstrained optimization. *Mathematical Programming*, 26:190–212, 1983.

[9] A. R. CONN, N. I. M. GOULD, A. SARTENAER, and Ph. L. TOINT. On iterated-subspace minimization methods for nonlinear optimization. Technical Report RAL-94-069, Rutherford Appleton Laboratory, Chilton, England, 1994.

[10] I. BONGARTZ, A. R. CONN, N. I. M. GOULD, and Ph. L. TOINT. CUTE: Constrained and Unconstrained Testing Environment. Technical Report TR/PA/93/10, CERFACS, Toulouse, France, 1993.

[11] M. ARIOLI, T. F. CHAN, I. S. DUFF, N. I. M. GOULD, and J. K. REID. Computing a search direction for large-scale linearly-constrained nonlinear optimization calculations. Technical Report TR/PA/93/34, CERFACS, Toulouse, France, 1993.

[12] A. R. CONN, N. I. M. GOULD, and Ph. L. TOINT. Improving the decomposition of partially separable functions in the context of large-scale optimization: a first approach. In D. W. Hearn W. W. Hager and P.M. Pardalos, editors, *Large Scale Optimization: State of the Art*. Kluwer Academic Publishers B.V, 1994.

[13] M. J. DAYDÉ, J.-Y. L'EXCELLENT, and N. I. M. GOULD. On the preprocessing of sparse unassembled linear systems for efficient solution using element-by-element preconditioners. In A. Mignotte L. Bougé, P. Fraigniaud and Y. Robert, editors, *Proceedings of Euro-Par 96, Lyon*, pages 34–43. Springer Verlag, 1996. Vol. 1124 of Lecture Notes in Computer Science.

[14] M. J. DAYDÉ, J.-Y. L'EXCELLENT, and N. I. M. GOULD. Solution of structured systems of linear equations using element-by-element preconditioners. In *Proceedings 2nd IMACS International Symposium on Iterative Methods in Linear Algebra*, pages 181–190, Toulouse, France, 1995.