Theor Chem Acc (1998) 99:197–206

Theoretical
Chemistry Accounts

Regular article

A scalable divide-and-conquer algorithm combining coarse and fine-grain parallelization

Sor Koon Goh[1], **Carlos P. Sosa**[2], **Alain St-Amant**[1]

[1] Department of Chemistry, University of Ottawa, 10 Marie Curie, Ottawa, Ontario, Canada K1N 6N5
[2] Silicon Graphics, Inc./Cray Research, Inc., 655E Lone Oak Drive, Eagan, MN 55121, USA

Received: 15 September 1997 / Accepted: 21 January 1998

Abstract. We describe an efficient algorithm for carrying out a "divide-and-conquer" fit of a molecule's electronic density on massively parallel computers. Near linear speedups are achieved with up to 48 processors on a Cray T3E, and our results indicate that similar efficiencies could be attained on an even greater number of processors. To achieve optimum efficiency, the algorithm combines coarse and fine-grain parallelization and adapts itself to the existing ratio of processors to subsystems. The subsystems employed in our divide-and-conquer approach can also be made smaller or bigger, depending on the number of processors available. This allows us to further reduce the wallclock time and improve the method's overall efficiency. The strategies implemented in this paper can be extended to any other divide-and-conquer method used within an ab initio, density functional, or semi-empirical quantum mechanical program.

Key words: Scalable algorithms – Supercomputing – Quantum mechanics – Divide-and-conquer – Large molecules

1 Introduction

Several quantum mechanical methods that scale linearly with system size have recently been introduced [1–27]. These new approaches hold great promise for calculations on truly large systems. Even though linear scaling has been achieved, such quantum mechanical applications will still require vast amounts of computer time. We can only hope to complete these calculations within a reasonable amount of real, or wallclock, time if the calculation can be run effectively on a large number of computer processors. Consequently, a new algorithm's potential usefulness is largely determined by its scalability, i.e. its ability to run as efficiently on a large number of processors as it does on a small number of processors. Amdahl's law [28] tells us, however, that it is extremely difficult to construct software that will scale efficiently as the number of processors is increased. Eventually, a point is reached where the speedups obtained increase by very little as the number of processors is increased, and little is gained by throwing more of the computer's resources towards the problem at hand. Though some computer programs can achieve excellent results on massively parallel computers, using sometimes hundreds of processors, the speedup plateau will inevitably be reached [28–32].

One class of linear scaling methods are the so-called divide-and-conquer (DAC) techniques [11–17]. Rather than performing the conventional quantum mechanical procedure over the entire molecule, the molecule is first divided into a collection of subsystems, and analagous procedures are independently carried out on each and every one. A subsystem calculation itself exhibits the same scaling behaviour as its global counterpart when the size of the subsystem problem is increased. However, the cost of a subsystem calculation quickly becomes independent of the overall system size as we need only concern ourselves with those basis functions which are either centred on atoms that are actually in, or positioned very near to, the subsystem in question. Once these subsystem calculations are localized to regions covering only a fraction of the molecule's entire extent, the total computational effort required will only rise linearly with the number of subsystems which, in turn, rises linearly with overall system size. In this fashion, DAC schemes achieve linear scaling. DAC procedures for the construction of the electronic density have been used within both density functional [11, 12] and semi-empirical calculations [13, 14]. DAC procedures for the fits of electronic density [15, 16] and exchange-correlation potentials [17] have also been used within a density functional program. Though they have yet to be applied to traditional ab initio methods, several of the DAC techniques so far proposed are equally well suited to Hartree-Fock (HF) and post-HF applications. Linear

Correspondence to: A. St-Amant

scaling is achievable [13, 14, 16, 17], and provided that the system under study is large enough, the DAC approach outperforms the conventional approach when run on a single processor [13, 14, 16, 17].

Since the subsystem calculations within a DAC application can be carried out independently, the DAC philosophy is inherently parallelizable. Coarse-grain parallelization is easily introduced by assigning each processor to a specific subsystem and carrying out the subsystem calculations in unison. However, achieving a high efficiency on a massively parallel computer architecture is not that simple since the CPU times required by these subsystem calculations vary considerably. Improper load balancing will result if we assign equal numbers of processors to each subsystem. The shorter subsystem calculations will be completed long before the larger ones, and their processors will be needlessly idle for considerable periods of time. Even if we allow for an unequal distribution of processors among the subsystems, proper load balancing is still difficult to achieve unless we have far more processors than subsystems.

In this paper, we show how coarse-grain parallelism made available to us by a DAC scheme can be efficiently exploited over both small and large numbers of processors. The key to our algorithm is its flexibility but the precise approach used will depend on the number of processors available and the number of subsystems to be treated. Its success will also depend on the efficient fine-grain parallelization of the individual subsystem calculations. Provided we can efficiently run a single subsystem over 10–20 processors, our combined coarse/fine-grain approach should be scalable up to several hundreds, if not thousands, of processors when working with truly large systems. The ability to efficiently exploit massively parallel computers will make DAC approaches even more attractive on these architectures than they already are on conventional single processor workstations.

Although the algorithm we present can be easily extended to any DAC approach, we focus specifically on the DAC fit of a molecule's electronic density [15, 16]. The minor issues that must be addressed before the algorithm is extended to other DAC calculations are also discussed. We also show that the efficiency of a particular parallelization scheme for a given number of processors and subsystems can be predicted and that rough guidelines can be established to develop software that automatically determines which specific approach is best suited for the problem and architecture at hand. The subsystems of a molecule can be redefined to achieve even lower wallclock times given a certain number of processors. Just as no single parallelization scheme is the best for all numbers of processors, no scheme for partitioning the molecule into subsystems is the best for all numbers of processors.

2 Algorithm

We demonstrate our new algorithm using the DAC fit of the electronic density, $\rho(\mathbf{r})$, as an example. The conventional, non-DAC, variational $\rho(\mathbf{r})$ fitting procedure was first introduced by Dunlap et al. [33, 34]. In a linear combination of Gaussian-type orbitals (LCGTO) approach, $\rho(\mathbf{r})$ is given by

$$\rho(\mathbf{r}) = \sum_{\mu}^{N} \sum_{\nu}^{N} P_{\mu\nu} \chi_{\mu}(\mathbf{r}) \chi_{\nu}(\mathbf{r}) \tag{1}$$

where $P_{\mu\nu}$ is an element of the density matrix and $\chi_{\mu}(\mathbf{r})$ and $\chi_{\nu}(\mathbf{r})$ are among the N basis functions in the orbital basis [35]. To reduce the number of two-electron integrals, $\rho(\mathbf{r})$ is fitted within an auxiliary basis of M (where M is of the same order as N) uncontracted Gaussians, $\{\chi'_k(\mathbf{r})\}$,

$$\tilde{\rho}(\mathbf{r}) = \sum_{k}^{M} c_k \chi'_k(\mathbf{r}) \ . \tag{2}$$

The set of expansion coefficients, $\{c_k\}$, is obtained by minimizing [33, 34]

$$\iint \frac{[\rho(\mathbf{r}) - \tilde{\rho}(\mathbf{r})][\rho(\mathbf{r}') - \tilde{\rho}(\mathbf{r}')]}{|\mathbf{r} - \mathbf{r}'|} \, d\mathbf{r} \, d\mathbf{r}' \tag{3}$$

while constraining the fitted density, $\tilde{\rho}(\mathbf{r})$, to be normalized to the total number of electrons, N_e,

$$\int \tilde{\rho}(\mathbf{r}) \, d\mathbf{r} = N_e \ . \tag{4}$$

This leads to the following matrix equation

$$\mathbf{c} = \mathbf{S}^{-1}(\mathbf{t} + \lambda \mathbf{n}) \tag{5}$$

where

$$S_{ij} = \iint \frac{\chi'_i(\mathbf{r}) \chi'_j(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} \, d\mathbf{r} \, d\mathbf{r}' \ , \tag{6}$$

$$\begin{aligned} t_i &= \iint \frac{\chi'_i(\mathbf{r}) \rho(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} d\mathbf{r} \, d\mathbf{r}' \\ &= \sum_{\mu}^{N} \sum_{\nu}^{N} P_{\mu\nu} \iint \frac{\chi'_i(\mathbf{r}) \chi_{\mu}(\mathbf{r}') \chi_{\nu}(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} \, d\mathbf{r} \, d\mathbf{r}' \ , \end{aligned} \tag{7}$$

$$n_i = \int \chi'_i(\mathbf{r}) \, d\mathbf{r} \ , \tag{8}$$

and

$$\lambda = N_e - \frac{\mathbf{n}\mathbf{S}^{-1}\mathbf{t}}{\mathbf{n}\mathbf{S}^{-1}\mathbf{n}} \ . \tag{9}$$

By reducing the number of charge distributions from N^2 to M, a fitted $\rho(\mathbf{r})$ allows us to drastically reduce the number of two-electron integrals required to construct the Fock or Kohn-Sham matrix elements. Unfortunately, the conventional $\rho(\mathbf{r})$ fitting procedure is not ideally suited for large systems as Eq. (5) requires a matrix inversion that scales cubically with system size. However, this problem is avoided when a DAC approach is employed [15, 16]. Following Yang and Lee [12], the system is divided into a collection of subsystems, and $\rho(\mathbf{r})$ is divided into a sum of subsystem contributions, $\{\rho^{\alpha}(\mathbf{r})\}$,

$$\rho(\mathbf{r}) = \sum_{\alpha}^{\text{subsystems}} \rho^{\alpha}(\mathbf{r}) = \sum_{\alpha}^{\text{subsystems}} p^{\alpha}_{\mu\nu} P_{\mu\nu} \chi_{\mu}(\mathbf{r}) \chi_{\nu}(\mathbf{r}) \qquad (10)$$

where $p^{\alpha}_{\mu\nu}$ is a partitioning function [12] that equals 1 if both $\chi_{\mu}(\mathbf{r})$ and $\chi_{\nu}(\mathbf{r})$ are centred on atoms in subsystem α, $\frac{1}{2}$ if either of the two basis functions is so centred, and 0 if neither is so centred. Rather than carrying out a single, global, fit of $\rho(\mathbf{r})$, a series of subsystem fits are carried out [15, 16]. Each subsystem fit now minimizes

$$\iint \frac{[\rho^{\alpha}(\mathbf{r}) - \tilde{\rho}^{\alpha}(\mathbf{r})][\rho^{\alpha}(\mathbf{r}') - \tilde{\rho}^{\alpha}(\mathbf{r}')]}{|\mathbf{r} - \mathbf{r}'|} \, d\mathbf{r} \, d\mathbf{r}' \qquad (11)$$

subject to the constraint that $\tilde{\rho}^{\alpha}(\mathbf{r})$ is normalized to the Mulliken population of subsystem α [15]. The subsystem fits are carried out in exactly the same way as the global fit. However, only auxiliary basis functions on actual subsystem atoms and nearby "buffer" atoms are used. In other words, Eq. (5) is solved independently for each subsystem, however, the vectors and matrices now span only the M^{α} auxiliary functions positioned on atoms within the subsystem in question or its associated buffer space. The global fitted $\rho(\mathbf{r})$ is obtained by summing up the coefficients from the subsystem fits, $\{c^{\alpha}_{k}\}$ [15, 16]. It is found that errors can be kept to less than 0.01 kcal mol^{-1} when atoms within 5.0 Å are added to a subsystem's buffer space [16]. The number of auxiliary functions employed in a subsystem fit thus becomes independent of overall system size once the system adds new atoms beyond that subsystem's 5.0 Å cutoff. At that point, the total CPU time associated with the DAC fit of $\rho(\mathbf{r})$ rises linearly with the number of subsystems, or equivalently, the total size of the system under study. Of course, the fitting procedure is only part of the entire Coulomb problem. When constructing the Fock or Kohn-Sham matrix elements, fast multipole methods [1, 6] must still be used to evaluate long-ranged interactions so as to achieve linear scaling. Despite the fact that linear scaling can be achieved with or without the use of a fitted $\rho(\mathbf{r})$, it has been shown that the remaining number of two-electron integrals requiring explicit evaluation can be cut by some two orders of magnitude with a fitted $\rho(\mathbf{r})$ [16]. Since the explicit evaluation of interactions with charge distributions that are not sufficiently "well-separated" remains the most time-consuming task for the Coulomb problem [6], then clearly, the overall CPU time can be dramatically reduced if a DAC $\rho(\mathbf{r})$ fitting procedure is adopted.

This DAC $\rho(\mathbf{r})$ fitting procedure has been implemented within the LCGTO density functional theory (DFT) program DeFT [36]. The conventional LCGTO-DFT $\rho(\mathbf{r})$ fitting procedure [33, 34] is also implemented in such programs as DGauss [37] and deMon [38]. For the purposes of this paper, the key aspect of the DAC $\rho(\mathbf{r})$ fitting procedure is that the subsystem fits can be carried out independently. It is also important to note that the CPU times associated with the subsystem fits span a considerable range of values [16]. The results in the next section show that an order of magnitude can easily separate the longest subsystem fit from the shortest. Finally, the efficiency of the DAC approach is little affected by the size of the subsystems [16], provided that the subsystems are not so large that they encompass a too larger fraction of the overall system. This means that we have considerable flexibility in the number of subsystems with which we can work. The importance of this point will be made clear further on.

We are currently in the midst of creating a parallel distributed memory version of DeFT. This is being done with the message-passing interface (MPI) library [39]. When running on a single processor, each subsystem fit is carried out one after the other. When running on a collection of processors, this same strategy can be employed. We call this stategy the fine-grain approach, as only the parallelism within a given subsystem calculation is being exploited. The most computationally intensive portion of the density fitting procedure is the evaluation of the three-centred two-electron integrals of Eq. (7). In this part of the code, fine-grain parallelism is established by partitioning the list of non-vanishing primitive pairs arising from the orbital basis sets over the processors. This is done separately for each combination of angular momenta in the orbital basis (in our tests, this would mean *ss*, *ps*, *pp*, *ds*, *dp* and *dd*) so that the varying CPU requirements of each [40] do not hinder proper load balancing. Limiting ourselves to fine-grain parallelization only is the "straightforward" approach, as no attempt is made to exploit the coarse-grain parallelism afforded by our DAC scheme. This straightforward approach can work well if the number of processors is not too large. Otherwise, Amdahl's law [28],

$$T_{\text{total}} = T_{\text{serial}} + \frac{T_{\text{parallel}}}{\text{No. of processors}} \ , \qquad (12)$$

indicates that the sections of code which cannot be carried out in parallel (requiring time T_{serial}) will begin to dominate the overall time, T_{total}, when the sections of code that can be parallelized (requiring time T_{parallel} on a single processor) are being performed on a sufficiently large number of processors. No matter how large the ratio $\frac{T_{\text{parallel}}}{T_{\text{serial}}}$ may appear, it is very difficult to achieve high efficiency on a very large number of processors. For example, even if 99.5% of a job is parallelized, a speedup of only 38.9 is seen if it is run on 48 processors. The speedup increases to only 65.1 if we double the number of processors to 96. Clearly, the fine-grain approach is not the method of choice when trying to exploit machines with hundreds, or someday thousands, of processors.

To exploit the coarse-grain parallelism inherently provided by the DAC philosophy, we use a combined coarse/fine-grain approach. Rather than carrying out the subsystem calculations one after another, we carry them out in unison, assigning each subsystem calculation to a group of processors. Since the computational burden associated with a subsystem calculation can be many times larger, or smaller, than its counterparts, assigning similar numbers of processors to each subsystem calculation would be extremely inefficient. Rather, we assign to a subsystem calculation a fraction of the total number of processors as similar as possible to the fraction of the overall time required by that subsystem calculation. Since there is such a discrepancy in the individual sub-

system times, efficient load balancing is difficult to achieve unless a reasonably large number of processors is available. For example, if one subsystem calculation is 10 times longer than another, at least 11 processors among these two subsystems are needed to achieve optimal load balancing. If faced with a smaller number of processors than 11, processors would necessarily have to be removed from the larger subsystem calculation since the smaller subsystem must always have at least a single processor at its disposal. Optimal load balancing can no longer be achieved.

If the ratio of processors to subsystems becomes too small, better load balancing can be achieved by performing multiple passes within the combined coarse/fine-grain approach. Rather than performing all the subsystem calculations in unison during a single pass, calculations are concurrently performed on only a fraction of the total number of subsystems. The remaining subsystem calculations are handled in subsequent passes. In so doing, the ratio of processors to subsystems is greatly increased within any one pass. This provides the flexibility required to achieve optimal load balancing within these passes. In our simple algorithm, the subsystems are reordered, in descending order, based on the time expected to complete them on a single processor. The first n_1 subsystems are then performed in unison within the first pass, the next n_2 within the second, and so on. By placing them in such an order, we ensure that subsystem calculations carried out within any one pass are as similar in cost as possible. Proper load balancing within a pass is, in general, more easily achieved if the ratio of times between the longest and shortest subsystem calculations is kept as small as possible. In our simple algorithm, the numbers of subsystems treated within the passes are chosen so that the times spent in the passes are as similar as possible. This is not an essential aspect of our algorithm, and this decision is made only to simplify the coding of an automated procedure. In testing the efficiency of our combined coarse/fine-grain approach, we only consider using either a single pass or two passes. However, situations in which it may become advantageous to go to three or more passes are discussed towards the end of this report. Note that the straightforward scheme using only fine-grain parallelization is equivalent to the combined coarse/fine-grain scheme where the number of passes is equal to the number of subsystems.

In this approach, it is essential to know in advance the approximate cost of each subsystem calculation. Otherwise, optimal load balancing cannot be achieved through the coarse-grain distribution of the subsystem calculations over appropriately sized clusters of processors. Fortunately, these DAC methods are being used within an iterative self-consistent field (SCF) procedure. Information from the previous iteration can be used to establish how much time will be required for each subsystem in the current iteration. Subsystem times will be little affected from iteration to iteration. For the first iteration, it should be fairly simple to establish a reasonably accurate estimate of the times required by each subsystem calculation. In our example, the DAC fit of $\rho(\mathbf{r})$, a reasonably accurate estimate of the time required by a subsystem calculation can be made given the number of auxiliary fitting functions being used and the number of charge distributions (that arise when there is appreciable overlap between a pair of primitive Gaussians in the orbital basis) being fitted. And since only the ratios of the subsystem times are ultimately important to the success of the combined coarse/fine grain scheme, the formulae used to make these estimates need not be changed upon going from one computer platform to another. In this work, we simply use subsystem times obtained from a previous run on a single processor. Also, when assigning processors to subsystems, we could further refine our algorithm by using Amdahl's law to take into account the fact that each subsystem calculation is not 100% parallelized. This begins to alter the actual numbers of processors assigned to subsystems only when the ratio of processors to subsystems is exceedingly large. However, the goal of our flexible algorithm is to avoid such a situation, whenever possible. Therefore, when assigning processors to subsystems, we assume 100% efficiency over the cluster of processors assigned to each subsystem.

A combined coarse/fine grain approach to carry out a DAC procedure is easily implemented within an existing code. It has already been established that DAC schemes can be easily introduced into existing computer codes [12, 15]. Once an MPI version of the DAC computer code is created, our coarse/fine-grain approach is implemented by simply replacing a loop over the subsystems by a loop over the number of passes. Within each cycle of this loop over the passes, the appropriate number of processors is assigned to each subsystem calculation that is to be performed within that pass. These processors are first enrolled within an MPI group [39], and the group is then assigned an MPI communicator [39]. After this has been done, the same MPI code used for the straightforward fine-grain approach can be used virtually unchanged. All that need be done is to replace the communicator associated with the entire collection of processors, MPI_COMM_WORLD [39], by the communicator associated with the subset of processors assigned to the subsystem calculation. At the end of a loop cycle, the MPI groups and communicators created for this pass are destroyed. New ones are created, as appropriate, for the next pass.

Finally, our code has been modified to exploit the vast amount of memory available when running on many processors. The original code for the DAC fit of $\rho(\mathbf{r})$ stored much of the information necessary for a subsystem calculation (most importantly, the inverse of the auxiliary basis overlap matrix) on disk. On a single processor with 64 or 128 MB of memory, it would be impossible to store all this information in memory for anything other than a relatively small molecule. However, when running on a number of processors greater than or equal to the number of subsystems, we now store all this information in memory, rather than disk. A subsystem's requirements are independent of the overall system size, and therefore, this approach is entirely scalable. This modification in the code will be responsible for the superlinear speedups often observed in the following section. Similar modifications have been made

in other scientific codes and are responsible for the superlinear speedups which are often seen when they are ported to massively parallel platforms [39].

3 Results and discussion

Benchmark DAC fits of $\rho(\mathbf{r})$ are performed on the PM3 [41] optimized extended conformations of the glycine pentapeptide (34 atoms), heptapeptide (48 atoms) and nonapeptide (62 atoms). In partitioning scheme A, these systems are divided into -$NHCH_2CO$- subsystems, a terminal HCO- subsystem, and a terminal -NH_2 subsystem. In partitioning scheme B, each one of the -$NHCH_2CO$- groups is further subdivided into NH, CH_2 and CO fragments. Regardless of the partitioning scheme, a 5.0 Å cutoff is used for the DAC fit of $\rho(\mathbf{r})$. A $(7111/411/1^*)$ orbital basis is used for heavy atoms, while a $(41/1^*)$ basis is used for hydrogen atoms [42]. To fit $\rho(\mathbf{r})$, an auxiliary basis of seven *s* functions, three sets of *p* functions, and three sets of *d* functions is assigned to each heavy atom [42]. The three *p* and *d* sets of functions, along with three of the *s* functions, are constrained to having the same exponents so as to make integral evaluation more efficient [42]. For hydrogen atoms, such a constraint is also in place, but the auxiliary basis is now reduced to four *s* functions, a single set of *p* functions, and a single set of *d* functions [42].

Table 1 lists the wallclock times required by these subsystem fits when run on a single Cray T3E processor. Results for partitioning scheme B are only included for the heptapeptide. These subsystem times include both the CPU time and the disk I/O time required to read the inverse of the overlap matrix for that subsystem's auxiliary basis. Within partitioning scheme A, the sums of the subsystem I/O times are roughly 2.0 s, 4.3 s and 5.6 s for the pentapeptide, heptapeptide and nonapeptide, respectively. The I/O time increases with system size in an approximately linear fashion, reflecting the fact that the maximum number of auxiliary basis functions within any one subsystem fit has been attained and I/O costs are now rising only with the addition of more subsystems. Within partitioning scheme A, the amount of time associated with any one subsystem calculation has almost attained a maximum value. The longest subsystem time seen for the heptapeptide is far more similar to that seen in the nonapeptide than it is to that seen in the pentapeptide. Though many of the central -$NHCH_2CO$- subsystems have exactly the same number of auxiliary basis functions assigned to their fits, those at the very centre of the polypeptide chain have slightly longer wallclock times since a greater number of primitive pairs having appreciable overlap arise from the orbital bases [15]. In a more general globular peptide, the maximum time seen for any one subsystem calculation would be considerably higher. However, the key point, that this time would quickly attain a maximum value and the overall computer time would rise linearly with system size, remains unchanged. Within partitioning scheme B, the subsystem times are naturally much smaller. The sum of the times for the three fragments of each -$NHCH_2CO$- group is roughly equal to the time seen for the corresponding -$NHCH_2CO$- subsystem within partitioning scheme A. Overall, partitioning scheme B would seem to be slightly more efficient. However, it is

Table 1. Wallclock time (in seconds, single Cray T3E processor) required by each subsystem in a divide-and-conquer fit of $\rho(\mathbf{r})$ or a series of extended glycine polypeptides

Subsystem #	Pentapeptide[a]	Heptapeptide[a]	Heptapeptide[b]	Nonapeptide[a]
1	3.0	3.1	3.1	3.2
2	13.4	13.5	2.5	13.6
3	21.1	21.9	4.8	22.1
4	20.8	23.5	6.9	23.7
5	12.4	23.3	4.9	24.1
6	2.0	21.5	7.2	23.7
7	–	12.2	8.6	23.3
8	–	2.0	5.2	21.3
9	–	–	7.7	12.3
10	–	–	9.9	2.0
11	–	–	5.4	–
12	–	–	7.7	–
13	–	–	8.8	–
14	–	–	5.1	–
15	–	–	6.6	–
16	–	–	7.8	–
17	–	–	4.4	–
18	–	–	5.0	–
19	–	–	3.7	–
20	–	–	2.0	–
Total	72.7	121.0	117.3	169.3
Conventional[c]	73.2	163.8	163.8	281.4

[a] Using partitioning scheme A
[b] Using partitioning scheme B
[c] Time required for a conventional LCGTO-DFT fit of $\rho(\mathbf{r})$

clear that the efficiency of the DAC fit of $\rho(\mathbf{r})$ on a single processor is fairly insensitive to our choice of subsystems (provided the subsystems do not become so large as to cover a significant fraction of the total system, in which case, the cubic scaling of the conventional non-DAC approach would be recovered). This insensitivity to the chosen partitioning scheme, A or B, has been previously observed over a range of sizes for these polypeptides [16]. It is important to note that the individual CPU times for the subsystem fits within any polypeptide can span a full order of magnitude. This presents a challenging load balancing problem. Finally, within partitioning scheme A, the overall wallclock times associated with the DAC fits display essentially perfect linear scaling as an extra 48.3 s of total CPU time are required each time another two -$NHCH_2CO$- subsystems are added. Similar results (not shown) are also seen using scheme B. The conventional non-DAC fits of $\rho(\mathbf{r})$ require 73.2 s, 163.8 s and 281.4 s, respectively. These times scale slightly worse than quadratically. However, all these times neglect the time spent inverting the overlap matrices of the auxiliary bases before the start of the SCF procedure. The conventional non-DAC calculation will see the cost of this step rise cubically with system size. For the DAC calculation, it will only rise linearly because the number of auxiliary basis functions in a subsystem calculation has already reached a maximum value.

Figures 1–4 display the speedups obtained on a Cray T3E computer for the four systems listed in Table 1. Note that data for the combined coarse/fine-grain approach with a single pass does not begin until the number of processors is greater than or equal to the number of subsystems. This is a necessary condition when only a single pass is performed. In general, we need a number of processors which is at least equal to the maximum number of subsystems treated within a pass. Therefore the speedups for the combined coarse/fine-grain approach with two passes begins when the number of processors is at least one half the total number of subsystems. Also, a significant improvement in the efficiency of the straightforward fine-grain scheme is observed each time the number of processors is equal to or greater than the number of subsystems. These superlinear speedups arise from the program's switch to using memory, rather than disk, to store all the necessary information for the subsystem fits.

Figure 1 shows the speedups provided by various approaches when applied to the smallest system, the pentapeptide, using partitioning scheme A. If we look at the speedups obtained through fine-grain parallelization only, we see that a reasonable level of efficiency is maintained up to 24 processors. This number of processors provides a speedup of 23.3, for an efficiency of 97.8% . Beyond this number of processors, the efficiency of the fine-grain approach drops dramatically. A speedup of only 33.8, for an efficiency of only 70.4%, is obtained when running the code on 48 processors.

The single pass coarse/fine-grain approach on the pentapeptide starts off with a very poor speedup over eight processors. This, however, is to be expected. With six subsystems and only eight processors, the best that we can do to reduce the wallclock time is assign an extra processor to each of the two most time-consuming subsystem fits. Even if these two subsystem calculation times are halved, they will still take just over 10 s each, as opposed to only 2 s for the shortest subsystem calculation. Optimal load balancing is far from being achieved. As more processors are added, the efficiency of the single pass coarse/fine-grain approach improves dramatically. Starting with an efficiency of only 68.8% on 8 processors, the efficiency rapidly rises to roughly 90% for 16,

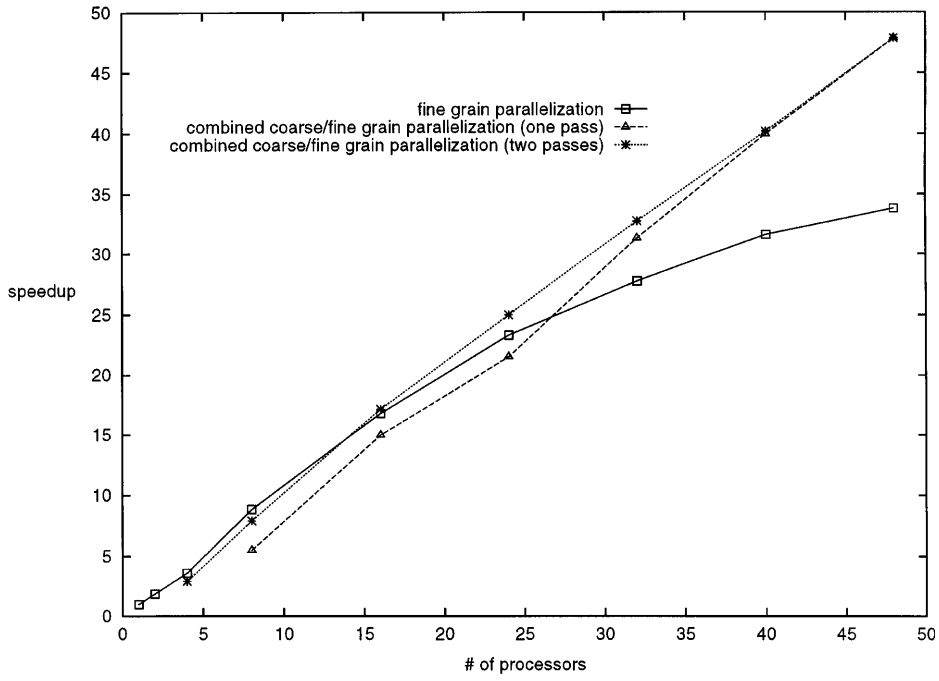

Fig. 1. Speedup on a Cray T3E for the divide-and-conquer (DAC) fit of $\rho(\mathbf{r})$ for the glycine pentapeptide using partitioning scheme A

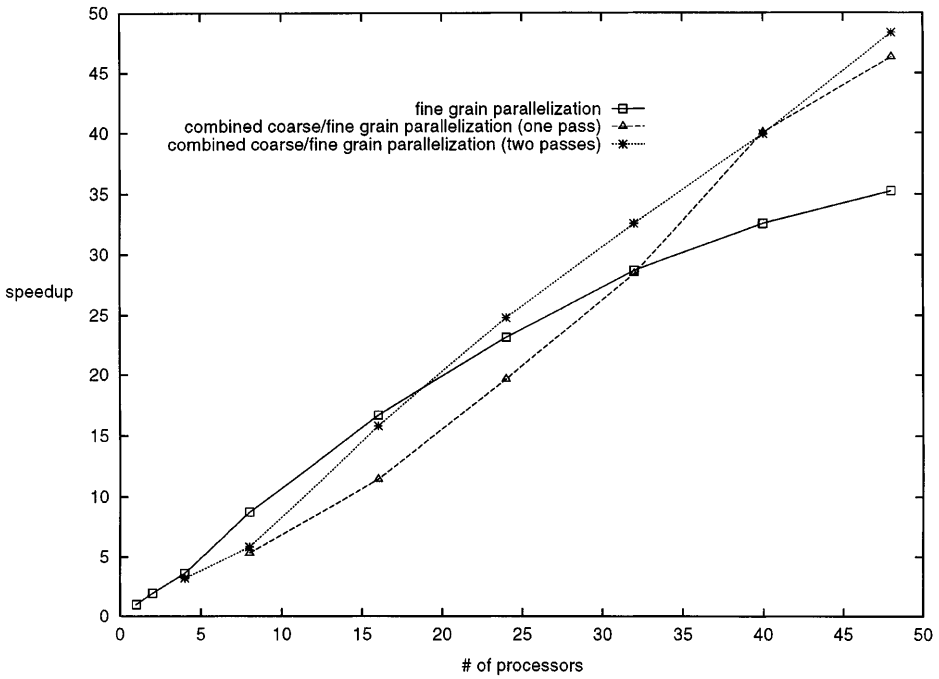

Fig. 2. Speedup on a Cray T3E for the DAC fit of $\rho(\mathbf{r})$ for the glycine heptapeptide using partitioning scheme A

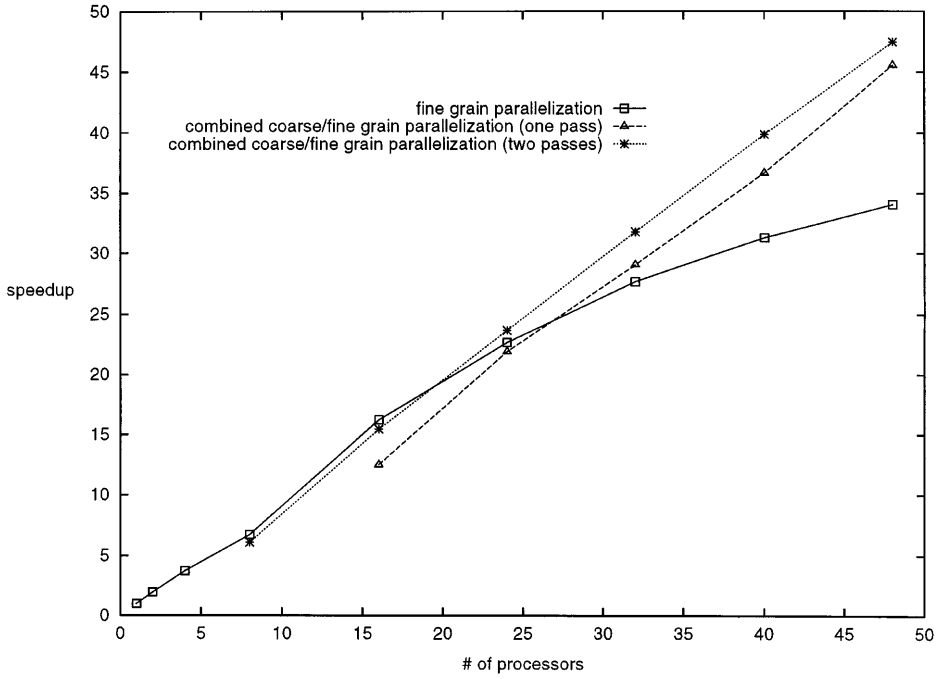

Fig. 3. Speedup on a Cray T3E for the DAC fit of $\rho(\mathbf{r})$ for the glycine nonapeptide using partitioning scheme A

24 and 32 processors. The exact efficiencies depend on what limited load balancing we can achieve through coarse-grain parallelism with the number of processors at our disposal. In our algorithm, a larger number of processors does not necessarily mean that a higher efficiency will be achieved. As a simple illustration of this fact, if we have two subsystem fits of equal complexity, 100% efficiency can be theoretically achieved on 16 processors, but not on 17, as our method only allows an integer number of processors to be assigned to a subsystem. However, in general, more processors allow for a better coarse-grain distribution of subsystems over groups of processors and a higher efficiency is most often observed. When going to 40 and 48 processors, there are now enough processors available to achieve near optimal load balancing. Speedups of 40.0 and 47.8 are observed. Had we been able to store all the information for all the subsystem fits in memory for the single processor run and eliminate all disk I/O time (this is, however, quite impossible with the limited memory available on a single node), our speedups would only drop to 38.8 and 46.5. These are still excellent results. When running over 48 processors, no subsystem calculation runs over more than 14 processors. In descending order, 14, 13, 9, 8, 2

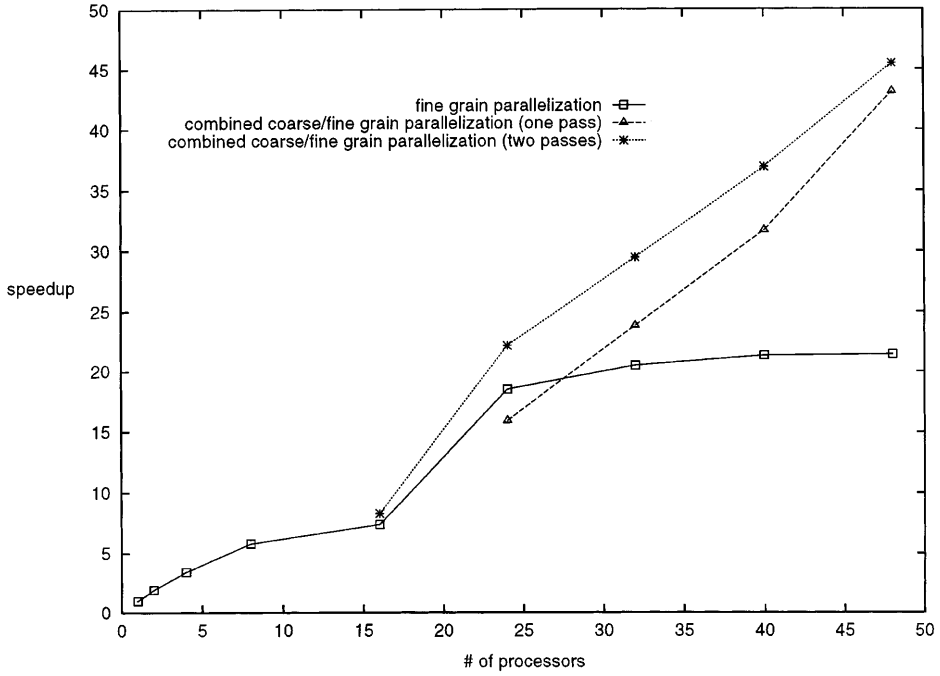

Fig. 4. Speedup on a Cray T3E for the DAC fit of $\rho(\mathbf{r})$ for the glycine heptapeptide using partitioning scheme B

and 2 processors are assigned to the subsystems once we have ordered them based on their wallclock times on a single processor. Amdahl's law should therefore not have a great impact on any one subsystem calculation, particularly since the smaller subsystems, which are most susceptible to poor scalability, have the least number of processors assigned to them.

When we begin to perform two passes within the combined coarse/fine-grain approach, our processor to subsystem ratio within an individual pass is higher, and proper load balancing is far easier to achieve when working with a limited number of processors. Speedups of 2.9, 7.9, 17.2, 25.0, 32.8, 40.2 and 47.8 are obtained over 4 to 48 processors. Superlinear speedups arise from the elimination of disk I/O. Excellent speedups are still observed out to 48 processors. However, at this point, the first pass is performing the largest two subsystem calculations in tandem, each one run over 24 processors. We are very near the point where Amdahl's law will begin to cause a noticeable reduction in the efficiency of a single subsystem fit. The second pass is also a problem since 21, 19, 5 and 3 processors are being assigned to the four least time-consuming subsystems. Granted, we are not assigning as many processors to an individual subsystem. However, we are assigning nearly as many to a much smaller subsystem calculation, and Amdahl's law may play an even more important role given the subsystem's smaller size. Beyond 48 processors, we would therefore expect our efficiency to drop at a rate similar to that seen when going beyond 24 processors in the straightforward fine-grain approach. With 48 processors, the efficiencies of the combined coarse/fine-grain approach with either one pass or two are identical for the pentapeptide. Beyond 48, the single pass approach should become the more efficient of the two. Using a single pass, no individual subsystem calculation will be assigned more than 24 processors until we reach nearly 90 processors. It is reasonable to expect that we will maintain an efficiency very much similar to that seen with the two pass approach over 48 processors were we to attempt a single pass over 90 processors.

Figures 2 and 3 show the speedups obtained for the heptapeptide and the nonapeptide using partitioning scheme A. Exploiting fine-grain parallelism only, overall efficiency is little affected upon going to these larger systems. This is to be expected. The computational burden of the individual subsystem calculations themselves are little affected by the overall system size and the ratio $\frac{T_{\text{parallel}}}{T_{\text{serial}}}$ remains virtually unchanged. For the combined coarse/fine-grain approach with a single pass, the speedups are not as good as those seen for the pentapeptide on 48 or less processors. The processor to subsystem ratios are smaller and optimal load balancing becomes considerably harder to achieve. On 48 processors, the heptapeptide has a speedup of 46.3, as opposed to 47.8 for the pentapeptide. The speedup for the nonapeptide falls even further, to 45.6. However, for this last system, it will be easier to achieve proper load balancing once more processors are added, and we would expect to get near ideal speedups with up to 170 processors. It would only be at this point that the larger individual subsystem calculations would start having more than 24 processors assigned to them and that Amdahl's law would thus begin to take its toll on their efficiencies. If two passes are used, drops in efficiency are not really observed upon going to larger systems. On 48 processors, speedups of 48.3 and 47.5 are seen, respectively, for the heptapeptide and nonapeptide. Within any pass, the processor to subsystem ratio is sufficiently large enough to achieve adequate load balancing. In fact, better results are obtained for the heptapeptide than the pentapeptide, despite the smaller processor to subsystem

ratios. This may result from the fact that less processors are assigned to each subsystem calculation. For the pentapeptide, the first pass assigns 24 processors to each of the two largest subsystems. For the heptapeptide, the first pass assigns 17 processors to the largest subsystem, 16 to the second largest, and 15 to the third largest. In the second pass, the largest number of processors assigned to any one subsystem is 21 for the pentapeptide and 19 for the heptapeptide. The effects of Amdahl's law within the individual subsystem calculations are therefore less dramatic when working with the heptapeptide. The effects of Amdahl's law are even less dramatic within the nonapeptide calculation. Nevertheless, the nonapeptide has a somewhat smaller speedup, and it is no doubt due to its smaller processor to subsystem ratio and its subsequent failure to achieve a better load balance when assigning processors to subsystems.

Figure 4 shows the speedups obtained for the heptapeptide using partitioning scheme B. Though Table 1 indicates that the use of smaller subsystems makes scheme B marginally more efficient than scheme A when running on a single processor, it is evident that scheme B is not preferred when running on a large cluster of processors. When using fine-grain parallelization only, a speedup of only 21.4 is obtained over 48 processors. The increase in speedup between 40 and 48 processors is so marginal that it is doubtful that a speedup greater than 22 would happen no matter how many processors were made available. This is considerably worse than the speedups seen for the heptapeptide using partitioning scheme A. But this is to be expected since the subsystem calculations within scheme B are all much smaller than within scheme A. The fine-grain approach has considerable difficulty achieving high efficiency since each subsystem problem requires very little CPU time, and the overhead involved with the serial code at the beginning of each calculation begins to play a more important role. This will not change no matter how large the overall system becomes. When using the combined coarse/fine-grain approaches, the results are not nearly as bad. They are, however, still not nearly as good as those seen with scheme A. The reason for the decrease is due to the fact that the processor to subsystem ratio is considerably lower within scheme B than it is within scheme A. When performing a single pass, a considerable improvement in efficiency, 79.3% to 89.9%, is seen upon going from 40 to 48 processors. This indicates that overall efficiency is still highly sensitive to the processor to subsystem ratio and many more processors are still required before calculations using partitioning scheme B achieve their greatest efficiency. Unfortunately, the available resources do not allow us to test out this hypothesis. However, employing partitioning scheme B, and assuming that Amdahl's law will only begin to hamper the efficiency of a single subsystem calculation when 12 processors are assigned to it, near linear speedups should be observed out to nearly 140 processors with the combined coarse/fine-grain approach using a single pass. Within scheme A, near linear speedups are only projected out to roughly 125 processors, assuming Amdahl's law becomes a problem when 24 subsystems are assigned to any one of its larger subsystem calculations. And after taking into account the fact that scheme B provided a slightly lower overall time when run on a single processor, scheme B should become the method of choice when such a number of processors does become available.

4 Conclusion

These preliminary benchmark calculations (the actual results on up to 48 processors and our extrapolations beyond this number) have established that a DAC philosophy within a quantum mechanical program can be exploited to achieve high efficiencies on a very large number of processors. However, the precise method of choice depends entirely on the number of processors currently available. With a small number of processors, the combined coarse/fine-grain schemes are not as efficient as the straightforward fine-grain approach. However, when more processors are added, the combined coarse/fine-grain schemes become far more efficient. Meanwhile, the poor scalability of the straightforward fine-grain scheme becomes apparent. Multiple passes are preferred when the processor to subsystem ratio is still relatively small. Better load balancing is thus achieved within the individual passes. However, when proceeding to larger numbers of processors, it is best to reduce the number of passes so that as few processors as possible need be assigned to any one subsystem calculation. The benchmarks of our various parallelization schemes provide us with insights into how many processors are required before one particular approach becomes more efficient than another. These results may be used to design a section of computer code that will automatically determine which approach is preferred under the conditions in which the program is currently being run. This will allow us to achieve near linear speedups on any number of processors without resorting to user intervention. We are in the process of doing this within DeFT. Also, the relative insensitivity of the total CPU time to the particular partitioning scheme being used provides us with an extra degree of flexibility if the ratio of processors to subsystems is either too small or too large.

Our preliminary results were obtained for relatively small systems, which are a more severe test of a code's scalability over a large number of processors than large systems. Calculations on much larger molecules, with many more subsystems, are planned (memory constraints, in some of DeFT's modules unrelated to the DAC procedures, must first be addressed). For this same reason, linear systems were studied so as to establish linear scaling with as small systems as possible. The same would have been achieved for 3D systems, though much larger systems would have had to be studied before linear scaling became evident. With the number of processors available in this study (48), adequate load balancing for much larger systems can be achieved by performing multiple passes. This is a straightforward extension of the combined coarse/fine- grain approach with two passes that has been extensively studied in this report. However, when more processors become avail-

able, we should be able to achieve near linear speedups on up to several hundreds, if not thousands, of processors. We can easily make a reliable estimate of the number of processors that can be used in an efficient manner. We need only find the point at which any one subsystem in the combined coarse/fine-grain approach with a single pass will be assigned more than N processors, where N is the number of processors where we feel that the efficiency of the straightforward fine-grain approach will begin to suffer appreciably under Amdahl's law. The scalability of the combined coarse/fine-grain approach will thus be greatly extended each time we increase N's value, as more processors can now be efficiently exploited in each and every subsystem calculation. Slight improvements in the fine-grain parallelization can thus bring about huge improvements within our new scheme.

We have chosen to concentrate this paper on the DAC fit of the electronic density. However, the combined coarse/fine-grain approach can be extended, just as well, to the DAC fit of the exchange-correlation potential [17] and the DAC construction of the electronic density [11, 12, 43]. The DAC philosophy should also be amenable to evaluate energy gradients as well [12]. This work is now under way. It is also important to note that the DAC partitioning scheme used for the fit of the electronic density need not necessarily be the same as those chosen for other DAC procedures. A more beneficial partitioning scheme may be adopted for these. Ultimately, the efficiency of any combined coarse/fine-grain approach will depend entirely on how efficiently the individual subsystem calculations can be carried out using fine-grain parallelization only. The less efficiently this can be done, the more important it becomes to exploit a DAC method's inherent ability to establish coarse-grain parallelism. This, combined with the fact that a DAC algorithm can provide linear scaling in total CPU time with respect to overall system size, means that our DAC combined coarse/fine-grain approach is ideally suited for applications on truly large molecular systems using today's massively parallel supercomputers.

Acknowledgements. We would like to thank Cray Research for donating time on their Cray T3E system. We wish to thank the Natural Sciences and Engineering Research Council of Canada and the University of Ottawa for financial support. A research grant from the Merck Frosst Centre for Therapeutic Research is also gratefully acknowledged.

References

1. Strain MC, Scuseria GE, Frisch MJ (1996) Science 271: 51
2. Stratmann RE, Scuseria GE, Frisch MJ (1996) Chem Phys Lett 257: 213
3. Burant JC, Scuseria GE, Frisch MJ (1996) J Chem Phys 105: 8969
4. Xu CH, Scuseria GE (1996) Chem Phys Lett 262: 219
5. Millam JM, Scuseria GE (1997) J Chem Phys 106: 5569
6. White CA, Johnson BG, Gill PMW, Head-Gordon M (1994) Chem Phys Lett 230: 8
7. White CA, Johnson BG, Gill PMW, Head-Gordon M (1996) Chem Phys Lett 253: 268
8. Schwegler E, Challacombe M (1996) J Chem Phys 105: 2726
9. Challacombe M, Schwegler E (1997) J Chem Phys 106: 5526
10. Kutteh R, Apra E, Nichols J (1995) Chem Phys Lett 238: 173
11. Yang WT (1992) J Mol Struct (Theochem) 87: 461
12. Yang WT, Lee TS (1995) J Chem Phys 103: 5674
13. Lee TS, York DM, Yang WT (1996) J Chem Phys 105: 2744
14. Dixon SL, Merz KM (1996) J Chem Phys 104: 6643
15. Gallant RT, St-Amant A (1996) Chem Phys Lett 256: 569
16. Goh SK, St-Amant A (1997) Chem Phys Lett 264: 9
17. Goh SK, Gallant RT, St-Amant A, Int J Quantum Chem (accepted)
18. Stewart JJP (1996) Int J Quantum Chem 58: 133
19. Galli G, Parrinello M (1992) Phys Rev Lett 69: 3547
20. Mauri F, Galli G, Car R (1993) Phys Rev B 47: 9973
21. Li XP, Nunes RW, Vanderbilt D (1993) Phys Rev B 47: 10891
22. Daw MS (1993) Phys Rev B 47: 10895
23. Stechel EB, Williams AR, Feibelman PJ (1994) Phys Rev B 49: 10088
24. Hernandez E, Gillan MJ (1994) Phys Rev B 51: 10157
25. Chen X, Langlois JM, Goddard WA III (1995) Phys Rev B 52: 2348
26. Ordejon P, Drabold DA, Martin RM, Grumbach MP (1995) Phys Rev B 51: 1456
27. Kohn W (1996) Phys Rev Lett 76: 3168
28. Kendall RA, Harrison RJ, Littlefield RJ, Guest MF (1995) In: Lipkowitz KB, Boyd DB (eds) Reviews in computational chemistry, vol 6. VCH, New York, pp 209–316
29. Bernholdt DE, Apra E, Fruchtl HA, Guest MF, Harrison RJ, Kendall RA, Kutteh RA, Long X, Nicholas JB, Nichols JA, Taylor HL, Wong AT, Fann GI, Littlefield RJ, Nieplocha J (1995) Int J Quantum Chem Symp 29: 475
30. Schmidt MW, Baldridge KK, Boatz JA, Elbert ST, Gordon MS, Jensen JH, Koseki S, Matsunaga N, Nguyen KA, Su S, Windus TL, Dupuis M, Montgomergy JA Jr (1993) J Comput Chem 14: 1347
31. Feyereisen M, Kendall RA (1993) Theor Chim Acta 84: 289
32. Colvin ME, Janssen CL, Whiteside RA, Tong CH (1993) Theor Chim Acta 84: 301
33. Dunlap BI, Connolly JWD, Sabin JR (1979) J Chem Phys 71: 3396
34. Dunlap BI (1996) Int J Quantum Chem 58: 123
35. Hehre WJ, Radom L, Schleyer P, von R, Pople JA (1986) Ab initio molecular orbital theory. Wiley, New York
36. The complete source code and documentation for an earlier version of the DeFT program package may be downloaded from http://www.chem.uottawa.ca/DeFT.html
37. Andzelm J, Wimmer E (1992) J Chem Phys 96: 1280
38. Salahub D, Fournier R, Mlynarski P, Papai I, St-Amant A, Ushio J (1991) In: Labanowski JK, Andzelm JW (eds) Density functional methods in chemistry. Springer, Berlin Heidelberg New York, pp 77–100
39. Gropp W, Lusk E, Skjellum A (1995) Using MPI: portable parallel programming with the message-passing interface. MIT Press, Cambridge
40. Obara S, Saika A (1986) J Chem Phys 84: 3963
41. (a) Stewart JJP (1989) J Comput Chem 10: 209; (b) Stewart JJP (1989) J Comput Chem 10: 221
42. Godbout N, Salahub DR, Andzelm J, Wimmer E (1992) Can J Chem 70: 560
43. Goh SK, St-Amant A (1997) Chem Phys Lett 274: 429