# Integral-direct and parallel implementation of the CCSD(T) method: algorithmic developments and large-scale applications

László Gyevi-Nagy,[*] Mihály Kállay, and Péter R. Nagy[*]

*Department of Physical Chemistry and Materials Science, Budapest University of Technology and Economics, H-1521 Budapest, P.O.Box 91, Hungary*

E-mail: gyevi-nagy.laszlo@mail.bme.hu; nagyrpeter@mail.bme.hu

**Abstract**

A completely integral-direct, disk I/O and network traffic economic coupled-cluster singles, doubles, and perturbative triples [CCSD(T)] implementation has been developed relying on the density-fitting approximation. By fully exploiting the permutational symmetry the presented algorithm is highly operation-count and memory efficient. Our measurements demonstrate excellent strong scaling achieved via hybrid MPI/OpenMP parallelization and a highly competitive, 60-70% utilization of the theoretical peak performance on up to hundreds of cores. The terms whose evaluation time becomes significant only for small- to medium-sized examples have also been extensively optimized. Consequently, high performance is also expected for systems appearing in extensive data sets used, e.g., for density functional or machine learning parametrizations, and in calculations required for certain reduced-cost or local approximations of CCSD(T), such as in our local natural orbital scheme [LNO-CCSD(T)]. The efficiency

[*]To whom correspondence should be addressed

of this implementation allowed us to perform some of the largest CCSD(T) calculations ever presented for systems of 31-43 atoms and 1037-1569 orbitals using only 4-8 many-core CPUs and 1-3 days of wall time. The resulting 13 correlation energies and the 12 corresponding reaction energies and barrier heights are added to our previous benchmark set collecting reference CCSD(T) results of molecules at the applicability limit of current implementations.

## 1 Introduction

The coupled-cluster (CC)[1–4] family of methods has become one of the most accurate and versatile theoretical tools to simulate molecules and solids at the atomic scale. The size-extensivity and the systematically improvable character of the CC methods are highly advantageous for computing energies and other molecular properties.[5–7] For single reference cases, assuming that sufficient convergence is achieved also for the single-particle orbital basis sets, the CC model with single and double excitations (CCSD) augmented with perturbative triples correction [CCSD(T)][8] is regarded as the "gold standard" of quantum chemistry. For most properties, CCSD(T) results are expected to agree with experiments within their respective uncertainties.[2,4] The price of such beneficial properties, at least for conventional implementations, is the steep, sixth- and seventh-power-scaling operation count complexity for CCSD and CCSD(T), respectively, and fourth-power-scaling data complexity for both methods. This usually restricts the reach of conventional CCSD(T) codes to systems of up to 20-25 atoms.

A straightforward way to extend this limitation is to employ the tools of modern high-performance computing, such as parallel execution. Several recent CC implementations employ Message Passing Interface (MPI) to distribute the workload between multiple compute nodes.[9–20] Efficient strong scaling performance was demonstrated for CCSD(T) up to hundreds or thousands of processor cores with the PQS,[16] GAMESS,[13] and MOLCAS[12] packages, while the implementations in the MPQC,[17,19] ACESIII,[14] Aquarius,[18] FHI-aims,[20] and NWChem[10,11] program suits are shown to scale well even up to many thousands of cores. Compared to the availability of well-parallelized implementations only a relatively limited number of large-scale CCSD(T) applications were presented for systems including up to 1000 orbitals.[13,14,19–21] To our knowledge only a few conventional CCSD(T) calculations reached 1500 orbitals to date. The CCSD(T)/aug-cc-pVQZ calculation of Janowski and Pulay performed for the benzene dimer involved 1512 orbitals,[16] but the $C_{2h}$ symmetry was fully exploited to reduce the tremendous operation count and storage demand. Xantheas and coworkers employed the NWChem package to obtain CCSD(T)/aug-cc-pVTZ energies for water clusters including up to 17 molecules and 1564 orbitals, for which they employed 120,000 compute cores.[22]

Considering the steep-scaling arithmetic demand of CCSD(T) the use of multi-node parallelism or accelerators[19,23–25] alone can only moderately extend the applicability limits of CCSD(T). Nevertheless, many approximations have been developed aiming to preserve the intrinsic accuracy of CCSD(T), while reducing the scaling or prefactor of the operation count and/or space complexity. Regarding the latter, the storage and communication challenges posed by the two-electron four-center electron repulsion integrals (ERIs) can be significantly decreased via density-fitting (DF, often also referred to as resolution-of-identity),[19,20,26–28] Cholesky decomposition,[12,26,29] or alternative tensor factorization[30–36] approaches. For instance, Peng, Valeev, and coworkers recently presented a DF-CCSD implementation where all four-center molecular orbital (MO) ERIs with more than two unoccupied indices are reassembled when needed to avoid the allocation and communication of fourth-power-scaling arrays with high prefactor.[19] Scheffler, Shen, and coworkers also chose to reassemble the integrals with four unoccupied MO indices and the ERIs with three unoccupied indices which are not available in the local memory in each iteration as needed.[20] Compared to this repeated integral assembly it is negligible to also recreate the Coulomb integrals with fewer than three virtual indices in each iteration. In recognition of this, DePrince and Sherrill showed that the faster, DF-based atomic orbital (AO) to MO integral transformation algorithm can be utilized to design an efficient CCSD algorithm using the $t_1$-transformed Hamiltonian,[28] which technique is often utilized also in the context of second- and higher-order CC theories employed for excited-state calculations.[37–41] The DF approximation is thus highly useful to break down memory and bandwidth bottlenecks, but it does not noticeably reduce the operation count of CCSD(T), only one of the less expensive, sixth-power-scaling terms can be accelerated via more efficient factorization.[28]

Active development is also aimed towards reduced-cost and reduced-scaling CCSD(T) approximations. For instance, orbital transformation techniques[12,42–44] relying on natural orbitals (NOs) constructed at the level of second-order Møller–Plesset (MP2) perturbation theory can effectively compress the unoccupied MO space while retaining high accuracy. Recent cost-reduction efforts considered promising ideas utilizing, for example, sparsity exploitation,[45] mixed single and double precision operations,[46] or stochastic approaches.[47,48] The most successful methods to date combine multiple strategies, such as DF and NOs, with local approximations.[21,49–51] For instance, we have recently demonstrated with our local natural orbital (LNO)[21,52–57] scheme that, while retaining high accuracy, LNO-CCSD(T) calculations can be performed up to a few thousand atoms and 45,000 orbitals even with a single CPU.[21,57] As CC methods with local and NO approximations become increasingly accepted and trusted in the literature,[58] tightly converged approximations have mostly taken over the role of massively-parallel implementations in large-scale CCSD(T) applications.

Considering this shift we identify three use cases and the corresponding algorithmic properties for which our optimization efforts are aimed at. First, extensive conventional CCSD(T) calculations are still vital to compute references for the accuracy assessment of reduced-cost approximations.[21] Second, as current local CC methods are not necessarily more efficient for small systems, canonical CC implementations often provide benchmark data for the construction of density functional approximations[59,60] or for the training step in machine learning approaches.[61–68] Such data sets often collect thousands[59,60,67,68] or even hundreds of thousands[62,69] of CC results obtained for relatively small systems, say below about a dozen non-hydrogen atoms. Third, certain local CC approximations,[70–75] including also the LNO-CC scheme,[21,52–57] obtain the final CC correlation energy from the contributions of a number of independent, smaller-scale "domain" CC computations, for which often conventional, or slightly modified CC implementations are invoked.

We report a CCSD(T) algorithm and implementation designed for the above three goals. For efficient execution also on computer clusters or in a cloud compute environment where local disks are not available or are relatively small, the algorithm is completely integral-direct utilizing an effective DF-based and batched approach for the evaluation of $t_1$-transformed four-center integrals.[28,76] At the same time, in preparation for moderate per-node memory cases only the absolutely necessary four-dimensional tensors are kept during the CCSD iteration (doubles CC amplitudes and residuals). Low per-node memory footprint is also ensured by a shared-memory intra-node (OpenMP) parallelization strategy and symmetry-packed array formats. Consequently, disk input/output (I/O) and network use are completely avoided within a CCSD iteration and the evaluation of the (T) correction. In order to minimize the operation count the highest possible permutational symmetry is exploited throughout and, an additional layer of inter-node (MPI) parallelization is implemented on top of OpenMP. We have found that, for a significant portion of the target use cases, when the virtual/occupied ($n_\mathrm{v}/n_\mathrm{o}$) ratio is in the range of 5–10, the usual assumption that the $\mathcal{O}(n_\mathrm{v}^4 n_\mathrm{o}^2/4)$-scaling particle-particle ladder (PPL) term dominates the cost of CCSD does not hold. In such cases the remaining $\mathcal{O}(4n_\mathrm{v}^3 n_\mathrm{o}^3)$-scaling terms also take comparable time, for which a relatively limited attention have been devoted in the literature. Such cases occur with relatively small, e.g., double-$\zeta$ quality basis sets, small molecules with only a few hundred MOs, or when the virtual basis is successfully compressed via NO approximations, such as in our LNO-CC scheme.[21,57] Thus, we also developed hand-optimized, memory-efficient, in-core, and well-parallelized algorithms to all terms appearing in the $t_1$-transformed CCSD equations.[28,76] Our recent OpenMP parallel (T) algorithm[56] was also updated to match the minimum memory demand of the new CCSD code, while making it also MPI/OpenMP parallel and free from disk I/O and network use.

Some of our algorithmic choices are inspired by the $t_1$-transformed DF-CCSD implementation of DePrince and Sherrill[28] with notable improvements regarding the MPI parallelism, optimized non-PPL terms in CCSD, more than three times smaller minimum memory requirement, and the rigorous avoidance of disk I/O during a CCSD iteration step. The most recent CCSD(T) programs of the MPQC[19] and FHI-aims[20] packages are also similar to ours in some aspects, like the DF-based (semi-)integral-direct execution and high parallel efficiency. The main deviation in our algorithm design is that here the full permutational symmetry is retained for all terms of the CCSD equations, while refs 19 and 20 employ (partially) redundant solutions for some of the contractions and data structures appearing, for instance, in the PPL term. Since the packing/unpacking operations required for the efficient matrix-matrix multiplication-based formulation of the most demanding contractions do not scale well with the number of cores, the solutions of Peng et al.[19] and Shen et al.[20] are expected to perform better with thousands of cores. Since massive parallelism is already provided by the completely independent evaluation of numerous moderate-sized CCSD(T) problems in the second and third target applications, and we were able to perform large-scale CCSD(T) calculations in the 1000-1500 orbital region with a few hundred cores, the presented algorithm matches our goals well. It is also worth noting that while, the scheme of Scheffler and coworkers[20] and ours employ a hybrid MPI/OpenMP strategy and compute the (T) correction via the "*ijkabc*" approach,[77,78] Valeev and coworkers[19] implemented a purely MPI-based framework with extensions to graphics processing units and utilization of many integrated cores, and evaluate an "*abcijk*" type (T) expression.[78,79] Additionally, refs 19 and 20 rely on a distributed-memory model suitable for massive-parallelism, whereas our low-memory, batched, and fully integral-direct algorithms provide more bandwidth-economic replicated memory solutions for systems of up to about 2000 orbitals.

After providing the theoretical background in Section 2 and the detailed introduction of the novel algorithms in Section 3, extensive benchmarks are presented for both intra-node and inter-node parallel scaling in Section 5. The measurements representing typical use cases, e.g., appearing within an LNO-CCSD(T) calculation, demonstrate excellent strong scaling comparable to that of state-of-the-art implementations,[19,20,28,80] while close-to-ideal absolute efficiency is also displayed in terms of peak performance utilization. Section 6 presents illustrative applications at the applicability limit of current CCSD(T) codes for three reactions taken from recent mechanistic studies.[81–83] The resulting 13 correlation energies and 12 reaction energies and barrier heights are added to our recent 26-item correlation energies of medium-sized systems (CEMS26) compilation[21] and will be employed for the accuracy assessment of CCSD(T) approximations.

## 2 Theoretical background

### 2.1 CCSD energy and amplitude equations

Assuming a closed-shell reference determinant consisting of spatial orbitals, the CCSD energy can be written as

$$E_{\mathrm{CCSD}} = 2 \sum_{ia} f_{ai}\, t_i^a + \sum_{ijab} L_{ij}^{ab}\, \left(t_{ij}^{ab} + t_i^a t_j^b\right), \tag{1}$$

where indices $i,\ j,\ \ldots (a,\ b,\ \ldots)$ denote occupied (virtual) orbitals, $t_i^a$ and $t_{ij}^{ab}$ represent the singles and doubles amplitudes, respectively, $f_{pq}$ are elements of the Fock matrix with general orbital indices $p$, $q$, …, and

$$L_{ij}^{ab} = 2(ai|bj) - (aj|bi), \tag{2}$$

with $(pq|rs)$ as a two-electron repulsion integral in the Mulliken convention.

The equations determining the excitation amplitudes are derived by projecting the Schrödinger equation on the space of excited determinants, e.g.,

$$\left\langle\, \Phi_{ij}^{ab} \,\middle|\, \mathrm{e}^{-(T_1+T_2)}\, H\, \mathrm{e}^{T_1+T_2} \,\middle|\, \Phi_0 \right\rangle = 0. \tag{3}$$

Here, $T_1$ and $T_2$ are the cluster operators corresponding to single and double excitations, $\left|\Phi_0\right\rangle$ and $\left|\Phi_{ij}^{ab}\right\rangle$ denote the Hartree–Fock (HF) and a doubly excited determinant, and $H$ is the Hamiltonian operator of the system.

The four-center ERIs, collected in matrix $\mathbf{K}$ with elements $K_{pq,rs} = (pq|rs)$, will be evaluated relying on the DF approximation[84–86] as

$$\mathbf{K} = \mathbf{I}\mathbf{V}^{-1}\mathbf{I} = \mathbf{J}\mathbf{J}^{\mathrm{T}}, \tag{4}$$

where $\mathbf{I}$ collects the $I_{pq,Q} = (pq|Q)$ three-center two-electron integrals with $Q$ indexing the functions of the auxiliary basis. The two-center two-electron integral matrix of the auxiliary basis functions, with elements $V_{P,Q} = (P|Q)$, is factorized using Cholesky-decomposition as $\mathbf{V} = \mathbf{L}\mathbf{L}^{\mathrm{T}}$, where $\mathbf{L}$ is a lower triangular matrix. After the decomposition the inverse of $\mathbf{V}$ can be recast as $\mathbf{V}^{-1} = \left(\mathbf{L}^{-1}\right)^{\mathrm{T}}\mathbf{L}^{-1}$, and the inverse of $\mathbf{L}$ can be contracted with $\mathbf{I}$, forming $\mathbf{J}$ as

$$\mathbf{J} = \mathbf{I}(\mathbf{L}^{-1})^{\mathrm{T}}. \tag{5}$$

The CCSD amplitude equations can greatly be simplified by absorbing the effect of the single excitations into the Hamiltonian[87] via defining

$$\hat{H} = \mathrm{e}^{-T_1}\ H\ \mathrm{e}^{T_1}. \tag{6}$$

The above $t_1$-transformed Hamiltonian is expressed by the elements of the corresponding transformed Fockian ($\hat{f}_{pq}$) and the elements of the transformed three-center ERIs ($\hat{J}_{pq}^{Q}$) for which eqs 29-31 of the Appendix collect the explicit expressions.

Substituting the $t_1$-transformed Hamiltonian of eq 6 into the CC equations, the singles

($R_i^a$) and doubles ($R_{ij}^{ab}$) residuals can be written as[28,87]

$$R_i^a = \hat{f}_{ia} + A_i^a + B_i^a + C_i^a \tag{7}$$

$$\begin{aligned} R_{ij}^{ab} &= \sum_P \hat{J}_{ai}^P \hat{J}_{bj}^P + A_{ij}^{ab} + B_{ij}^{ab} \\ &\quad + P_{ij}^{ab}\left(\frac{1}{2} C_{ij}^{ab} + C_{ji}^{ab} + D_{ij}^{ab} + E_{ij}^{ab} + G_{ij}^{ab}\right), \end{aligned} \tag{8}$$

where the permutation operator, $P_{ij}^{ab}$, is defined as $P_{ij}^{ab}\binom{ab}{ij} = \binom{ab}{ij} + \binom{ba}{ji}$. The intermediates appearing in the $t_1$-transformed CCSD residual equations[28,76] read as

$$A_i^a = \sum_{kcd} u_{ki}^{cd} \sum_P \hat{J}_{kc}^P \hat{J}_{ad}^P \tag{9}$$

$$B_i^a = -\sum_{klc} u_{kl}^{ac} \sum_P \hat{J}_{ki}^P \hat{J}_{lc}^P \tag{10}$$

$$C_i^a = \sum_{kc} \hat{f}_{kc}\, u_{ik}^{ac} \tag{11}$$

$$A_{ij}^{ab} = \sum_{cd} t_{ij}^{cd} \sum_P \hat{J}_{ac}^P \hat{J}_{bd}^P \tag{12}$$

$$B_{ij}^{ab} = \sum_{kl} t_{kl}^{ab} \left(\sum_P \hat{J}_{ki}^P \hat{J}_{lj}^P + \sum_{cd} t_{ij}^{cd} \sum_P \hat{J}_{kc}^P \hat{J}_{ld}^P\right) \tag{13}$$

$$C_{ij}^{ab} = -\sum_{kc} t_{kj}^{bc} \left(\sum_P \hat{J}_{ki}^P \hat{J}_{ac}^P - \frac{1}{2}\sum_{ld} t_{li}^{ad} \sum_P \hat{J}_{kd}^P \hat{J}_{lc}^P\right) \tag{14}$$

$$D_{ij}^{ab} = \frac{1}{2}\sum_{kc} u_{jk}^{bc} \left(\hat{L}_{ic}^{ak} + \frac{1}{2}\sum_{ld} u_{il}^{ad} \hat{L}_{dc}^{lk}\right) \tag{15}$$

$$E_{ij}^{ab} = \sum_{c} t_{ij}^{ac} \left(\hat{f}_{bc} - \sum_{kld} u_{kl}^{bd} \sum_P \hat{J}_{ld}^P \hat{J}_{kc}^P\right) \tag{16}$$

$$G_{ij}^{ab} = -\sum_{k} t_{ik}^{ab} \left(\hat{f}_{kj} + \sum_{lcd} u_{lj}^{cd} \sum_P \hat{J}_{kd}^P \hat{J}_{lc}^P\right), \tag{17}$$

where the following shorthand notations are also exploited:

$$u_{ij}^{ab} = 2\,t_{ij}^{ab} - t_{ij}^{ba} \tag{18}$$

$$\hat{L}_{rs}^{pq} = \sum_P \left(2\hat{J}_{pr}^P\,\hat{J}_{qs}^P - \hat{J}_{ps}^P\,\hat{J}_{qr}^P\right). \tag{19}$$

## 2.2 The perturbative (T) correlation energy

The closed shell (T) energy expression in the canonical orbital basis can be written as[8,79]

$$E_{(\mathrm{T})} = \frac{1}{3}\sum_{ijk}\sum_{abc}(4W_{ijk}^{abc} + W_{ijk}^{bca} + W_{ijk}^{cab})(V_{ijk}^{abc} - V_{ijk}^{cba})/D_{ijk}^{abc}, \tag{20}$$

where $D_{ijk}^{abc} = f_{ii} + f_{jj} + f_{kk} - f_{aa} - f_{bb} - f_{cc}$ is the energy denominator with $f_{pp}$ as a diagonal element of the Fock matrix. The $W$ and $V$ intermediates are defined as

$$W_{ijk}^{abc} = P_{ijk}^{abc}\left(\sum_d (bd|ai)\;t_{kj}^{cd} - \sum_l (ck|jl)\;t_{il}^{ab}\right) \tag{21}$$

and

$$V_{ijk}^{abc} = W_{ijk}^{abc} + (bj|ck)t_i^a + (ai|ck)t_j^b + (ai|bj)t_k^c. \tag{22}$$

Here, the permutation operator, $P_{ijk}^{abc}$ is introduced as

$$P_{ijk}^{abc}\begin{pmatrix}abc\\ijk\end{pmatrix} = \begin{pmatrix}abc\\ijk\end{pmatrix} + \begin{pmatrix}acb\\ikj\end{pmatrix} + \begin{pmatrix}cab\\kij\end{pmatrix} + \begin{pmatrix}cba\\kji\end{pmatrix} + \begin{pmatrix}bca\\jki\end{pmatrix} + \begin{pmatrix}bac\\jik\end{pmatrix}. \tag{23}$$

Let us note that eqs 21 and 22 do not depend on the transformed $\hat{\mathbf{f}}$ and $\hat{\mathbf{J}}$ since the evaluation of the (T) correction cannot be accelerated by the $t_1$-transformation.

The sixfold permutational symmetry of the intermediate quantities can be utilized to evaluate the perturbative triples energy expression. When working in the canonical orbital basis of a closed-shell system, two suitable energy expressions can be derived[77,78] for this end. Here, we limit our discussion to the case when the outermost loops run over the restricted occupied indices. For the explicit energy formula corresponding to this so-called "*ijkabc*" algorithm, we refer to eq 5 of ref 56.

## 3 Algorithm

This section presents algorithmic developments and parallelization efforts devoted to the CCSD iteration and then to the perturbative triples correction. According to our target application types the following priorities are set for the new DF-CCSD(T) code: minimal memory footprint, optimal operation count in the sixth- and seventh-power-scaling terms, negligible hard disk and network use, and good parallel scaling.

### 3.1 CCSD algorithm: data structures and integral assembly

Dealing with limited per node memory and data traffic bandwidths is one of the cornerstones of current algorithm design. Hence we prioritize the minimization of data storage and hard disk/network use to increase CPU efficiency. For that purpose our aim is to minimize the storage requirement with minor sacrifices regarding the operation count so that all necessary quantities will be available in local memory or can be effectively recomputed when needed for as large orbital spaces as possible.

Since the best way to effectively factorize or compress the doubles amplitudes and residuals on a production level is still under active development,[32,35,36] they are currently kept in four-dimensional tensors. All other potentially sizable quantities are either factorized (ERIs) or split into at most $\mathcal{O}(N^3)$-scaling batches (all intermediates). Thus, the only fourth-power-scaling arrays, stored in their permutational symmetry-packed form (c.f., $t_{ij}^{ab} = t_{ji}^{ba}$), take $16\, n_\mathrm{v}^2\, n_\mathrm{o}(n_\mathrm{o}+1)/2$ bytes in double-precision, where $n_\mathrm{v}$ and $n_\mathrm{o}$ denote the number of virtual and occupied orbitals, respectively. The benefit is a factor of two saving in the size of the largest arrays compared to the more convenient unpacked format. The drawback is that repeated packing/unpacking operations are needed during each CCSD iterations, which is, however, a low operation count task, hence it can be traded for memory efficiency.

Our integral-direct, DF approach brings down the scaling of all integral tensor sizes from $\mathcal{O}(N^4)$ to $\mathcal{O}(N^3)$, with $N = n_\mathrm{o} + n_\mathrm{v}$ but necessitates the assembly of the required two-electron integrals in every iteration. Since the corresponding operation is only of $\mathcal{O}(N^5)$-scaling and can be handled with efficient, well-parallelizable tasks, this strategy effectively trades space requirement for increased operation count, as also recognized previously in similar contexts.[19,20,28] The alternative would be to store the extensive $\mathcal{O}(N^4)$-scaling arrays on hard disk or use distributed memory and network communication, both of which are highly challenging to perform efficiently with good scaling. The cubic-scaling DF integral tensors do not cause memory or data traffic bottlenecks, and we found that the reduced I/O demand and the better parallelizability of the integral assembly compensate well for the increased operation count of the DF approximation.

In order to minimize the storage requirement of the intermediates, we applied a batched evaluation strategy to each term in the residual equations. Namely, the two-electron integrals and intermediates are calculated in at most cubic-scaling blocks. Then their contribution is immediately cumulated into the residuals, and the intermediate data is discarded. This solution is again necessary to keep the $\mathcal{O}(N^4)$-scaling memory demand at minimum. In turn it leads to more complicated algorithms since the batch dimensions have to be set as large as possible to avoid performance loss when invoking vendor optimized BLAS routines for the corresponding contractions.

It is worth noting that the above memory reduction techniques do not increase the arithmetic complexity of the algorithm, that is, the $\mathcal{O}(N^6)$-scaling is retained for CCSD. Compared to that the $\mathcal{O}(N^4)$ and $\mathcal{O}(N^5)$ complexity of unpacking the doubles amplitudes and the integral assembly is affordable. It is still worth decreasing the number of integral assemblies within an iteration as much as possible. As the $t_1$-transformation incorporates the effect of the singles amplitudes into the Hamiltonian, all terms containing the singles amplitudes in the conventional CCSD equations are merged into terms reminiscent of the ones that appear in the coupled-cluster doubles (CCD) equations.[28] This simplification makes the algorithm more amenable to hand-optimization and parallelization. One apparent benefit is that one of the sixth-power-scaling terms can be more efficiently refactorized leading to a small prefactor decrease in the overall CCSD algorithm.[28] Additionally, a formulation can be chosen that does not require the assembly of the three-external four-center integrals at all, and the all virtual four-center integrals are only needed once per iteration.[28] In turn, the $t_1$-transformation itself would correspond to an $\mathcal{O}(N^5)$-scaling transformation of the ERI tensor and the Fock matrix in each iteration if DF or other factorization techniques were not employed. However, once we commit to a fully integral-direct, DF-based algorithm, the overhead of the $t_1$-transformation becomes a negligible, $\mathcal{O}(N^4)$-scaling operation.

In the Appendix we introduce storage and operation effective solutions to carry out the $t_1$-transformation, which deviates from the algorithms of ref 28 in multiple aspects. For instance, we transform the DF integrals from the AO to the MO basis only once and store only the $t_1$-transformed MO integrals corresponding to the singles amplitudes of the actual iteration, while the approach of ref 28 employs a disk-based AO to MO transformation in each iteration.

## 3.2 Algorithms for the CCSD residual equations

After finding a satisfactory solution for the representation of the integrals and wavefunction parameters, we turn our attention to the optimization of wall times within the set constraints. This is demonstrated by the detailed analysis of the individual terms.

First, we consider the supposedly most expensive $\mathcal{O}(N^6)$-scaling term, the $A_{ij}^{ab}$ contribution of eq 12, also known as the particle-particle ladder or PPL term. The total operation count of a naive implementation of this term is $2\,n_\mathrm{v}^4\,n_\mathrm{o}^2$, where we consider both the addition and multiplication operations in order to have accurate floating point operation (FLOP) counts for performance analysis. It is well known that this operation count can be reduced by symmetrizing the corresponding amplitude and ERI tensors,[88,89] which is, however, still challenging to implement with massive parallel scaling.[10,11,19,20] Presently we take advantage of the symmetrization as fourfold and twofold improvement can be achieved for the contraction and the four-external assembly steps, respectively. Following the DF-based formulation of ref 28 the $A_{ij}^{ab}$ term is evaluated as

$$A_{i\leq j}^{ab} = \begin{cases} {}^{(+)}A_{ij}^{ab} + {}^{(-)}A_{ij}^{ab}, & \text{if } a \leq b \\ {}^{(+)}A_{ij}^{ab} - {}^{(-)}A_{ij}^{ba} & \text{otherwise} \end{cases} \tag{24}$$

$$ {}^{(\pm)}A_{ij}^{ab} = \frac{1}{2} \sum_{c\leq d} {}^{(\pm)}I_{cd}^{ab} \; {}^{(\pm)}t_{ij}^{cd} \tag{25}$$

$$ {}^{(\pm)}I_{cd}^{ab} = I_{cd}^{ab} \pm I_{dc}^{ab} = \sum_P \hat{J}_{ac}^P \, \hat{J}_{bd}^P \pm \sum_P \hat{J}_{ad}^P \, \hat{J}_{bc}^P \tag{26}$$

$$ {}^{(+)}t_{ij}^{ab} = t_{ij}^{ab} + t_{ij}^{ba} \left(1 - \delta_{ab}\right) \tag{27}$$

$$ {}^{(-)}t_{ij}^{ab} = t_{ij}^{ab} - t_{ij}^{ba}, \tag{28}$$

where superscripts (+) and (−) denote symmetric and antisymmetric combinations, respectively, and $\delta_{ab}$ stands for the Kronecker delta symbol. Compared to that of ref 28 our algorithm for this term, presented in Algorithm 1, includes several improvements: it is more memory economic, I/O-free, and massively parallel via an OpenMP/MPI strategy, while retaining the use of effective matrix-matrix multiplications. The evaluation of the PPL term

**Algorithm 1** Evaluation of the $A_{ij}^{ab}$ (or PPL) term.

```
1: for $n_{\text{block}} = 1, \lceil (n_{\text{v}}(n_{\text{v}}+1)/2) / n_{\text{B}} \rceil$ // MPI and OpenMP parallel
2:     for $\overline{ab} = (n_{\text{block}} - 1)n_{\text{B}} + 1, n_{\text{block}}\, n_{\text{B}}$
3:         $I_{cd}^{\overline{ab}} = \hat{J}_{c,P}^{a} \left(\hat{J}_{d,P}^{b}\right)^{\dagger}$
4:         ${}^{(\pm)}I_{\overline{cd}\,\overline{ab}} = I_{cd}^{\overline{ab}} \pm I_{dc}^{\overline{ab}}$
5:     end for
6:     ${}^{(\pm)}A_{\overline{ab}\,\overline{ij}} = 1/2 \left({}^{(\pm)}I_{\overline{cd},\overline{ab}}\right)^{\dagger} \, {}^{(\pm)}T_{\overline{cd},\overline{ij}}$
7:     for $a \leq b$:   $R_{ab\overline{ij}} \leftarrow {}^{(\pm)}A_{\overline{ab}\,\overline{ij}}$
8:     for $a > b$:   $R_{ab\overline{ij}} \leftarrow \pm {}^{(\pm)}A_{\overline{ba}\,\overline{ij}}$
9: end for
```

is carried out in batches of restricted virtual index pairs, $\overline{ab}$ (see the loop over $n_{\text{block}}$ in line 1 of Algorithm 1), where $\overline{ab}$ denotes a hyperindex with $a \leq b$, and there is $n_{\text{B}}$ number of $ab$ pairs in a batch. First, the $(ac|bd)$ integrals are assembled for each $\overline{ab}$ hyperindex in the block and stored in array $I_{cd}^{\overline{ab}}$ (line 3), where upper indices are considered fixed. Here, $\hat{J}_{c,P}^{a}$ is an $n_{\text{v}} \times N_{\text{a}}$ array built from the corresponding elements of the three-index integral tensor for a particular value of $a$, and $N_{\text{a}}$ stands for the number of auxiliary functions. Summation over repeated indices, $P$ in this case, is implied throughout this section. Additionally, the comma separating the (hyper)indices of work arrays is used to denote the row and column dimensions appearing in the matrix-matrix multiplications. In line 4 the assembled block of integrals is (anti)symmetrized and then it is immediately discarded. Finally, the entire (anti)symmetrized integral batch is contracted with the packed (anti)symmetric doubles amplitudes $({}^{(\pm)}T)$ and accumulated into the residual array $(R)$.

The main drawback of utilizing the symmetrization in the PPL term is the appearance of multiple symmetry packing operations, which are usually hard to vectorize and parallelize, and bound by the bandwidth of memory operations. Let us note that the symmetrization of the doubles amplitudes needed in line 6 is performed in place to avoid the increase of the fourth-power-scaling memory footprint. Such redundant (un)packing operations naturally alter the efficiency but the reduced operation count of the contraction and assembly steps usually amply compensate this performance loss.

When employing some sort of NO approximation,[42–44] as in the LNO-CC local correlation methods,[21,52–57] or when working with a relatively small, double-$\zeta$ quality AO basis set, the virtual/occupied orbital ratio is too small to assume the cost dominance of the PPL term. For such cases the computation time of the $\mathcal{O}(n_{\text{v}}^3 n_{\text{o}}^3)$- and $\mathcal{O}(n_{\text{v}}^2 n_{\text{o}}^4)$-scaling terms ($B_{ij}^{ab}$, $C_{ij}^{ab}$, and $D_{ij}^{ab}$) also become significant compared to the $\mathcal{O}(n_{\text{v}}^4 n_{\text{o}}^2)$-scaling particle-particle ladder term. This is particularly true if the above symmetrization idea is used to reduce the operation count of the PPL term by a factor of 4. Let us consider an example with the cc-pVDZ basis set or with a average domain LNO orbital space dimensions a the LNO-CC calculation,[21,56,57] where, for a typical organic molecule, the $n_\mathrm{v}/n_\mathrm{o} \approx 5$ and $N_\mathrm{a}/n_\mathrm{v} \approx 4$ dimension ratios occur. Considering only the most demanding terms the operation count of PPL (including the on-the-fly integral assembly) is $n_\mathrm{v}^4\, n_\mathrm{o}^2/2 + n_\mathrm{v}^4\, N_\mathrm{a}$, and the overall FLOP count of the remaining $\mathcal{O}(N^6)$-scaling terms amounts to $8\, n_\mathrm{v}^3\, n_\mathrm{o}^3 + 4\, n_\mathrm{v}^2\, n_\mathrm{o}^4$. Substituting the assumed $n_\mathrm{v}/n_\mathrm{o} \approx 5$ and $N_\mathrm{a}/n_\mathrm{v} \approx 4$ relations the FLOP count ratio of the B, C, and D terms is about $(41\, n_\mathrm{o})/(12.5\, n_\mathrm{o}{+}500)$ relative to the PPL. Thus, with these orbital space dimensions the evaluation of the non-PPL terms becomes more demanding than that of the PPL term even for relatively small systems with more than about 17 occupied orbitals. Consequently, the optimization of the other $\mathcal{O}(N^6)$-scaling terms is equally important for an efficient CCSD algorithm, especially when small or compressed basis sets are used. One could argue that in these cases the (T) correction would still dominate the cost of a CCSD(T) calculation. That argument is, however, not valid for our recent LNO-CCSD(T) algorithms,[21,56,57] where the LNO-CCSD and the LNO-(T) calculations take comparable time. This is explained by the fact that a Laplace-transform based (T) expression[56] allows the redundancy free evaluation of the triples amplitudes, but the CCSD iteration is not accelerated by this strategy. Consequently, both the LNO-CCSD and LNO-(T) domain calculations scale as $\mathcal{O}(n_\mathrm{v}^4\, n_\mathrm{o}^2)$ with the LNO dimensions, and both have comparable prefactors.

Due to the increased importance of these terms for our target applications and the fact that these terms are mostly omitted in the algorithm optimization efforts in the literature, we decided to devote more attention to these contributions. It turned out that there are parts where a clearly optimal choice did not present itself because, for instance, the low-memory symmetry-packed route deviates from the one that is best for parallelization. For this reason, we found it interesting to present our thoughts on these terms in more detail than usual since other preferences might lead to alternative algorithmic choices, and this could contribute to a useful discussion in the literature of CCSD.

The evaluations of terms depending on the same integrals and intermediates are merged in order to reduce the number of integral assembly steps via exploiting reusable intermediates.

Namely $C_{ij}^{ab}$ is calculated together with $B_{ij}^{ab}$, and it is also beneficial to group $E_{ij}^{ab}$, $G_{ij}^{ab}$ with the terms contributing to the singles amplitudes equations ($A_i^a$, $B_i^a$, and $C_i^a$). The proposed algorithms share the strategy that the operations are divided into blocks over an occupied index (see the outermost loops over $n_{\text{block}}$ in Algorithms 2-4). For instance, the analogous $C_{ij}^{ab}$ and $D_{ij}^{ab}$ terms are computed for one occupied index at a time as apparent from the loops over index $i$ in lines 7 and 8 of Algorithms 2 and 3, respectively. Then the cubic-scaling $C_{abj}^{i}$ and $D_{abj}^{i}$ intermediates are gathered into the array of the residual inside the loops over index $i$ and discarded. Such batching obviates the storage of full-sized $\mathcal{O}(N^4)$-scaling intermediates but allows the use of effective matrix algebra with sufficiently large work arrays. The otherwise redundant assembly and unpacking of those four-center integrals and doubles amplitudes that are independent of index $i$ are carried out in advance for each block (see the inner loops over index $k$ at line 2 of Algorithms 2 and 3).

Note that the $I_{dlck}$ four-center integrals, assembled from the $t_1$-transformed three-center ERIs in line 3 of Algorithm 2, are required for the evaluation of the $C_{ij}^{ab}$ term, and they are also reused to calculate the $\alpha_{lkij}$ intermediate of the $B_{ij}^{ab}$ term (see line 15 of Algorithm 2). Thus some of DF-direct integral assembly operations can be spared in this way. The contraction of the doubles amplitudes with $\alpha_{lkij}$ is also performed in blocks of occupied indices ensuring that the unpacked amplitudes at line 20 of Algorithm 2 require at most cubic scaling storage space.

The remaining terms, scaling as $\mathcal{O}(N^5)$, do not require such exhaustive optimization. It is, however, ensured that none of those terms lead to bottlenecks related to inefficient, bandwidth-bound operations when many CPU cores are used, and that the minimal memory requirement is not increased. For this reason, it is worth decreasing the number of these $\mathcal{O}(N^5)$-scaling steps. The algorithm for these terms is shown in Algorithm 4. First, we should recognize that almost all of these terms depend on the same $u \cdot \hat{J}$ product. The size of this intermediate is cubic-scaling, so the entire $uJ$ array can be stored in memory (see line 7 of Algorithm 4). The advantage of calculating $uJ$ first is that it reduces the

**Algorithm 2** Evaluation of the $C_{ij}^{ab}$ and the $B_{ij}^{ab}$ terms.

```
1: for $n_{\text{block}} = 1, \lceil n_\text{o}/n_\text{B} \rceil$  // MPI and OpenMP parallel
2:     for $k = (n_{\text{block}} - 1)n_\text{B} + 1, n_{\text{block}}\ n_\text{B}$                $\triangleright\ C_{ij}^{ab}$
3:         $I_{dlck} = \hat{J}_{kd,P}\ \left(\hat{J}_{lc,P}\right)^\dagger$
4:         $I_{acik} = \hat{J}_{ac,P}\ \left(\hat{J}_{ki,P}\right)^\dagger$
5:         $T_{bjck} = T_{bc\overline{kj}}$
6:     end for
7:     for $i = 1, n_\text{o}$
8:         $Z_{ack}^i = -1/2\ T_{a,dl}^i\ I_{dl,ck}$
9:         $Z_{ack}^i \leftarrow I_{ack}^i$
10:        $C_{abj}^i = -Z_{a,ck}^i\ (T_{bj,ck})^\dagger$
11:        for $j \le i$:   $R_{ab\overline{ji}} \leftarrow C_{abj}^i + 1/2\ C_{baj}^i$
12:        for $j \ge i$:   $R_{ab\overline{ij}} \leftarrow 1/2\ C_{abj}^i + C_{baj}^i$
13:    end for
14:    $I_{dclk} = I_{dlck}$                $\triangleright\ B_{ij}^{ab}$
15:    $\alpha_{lk\overline{ij}} \leftarrow (I_{dc,lk})^\dagger\ T_{dc,\overline{ij}}$
16: end for
17: for $i, j$:   $\alpha_{lk\overline{ij}} \leftarrow \hat{J}_{l,P}^j\ \left(\hat{J}_{k,P}^i\right)^\dagger$
18: for $n_{\text{block}} = 1, \lceil n_\text{o}/n_\text{B} \rceil$
19:     for $k = (n_{\text{block}} - 1)n_\text{B} + 1, n_{\text{block}}\ n_\text{B}$
20:         $R_{ab\overline{ij}} \leftarrow T_{ab,lk}\ \alpha_{lk,\overline{ij}}$
21:     end for
22: end for
```

FLOP count of some of the remaining operations to quartic-scaling eliminating 7 out of the overall 10 quintic-scaling operations. The only other intermediate with a significant storage requirement is the $G_{ij}^{ab}$ array, which is again batched into intermediates of cubic-scaling memory requirement (see loop over index $i$ in line 18 of Algorithm 4). The blocks of $u$ can also be reused to calculate $C_i^a$ (line 8 of Algorithm 4) before discarding them.

## 3.3 Parallelization of the residual equations

The intra-node parallelization utilizing multi-core CPUs is performed using the OpenMP application programming interface, which facilitates multi-threaded execution with a many-

**Algorithm 3** Evaluation of the $D_{ij}^{ab}$ term.

```
1: for $n_{\text{block}} = 1, \lceil n_o/n_B \rceil$ // MPI and OpenMP parallel
2:     for $k = (n_{\text{block}} - 1)n_B + 1, n_{\text{block}}\ n_B$
3:         $L_{dl,ck} = \hat{J}_{ld,P}\ \left(\hat{J}_{kc,P}\right)^\dagger$
4:         $L_{dl,ck} = 2\ L_{dl,ck} - L_{cl,dk}$
5:         $I_{acik} = \hat{J}_{ac,P}\ \left(\hat{J}_{ki,P}\right)^\dagger$
6:         $U_{bjck} = 2\ T_{bc\overline{jk}} - T_{cb\overline{jk}}$
7:     end for
8:     for $i = 1, n_o$
9:         $u^i_{a,dl} = 2\ T_{ad\overline{il}} - T_{da\overline{il}}$
10:        $uL^i_{a,ck} = 1/2\ u^i_{a,dl}\ L_{dl,ck}$
11:        $uL^i_{a,ck} \leftarrow 2\hat{J}^i_{a,P}\ \left(\hat{J}_{ck,P}\right)^\dagger - I^i_{ack}$
12:        $D^i_{abj} = 1/2\ uL^i_{a,ck}\ (U_{bj,ck})^\dagger$
13:        for $j \leq i$:   $R_{ab\overline{ji}} \leftarrow D^i_{baj}$
14:        for $j \geq i$:   $R_{ab\overline{ij}} \leftarrow D^i_{abj}$
15:    end for
16: end for
```

core processor. On top of that, the MPI protocol is employed to distribute the workload among multiple nodes of a computer cluster. It is beneficial to combine the two parallelization strategies because MPI tasks cannot access the memory space of the other processes without introducing additional communication operations, while OpenMP threads running on the same node can directly access data in a shared memory environment. Thus, in the implemented hybrid MPI/OpenMP approach the memory of a non-uniform memory access (NUMA) node, or optionally the entire memory of a multi-processor node is shared among the threads of the OpenMP layer, and the MPI layer distributes the workload between the NUMA or complete compute nodes.

Since the individual terms in the CCSD residual equations are already partitioned into blocks, it is intuitive to spread the batches among nodes. Accordingly, the MPI parallelization is performed by distributing the blocks of $\overline{ab}$ indices for the PPL term and the blocks of the occupied $k$ indices for the other terms across the MPI processes. This strategy is benefi-

**Algorithm 4** Evaluation of $E^{ab}_{ij}$, $G^{ab}_{ij}$, $A^a_i$, $B^a_i$, and $C^a_i$ terms.

```
1: // MPI parallelization: indices $k$ and the
2:       appropriate blocks are distributed across processes
3: for $n_{\text{block}} = 1, \lceil n_\text{o}/n_\text{B} \rceil$
4:     for $k = (n_{\text{block}} - 1)n_\text{B} + 1, n_{\text{block}}\ n_\text{B}$
5:         $u_{bkld} = 2\ T_{bd\overline{kl}} - T_{db\overline{kl}}$
6:     end for
7:     $uJ_{bkP} = u_{bk,ld}\ \hat{J}_{ld,P}$
8:     $R_{bk} \leftarrow u_{bk,ld}\ \hat{F}_{ld}$                                   ▷ $C^a_i$
9: end for
10: $X_{bc} = -uJ_{b,kP}\ \left(\hat{J}_{c,kP}\right)^\dagger$                       ▷ $E^{ab}_{ij}$
11: $X_{bc} \leftarrow \hat{F}_{bc}$
12: $R_{ab\overline{ij}} \leftarrow X_{a,c}\ T_{c,b\overline{ij}}$
13: for $\overline{ij}$:   $R_{ab\overline{ij}} \leftarrow T^{\overline{ij}}_{a,c}\ (X_{b,c})^\dagger$
14: $R_{bi} \leftarrow -uJ_{b,kP}\ \left(\hat{J}_{i,kP}\right)^\dagger$                ▷ $B^a_i$
15: $R_{ak} \leftarrow \hat{J}_{a,bP}\ (uJ_{k,bP})^\dagger$                         ▷ $A^a_i$
16: $\beta_{jk} = \hat{J}_{j,bP}\ \left(u\hat{J}_{k,bP}\right)^\dagger$                ▷ $G^{ab}_{ij}$
17: $\beta_{jk} \leftarrow \hat{F}_{jk}$
18: for $i = 1, n_\text{o}$
19:     $G^i_{adk} = T^i_{ad,j}\ \beta_{j,k}$
20:     for $k \le i$:   $R_{da\overline{ki}} \leftarrow -G^i_{adk}$
21:     for $k \ge i$:   $R_{ad\overline{ik}} \leftarrow -G^i_{adk}$
22: end for
```

cial because, as long as the key quantities, e.g., the three-center two-electron MO integrals, the amplitudes, and residuals, fit into the memory of a single node, their inter-node communication is not needed. It is only necessary to gather the residual contribution of each process at the end of an iteration. Alternatively, the four-index arrays and the three-center integrals could be distributed among the nodes.[19,20] This strategy decreases the memory demand of the MPI tasks running on a single node but introduces a potentially limiting communication cost. The applications of the present report were performed with the replicated memory model since this choice was allowed by our highly memory-efficient implementation. The workload is distributed statically for each term except for the PPL. The distribution of blocks in the PPL term occurs dynamically, and its evaluation takes place at the end of the iteration for load balancing. This dynamic load distribution strategy also facilitates the use of a heterogeneous hardware environment, e.g., nodes with different processor types or memory amounts can also be utilized effectively up to a reasonable degree of heterogeneity.

The poor performance and parallel scaling of the bandwidth-bound operations, like the (un)packing of amplitudes and residuals, e.g., line 4 in Algorithm 1 or lines $11-12$ in Algorithm 2, can be masked by overlapping them with compute-bound operations like matrix-matrix multiplications. For this reason, we adopted a nested OpenMP parallel scheme. On the outer OpenMP level several threads work on different blocks of the batched loops. Within these blocks, on the inner level the intrinsic parallelism of level 3 threaded BLAS routines is exploited. For example, this nested OpenMP strategy exhibits excellent scaling and a 7.6 speedup for a $(H_2O)_{10}$ cluster with the cc-pVDZ basis set on a 16-core node compared to our previous CCSD program.[53] Most of this gain can, however, be attributed to the fact that the previous implementation was not optimized to run efficiently on a large number of threads.

## 3.4 Integral-direct and parallel (T) algorithm

The so-called "*ijkabc*" type implementations of the (T) correction usually compute the $\mathbf{W}$ and $\mathbf{V}$ intermediates of eqs 21 and 22 for all virtual orbital index triples in nested loops over fixed "*ijk*" triplets. In order to cumulate all contributions of $\mathbf{W}$ into the same three-dimensional array, the indices of the intermediates resulting from the contractions of eq 21 are usually permuted to a matching order, say "*abc*". We have shown previously that this permutation, despite its lower, sixth-power-scaling operation count, has a highly bandwidth-bound nature as opposed to the effective, compute-bound contractions.[56] For this reason we proposed an algorithm for multi-threaded use which exploits the permutational symmetry of the Coulomb integrals and the doubles amplitudes in order to eliminate all but one of the permutations needed.[56] Here, we report a further improved "*ijkabc*" (T) algorithm by eliminating the one remaining permutation, as well as any remaining disk usage. Additionally, the minimal memory requirement is compressed to be comparable to that of the new CCSD algorithm, and the code is also parallelized via a similar MPI/OpenMP route.

The updated *ijkabc* (T) algorithm is presented in Algorithm 5. To avoid the redundant unpacking of the doubles amplitudes inside the three loops over the $i \geq j \geq k$ indices, it is beneficial to perform this operation outside all the occupied loops. Since the residual array from the CCSD calculation is no longer needed, the unpacked doubles amplitude tensor can be stored in a space comparable to that allocated for the CCSD calculation. The storage of the doubles amplitudes with permuted indices, as shown in array $R$ in line 1 of Algorithm 5, is also beneficial for the effective elimination of the index permutations of the intermediates (see lines 12 and 15 of Algorithm 5). Since array $R$ takes as much memory as the array holding the double amplitudes, the storage of the entire $R$ would almost double the minimal memory requirement. Thus, if there is limited memory, we do not store the entire $R$ and perform the necessary amplitude index permutations inside the loops over indices $i$ and $j$. The two- and three-external index four-center ERIs can be assembled inside the occupied loops for a single value of $i$, $j$, and $k$ as needed (see lines 3, 6-8, 10, and 18 of Algorithm 5). Consequently, the size of all intermediates can be kept at most cubic-scaling.

All in all, the minimal memory requirement for the (T) correction is $8\, n_\mathrm{v}^2\, n_\mathrm{o}^2$ bytes for the unpacked doubles amplitudes, $8\,(n_\mathrm{o}^2 + n_\mathrm{v}^2 + n_\mathrm{o}\, n_\mathrm{v})\, N_\mathrm{a}$ bytes for the unpacked three-center integrals, and $8 \cdot 5\, n_\mathrm{v}^3$ bytes for the three-external two-electron integrals, i.e., $(ac|bi)$, $(ac|bj)$, and $(ac|bk)$ (for a given $i$, $j$ or $k$ index), as well as for the $\mathbf{W}$ and the $\mathbf{V}$ intermediates. This minimal memory usage is comparable to that of the above described CCSD algorithm, which does not include the storage of the two-external four-center ERIs. That is feasible since, if there is insufficient per-node memory, the $(ai|bj)$ integrals can be obtained when needed via an effective integral-direct approach as shown in lines 6, 7, and 18 of Algorithm 5.

With a sufficiently low memory requirement, the next task is the wall time optimization of the (T) correction because of its even steeper scaling than that of the CCSD method. The

**Algorithm 5** Integral-direct and parallel $ijkabc$ (T) algorithm.

```
1: $R_{baij} = T_{abij}$
2: for $l = n_\mathrm{o}, m$
3:     $I^l_{abc} = J_{ab,P}\,(J^l_{c,P})^\dagger$ // assembly as far as memory allows until index $m$
4: end for
5: for $k = 1, n_\mathrm{o}$  // MPI parallel
6:     $I^k_{bcj} = J_{bj,P}\,(J^k_{c,P})^\dagger$
7:     $I^k_{aci} = J_{ai,P}\,(J^k_{c,P})^\dagger$
8:     $I^k_{abc} = J_{ab,P}\,(J^k_{c,P})^\dagger$ // if unavailable
9:     for $ij$ $(i \geq j \geq k)$  // OpenMP parallel
10:        $I^l_{abc} = J_{ab,P}\,(J^l_{c,P})^\dagger$ // if unavailable for $l = i$ or $l = j$
11:        for $b = 1, n_\mathrm{v}$  // OpenMP parallel
12:            $w^b_{ac} = I^{jb}_{a,d}\ T^{ik}_{d,c} + (I^{bi}_{d,a})^\dagger\ T^{jk}_{d,c} + T^{jb}_{a,l}\ (I^{ki}_{c,l})^\dagger + R^{bi}_{a,l}\ (I^{kj}_{c,l})^\dagger$
13:        end for
14:        for $c = 1, n_\mathrm{v}$  // OpenMP parallel
15:            $w^c_{ab} \leftarrow T^{ik}_{a,d}\ I^{cj}_{d,b} + T^{ij}_{a,d}\ (I^{ck}_{b,d})^\dagger + I^{ck}_{a,d}\ T^{ij}_{d,b} + (I^{ci}_{d,a})^\dagger\ T^{kj}_{d,b} +$
                   $I^{ij}_{a,l}\ (T^{ck}_{b,l})^\dagger + I^{ik}_{a,l}\ (R^{cj}_{b,l})^\dagger + T^{ck}_{a,l}\ (I^{ji}_{b,l})^\dagger + R^{ci}_{a,l}\ (I^{jk}_{b,l})^\dagger$
16:            $v^c_{ab} = w^c_{ab} + T^i_a\ I^{cjk}_b + I^{cik}_a\ T^j_b$
17:        end for
18:        $I^{ij}_{ab} = J^i_{a,P}\,(J^j_{b,P})^\dagger$
19:        for $b = 1, n_\mathrm{v}$  // OpenMP parallel
20:            $v^b_{ac} \leftarrow I^{bij}_a\ T^k_c$
21:        end for
22:        Calculate energy contribution according to eq 5 of ref 56
23:    end for
24: end for
```

FLOP count for the naive, fully integral direct construction of the three-external two-electron integrals would be about $n_\mathrm{o}^3\,n_\mathrm{v}^3\,N_\mathrm{a}/6$, because all $(ab|ci)$ integrals have to be assembled for all independent $a$, $b$, and $c$ indices and for all $i \geq j \geq k$ triplets. Since that is too expensive to be practical, as many of these integrals as possible are assembled and stored in memory outside of the three outer loops (see lines 2-4). So far there is no overhead in the assembly of $(ab|ci)$ compared to alternative implementations, since the assembly of the three-external integrals is not needed for the $t_1$-transformed CCSD equations. Because of the restrictions $i \geq j \geq k$, the reusability of the pre-assembled integrals is optimal if they are stored for the highest occupied indices. Then only the missing integrals are assembled (redundantly) inside the occupied loops when they are not available in memory (line 10). Alternatively, the three-external two-electron integrals could be distributed across the memory of all nodes, but this strategy results in a higher inter-node communication cost. The choice of assembling the required integrals on each node obviates the communication at the expense of an increased operation count. This cost is, however, acceptable, especially because the assembly can be performed via highly-efficient and well-scaling BLAS level 3 *gemm* operations.

Regarding the multi-threaded scaling, we improved upon the contraction of the doubles amplitudes and the two-electron integrals. By introducing a loop over a virtual index that is not a summation index in the contraction, e.g., over $b$ or $c$ in lines 11 and 14 of Algorithm 5, it became possible to cumulate the resulting $\mathbf{W}$ contribution immediately with the appropriate index order for all six terms. Thus, the one remaining, poorly scaling permutation operation is also eliminated from the present algorithm. Our measurements indicate that, especially with high number of threads, this new contraction strategy is usually more efficient even when the index order resulting from the contraction matches that of the array used for the $\mathbf{W}$ intermediate. Therefore, we adopted this approach for the calculation of all six types of terms contributing to the $\mathbf{W}$ intermediate.

Furthermore, via the introduction of OpenMP directives for the multi-threaded parallelization of the $j$ and $i$ loops, it was possible to overlap the low arithmetic intensity operations, i.e., the dyadic products needed for the $\mathbf{V}$ intermediate (lines 16 and 20 of Algorithm 5) and the calculation of the energy contribution (line 22 of Algorithm 5), with compute-bound operations. To reduce the additional memory requirement originating from the thread-local intermediates and integrals, we employ nested OpenMP parallelization. For that end, either the loops over the virtual indices $b$ and $c$ can be parallelized (see lines 11, 14, and 19 of Algorithm 5) or threaded BLAS routines can be called for the *gemm* operations. We have found both solutions similarly effective. Consequently, by assigning some of the threads to the inner OpenMP layer, the number of threads in the outer layer and hence the number of thread-local arrays can be kept small. As a result of the aforementioned improvements, in comparison to the previous, already multi-threaded algorithm,[56] we measure an overall decrease of about 40% in the wall times with 16 cores for the example of the $(H_2O)_{10}$ cluster with the cc-pVDZ basis set.

On top of that, inter-node parallelization is also implemented by distributing the $k$ indices of the outer loop (line 5 of Algorithm 5) across the compute nodes using the MPI protocol. The distribution of $k$ indices is dynamic to achieve a balanced load on each compute node. This task distribution is very effective and fits well into our data allocation strategy. Obviously, this choice does not scale any more if the number of MPI tasks exceeds the number of occupied orbitals, but excellent speedups are measured up to a few hundreds of cores (see Section 5).

It is also worth noting that relatively frequent checkpointing is implemented for both the CCSD and the (T) parts of the calculation. For example, in the case of unexpected power failure or expired wall time limit, the CCSD iteration can be restarted from the last completed iteration. The more costly (T) correlation energy evaluation is checkpointed at each completed iteration of the innermost occupied loop (over index $i$ in Algorithm 5). When restarting the (T) calculation the job simply skips the converged CCSD iteration, reads the integrals and amplitudes from disk and continues the innermost occupied loop with the next incomplete $ijk$ index triplet.

## 4 Computational details

The above CCSD(T) approach has been implemented in the MRCC suite of quantum chemical programs, and it will be made available in a forthcoming release of the package.[90]

The benchmark tests were carried out with compute nodes containing two Intel Xeon E5-2650 v2 CPUs with 8 physical cores per CPU. The theoretical peak performance of these nodes (332.8 GFLOP/s) is identical to the ones used in previous benchmark studies assessing alternative CCSD(T) implementations.[17,19,20] Hence this CPU choice facilitates the direct comparison of the performance of our code with alternative implementations. The compute nodes have 125 GB RAM, and their interconnection is established with an InfiniBand FDR network. The more extensive calculations of Section 6 were carried out using Intel Xeon Platinum 8180M processors equipped with 28 physical cores each. The theoretical peak performance of one such CPU is 2240 GFLOP/s.

For the benchmark calculations correlation consistent basis sets, cc-pV$XZ$,[91] were utilized with the corresponding DF auxiliary bases, cc-pV$X$Z-RI.[92] The calculations in Section 6 utilize triple- and quadruple-$\zeta$ valence basis sets including polarization[93] and diffuse functions[94] (def2-TZVPPD and def2-QZVPPD) with the corresponding auxiliary basis sets.[95] The core electrons were not correlated in any of the presented cases.

## 5 Performance analysis

In this section we analyze the multi-threaded and multi-node parallel performance of the introduced DF-CCSD(T) program. The benchmark calculations were performed on the same molecules and identical CPUs as in recent studies[17,19,20] so that it will also be meaningful to compare wall time measurements with the ones obtained with different programs. Namely, the test molecules adopted from these studies are the $(H_2O)_{10}$ cluster[20] for the study of the multi-threaded performance of CCSD(T), and the uracil trimer[19] as well as the $(H_2O)_{14}$ cluster[19] for the multi-node performance analysis of CCSD and (T), respectively. To determine the basis set dependence of the parallel efficiency, calculations were performed on the $(H_2O)_6$ cluster with basis sets of increasing size following the analogous test of ref 20. Various choices for the number of MPI tasks and for nested OpenMP parallelism were examined on the $(H_2O)_{10}$ cluster. On top of the above, we also find these examples fortunate because their $n_\mathrm{v}/n_\mathrm{o}$ ratio and the number of correlated orbitals are highly representative of the average/maximum number of correlated natural orbitals in a single domain CCSD(T) calculation of our LNO-CCSD(T) method.[21,56,57] The benchmarked molecules with the applied basis sets and the number of basis functions are collected in Table 1.

Table 1: Systems, basis sets, and the number of basis functions selected for the benchmark calculations.

| System | Basis | $n_o$ | $n_v$ | $N_a$ |
|---|---|---|---|---|
| $(H_2O)_6$ | cc-pVDZ | 24 | 114 | 468 |
| | cc-pVTZ | 24 | 318 | 846 |
| | cc-pVQZ | 24 | 660 | 1452 |
| $(H_2O)_{10}$ | cc-pVDZ | 40 | 190 | 780 |
| $(H_2O)_{14}$ | cc-pVDZ | 56 | 266 | 1176 |
| Uracil trimer | cc-pVDZ | 63 | 309 | 1512 |

## 5.1 Multi-threaded performance of CCSD(T)

The multi-threaded scaling of the CCSD algorithm is depicted in Figure 1. The calculations were performed on the $(H_2O)_{10}$ cluster with the cc-pVDZ basis set using a single compute node. The wall times and the speedups are plotted for five implementations (MPQC,[19] ORCA,[80] PSI4,[28] FHI-aims,[20] and MRCC[90]). Speedup values of the middle panel are obtained using the single core measurement as reference. The performance value is obtained as the ratio of the required double precision operations of our algorithm and the measured wall times. For the relative performance shown in the right panel, this is divided by the peak performance corresponding to the given number of cores and the CPU's base frequency. In some benchmark calculations more than 100% CPU utilization can be observed, which is attributed to the use of the Intel Turbo Boost (ITB) technology. The theoretical peak performance is calculated using the 2.6 GHz base frequency of the CPU, while with ITB the clock rate can be significantly higher, up to 3.1 GHz above 4 threads or even 3.4 GHz with only 1 thread. Unfortunately, this effect cannot be accounted for in our relative performance expressions since the actual operating frequencies are not known. From this perspective it might have been more fortunate to turn off ITB, but that was out of our control. In turn, the measurements represent more realistic scenarios where ITB is operating. The wall times obtained with the ORCA, PSI4, and MPQC packages in our measurements are somewhat different from those presented in refs 19, 20, and 17. This could probably be explained by the different configuration of the clusters (e.g., network, file system) used for the measurements. Therefore, we acknowledge that wall time measurements have observable uncertainties even if the same CPU type is used and we focus on the speedup values and the relative performances compared to the theoretical peak performance, which are supposedly more independent of the actual hardware.

Regarding the results for CCSD (Figure 1), all five investigated implementations show an excellent speedup, mostly between 8 and 11 with 16 threads. Our implementation also demonstrates an efficient CPU utilization with about 65% of the theoretical peak performance with 16 threads.

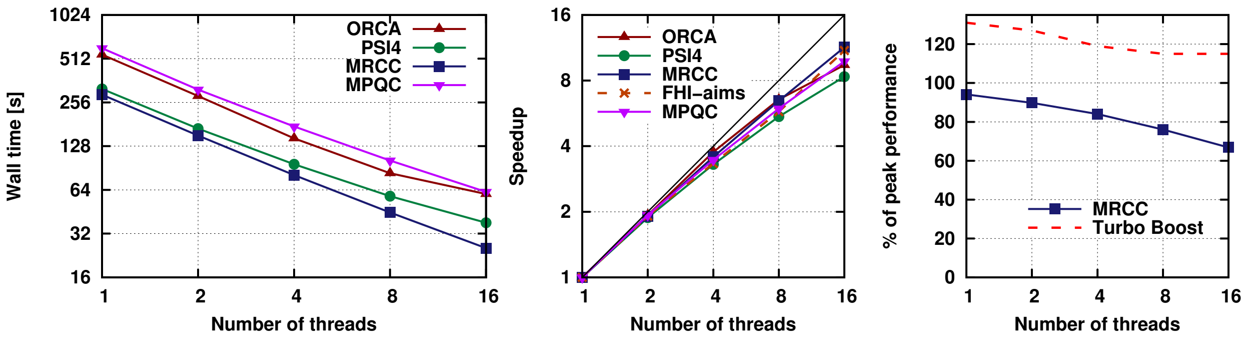


Figure 1: Multi-threaded performance of a CCSD iteration of various implementations for a $(H_2O)_{10}$ cluster using the cc-pVDZ basis set. Speedup values obtained for the FHI-aims software, illustrated with a dashed line, are taken from ref 20. The left panel shows wall-clock times, the middle panel depicts speedup values compared to the measurement with 1 thread, and the right panel illustrates performance values as the percentage of the theoretical peak performance of the corresponding number of cores at their base frequency.

The scaling of the individual terms in CCSD is shown in detail in Figure 2. The shorthand notation of Figure 2 refers to a contribution to the doubles amplitudes (e.g., A2 $\equiv \mathrm{A}_{ij}^{ab}$). Only the terms with a wall time of at least 1% of a CCSD iteration are presented. In accordance with Section 3, the cumulative wall time of the $\mathrm{B}_{ij}^{ab}$, the $\mathrm{C}_{ij}^{ab}$, and the $\mathrm{D}_{ij}^{ab}$ terms is about twice the time of PPL, representing about 2/3 of the total elapsed time of the calculation. This measurement also emphasizes the importance of optimal algorithms in the case of the sixth-power-scaling terms apart from PPL, at least when small basis sets are utilized. It can be seen that the $\mathcal{O}(N^6)$-scaling terms (i.e., $\mathrm{A}_{ij}^{ab}$, $\mathrm{B}_{ij}^{ab}$, $\mathrm{C}_{ij}^{ab}$, and $\mathrm{D}_{ij}^{ab}$) that contain mostly compute bound operations scale very well with the number of threads. Only the $\mathcal{O}(N^5)$-scaling $\mathrm{E}_{ij}^{ab}$ and $\mathrm{G}_{ij}^{ab}$ terms exhibit worse speedup because of the more bandwidth-bound nature of the involved operations. These two terms represent only $\sim$ 4% of the total run time with 16 threads, and their moderate scaling should make the lower speedup even less influential with larger molecules or basis sets. Therefore, with larger systems a better overall speedup of CCSD can be expected (c.f., Figure 7 below).

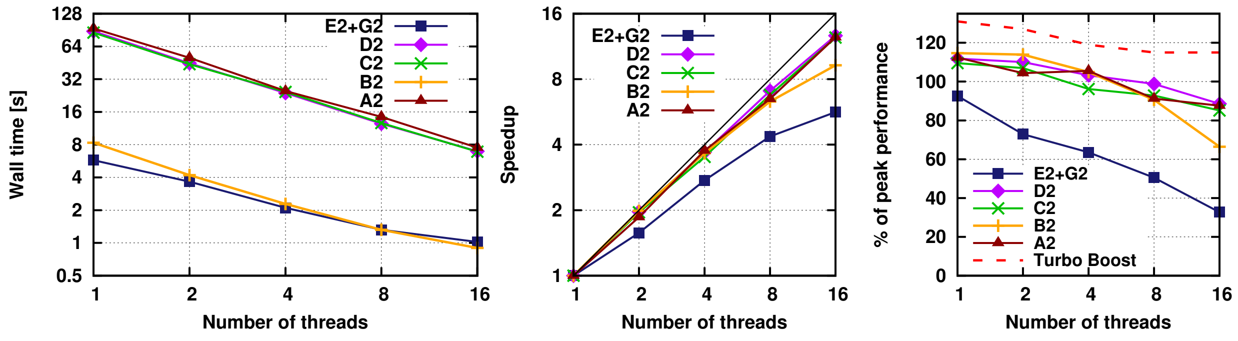


Figure 2: Scaling of the computationally most expensive terms in CCSD. The calculation was performed on a $(H_2O)_{10}$ cluster with the cc-pVDZ basis set. See the caption of Figure 1 for further details.

Similar conclusions can be drawn from the (T) measurements depicted in Figure 3. The speedups with the three implementations with which we performed measurements (ORCA, PSI4, and MRCC) are close to each other, on the average around 10-12 with 16 threads, which is probably very close to the limit that is achievable with the given hardware. The performance of MRCC is better than in the case of CCSD, i.e., about 80% of the theoretical peak is achieved with 16 threads.

The scaling of the computationally most expensive terms of the (T) correction are depicted in Figure 4. The operations needed for the steeper, $\mathcal{O}(N^7)$-scaling terms scale considerably better (the speedup is about 12 on 16 cores) with the number of threads than those required for the $\mathcal{O}(N^6)$-scaling calculation of the $\mathbf{V}$ intermediate. However, the evaluation of $\mathbf{V}$ takes only about 7% of the (T) correction even with 16 threads and is expected to become even shorter relative to the total calculation for larger systems because of the less

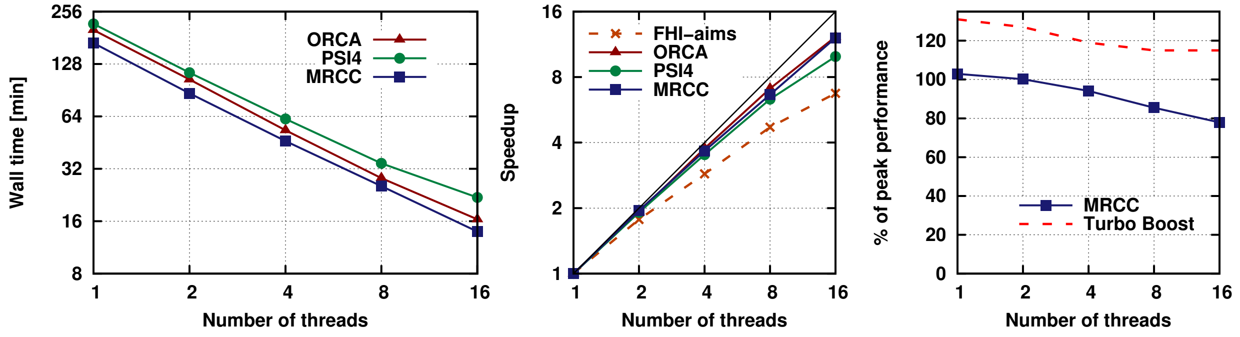


Figure 3: Multi-threaded performance of the (T) correction of various implementations for a $(H_2O)_{10}$ cluster with the cc-pVDZ basis. Speedup values obtained with the FHI-aims package, illustrated with a dashed line, are taken from refs 19 and 20, respectively. See the caption of Figure 1 for further details.

steep scaling of its operation count.

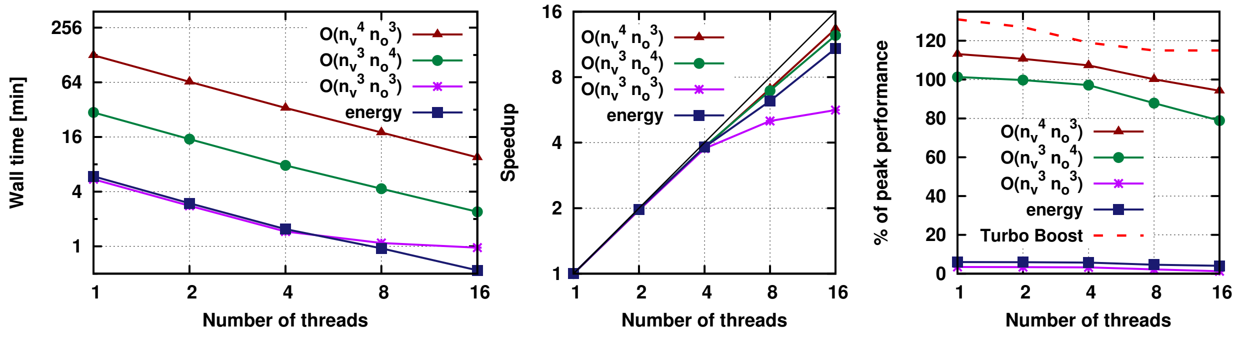


Figure 4: Scaling of the computationally most expensive terms of the (T) correction. The calculation was performed for a $(H_2O)_{10}$ cluster with the cc-pVDZ basis set. See the caption of Figure 1 for further details.

We also determined the dependence of the performance (still within a single node) on the number of MPI tasks and the number of threads in the outer parallel region [outside of BLAS calls in CCSD or the inner virtual loops in the (T) algorithm] in case of nested OpenMP parallelism (see line 1 of Algorithms 1, 2, and 3, and line 5 of Algorithm 5) for the $(H_2O)_{10}$ cluster. The results are plotted in Figures 5 and 6. For better visibility we show the decrease in wall times (left panels) and improvements in speedups (right panels) in comparison to the single task measurement performed without nested OpenMP or MPI. It is observed that the introduction of both the higher number of MPI tasks and outer OpenMP threads increase the performance. The better performance obtained in the case of nested OpenMP parallelism can be explained by the overlap of the memory-intensive operations with other, more arithmetic-intensive ones as described in Section 3. For instance, nested parallelism improves the relatively poorly scaling evaluation of the **V** intermediate. According to our measurements, the speedup with more MPI processes, on the other hand, can mostly be attributed to the better utilization of the NUMA architecture of the compute nodes. This was verified by running two computations for the $(H_2O)_{10}$/cc-pVDZ example. First, all threads and data allocations were assigned to the same NUMA node, and then the threads and the data were fixed on different NUMA nodes. The wall time measured with the data in the non-local memory was found about 13% longer compared to the one with only local memory access. However, this is clearly the limiting case and, in realistic applications, when only one MPI task is running on a node, the data is distributed between the local and non-local memory. When there is enough memory, it is advisable to run at least as many MPI processes per node as the number of NUMA nodes to avoid this slower memory access.

The CCSD calculation benefits more from the higher number of threads on the first (outer) OpenMP level (line 1 of Algorithms 1, 2, and 3; denoted by 2OMP in the figures), while (T) is better accelerated via MPI tasks (denoted by 2MPI). The cumulative effect of the nested OpenMP and MPI parallelism (2MPI-2OMP) is smaller in both cases. Using both nested OpenMP and MPI parallelism an overall 16% and 21% decrease of wall times could be achieved for the CCSD iteration and the (T) correction, respectively, for these single-node 16-core calculations.

In Figure 7 the scalings of CCSD and (T) are illustrated as the function of the basis set size for a $(H_2O)_6$ cluster. While the speedup measured for the (T) correction is nearly independent of the applied basis set within the range of cc-pVDZ to cc-pVQZ, the CCSD iteration scales better with larger basis sets. The speedup of CCSD with the cc-pVDZ basis is somewhat lower because the small number of basis functions make the sequential $\mathcal{O}(N^4)$-scaling terms, e.g., the unpacking of the doubles amplitudes, noticeable compared to the most expensive but well-scaling $\mathcal{O}(N^6)$ terms.

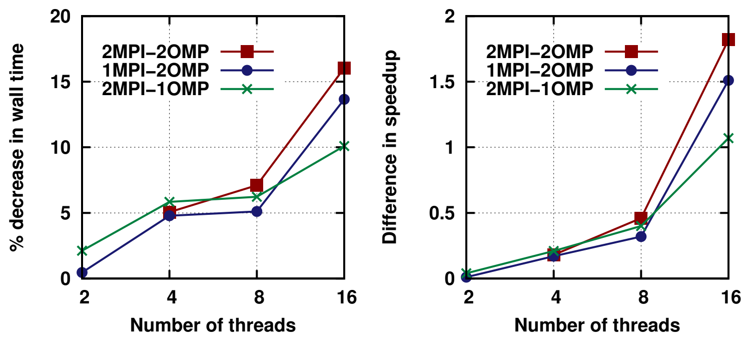


Figure 5: Performance of a CCSD iteration as the function of the number of MPI tasks and level 1 OpenMP threads for the $(H_2O)_{10}$ cluster. The left panel shows the decrease of wall times as the percentage of the measurement with 1 MPI task and without nested OpenMP parallelism. The right panel depicts the relative speedup values with the same reference.

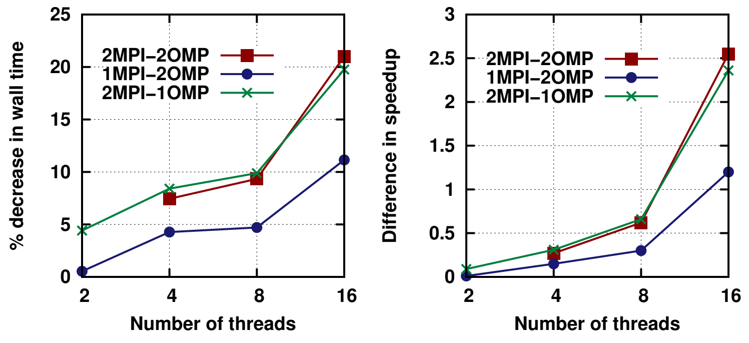


Figure 6: Performance of the (T) correction as the function of the number of MPI tasks and level 1 OpenMP threads for the $(H_2O)_{10}$ cluster. The left panel shows the decrease of wall times as the percentage of the measurement with 1 MPI task and without nested OpenMP parallelism. The right panel depicts the relative speedup values with the same reference.

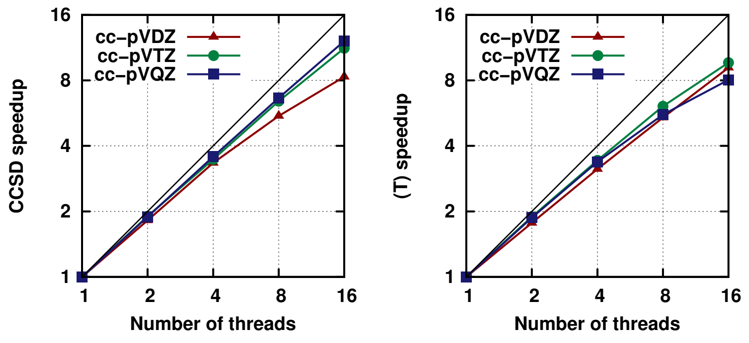


Figure 7: Speedup of the CCSD iteration (left panel) and the (T) correction (right panel) relative to the measurement with 1 thread as the function of the basis set size for the $(H_2O)_6$ cluster.

## 5.2 Multi-node performance of CCSD(T)

The multi-node performance was determined on the example of the uracil trimer for CCSD and for the $(H_2O)_{14}$ cluster for (T), both with the cc-pVDZ basis set. The multi-node scaling of the CCSD and (T) parts are shown in Figures 8 and 9, respectively.

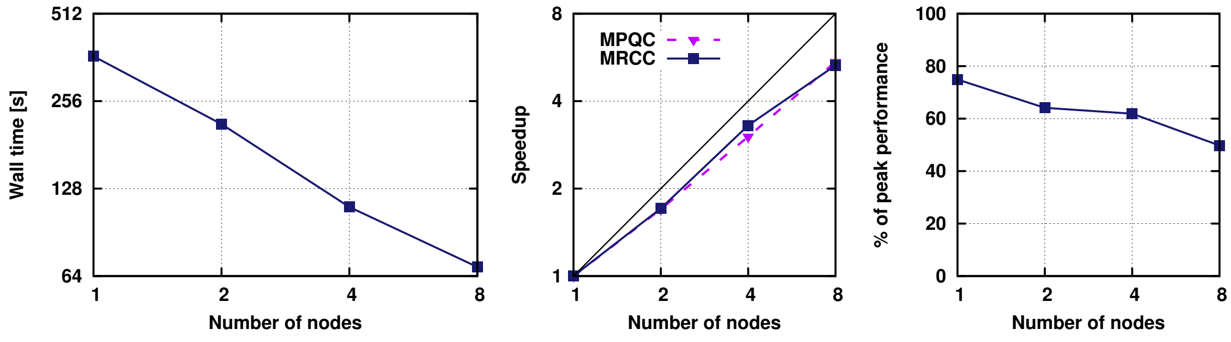


Figure 8: Multi-node performance of a CCSD iteration for the uracil trimer with the cc-pVDZ basis set. The speedup values obtained with the MPQC package are taken from ref 19. See the caption of Figure 1 for further details.

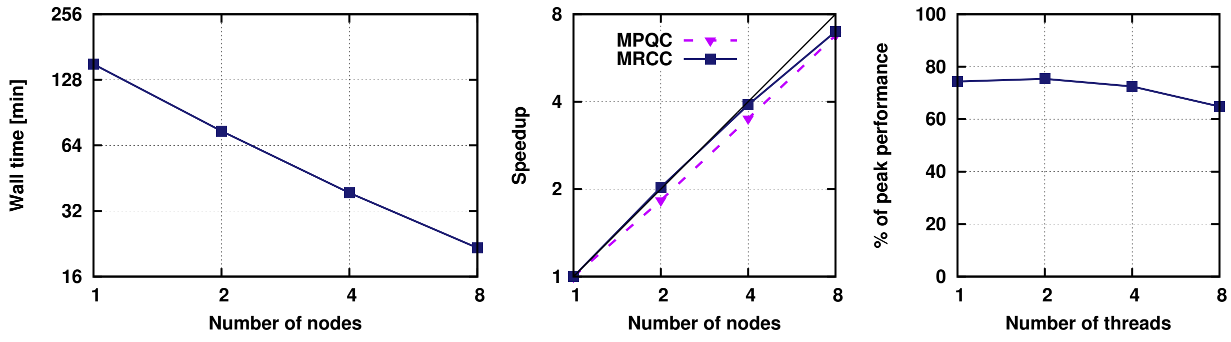


Figure 9: Multi-node performance of the (T) correction for a $(H_2O)_{14}$ cluster with the cc-pVDZ basis. See the caption of Figure 1 and 8 for further details.

Both the CCSD calculation and the (T) correction show an excellent speedup that is comparable to the performance of the recent MPQC implementation. The parallel efficiency of the CCSD iteration and the (T) correction is 66% and 88% on 8 nodes, respectively. The relative performance of CCSD and (T) is still at least about 50% and 65% of the combined theoretical peak performance of the 8 compute nodes (see right panels of Figures 8 and 9).

The individual terms exhibit behavior analogous to the case of the single node measurements. The steepest scaling contributions [$\mathcal{O}(N^6)$ for CCSD and $\mathcal{O}(N^7)$ for the (T) correction] scale well with the number of nodes. Compared to that the scaling of the less expensive [$\mathcal{O}(N^5)$ for CCSD and $\mathcal{O}(N^6)$ for the (T) correction], more bandwidth-bound terms start to deteriorate from the ideal scaling when the number of nodes increases above 8-16.

That is, however, satisfactory from the perspective of the planned applications since a few hundred compute cores can still be efficiently exploited by using up to a few tens of nodes equipped with many-core CPUs.

Further performance benchmarks are given in Section 6.2 where various MPI/OpenMP strategies are compared for large-scale examples.

## 6 Applications

In this section, we illustrate the capabilities of the presented CCSD(T) code on chemical problems, which would be out of the scope of conventional implementations. The performance and efficiency of the program is also analyzed for these large-scale calculations executed with a few hundred cores.

### 6.1 Expansion of the CEMS26 test set of CCSD(T) references

According to the first goal of providing valuable reference data for the assessment of reduced-scaling methods, we expand our previous CCSD(T) reference compilation[21] with correlation energies, reaction energies, and barrier heights obtained for real-life catalytic reactions.[81–83] The first version of the correlation energies of medium-sized systems list contains 26 entries, hence the abbreviation of CEMS26.[21] First, in order to consider realistic test systems which are also capable to assess the accuracy of various local approximations, each molecule of CEMS26 contains at least 30 atoms. Second, all correlation energies are obtained with at least triple-$\zeta$ quality basis sets to provide some flexibility in the one-particle basis set. Third, high reproducibility is required for these benchmarks, thus, for example, reduced-cost NO-based calculations were excluded. Together these three criteria represent a very strong set of limitations. After an extensive literature search we were only able to find a handful of suitable calculations to add to the test set. Aiming at a more representative data set size we also performed a number of extensive CCSD(T) calculations, but a large portion of those had to exploit spatial symmetry to have an affordable computation cost. The higher than average portion of spatially symmetric molecules is thus not representative, which is corrected here by adding 12 asymmetrical molecules of $31 - 43$ atoms to the list. Another unfavorable feature of the test set is the relatively low number of results calculated with basis sets other than Dunning's correlation-consistent basis sets[91] or with ones including diffuse functions. To improve upon this aspect at least triple-$\zeta$, and for one entry a quadruple-$\zeta$, basis set were employed including both polarization and diffuse functions (def2-TZVPPD and def2-QZVPPD).[93,94] The previous entries were added using their ground state global minimum structures. The current selections thus include also local minima and transition state structures along the previously explored reaction paths.[81–83]

Three reactions are considered: an organocatalytic Michael-addition reaction,[81] the hydrogen activation by a frustrated Lewis pair (FLP),[82] and a palladium catalyzed C–H activation reaction,[83] as shown in Figures 10-12. The size of the molecules, the applied basis sets, and the corresponding orbital dimensions, as well as the calculated CCSD(T) correlation energies are collected in Table 2. These correlation energies are also useful to expand the CEMS26 set with 8 new reaction energies and 4 barrier heights, which are collected in Table 3. The complete list of species, HF, MP2, CCSD, and CCSD(T) energies and the employed Cartesian coordinates are available in the Supporting Information (SI).

Having a closer look at the investigated species and reactions the first example is an organocatalytic Michael-addition reaction[81] in which propanal and $\beta$-nitrostyrene (NS) react in a diphenylprolinol silyl ether catalyzed reaction with a *p*-nitrophenol cocatalyst. Besides the main enamine (en-trans) and iminium intermediates a stable cyclobutane (CB) and a dihydrooxazine *N*-oxide (OO) intermediate also have important roles in the reaction mechanism.[81] The enamine intermediate and $\beta$-nitrostyrene react through a transition state (TS) denoted as TS1. The intermediates, OO and CB, are separated by another TS labeled by TS2 (see Figure 10). The overall stereochemistry and the reaction rate of these reactions are governed by delicate interactions between the reactants and the catalyst. Moreover, various

Table 2: The species, the utilized basis sets, and the CCSD(T) correlation energies added to the CEMS26 test set.

| Species | Atoms | Basis set | no. of AOs | $n_o$[a] | $n_v$ | $N_a$ | $E^{\mathrm{CCSD(T)}}$ [$E_h$] |
|---|---|---|---|---|---|---|---|
| OO | 40 | def2-TZVPPD | 1089 | 54 | 1015 | 2620 | −3.854401 |
| CB | 40 | def2-TZVPPD | 1089 | 54 | 1015 | 2620 | −3.858204 |
| TS1 | 40 | def2-TZVPPD | 1089 | 54 | 1015 | 2620 | −3.878783 |
| TS2 | 40 | def2-TZVPPD | 1089 | 54 | 1015 | 2620 | −3.861682 |
| FLPD | 41 | def2-TZVPPD | 1037 | 46 | 974 | 2500 | −3.122023 |
| FLPO | 41 | def2-TZVPPD | 1037 | 46 | 974 | 2500 | −3.096183 |
| $TS_{add}$ | 43 | def2-TZVPPD | 1071 | 47 | 1007 | 2578 | −3.146588 |
| FLPA | 43 | def2-TZVPPD | 1071 | 47 | 1007 | 2578 | −3.162500 |
| $S_1$ | 34 | def2-TZVPPD | 992 | 54 | 916 | 2417 | −3.929478 |
| $S_2$ | 34 | def2-TZVPPD | 992 | 54 | 916 | 2417 | −3.926367 |
| $TS_{Pd}$ | 34 | def2-TZVPPD | 992 | 54 | 916 | 2417 | −3.937428 |
| ABP | 31 | def2-TZVPPD | 893 | 45 | 830 | 2163 | −3.238854 |
| ABP[b] | 31 | def2-QZVPPD | 1569 | 45 | 1506 | 3671 | −3.405137 |

[a] Number of correlated occupied orbitals.
[b] The CCSD(T) correlation energy of ABP extrapolated from the def2-TZVPPD and def2-QZVPPD energies is −3.528309 $E_h$.

paths are found in a fairly narrow energy range. Thus, highly accurate calculations are required for the reliable characterization of the reaction mechanism.[81] The species added to the CEMS26 test set are the OO and the CB intermediates as well as the TS1 and TS2 transition states (Table 2). All 4 structures contain 40 atoms. Due to the extended def2-TZVPPD basis set choice a rather large number of basis functions (1089) are involved.

The second example is the first step of a hydrogenation reaction catalyzed by an FLP, namely the addition of $H_2$ to the FLP catalyst.[82] In FLP catalysis the system contains both a Lewis acid and a Lewis base but the formation of a classical Lewis adduct is prohibited, usually because of steric effects. In the example of ref 82 the heterolytic bond breaking of $H_2$ is catalyzed (see Figure 11). The species added to the CEMS26 list are the datively bound FLP catalyst (FLPD), its open form isomer (FLPO), the transition state $TS_{add}$ and

Table 3: CCSD(T) reaction energies and barrier heights for the reactions of Section 6.1.

| Reaction | Basis set | $\Delta E^{\mathrm{CCSD(T)}}$ [kcal mol$^{-1}$] |
|---|---|---|
| en-trans + NS $\rightarrow$ TS1 | def2-TZVPPD | 4.89 |
| en-trans + NS $\rightarrow$ OO | def2-TZVPPD | −23.75 |
| en-trans + NS $\rightarrow$ TS2 | def2-TZVPPD | 5.06 |
| en-trans + NS $\rightarrow$ CB | def2-TZVPPD | −26.39 |
| FLPD $\rightarrow$ FLPO | def2-TZVPPD | 7.92 |
| FLPD + $H_2$ $\rightarrow$ $TS_{add}$ | def2-TZVPPD | 12.89 |
| FLPD + $H_2$ $\rightarrow$ FLPA | def2-TZVPPD | −6.19 |
| AA + $Pd(OAc)_2$ $\rightarrow$ $S_1$ | def2-TZVPPD | 2.92 |
| AA + $Pd(OAc)_2$ $\rightarrow$ $S_2$ | def2-TZVPPD | −14.05 |
| AA + $Pd(OAc)_2$ $\rightarrow$ $TS_{Pd}$ | def2-TZVPPD | 8.51 |
| Reaction 3[a] | def2-TZVPPD | −74.42 |
| Reaction 3[b] | def2-QZVPPD | −73.82 |

[a] AA + BA + TBHP $\rightarrow$ ABP + TBA + $H_2O$.

[b] The reaction energy extrapolated using the def2-TZVPPD and def2-QZVPPD results is -73.50 kcal/mol.

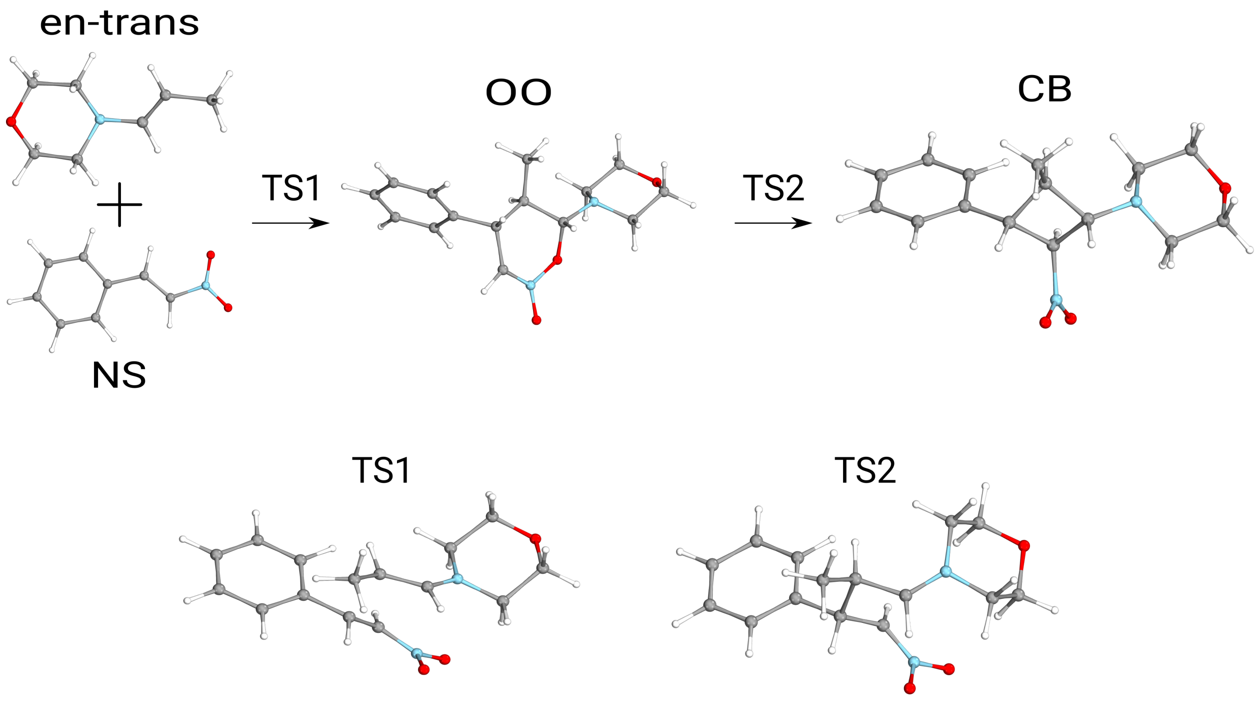


Figure 10: Schematic representation of the diphenylprolinol silyl ether catalyzed Michael-addition reaction.[81]

the product of the FLP-mediated $H_2$ activation reaction (FLPA). The catalyst contains 41 atoms, whereas the TS and the adduct consist of 43 atoms. The utilized def2-TZVPPD basis set contains 1037 and 1071 functions for the FLP and the TS/adduct species, respectively.

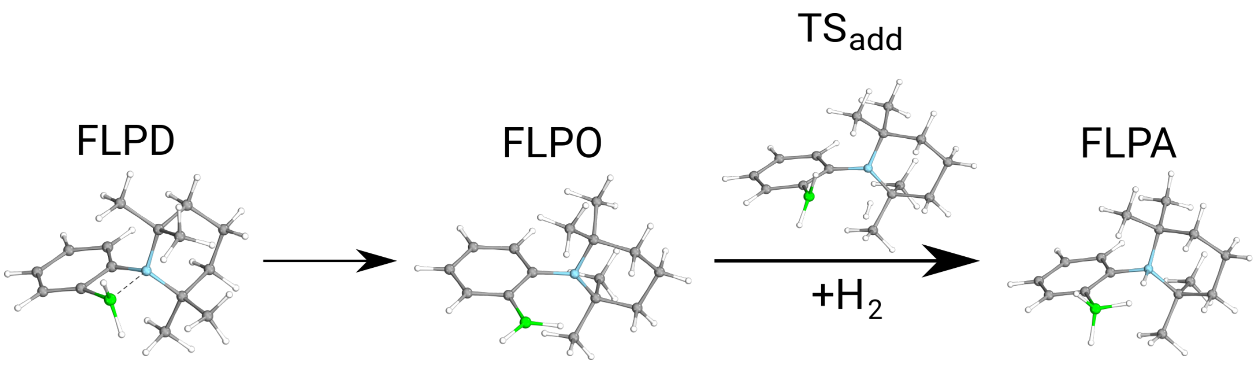


Figure 11: Addition of dihydrogen to the frustrated Lewis pair catalyst.[82]

Finally, we consider a palladium catalyzed C–H bond activation reaction (see Figure 12).[83] In the reaction, palladium catalyzes the cross-dehydrogenative coupling between anilides, like acetanilide (AA), and aromatic aldehydes, like benzaldehyde (BA), in the presence of tert-butyl hydroperoxide (TBHP) forming tert-butyl alcohol (TBA) and 2-aminobenzophenon (ABP). The product, 2-acetaminobenzophenon, containing 31 atoms, the transition state ($TS_{Pd}$), and two intermediates ($S_1$ and $S_2$), containing 34 atoms fall within the size range of the molecules in the CEMS26 test set. For $TS_{Pd}$ and the intermediates $S_1$ and $S_2$ the def2-TZVPPD basis set was utilized, which consists of 992 AOs. The somewhat smaller size of ABP also enabled a CCSD(T) calculation to be carried out with the def2-TZVPPD as well as the def2-QZVPPD basis sets, which contained 893 and 1569 basis functions, respectively. The energies calculated for ABP with the def2-TZVPPD and the def2-QZVPPD bases were also extrapolated to the basis set limit. The extrapolation was carried out utilizing the two-point expression of Karton and Martin[96] for the HF energies using the parameters suggested by Neese and Valeev.[97] Correlation energies was extrapolated using the formula introduced by Helgaker *et al.*[98] As shown in Table 3, at least for this particular example, the reaction energy converges relatively rapidly with respect to the basis set size. Namely, -74.4, -73.8, and -73.5 kcal/mol were obtained, respectively, with the def2-TZVPPD and the def2-QZVPPD basis sets, and as the result of the extrapolation.

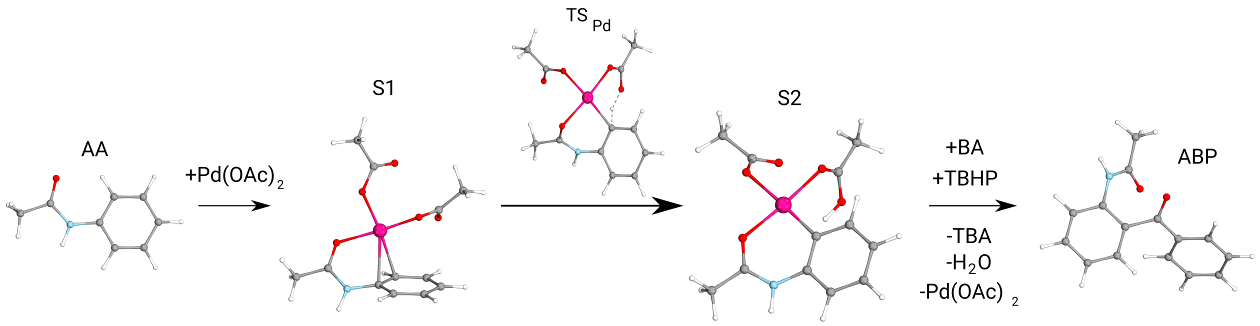


Figure 12: Schematic representation of the palladium catalyzed C–H activation reaction.[83]

With the above computations the CEMS26 set has been extended with the 12 structures and the 13 corresponding CCSD(T) correlation energies of Table 2 leading to the new, 39-element compilation called CEMS39. The new list contains results for local minima and transition states together with commonly utilized basis sets including diffuse functions offering a greater variety of systems. Besides the already present C, H, N, O, P, Cl, S, Si, Na, Mg, Li elements now B and Pd are also represented. The systems of the new CEMS39 set contain 38.5 atoms and 999 atomic orbitals on the average, and with that CEMS39 is currently the most realistic test set aimed at the representative assessment of local correlation methods. We will employ CEMS39 for that purpose in a forthcoming publication in the context of our LNO-CCSD(T) method.[21,57]

## 6.2 Performance for large-scale examples

To characterize the efficiency of the program also for large examples, we employed various settings for the number of MPI processes and OpenMP threads as well as for the parallelism of the inner parallel region (threaded or sequential BLAS library). We measured the efficiency for these calculations relative to the theoretical peak performance of the 4 Intel Xeon Platinum 8180M processors utilized in these numerical experiments. For this particular CPU the theoretical peak performance can be calculated with 1.7 GHz base frequency, which is the limit when the AVX-512 instruction set and all cores are employed. The measurements are summarized in Table 4.

Table 4: Performance values in percentage of the theoretical maximum obtained with various settings for the species of the Michael-addition reaction[81] and the hydrogen addition to FLP.[82]

| Species | No. of AOs | $N_\mathrm{a}$ | MPI tasks[a] | Outer OpenMP threads[a] | BLAS[b] | Wall time [h] | % performance[c] | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | CCSD | (T) |
| OO | 1089 | 2620 | 4 | 4 | sequential | 31.8 | 56 | 64 |
| CB | 1089 | 2620 | 8 | 2 | threaded | 31.0 | 58 | 65 |
| TS1 | 1089 | 2620 | 4 | 4 | threaded | 30.5 | 56 | 69 |
| TS2 | 1089 | 2620 | 4 | 2 | sequential | 32.0 | 62 | 65 |
| FLPD | 1037 | 2500 | 4 | 2 | threaded | 21.1 | 47 | 50 |
| FLPO | 1037 | 2500 | 8 | 2 | sequential | 24.1 | 40 | 46 |
| $\mathrm{TS_{add}}$ | 1071 | 2578 | 4 | 1 | threaded | 21.3 | 61 | 62 |
| FLPA | 1071 | 2578 | 8 | 1 | threaded | 21.8 | 63 | 58 |

[a] The total number of threads (i.e., MPI × outer OpenMP × inner OpenMP) is 112.
[b] Parallelism of the BLAS library.
[c] Percentage of the theoretical peak performance.

Note that these large-scale computations were performed primarily for the purpose of producing reference CCSD(T) correlation energies. Since the number of MOs, and hence the total number of operations required for the computations, are fairly close for the systems of Table 4, the comparison of these relative parformances is informative even if they were measured with slightly different orbital dimensions. It is apparent that for these systems and this particular CPU choice the performance of CCSD and the (T) correction is to a large extent independent of the above settings within the investigated range of MPI tasks and OpenMP threads. While keeping the total number of CPUs and cores fixed at 4 and 112, respectively, there is freedom to vary the number of MPI tasks and the number of outer OpenMP threads. For most of the inspected setting combinations (i.e., 4-8 MPI tasks and 1-4 outer OpenMP threads) the performances of the CCSD and (T) steps were found highly stable in the very satisfactory range of 56-63% and 58-69%, respectively. As explained in Section 5, it is advisable to set the number of the MPI tasks to the number of NUMA nodes. Based on the values of Table 4, if the total number of employed cores is kept fixed (112 in this case), the further increase of the number of MPI processes form 4 to 8 does not increase the performance, at least for systems of this size and with modern many-core CPUs. The optimal number of inner and outer OpenMP threads is not as obvious to determine. In the case of the above calculations, the number of outer threads does noticeably not affect the performance in the range of 1-4. Similarly, switching from threaded to sequential BLAS routines in combination with hand-coded inner layer OpenMP instructions for the latter has a negligible impact on the performance. The almost uniformly good performance with a relatively wide range of settings is certainly beneficial from the perspective of applications.

Considering the measured wall times, for the systems with 1037-1089 AOs the CCSD calculation took about 4-7 hours, while the (T) part required 17-25 hours, both with 112 cores. Compared to that our largest CCSD(T) calculation performed with 1569 AOs required 68 hours using 224 cores. In terms of the number of AOs or in terms of the number of atoms, in combination with quadruple-$\zeta$ basis sets, to our knowledge, that is the largest CCSD(T) calculation ever carried out. Considering that a few days of compute time on a few hundred cores, or in other words about 15,000 core hours, is easily accessible in many computer centers, such large-scale calculations can now be considered relatively routine for a much wider audience. Alternatively, smaller number of cores can be traded for longer execution time. Since the implementation is frequently checkpointed and effectively restartable, this is also a viable option even if strict wall times limits are implemented.

## 7 Summary and outlook

A completely integral-direct, operation-count and storage economic, well-parallelized DF-CCSD(T) algorithm and implementation have been presented. The on-the-fly and blocked assembly of all four-center ERIs allows us to minimize the $\mathcal{O}(N^4)$-scaling storage requirement to the symmetry-packed doubles amplitudes and residuals and avoid potentially limiting disk I/O or network communication both during a CCSD iteration and for the (T) correction up to the range of 1000-2000 orbitals. We also improved upon a previous $t_1$-transformed CCSD algorithm,[28] for instance, by optimizing and parallelizing all contractions besides the usually emphasized particle-particle ladder term. As the highest possible permutational symmetry and lowest operation count are ensured for the PPL term, some of which is sacrificed in alternative parallel CCSD implementations,[10,11,19,20] the remaining four $\mathcal{O}(N^6)$-scaling terms are found to be comparably time-consuming in some of our target applications. Thus, hand-optimized and well-scaling algorithms are presented also for those terms appearing in the $t_1$-transformed CCSD equations to which limited attention has been payed in the literature previously. Our recent, OpenMP-parallel (T) algorithm[56] has also been improved by making it completely integral-direct, I/O-free, and MPI/OpenMP parallel, while decreasing its minimal memory requirement to match that of the CCSD program and retaining the full permutational symmetry of the contractions.

Detailed wall time measurements performed with the presented CCSD and (T) codes demonstrate excellent strong scaling comparable to the performance of state-of-the-art implementations[11,19,20,28,80] for a wide range of systems including 100 to 1600 orbitals. At multi-threaded use on a single node about 65 and 80%, while for hybrid MPI/OpenMP use on up to a few hundred cores about 60 and 70% of the theoretical peak performance is utilized by the CCSD and (T) codes, respectively. The combination of the optimal operation count algorithms and the outstanding efficiency allowed us to perform 13 large-scale DF-CCSD(T) calculations at the applicability limit of the CCSD(T) method in a relatively routine manner using only 4-8 many-core CPUs. With those results we have extended our recent CCSD(T) benchmark set[21] with 13 new correlation energies and 12 new reaction energies and barrier heights characterizing three reaction mechanisms taken from contemporary chemistry.[81–83] Each calculation, involving 1037-1089 AOs, took only about one day with 112 cores, while the largest example of 1569 orbitals ran for about three days with 224 cores. To our knowledge the latter is one of the largest CCSD(T) application ever presented.

Due to the balanced performance obtained also for relatively small systems appearing also in popular data sets used for parametrizing DFT functionals[59,60] or machine-learning models[61–69] and to the minimal memory and disk footprint, we believe that the present code is well suited to produce such large-scale benchmark CCSD(T) data. These properties are especially useful if only a network file system and limited per-core memory is available, as in many current computer clusters, allowing for the almost independent evaluation of hundreds of medium-sized CCSD(T) calculations without introducing restrictive amount of network use. The disk-, memory-, and communication-economic algorithms and the good portability of the implementation also allowed us to perform CCSD(T) calculations both on various supercomputer centers and in a cloud environment.

Future directions of development could benefit from promising tensor factorization approaches to further reduce the memory and operation count requirements.[30–36] Additionally, the presented MPI/OpenMP parallel DF-CCSD(T) represents a significant step towards the development of a massively parallel LNO-CCSD(T) implementation. The LNO strategy, while drastically reducing the computational cost of CCSD(T) via local, natural orbital, and other approximations, estimates the correlation energy using orbital-specific contributions obtained via independent CCSD(T) calculations.[21,53–57] In rare cases when exceptionally high accuracy is required for systems of complicated, moderately truncatable wavefunctions, a potentially large number of LNOs has to be handled in some of the independent CCSD(T) runs. The new DF-CCSD(T) algorithm is an excellent tool to accelerate such extensive domain calculations. As LNO-CCSD(T) calculations were already feasible for entire proteins in the range of 1023-2380 atoms and up to 45,000 basis functions[21,57] using a single CPU, the extension with the present high-performance CCSD(T) algorithm could lead to accurate and efficient CCSD(T) calculations for systems of previously unreachable size and complexity.

## Acknowledgement

The authors are grateful for the financial support from the National Research, Development, and Innovation Office (NKFIH, Grant No. KKP126451). This work was also supported by the BME-Biotechnology FIKP grant of EMMI (BME FIKP-BIO). The work of PRN is supported by the ÚNKP-18-4-BME-257 and ÚNKP-19-4-BME-418 New National Excellence Program of the Ministry for Innovation and Technology and the János Bolyai Research Scholarship of the Hungarian Academy of Sciences. We acknowledge that the results of this research have been achieved using the DECI resource Saga based in Norway at Trondheim with support from the PRACE aisbl, and the Hungarian HPC Infrastructure at NIIF Institute, Hungary.

## Supporting Information Available

See supporting information for the complete list of Cartesian coordinates employed for the CEMS39 set; and for the computed HF, MP2, CCSD, and CCSD(T) energies.

This material is available free of charge via the Internet at `http://pubs.acs.org/`.

# Appendix

The $t_1$-transformed Hamiltonian can be expressed via the transformed Fock matrix ($\hat{\mathbf{f}}$), the one electron Hamiltonian ($\hat{\mathbf{h}}$), and the three-center Coulomb integrals ($\hat{\mathbf{J}}$) if the DF approximation is employed. Performing the transformation leads to the following matrix elements:[28,76]

$$\hat{f}_{pq} = \hat{h}_{pq} + \sum_{iQ} \left( 2 \; \hat{J}^Q_{pq} \; \hat{J}^Q_{ii} - \hat{J}^Q_{pi} \; \hat{J}^Q_{iq} \right) \tag{29}$$

$$\hat{h}_{pq} = \sum_{rs} (\mathbf{1} - \mathbf{t}_1^{\mathrm{T}})_{rp} \; h_{rs} \; (\mathbf{1} + \mathbf{t}_1^{\mathrm{T}})_{sq} \tag{30}$$

$$\hat{J}^Q_{pq} = \sum_{rs} (\mathbf{1} - \mathbf{t}_1^{\mathrm{T}})_{rp} \; J^Q_{rs} \; (\mathbf{1} + \mathbf{t}_1^{\mathrm{T}})_{sq}. \tag{31}$$

Here, $\mathbf{t}_1$ denotes a matrix of dimension $n_{\mathrm{o}} + n_{\mathrm{v}}$, with $(\mathbf{t}_1)_{pq} = t^p_q$ for $p > n_{\mathrm{o}}$ and $q < n_{\mathrm{v}}$; and with $(\mathbf{t}_1)_{pq} = 0$ otherwise.[28] It is worth noting that, after the transformation defined by eq 6, $\hat{\mathbf{f}}$ and $\hat{\mathbf{J}}$ do not retain the permutational symmetry[28,76] of $\mathbf{f}$ and $\mathbf{J}$, that is, $\hat{J}^P_{pq} \neq \hat{J}^P_{qp}$ and $\hat{f}_{pq} \neq$

$\hat{f}_{qp}$. Furthermore, the occupied-virtual block of $\mathbf{J}$ is invariant to the $t_1$-transformation,[76] i.e., $\hat{J}^P_{ia} = J^P_{ia}$. The transformation of the individual blocks of $\mathbf{J}$ and $\mathbf{f}$ can be carried out according to the following equations:

$$\hat{J}^P_{ia} = J^P_{ia} \tag{32}$$

$$\hat{J}^P_{ij} = J^P_{ij} + \sum_c J^P_{ic} t^c_j \tag{33}$$

$$\hat{J}^P_{ab} = J^P_{ab} - \sum_k J^P_{kb} t^a_k \tag{34}$$

$$\hat{J}^P_{ai} = J^P_{ai} + \sum_c J^P_{ac} t^c_i - \sum_k J^P_{ki} t^a_k - \sum_{kc} J^P_{kc} t^a_k t^c_i \tag{35}$$

$$\hat{f}_{ia} = f_{ia} + \sum_{kc} \left[ 2 \sum_P J^P_{ai} J^P_{kc} - \sum_P J^P_{ic} J^P_{ka} \right] t^c_k \tag{36}$$

$$\hat{f}_{ij} = f_{ij} + \sum_c \hat{f}_{ic} t^c_j + \sum_{kc} \left[ 2 \sum_P J^P_{ij} J^P_{kc} - \sum_P J^P_{ic} J^P_{kj} \right] t^c_k \tag{37}$$

$$\hat{f}_{ab} = f_{ab} - \sum_l \hat{f}_{lb} t^a_l + \sum_{kc} \left[ 2 \sum_P J^P_{ab} J^P_{kc} - \sum_P J^P_{ac} J^P_{kb} \right] t^c_k \tag{38}$$

$$\hat{f}_{ai} = f_{ai} + \sum_c \hat{f}_{ac} t^c_i - \sum_l \hat{f}_{li} t^a_l + \sum_{kc} \left[ 2 \sum_P J^P_{ai} J^P_{kc} - \sum_P J^P_{ac} J^P_{ki} \right] t^c_k \; . \tag{39}$$

In contrast to the approach of ref 28, here the $t_1$-transformation is performed in the MO basis because in this case the three-center AO integrals do not have to be stored during the CCSD iteration. Let us also note that the auxiliary basis required for the correlated calculation is employed for the three-center integrals, thus our $t_1$-transformed expressions yield exactly the same numerical results as a DF-CC implementation without $t_1$-transformation. In that respect we also deviate from the algorithm of ref 28, where, to our understanding, the auxiliary basis of the SCF calculation is employed to form the $t_1$-transformed Fock-matrix. In order to save memory space during the CCSD iterations, we do not store the original MO integrals because they can be recovered from the $t_1$-transformed integrals via the inverse transformation. This can be achieved by inverting the transformation defined by eqs 32-35.

For example, the $\hat{J}_{ij}^{P}$ integrals can be calculated as

$$\begin{aligned}\hat{J}_{ij}^{P(n)} =& J_{ij}^{P} + \sum_{c} J_{ic}^{P} t_{j}^{c(n)} \\ =& \hat{J}_{ij}^{P(n-1)} - \sum_{c} J_{ic}^{P} t_{j}^{c(n-1)} + \sum_{c} J_{ic}^{P} t_{j}^{c(n)},\end{aligned} \tag{40}$$

where $\hat{J}_{ij}^{P(n)}$ and $t_{j}^{c(n)}$ stand for the $t_1$-transformed three-center integrals and singles amplitudes of the $n^{\text{th}}$ iteration. Note that only the original occupied-virtual integral block, $J_{ia}$ is needed for the inverse transformation. This block is readily available in every iteration since it is unaffected by the $t_1$-transformation in accordance with eq 32. Alternatively, the original integrals $J_{ic}$ can be pulled out from the last two terms of eq 40. This way the back-transformation can be avoided by performing the transformation on the $t_1$-transformed integrals of the previous iteration, $\hat{J}_{ij}^{P(n-1)}$, using $t_{j}^{c(n)} - t_{j}^{c(n-1)} = R_{j}^{c(n-1)}/(f_{jj} - f_{cc})$. However, the inverse transformation is preferable because the original integrals are also necessary for the $t_1$-transformation of the Fock matrix according to eqs 36-39. The transformation of the remaining three-center integrals can be carried out analogously.

## References

(1) Crawford, T. D.; Schaefer III, H. F. *Rev. Comp. Chem.* **1999**, *14*, 33.

(2) Helgaker, T.; Jørgensen, P.; Olsen, J. *Molecular Electronic Structure Theory*; Wiley: Chichester, 2000.

(3) Shavitt, I.; Bartlett, R. *Many-Body Methods in Chemistry and Physics: MBPT and Coupled-Cluster Theory*; Cambridge Molecular Science; Cambridge University Press, 2009.

(4) Bartlett, R. J.; Musiał, M. Coupled-cluster theory in quantum chemistry. *Rev. Mod. Phys.* **2007**, *79*, 291.

(5) Kállay, M.; Gauss, J. Analytic second derivatives for general coupled-cluster and configuration interaction models. *J. Chem. Phys.* **2004**, *120*, 6841.

(6) Kállay, M.; Gauss, J. Calculation of excited-state properties using general coupled-cluster and configuration-interaction models. *J. Chem. Phys.* **2004**, *121*, 9257.

(7) Helgaker, T.; Coriani, S.; Jørgensen, P.; Kristensen, K.; Olsen, J.; Ruud, K. Recent Advances in Wave Function-Based Methods of Molecular-Property Calculations. *Chem. Rev.* **2012**, *112*, 543–631.

(8) Raghavachari, K.; Trucks, G. W.; Pople, J. A.; Head-Gordon, M. A fifth-order perturbation comparison of electron correlation theories. *Chem. Phys. Lett.* **1989**, *157*, 479.

(9) Deegan, M. J. O.; Knowles, P. J. Perturbative corrections to account for triple excitations in closed and open shell coupled cluster theories. *Chem. Phys. Lett.* **1994**, *227*, 321.

(10) Kobayashi, R.; Rendell, A. P. A direct coupled cluster algorithm for massively parallel computers. *Chem. Phys. Lett.* **1997**, *265*, 1–11.

(11) Anisimov, V. M.; Bauer, G. H.; Chadalavada, K.; Olson, R. M.; Glenski, J. W.; Kramer, W. T. C.; Aprà, E.; Kowalski, K. Optimization of the Coupled Cluster Implementation in NWChem on Petascale Parallel Architectures. *J. Chem. Theory Comput.* **2014**, *10*, 4307–4316.

(12) Pitoňák, M.; Aquilante, F.; Hobza, P.; Neogrády, P.; Noga, J.; Urban, M. Parallelized implementation of the CCSD(T) method in MOLCAS using optimized virtual orbitals space and Cholesky decomposed two-electron integrals. *Collect. Czech. Chem. Commun.* **2011**, *76*, 713–742.

(13) Asadchev, A.; Gordon, M. S. Fast and Flexible Coupled Cluster Implementation. *J. Chem. Theory Comput.* **2013**, *9*, 3385–3392.

(14) Deumens, E.; Lotrich, V. F.; Perera, A.; Ponton, M. J.; Sanders, B. A.; Bartlett, R. J. Software design of ACES III with the super instruction architecture. *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **2011**, *1*, 895–901.

(15) Kaliman, I. A.; Krylov, A. I. New algorithm for tensor contractions on multi-core CPUs, GPUs, and accelerators enables CCSD and EOM-CCSD calculations with over 1000 basis functions on a single compute node. *J. Comput. Chem.* **2017**, *38*, 842–853.

(16) Janowski, T.; Pulay, P. Efficient Parallel Implementation of the CCSD External Exchange Operator and the Perturbative Triples (T) Energy Calculation. *J. Chem. Theory Comput.* **2008**, *4*, 1585–1592.

(17) Peng, C.; Calvin, J. A.; Pavošević, F.; Zhang, J.; Valeev, E. F. Massively Parallel Implementation of Explicitly Correlated Coupled-Cluster Singles and Doubles Using TiledArray Framework. *J. Phys. Chem. A* **2016**, *120*, 10231–10244.

(18) Solomonik, E.; Matthews, D.; Hammond, J. R.; Stanton, J. F.; Demmel, J. A massively parallel tensor contraction framework for coupled-cluster computations. *J. Parallel Distr. Com.* **2014**, *74*, 3176.

(19) Peng, C.; Calvin, J. A.; Valeev, E. F. Coupled-cluster singles, doubles and perturbative triples with density fitting approximation for massively parallel heterogeneous platforms. *Int. J. Quantum Chem.* **2019**, *119*, e25894.

(20) Shen, T.; Zhu, Z.; Zhang, I. Y.; Scheffler, M. Massive-parallel Implementation of the Resolution-of-Identity Coupled-cluster Approaches in the Numeric Atom-centered Orbital Framework for Molecular Systems. *J. Chem. Theory Comput.* **2019**, *15*, 4721.

(21) Nagy, P. R.; Samu, G.; Kállay, M. Optimization of the linear-scaling local natural orbital CCSD(T) method: Improved algorithm and benchmark applications. *J. Chem. Theory Comput.* **2018**, *14*, 4193.

(22) Yoo, S.; Aprà, E.; Zeng, X. C.; Xantheas, S. S. High-Level Ab Initio Electronic Structure Calculations of Water Clusters $(H_2O)_{16}$ and $(H_2O)_{17}$: A New Global Minimum for $(H_2O)_{16}$. *J. Phys. Chem. Lett.* **2010**, *1*, 3122–3127.

(23) Eriksen, J. J. Efficient and portable acceleration of quantum chemical many-body methods in mixed floating point precision using OpenACC compiler directives. *Mol. Phys.* **2017**, *115*, 2086.

(24) DePrince, A. E.; Kennedy, M. R.; Sumpter, B. G.; Sherrill, C. D. Density-fitted singles and doubles coupled cluster on graphics processing units. *Mol. Phys.* **2014**, *112*, 844.

(25) Aprà, E.; Kowalski, K. Implementation of High-Order Multireference Coupled-Cluster Methods on Intel Many Integrated Core Architecture. *J. Chem. Theory Comput.* **2016**, *12*, 1129.

(26) Epifanovsky, E.; Zuev, D.; Feng, X.; Khistyaev, K.; Shao, Y.; Krylov, A. I. General implementation of the resolution-of-the-identity and Cholesky representations of electron repulsion integrals within coupled-cluster and equation-of-motion methods: Theory and benchmarks. *J. Chem. Phys.* **2013**, *139*, 134105.

(27) Bozkaya, U.; Sherrill, C. D. Analytic energy gradients for the coupled-cluster singles and doubles with perturbative triples method with the density-fitting approximation. *J. Chem. Phys.* **2017**, *147*, 044104.

(28) DePrince, A. E.; Sherrill, C. D. Accuracy and Efficiency of Coupled-Cluster Theory Using Density Fitting/Cholesky Decomposition, Frozen Natural Orbitals, and a t1-Transformed Hamiltonian. *J. Chem. Theory Comput.* **2013**, *9*, 2687.

(29) Boström, J.; Pitoňák, M.; Aquilante, F.; Neogrády, P.; Pedersen, T. B.; Lindh, R. Coupled Cluster and Møller–Plesset Perturbation Theory Calculations of Noncovalent Intermolecular Interactions using Density Fitting with Auxiliary Basis Sets from Cholesky Decompositions. *J. Chem. Theory Comput.* **2012**, *8*, 1921.

(30) Kinoshita, T.; Hino, O.; Bartlett, R. J. Singular value decomposition approach for the approximate coupled-cluster method. *J. Chem. Phys.* **2003**, *119*, 7756.

(31) Hummel, F.; Tsatsoulis, T.; Grüneis, A. Low rank factorization of the Coulomb integrals for periodic coupled cluster theory. *J. Chem. Phys.* **2017**, *146*, 124105.

(32) Schutski, R.; Zhao, J.; Henderson, T. M.; Scuseria, G. E. Tensor-structured coupled cluster theory. *J. Chem. Phys.* **2017**, *147*, 184113.

(33) Peng, B.; Kowalski, K. Highly Efficient and Scalable Compound Decomposition of Two-Electron Integral Tensor and Its Application in Coupled Cluster Calculations. *J. Chem. Theory Comput.* **2017**, *13*, 4179.

(34) Parrish, R. M.; Sherrill, C. D.; Hohenstein, E. G.; Kokkila, S. I. L.; Martínez, T. J. Communication: Acceleration of coupled cluster singles and doubles via orbital-weighted least-squares tensor hypercontraction. *J. Chem. Phys.* **2014**, *140*, 181102.

(35) Parrish, R. M.; Zhao, Y.; Hohenstein, E. G.; Martínez, T. J. Rank reduced coupled cluster theory. I. Ground state energies and wavefunctions. *J. Chem. Phys.* **2019**, *150*, 164118.

(36) Benedikt, U.; Böhm, K.-H.; Auer, A. A. Tensor decomposition in post-Hartree–Fock methods. II. CCD implementation. *J. Chem. Phys.* **2013**, *139*, 224101.

(37) Christiansen, O.; Koch, H.; Jørgensen, P. The second-order approximate coupled cluster singles and doubles model CC2. *Chem. Phys. Lett.* **1995**, *243*, 409.

(38) Christiansen, O.; Koch, H.; Jørgensen, P.; Helgaker, T. Integral direct calculation of CC2 excitation energies: singlet excited states of benzene. *Chem. Phys. Lett.* **1996**, *263*, 530.

(39) Hättig, C.; Weigend, F. CC2 excitation energy calculations on large molecules using the resolution of the identity approximation. *J. Chem. Phys.* **2000**, *113*, 5154.

(40) Mester, D.; Nagy, P. R.; Kállay, M. Reduced-cost linear-response CC2 method based on natural orbitals and natural auxiliary functions. *J. Chem. Phys.* **2017**, *146*, 194102.

(41) Koch, H.; Christiansen, O.; Jørgensen, P.; Sánchez de Merás, A. M.; Helgaker, T. The CC3 model: An iterative coupled cluster approach including connected triples. *J. Chem. Phys.* **1997**, *106*, 1808.

(42) Taube, A. G.; Bartlett, R. J. Fozen Natural Orbital Coupled-Cluster Theory: Forces and Application to Decomposition of Nitroethane. *J. Chem. Phys.* **2008**, *128*, 164101.

(43) DePrince, A. E.; Sherrill, C. D. Accurate Noncovalent Interaction Energies Using Truncated Basis Sets Based on Frozen Natural Orbitals. *J. Chem. Theory Comput.* **2013**, *9*, 293.

(44) Rolik, Z.; Kállay, M. Cost-reduction of high-order coupled-cluster methods via active-space and orbital transformation techniques. *J. Chem. Phys.* **2011**, *134*, 124111.

(45) Brabec, J.; Yang, C.; Epifanovsky, E.; Krylov, A. I.; Ng, E. Reduced-cost sparsity-exploiting algorithm for solving coupled-cluster equations. *J. Comput. Chem.* **2016**, *37*, 1059.

(46) Pokhilko, P.; Epifanovsky, E.; Krylov, A. I. Double Precision Is Not Needed for Many-Body Calculations: Emergent Conventional Wisdom. *J. Chem. Theory Comput.* **2018**, *14*, 4088.

(47) Spencer, J. S.; Neufeld, V. A.; Vigor, W. A.; Franklin, R. S. T.; Thom, A. J. W. Large scale parallelization in stochastic coupled cluster. *J. Chem. Phys.* **2018**, *149*, 204103.

(48) Scott, C. J. C.; Di Remigio, R.; Crawford, T. D.; Thom, A. J. W. Diagrammatic Coupled Cluster Monte Carlo. *J. Phys. Chem. Lett.* **2019**, *10*, 925.

(49) Riplinger, C.; Pinski, P.; Becker, U.; Valeev, E. F.; Neese, F. Sparse maps—A systematic infrastructure for reduced-scaling electronic structure methods. II. Linear scaling domain based pair natural orbital coupled cluster theory. *J. Chem. Phys.* **2016**, *144*, 024109.

(50) Ma, Q.; Werner, H.-J. Explicitly correlated local coupled-cluster methods using pair natural orbitals. *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **2018**, *8*, e1371.

(51) Schmitz, G.; Hattig, C.; Tew, D. P. Explicitly correlated PNO-MP2 and PNO-CCSD and their application to the S66 set and large molecular systems. *Phys. Chem. Chem. Phys.* **2014**, *16*, 22167–22178.

(52) Rolik, Z.; Kállay, M. A general-order local coupled-cluster method based on the cluster-in-molecule approach. *J. Chem. Phys.* **2011**, *135*, 104111.

(53) Rolik, Z.; Szegedy, L.; Ladjánszki, I.; Ladóczki, B.; Kállay, M. An efficient linear-scaling CCSD(T) method based on local natural orbitals. *J. Chem. Phys.* **2013**, *139*, 094105.

(54) Kállay, M. Linear-scaling implementation of the direct random-phase approximation. *J. Chem. Phys.* **2015**, *142*, 204105.

(55) Nagy, P. R.; Samu, G.; Kállay, M. An integral-direct linear-scaling second-order Møller–Plesset approach. *J. Chem. Theory Comput.* **2016**, *12*, 4897.

(56) Nagy, P. R.; Kállay, M. Optimization of the linear-scaling local natural orbital CCSD(T) method: Redundancy-free triples correction using Laplace transform. *J. Chem. Phys.* **2017**, *146*, 214106.

(57) Nagy, P. R.; Kállay, M. Approaching the basis set limit of CCSD(T) energies for large molecules with local natural orbital coupled-cluster methods. *J. Chem. Theory Comput.* **2019**, *15*, 5275.

(58) Gordon, M., Ed. *Fragmentation: Toward Accurate Calculations on Complex Molecular Systems*; Wiley: New York, 2017.

(59) Mardirossian, N.; Head-Gordon, M. Thirty years of density functional theory in computational chemistry: an overview and extensive assessment of 200 density functionals. *Mol. Phys.* **2017**, *115*, 2315.

(60) Goerigk, L.; Hansen, A.; Bauer, C.; Ehrlich, S.; Najibi, A.; Grimme, S. A look at the density functional theory zoo with the advanced GMTKN55 database for general main group thermochemistry, kinetics and noncovalent interactions. *Phys. Chem. Chem. Phys.* **2017**, *19*, 32184.

(61) Cheng, L.; Welborn, M.; Christensen, A. S.; Miller, T. F. A universal density matrix functional from molecular orbital-based machine learning: Transferability across organic molecules. *J. Chem. Phys.* **2019**, *150*, 131103.

(62) McGibbon, R. T.; Taube, A. G.; Donchev, A. G.; Siva, K.; Hernández, F.; Hargus, C.; Law, K.-H.; Klepeis, J. L.; Shaw, D. E. Improving the accuracy of Møller–Plesset perturbation theory with neural networks. *J. Chem. Phys.* **2017**, *147*, 161725.

(63) Bartók, A. P.; De, S.; Poelking, C.; Bernstein, N.; Kermode, J. R.; Csányi, G.; Ceriotti, M. Machine learning unifies the modeling of materials and molecules. *Sci. Adv.* **2017**, *3*, e1701816.

(64) Nudejima, T.; Ikabata, Y.; Seino, J.; Yoshikawa, T.; Nakai, H. Machine-learned electron correlation model based on correlation energy density at complete basis set limit. *J. Chem. Phys.* **2019**, *151*, 024104.

(65) Chmiela, S.; Sauceda, H. E.; Müller, K.-R.; Tkatchenko, A. Towards exact molecular dynamics simulations with machine-learned force fields. *Nat. Commun.* **2018**, *9*, 3887.

(66) Mezei, P. D.; von Lilienfeld, A. O. Non-covalent quantum machine learning corrections to density functionals. *arXiv e-prints* **2019**, arXiv:1903.09010.

(67) Montavon, G.; Rupp, M.; Gobre, V.; Vazquez-Mayagoitia, A.; Hansen, K.; Tkatchenko, A.; Klaus-Robert, M.; von Lilienfeld, O. A. Machine learning of molecular electronic properties in chemical compound space. *New J. Phys.* **2013**, *15*, 095003.

(68) Ramakrishnan, R.; Dral, P. O.; Rupp, M.; von Lilienfeld, O. A. Big Data Meets Quantum Chemistry Approximations: The $\Delta$-Machine Learning Approach. *J. Chem. Theory Comput.* **2015**, *11*, 2087.

(69) Smith, J. S.; Nebgen, B. T.; Zubatyuk, R.; Lubbers, N.; Devereux, C.; Barros, K.; Tretiak, S.; Isayev, O.; Roitberg, A. E. Approaching coupled cluster accuracy with a general-purpose neural network potential through transfer learning. *Nat. Commun.* **2019**, *10*, 2903.

(70) Eriksen, J. J.; Baudin, P.; Ettenhuber, P.; Kristensen, K.; Kjærgaard, T.; Jørgensen, P. Linear-Scaling Coupled Cluster with Perturbative Triple Excitations: The Divide–Expand–Consolidate CCSD(T) Model. *J. Chem. Theory Comput.* **2015**, *11*, 2984.

(71) Li, W.; Ni, Z.; Li, S. Cluster-in-molecule local correlation method for post-Hartree–Fock calculations of large systems. *Mol. Phys.* **2016**, *114*, 1447.

(72) Friedrich, J.; Dolg, M. Fully Automated Incremental Evaluation of MP2 and CCSD(T) Energies: Application to Water Clusters. *J. Chem. Theory Comput.* **2009**, *5*, 287.

(73) Mochizuki, Y.; Yamashita, K.; Nakano, T.; Okiyama, Y.; Fukuzawa, K.; Taguchi, N.; Tanaka, S. Higher-order correlated calculations based on fragment molecular orbital scheme. *Theor. Chem. Acc.* **2011**, *130*, 515–530.

(74) Kobayashi, M.; Nakai, H. Divide-and-conquer-based linear-scaling approach for traditional and renormalized coupled cluster methods with single, double, and noniterative triple excitations. *J. Chem. Phys.* **2009**, *131*, 114108.

(75) Yuan, D.; Li, Y.; Ni, Z.; Pulay, P.; Li, W.; Li, S. Benchmark Relative Energies for Large Water Clusters with the Generalized Energy-Based Fragmentation Method. *J. Chem. Theory Comput.* **2017**, *13*, 2696–2704.

(76) Koch, H.; Christiansen, O.; Kobayashi, R.; Jørgensen, P.; Helgaker, T. A direct atomic orbital driven implementation of the coupled cluster singles and doubles (CCSD) model. *Chem. Phys. Lett.* **1994**, *228*, 233.

(77) Lee, T. J.; Rendell, A. P.; Taylor, P. R. Comparison of the quadratic configuration interaction and coupled-cluster approaches to electron correlation including the effect of triple excitations. *J. Phys. Chem.* **1990**, *94*, 5463.

(78) Rendell, A. P.; Lee, T. J.; Komornicki, A.; Wilson, S. Evaluation of the contribution from triply excited intermediates to the fourth-order perturbation theory energy on Intel distributed memory supercomputers. *Theor. Chem. Acc.* **1993**, *84*, 271.

(79) Rendell, A. P.; Lee, T. J.; Komornicki, A. A parallel vectorized implementation of triple excitations in CCSD(T): application to the binding energies of the $AlH_3$, $AlH_2F$, $AlHF_2$ and $AlF_3$ dimers. *Chem. Phys. Lett.* **1991**, *178*, 462.

(80) Neese, F. Software update: the ORCA program system, version 4.0. *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **2018**, *8*, e1327.

(81) Földes, T.; Madarász, Á.; Révész, Á.; Dobi, Z.; Varga, S.; Hamza, A.; Nagy, P. R.; Pihko, P. M.; Pápai, I. Stereocontrol in Diphenylprolinol Silyl Ether Catalyzed Michael Additions: Steric Shielding or Curtin–Hammett Scenario? *J. Am. Chem. Soc.* **2017**, *139*, 17052.

(82) Chernichenko, K.; Kótai, B.; Pápai, I.; Zhivonitko, V.; Nieger, M.; Leskelä, M.; Repo, T. Intramolecular Frustrated Lewis Pair with the Smallest Boryl Site: Reversible $H_2$ Addition and Kinetic Analysis. *Angew. Chem. Int. Ed.* **2015**, *54*, 1749.

(83) Szabó, F.; Daru, J.; Simkó, D.; Nagy, T. Z.; Stirling, A.; Novák, Z. Mild Palladium-Catalyzed Oxidative Direct ortho-C-H Acylation of Anilides under Aqueous Conditions. *Adv. Synth. Catal.* **2013**, *355*, 685.

(84) Boys, S. F.; Cook, G. B.; Reeves, C. M.; Shavitt, I. Automatic Fundamental Calculations of Molecular Structure. *Nature* **1956**, *178*, 1207.

(85) Whitten, J. L. Coulombic potential energy integrals and approximations. *J. Chem. Phys.* **1973**, *58*, 4496.

(86) Dunlap, B. I.; Connolly, J. W. D.; Sabin, J. R. On some approximations in applications of $X\alpha$ theory. *J. Chem. Phys.* **1979**, *71*, 3396.

(87) Lee, Y. S.; Kucharski, S. A.; Bartlett, R. J. A coupled cluster approach with triple excitations. *J. Chem. Phys.* **1984**, *81*, 5906.

(88) Pulay, P.; Saebø, S.; Meyer, W. An efficient reformulation of the closed-shell self-consistent electron pair theory. *J. Chem. Phys.* **1984**, *81*, 1901.

(89) Scuseria, G. E.; Janssen, C. L.; Schaefer III, H. F. An efficient reformulation of the closed-shell coupled cluster single and double excitation (CCSD) equations. *J. Chem. Phys.* **1988**, *89*, 7382.

(90) Mrcc, a quantum chemical program suite written by M. Kállay, Z. Rolik, J. Csontos, P. Nagy, G. Samu, D. Mester, J. Csóka, B. Szabó, I. Ladjánszki, L. Szegedy, B. Ladóczki, K. Petrov, M. Farkas, P. D. Mezei, and B. Hégely. See also Ref. 53 as well as http://www.mrcc.hu/ (Accessed June 15, 2018).

(91) Dunning Jr., T. H. Gaussian basis sets for use in correlated molecular calculations. I. The atoms boron through neon and hydrogen. *J. Chem. Phys.* **1989**, *90*, 1007.

(92) Weigend, F.; Köhn, A.; Hättig, C. Efficient use of the correlation consistent basis sets in resolution of the identity MP2 calculations. *J. Chem. Phys.* **2002**, *116*, 3175.

(93) Weigend, F.; Ahlrichs, R. Balanced basis sets of split valence, triple zeta valence and quadruple zeta valence quality for H to Rn: Design and assessment of accuracy integrals over Gaussian functions. *Phys. Chem. Chem. Phys.* **2005**, *7*, 3297.

(94) Rappoport, D.; Furche, F. Property-optimized Gaussian basis sets for molecular response calculations. *J. Chem. Phys.* **2010**, *133*, 134105.

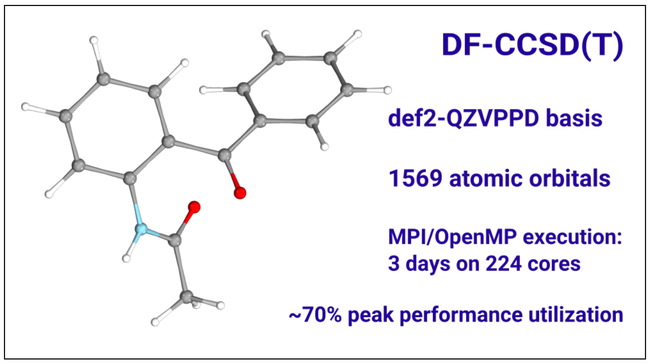
(95) Hellweg, A.; Rappoport, D. Development of new auxiliary basis functions of the Karlsruhe segmented contracted basis sets including diffuse basis functions (def2-SVPD, def2-TZVPPD, and def2-QVPPD) for RI-MP2 and RI-CC calculations. *Phys. Chem. Chem. Phys.* **2015**, *17*, 1010.

(96) Karton, A.; Martin, J. M. L. Comment on: "Estimating the Hartree–Fock limit from finite basis set calculations". *Theor. Chem. Acc.* **2006**, *115*, 330.

(97) Neese, F.; Valeev, E. F. Revisiting the Atomic Natural Orbital Approach for Basis Sets: Robust Systematic Basis Sets for Explicitly Correlated and Conventional Correlated ab initio Methods? *J. Chem. Theory Comput.* **2011**, *7*, 33.

(98) Helgaker, T.; Klopper, W.; Koch, H.; Noga, J. Basis-set convergence of correlated calculations on water. *J. Chem. Phys.* **1997**, *106*, 9639.

## Graphical TOC Entry


DF-CCSD(T)
def2-QZVPPD basis
1569 atomic orbitals
MPI/OpenMP execution:
3 days on 224 cores
~70% peak performance utilization