Title

A parallel, distributed memory implementation of the adaptive sampling configuration interaction method

Journal

The Journal of Chemical Physics, 158(21)

ISSN

0021-9606

Authors

Williams-Young, David B

Tubman, Norm M

Mejuto-Zaera, Carlos

et al.

Publication Date

2023-06-07

DOI

10.1063/5.0148650

A Parallel, Distributed Memory Implementation of the Adaptive Sampling Configuration Interaction Method

David B. Williams-Young,[1] Norm M. Tubman,[2] Carlos Mejuto-Zaera,[3] and Wibe A. de Jong[1]

[1)] *Applied Mathematics and Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, California , USA*[a)]

[2)] *NASA Ames Research Center, Moffett Field, California 94035, USA*

[3)] *Scuola Internazionale Superiore di Studi Avanzati (SISSA), Trieste TS, Italy*

(Dated: 13 March 2023)

Many-body simulations of quantum systems is an active field of research that involves many different methods targeting various computing platforms. Many methods commonly employed, particularly coupled cluster methods, have been adapted to leverage the latest advances in modern high-performance computing. Selected configuration interaction (sCI) methods have seen extensive usage and development in recent years. However development of sCI methods targeting massively parallel resources has been explored only in a few research works. In this work, we present a parallel, distributed memory implementation of the adaptive sampling configuration interaction approach (ASCI) for sCI. In particular, we will address key concerns pertaining to the parallelization of the determinant search and selection, Hamiltonian formation, and the variational eigenvalue calculation for the ASCI method. Load balancing in the search step is achieved through the application of memory-efficient determinant constraints originally developed for the ASCI-PT2 method. Presented benchmarks demonstrate parallel efficiency exceeding 95% for the variational ASCI calculation of Cr_2 (24e,30o) with $10^6, 10^7$, and $3 * 10^8$ variational determinants up to 16,384 CPUs. To the best of the authors' knowledge, this is the largest variational ASCI calculation to date.

[a)] Electronic mail: dbwy@lbl.gov

I. INTRODUCTION

The ability of *ab initio* quantum chemistry to solve challenging science problems strongly relies on its success in leveraging the latest advances in modern high-performance computing. As such, the development of massively parallel algorithms for quantum chemistry methods targeting large supercomputers has been of significant interest in recent years[1–6]. In particular, many-body methods hold among the highest potential for exploiting concurrency on modern machines (we refer the reader to Ref. 7 and references therein for a recent comprehensive review). There are several competitive methods in quantum chemistry used for many-body simulation, including density matrix renormalization group (DMRG)[8,9], coupled cluster[10,11], auxillary field quantum Monte Carlo[12–14] and several others[15,16]. It is an active field of research to understand the pros and cons of these various methods[16], and almost all of these methods are being actively developed.

Recently, the development of selected configuration interaction (sCI) methods has seen a renewed interest, and there has been a push to adapt the approach to modern computing architectures. This interest has been, in large part, driven by recent advances in the core components of sCI algorithms[17–19], which have in turn extensively broadened the reach of their applications [16,20–46]. In particular, some of the newer applications include simulation of vibrational/nuclear degrees of freedom[24,35], dynamical mean field theory[26,27], time evolution[31,32,47], Hamiltonian compression [43], new algorithms for quantum computing and benchmarking[44,48,49], machine learning approaches[22,33,41], and large-scale simulation of quantum circuits[50,51]. While there has been a lot of interest in sCI, there are still a number of technical aspects that are open research problems within the realm of high-performance computing. The lack of parallel sCI implementations has been somewhat visible as in a recent comparison of different algorithms for the application of the benzene molecule[16]. Several sCI methods were considered but none of them used anywhere close to the number of CPU hours that the most accurate methods required. Although there has been at least one simulation of sCI using more than a billion determinants[39], simulations of this scale have not been widely performed. This leads to the obvious question of what the limits of sCI approaches are when used in conjunction with modern high performance computing resources. In this work, we present a parallel, distributed memory implementation of the adaptive sampling configuration interaction (ASCI) method[18] for sCI simulations.

The ASCI algorithm was developed with the idea of making approximations to various aspects of sCI algorithms. The original ASCI paper[18] included approximations to the search algorithm that reduced the number of search terms while maintaining highly accurate results. Soon after the heatbath CI algorithm made further approximations to the search algorithm while still maintaining reasonable results in many systems[19]. Initial results on Cr_2 suggested these sCI algorithms are competitive with the most accurate contemporary many body methods, including DMRG. Many further developments increased the efficiency of different parts of the algorithm, including the development of a fast perturbation step with a deterministic constraint approach (ASCI-PT2)[23]. In this current work, we look to turn the fast ASCI-PT2 algorithm into a massively parallel search step, while also developing a scaleable parallel diagonalization step. These make up the variational part of the sCI algorithm, and thus we are hoping to advance the parallel capabilities of variational sCI.

While single core and single node approaches for sCI have been explored extensively with a focus on making use of modern computational tools[28], there has been much less focus on massive parallelism and the development of distributed memory algorithms. Most sCI implementations can make use of shared memory parallelism[28,36,39] during the computationally heavy parts of the simulation. In particular, leverage of multi-threaded[28,52,53] and GPU accelerated[23,28,54] sorting techniques, which are the primary algorithmic motifs of the ASCI method, have shown great promise on modern computing architectures. One recent study examined a distributed memory implementation of the Heat-Bath configuration interaction (HCI) method, particularly focusing on the details pertaining to the strong scaling of the second-order perturbative corrections (PT2), which dominated their simulation. While this study was able to exhibit excellent strong scaling for the PT2 step in HCI, the challenges associated with this development are somewhat unique to HCI and are not easily extended to other sCI algorithms. THe ASCI-PT2 algorithm, which was first posted in 2018[23], uses triplet constraints to facilitate the division of parallel tasks[23]. Some of the new ideas in the recent parallel HCI[38] approach are used to overcome issues that are bottlenecks for only HCI PT2 approaches[39], and are not directly applicable to ASCI-PT2 with triplet constraints. An even bigger issue that we address in this current work are the parallel bottlenecks in the variational part of an sCI algorithm, which appears to be a major bottleneck for all the sCI approaches we are aware of. This is evident in the parallel HCI paper as their method exhibited much less favorable strong scaling for the variational component of the sCI calculation.

Thus the primary focus of the following developments will be on the variational component of the ASCI algorithm, of which a fair amount can be applied to other sCI algorithms as well.

The remainder of this work will be organized as follows. In Section II A we review the salient features of the ASCI method relevant to the development of parallel algorithms. Section II B presents a work partitioning and load balancing scheme relevant to the development of scalable determinant selection on massively parallel architectures. Section II C addresses key considerations for the development of parallel algorithms for the solution of large-scale eigenvalue problems and the parallel construction of sparse Hamiltonian matrices within the sCI formalism. We examine the performance and scalability of the proposed methods for a challenging test case (Cr_2) in Sec. III and discuss future research directions in Sec. IV.

II. METHODS

The notation and common abbreviations used in this work are summarized in Table I.

A. Adaptive Sampling Configuration Interaction

The details and efficacy of the ASCI algorithm are presented elsewhere. The initial implementation can be found in Ref. 18, and many details and improvements can be found in Ref. 28. The general schematic for the ASCI algorithm is given in Alg. 1. In general, the ASCI algorithm, as most other sCI algorithms, consists of three major steps:

1. Hamiltonian construction and diagonalization (eigenvalue problem),
2. Determinant selection (search),
3. (optional) Perturbative (PT2) correction to the energy.

The first two steps are typically referred to as the variational steps for sCI algorithm and will be the primary focus of this work. However, in the development of parallel algorithms for the ASCI search step (Sec. II B), we will extend a work partitioning scheme originally developed for the ASCI-PT2 method[23]. For the variational steps, sCI algorithms primarily differ in their treatment of the determinant search, whereas once a set of determinants has been selected, the Hamiltonian construction and subsequent diagonalization steps are

Symbol	Explanation
ψ_k	The wave function in the current ASCI step
C_i	The coefficients of the ith detemi-nant D_i of ψ_k
D_i	The ith determinant in ψ_k
$\tilde{D}_j$	The jth determinant not in ψ_k
$\{D^{sd}\}$	The set of all single and double ex-citations that are connected to ψ_k
$\{D_i^{sd}\}$	The set of all single and double ex-citations that are connected to the determinant D_i
N_{tdets}	Number of determinants in the cur-rent wave function.
N_{cdets}	Core space size used for pruning in ASCI
PE	Processing element (independent compute context)

TABLE I. List of symbols used in this work.

largely the same. As such, the parallel algorithms developed in this work for the latter can straightforwardly be applied to other sCI algorithms as well. However, the details of the search step, which must be carefully considered in the development of scalable parallel algorithms, are typically unique to each sCI algorithm, and thus not straightforwardly extended between methods.

With the possible exception of a few approaches such as the machine learning sCI methods[22,33,34,41], the vast majority of sCI algorithms use the following formula for generating a ranking value of a determinant not currently considered in the trial wave function, ψ_k,

$$S_i = \sum_j S_i^{(j)}, \qquad S_i^{(j)} = \frac{H_{ij}C_j}{H_{ii} - E_k}, \qquad E_k = \langle \psi_k | H \psi_k \rangle. \tag{1}$$

Here, H is the many-body molecular Hamiltonian, H_{ij} is a matrix element of H between

1: **Input:** Start with a Hartree-Fock simulation

2: **Output:** Ground state energy of the system

3: Create a starting wave function which can be a Hartree-Fock determinant

4: ASCI-search (find important contributions not current in the wave function)

5: Sort and select the top determinants from ASCI-search

6: Diagonalize the Hamiltonian in the selected determinant space

7: Return to step 3, but now use the ground state result of the diagonalization process as the starting wave function. Repeat until stopping criteria is reached

8: Run a final ASCI-PT2 step (to improve the energy further)

Algorithm 1. ASCI algorithm

Slater determinants, C_j is the coefficient for $D_j \in \psi_k$, and E_k is the variational energy associated with ψ_k. S_i is the metric ("score") by which we will determine which determinants to add to the sCI subspace, and $S_i^{(j)}$ is the j-th partial-score which describes the contribution of D_j to S_i. The contracted index, j, runs over $D_j \in \psi_k$ whereas the free index, i, runs over $\tilde{D}_i \notin \psi_k$. In principle, i runs over the entire Hilbert space in the complement of the determinants in ψ_k. However, the elements of H_{ij} are sparse, and the maximally two-body nature of H for molecular Hamiltonians imposes $\tilde{D}_i \in \{D^{sd}\}$, where $\{D^{sd}\}$ is the set of determinants which are singly or doubly connected to a determinant in ψ_k. In practice, even consideration of all determinants in $\{D^{sd}\}$ is not needed, as there exists considerable numerical sparsity beyond that which is solely imposed by connections in the Hamiltonian. Thus, while calculating these equations is straightforward in principle, in practice it is important to identify the non-zero terms *a priori* because the large majority of terms are zero. Additionally, it is important to only use and store the non-zero terms when needed, as memory requirements for storing all the terms in these sums are extremely large when pushing to a large sCI calculation. Previous papers have reported examples pertaining to the number of connections of different examples. In a previous ASCI study, the number of such terms considered for a Cr_2 example had over 282 billion connection terms which were all generated on a single core[23] and in the recent parallel HCI paper, the authors were able to simulate over 89.5 billion connection terms. If all of these terms had to be stored in memory simultaneously, the memory requirements would make these approaches unfeasible.

Due to the immense cost of calculating H_{ij} and S_i, small differences in data structures and algorithmic design choices can yield large impacts on overall performance. Since its inception, the design approach of the ASCI algorithm has centered around the use of modern computing architectures. In particular, the ASCI method has adopted sorting as its primary algorithmic motif due to the favorable scaling and performance of sorting algorithms on contemporary CPU and GPU hardware[28]. However, current developments of the ASCI algorithm have focused on shared memory compute environments. Almost all ASCI simulations that we are aware of have been performed on a single node with the exception of a small scale MPI test with parallel ASCI-PT2[23]. In the following subsections, we examine the extension of the ASCI sorting motif to distributed memory compute environments and emergent challenges with load balancing on massively parallel architectures.

B. Parallel Determinant Search

The general procedure for the ASCI determinant search is given in Alg. 2. The search proceeds by considering single and double connections from a set of dominant configurations in the trial wave function[28]. We will refer to these determinants as the "core" determinants in the following. The number of core determinants to consider is a tunable parameter in the ASCI algorithm and is typically achieved by selecting the N_{cdets} determinants in ψ_k with the largest coefficient magnitudes. In a naive approach, one determines *a priori* the $\{D^{sd}\}$ associated with the core determinants, which we will refer to as the *target* determinants in the following, and subsequently calculates the contribution of each core determinant to each target configuration via Eq. (1). In many practical simulations, it is better to generate the possible target determinants as singles and doubles from each core configuration, rather than loop over all possible target configurations and then check for connections with the core determinants. The latter procedure generally performs excessive work as the majority of the core determinants only contribute to a few target determinants. In the shared memory ASCI algorithm, a single pass is taken over the core determinants, and the associated $\{D_j^{sd}\}$ and $S_i^{(j)}$ for $\tilde{D}_i \in \{D_j^{sd}\}$ are generated on the fly. If $|S_i^{(j)}| > \epsilon_{search}$, where ϵ_{search} is a tunable threshold, it is appended to an array as a pair $(\tilde{D}_i, S_i^{(j)})$, which we will refer to as an ASCI pair in the following. For the purposes of molecular calculations, bit-strings are typically the data structure of choice for the representation and manipulation of many-

Input: Trial wave function determinants, ψ_k and coefficients C; Search configuration cutoff N_{cdets}; Max wave function size N_{tdets}; ASCI pair threshold, ϵ_{search}
Output: New ASCI determinants ψ_{k+1}
$\psi_c \leftarrow$ Obtain N_{cdets} subset of ψ_k with largest coefficients
$P \leftarrow [\,]$ ▷ ASCI Pairs
for $D_j \in \psi_c$ **do**
    $\{D_j^{sd}\} \leftarrow$ Single and double connections from D_j
    for $\tilde{D}_i \in \{D_j^{sd}\}$ **do**
        $S_i^{(j)} \leftarrow$ Eq. (1)
        if $|S_i^{(j)}| > \epsilon_{thresh}$ **then**
            $P \leftarrow [P,\ (\tilde{D}_i, S_i^{(j)})]$
        end if
    end for
end for
$P \leftarrow$ Sort P on $\tilde{D}_i$.
$P \leftarrow \sum_j S_i^{(j)}$ for each unique i ▷ Each unique $p \in P$ now contains a complete S_i
$P \leftarrow$ Sort P on S_i
return $\psi_{k+1} \leftarrow$ top-N_{tdets} determinants in P

Algorithm 2. The ASCI Search Algorithm

body determinants as it allows for fast bit operations in e.g. calculating matrix elements on modern computing architectures[17,28,36,38,55]. As such, the ASCI pairs can be stored in a simple data structure consisting of the bitsring representing $\tilde{D}_i$ and the partial score $S_i^{(j)}$. Once all pairs have been generated after passing over each core configuration, the array is sorted on the determinant bitstring which in turn brings all partial scores associated with a particular target determinant contiguously in memory. With this new ordering, scores for individual target determinants can be straightforwardly assembled via Eq. (1). We refer to this procedure as "sort-and-accumulate" in the following.

Even for relatively aggressive ϵ_{search} cutoffs, this approach can constitute a large memory footprint for larger N_{cdets}. In the original ASCI method, a threshold for the maximum number of ASCI pairs to store in memory was set, and the pair list would be occasionally pruned

to ensure memory compactness. This pruning procedure has been demonstrated to provide approximate scores of sufficient accuracy to successfully rank determinants for the sCI wave function[28]. A similar approach could be taken for the calculation of the PT2 energy, E_{PT2}, which is closely related to the scores of Eq. (1)[20,23,39]. However, for accurate calculations of E_{PT2}, one cannot employ the aggressive cutoffs and pruning schemes employed to ensure memory compactness in the ASCI search. In the development of the ASCI-PT2 method[23], a solution was developed which allows for memory compactness while still leveraging the powerful nature of sorting algorithms on modern computing architectures. In this procedure, the set of target configurations is partitioned according to a constraint identified by their largest occupied orbitals. In the ASCI-PT2 study, target determinants were classified by triplet constraints, i.e., by their three highest-occupied orbitals. These constraints generate non-overlapping subsets of determinants such that every possible determinant belongs to exactly one constraint class.

At first glance, this approach appears similar to the naive approach for ASCI score calculation as it might imply that one would need to know the entire $\{D^{sd}\}$ associated with the core determinants to be able to partition the target determinants *a priori*. However, the major realization of ASCI-PT2 was to determining which target determinants arising from excitations of a particular core configuration belong to a particular constraint class. As such, in a similar manner to the original ASCI search algorithm, one can make multiple passes over the core configurations for each constraint and generate its associated target configurations on the fly. The power of this procedure is that the sort-and-accumulate steps can now be done at the constraint level, i.e., ASCI pairs can be generated and aggregated for each constraint and, due to the non-overlapping partition generated by the individual constraints, all partial scores associated with the target determinant belonging to that class are guaranteed to be generated by the end of each pass over core configurations. As the number of unique determinant scores is much less than partial score contributions, this procedure exhibits much lower memory usage than the original ASCI search algorithm.

To proceed with development of a parallel ASCI search algorithm we consider a similar approach to ASCI-PT2, where the work associated with the generation of pairs for individual constraints is assigned to different processing elements (PE) in the distributed memory environment. To ensure scalability of the pair generation and subsequent sort-and-accumulate steps, we have developed a static load balancing procedure presented in Alg. 3. Prior to

Input (global): Dominant core configurations, ψ_c , max constraint level L
Output (local): Local constraints C_{loc}
$W \leftarrow$ an array of size of the execution context (# PEs) ▷ Initialize to zero
$p \leftarrow$ index of PE in
$C_{loc} \leftarrow [\,]$
$C_t \leftarrow$ all unique triplet constraints
for $C \in C_t$ **do**
    $h \leftarrow 0$
    for $D_j \in \psi_c$ **do**
        $s \leftarrow$ singly connected determinants to D_j satisfying C.
        $d \leftarrow$ doubly connected determinants to D_j satisfying C.
        $h \leftarrow h + |s| + |d|$
    end for
    if $h > 0$ **then**
        $q \leftarrow \arg\min_i W_i$
        if $p = q$ **then**
            $C_{loc} \leftarrow [C_{loc},\ C]$
        end if
    end if
end for
return C_{loc}

Algorithm 3. Load Balancing for Parallel Constraint ASCI Search

any pairs being generated for the ASCI search, the number of determinants associated with individual constraints is precomputed on the fly to generate the rough amount of work for each constraint. The number of contributions is then used as a heuristic to assign work to individual PEs. It's important to note that the number of contributions is not an exact measure of the work associated with a particular constraint due to the fact that contributions are screened out according to ϵ_{search}. However, for the purposes of the ASCI search, this procedure has been demonstrated to generate sufficiently balanced work (see Sec. II B). The primary challenge of this work distribution scheme stems from the large variance of work

associated with any particular constraint, i.e. it is often the case that a few constraints yield a disproportionate of target determinants. As such, at a particular level of constraint (e.g. triplets, etc), there always exists an amount of indivisible work that the load balancing procedure cannot distribute over PEs. The work associated with a particular constraint can be further divided by considering addition constraints of higher order, e.g. triplets can be broken up into quadruplets, quadruplets into quintuplets, etc. However, the number of constraints at a particular level grows polynomially with the number of single particle orbitals, e.g. $O(n_{orb}^3)$ for triplets, $O(n_{orb}^4)$ for quadruplets, etc. Due to the fact that for each constraint there is an inner loop over N_{cdets} core configurations, it is highly desirable to limit the number of considered constraints. As such, it would be highly inefficient to simply consider all higher-level constraints if the work associated with a particular constraint level is deemed to exhibit excessive load imbalance. Instead, it is advantageous to only divide the constraints which are expected to exhibit excessive work into higher constraint levels rather than dividing all constraints. We will demonstrate the efficacy of this partitioning scheme in Sec. II B.

Given the locally constructed arrays of string-score pairs generated as a result of the constraint partitioning, the remaining aspect of the parallel ASCI search is to determine the dominant N_{tdets} determinants to keep for the next iteration. In principle, this can be achieved by performing a full sort of the pairs on the absolute value of their scores. For an array A such that $|A| = n$, full sorting of A scales $O(n \log n)$, which is generally unfavorable for the large sizes of arrays considered in this work (as would be generated by large values of N_{cdets}). For distributed memory implementations, this problem would be exacerbated due to the communication overhead of existing distributed memory sort algorithms[52,56]. For the vast majority of properties involving selected-CI wave functions, such as energies and density matrices, the *order* in which the determinant coefficients appear in the CI vector is irrelevant and all computed quantities will be invariant to arbitrary permutations $C \leftarrow PC$. As such, the sorting problem can be replaced with the *selection* problem which can be used to determine the largest (or smallest) $k \leq n$ values of an array with $O(n)$ complexity [57,58]. In addition, selection algorithms can be performed in parallel with nearly optimal speedup without having to communicate significant segments of A[59]. Rather than obtaining an absolute ordering of its input data, selection algorithms are designed to determining a *pivot*, $a_k \in A$, such that $|A^g| = k$, where $A^g = \{a > a_k \,|\, a \in A\}$. In cases where a_k appears multiple

times in A, this definition can be extended to indicate $|A^g| < k \leq |A^g \cup A^e|$, where $A^e \subset A$ is the indexed subset containing the dubplicate elements of a_k. We outline a duplicate-aware, distributed memory extension of the `quickselect` algorithm, with expected $O(n)$ runtime, in Alg. 4. For the ASCI search problem, Alg. 4 is used to determine the contribution pair with the N_{tdets}-largest score, p_{tdets}. We may then determine the contribution pairs with scores larger-than or equal-to p_{tdets} and subsequently gather them (collectively) to all participating PEs (via e.g. `MPI_Allgather(v)`). We examine the performance of this scheme, which introduces the only source of distributed memory communication in the presented ASCI search method, in Sec. III.

C. Parallel Eigensolver

After each search iteration of the ASCI procedure, we must obtain the ground state eigenpair of the many-body Hamiltonian projected onto the basis of newly selected determinants. The large basis dimensions (N_{tdets}) employed in accurate sCI applications precludes the use of direct eigensolvers (e.g. those implemented in dense linear algebra libraries such as (Sca)LAPACK[60] and ELPA[61,62]) due to their $O(N^2)$ dense storage requirement and steep $O(N^3)$ scaling with problem dimension. As such, Krylov-subspace methods, such as Davidson[63], LOBPCG[64,65], and Lanczos[66], are typically employed. Development of efficient and scalable Krylov methods on distributed memory computers is challenging due to the existence of many synchronization points arising from the serial nature of subspace construction. Significant research effort has been afforded to the development of distributed memory Krylov eigenvalue methods across many scientific computing domains[67–76]. In this work, due to the diagonally dominant nature of the Hamiltonian, we consider the parallel implementation of the Davidson method for sCI applications, although many of the same principles can be extended to other Krylov methods such as Lanczos or LOBPCG.

Given an algorithm to produce the action of the Hamiltonian onto a trial vector (σ formation), we present the general scheme for the distributed memory Davidson method in Alg. 5. The algorithm presented is general to an arbitrary preconditoner, although we will always adopt the shifted Jacobi (diagonal) preconditioner which has become standard in configuration interaction calculations[63]. Unlike exact diagonalization configuration interaction methods (e.g. full/complete active space CI), where implicit σ formation exhibits efficient

closed-form expressions[1,55], the vast majority of sCI methods require explicit construction of H_{ij} to perform the necessary contractions. We refer the reader to Ref. 28 for a comprehensive discussion of efficient algorithms for the assembly of H_{ij} for sCI wave functions. Although explicit storage of the Hamiltonian can be avoided via recalculation of the H_{ij} at each Davidson iteration, the immense cost of matrix element construction renders this procedure prohibitive for practical applications. Sparsity of the the Hamiltonian in the basis of determinants allow for its explicit storage using sparse matrix storage formats. In this work, we will use the distributed sparse matrix format depicted in Fig. 1, which has been employed for the development of parallel Krylov algorithms in many community software packages for sparse matrix algebra (e.g. Ref. 77). In this storage scheme, each PE owns a contiguous (not necessarily equal) row subset of the full sparse matrix divided into diagonal and off-diagonal blocks which are stored in standard sparse matrix formats. We will use the compressed sparse row (CSR) format for local matrix storage[78]. The partitioning into diagonal and off-diagonal blocks will ultimately determine the communication pattern of the σ formation in the Davidson method, as will be apparent in Sec. II C. Of the several methods for matrix element construction discussed in Ref. 28, we consider the double loop approach in this work. In the limit that each task only has a small portion of the Hamiltonian, the double loop approach is quick at generating all the matrix elements. There is likely time savings to be had by optimizing this further with consideration to other approaches of generating the matrix elements like with residue arrays and dynamic bitmasking[28]. However, the cost analysis must take into account inefficiencies due to load balancing. Thus a double loop approach is quite reasonable in terms of load balancing and we consider it a reasonable starting point for massively parallel approaches to sCI.

A naive approach for the distributed σ formation would replicate the basis vectors on each PE such that local sparse matrix-vector (SpMV) product could be done without communication. However, in the context of a Krylov method such as Davidson, this would require gathering the locally computed elements of the SpMV to *every* PE at each iteration. For large problem sizes, the connectivity of the full gather operation would become a bottleneck. Due to the sparsity of H, the computation of individual elements of σ does not require knowledge of the entire basis vector; in fact, the number of elements required for the contraction of *any* row of H is (pessimistically) bounded by the number of single and double exciations allowable from the single particle basis. As such, replication (or commu-

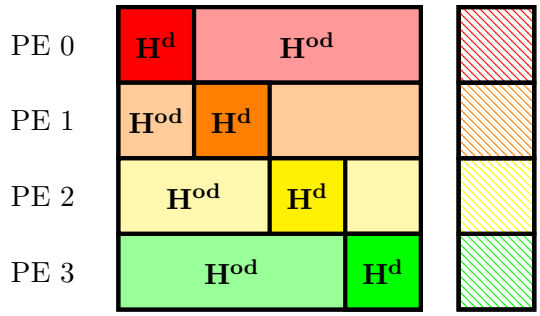

FIG. 1. Example distributed memory storage scheme for the Hamiltonian and Davidson trial vectors across 4 PEs. The diagonal ($\mathbf{H^d}$, darker) and off-diagonal ($\mathbf{H^{od}}$, lighter) blocks of H are stored separately on each PE as CSR matrices. For PEs 1 and 2, the off-diagonal blocks are stored as a single sparse matrix with elements on both sides of the diagonal block. Vectors (textured) are distributed according to the same row distribution as H.

nication) of the entire vector is not generally required. To minimize data communication in the Davidson iterations, we employ the vector partitioning scheme also depicted in Fig. 1. Using the sparsity pattern of the off-diagonal blocks of H, we can efficiently determine which elements of remote vector elements need to be communicated between PEs as the union of the row sparsities, i.e.

$$\mathcal{C}_I = \bigcup_{i \text{ on PE } I} \mathcal{R}_i, \qquad \mathcal{R}_i = \{j \,|H^{\text{od}}_{ij} \neq 0\}. \tag{2}$$

From this set, the owners of remote matrix elements can be looked-up using the row distribution of H. Another benefit of this precomputation is that it allows for the overlap of communication and computation[79]. Via this distribution, the diagonal SpMV can be done without communication as the vector elements for the local SpMV reside on the same PE by construction. Using non-blocking communication primitives in the MPI library, communication can be initiated prior to starting the diagonal SpMV and finalized only once the remote elements are required for the off-diagonal SpMV. The pseudocode for this distributed SpMV employed in this work is given in Alg. 6.

III. RESULTS AND DISCUSSION

In this section we examine the strong scaling behaviour of the proposed algorithms for Cr_2 / def-SVP[80] in an active space of (24e, 30e). This test system is one of the challenge

systems that has effectively been solved to chemical accuracy, but only in the last 10 years[9,18]. Therefore Cr_2 is a a good benchmark systems for testing and developing methods modern method. Molecular integrals were obtained using the Molpro[81,82] program and all ASCI calculations were performed on the Haswell partition of the Cray XC40 Cori supercomputer at the National Energy Research Scientific Computing Center (NERSC). Each Cori-Haswell node consists of 2x Intel Xeon Processor E5-2698 v3 16-core CPUs with 128GB DDR4 RAM. All jobs were run with 32 PEs (MPI ranks) per node. Energies for Cr_2 calculations presented in this work are given in Tab. II. All calculations were performed using $\epsilon_{search} = 10^{-10}$. Although not the primary focus of this work, we note that the variational energy determined at $N_{tdets} = 3{*}10^8$ is only 0.3 mEh higher in energy than the most accurate variational DMRG result for the same system in Ref. 83.

In the following, we present strong scaling results for the parallel ASCI methods proposed in this work.

	N_{cdets}	
E_0/ Eh	100k	300k
$N_{tdets} = 10^6$	-2086.40966	-2086.41018
$N_{tdets} = 10^7$	-2086.41769	-2086.41781
$N_{tdets} = 3 * 10^8$	–	-2086.42042
Reference DMRG[83]	-2086.42077	

TABLE II. ASCI Ground State energies for Cr_2 (24e,30o)

A. ASCI Search Performance

In Fig. 2, we present the component and overall strong scaling of the ASCI search for triplet, quadruplet, quintuplet, and hextuplet determinant constraints (Sec. II B) for Cr_2 with $N_{cdets} = 300k$ and $N_{tdets} = 10^6, 10^7$. The calculation is dominanted by the generation of ASCI pairs whereas the distributed determinant selection via Alg. 4 is relatively minor in comparison (< 1s). As such, the execution time of the ASCI search is largely independent of N_{tdets}. At the lowest determinant constraint level (triplets), the presented algorithm stagnates immediately with no further performance improvement with large PE counts.

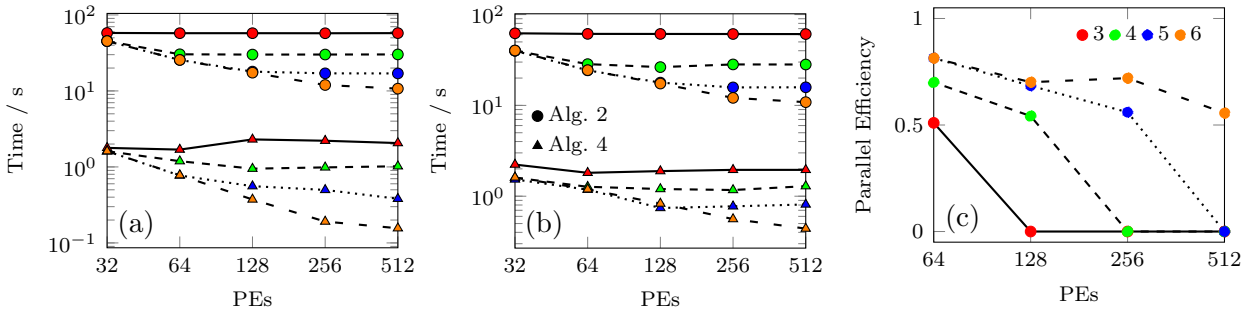

FIG. 2. Strong scaling of ASCI search components with different determinant constraints (triplet, 3; quadruplet, 4; qunituplet, 5; hextuplet, 6) for $N_{cdets} = 300k$ and (a) $N_{tdets} = 10^6$ (b) $N_{tdets} = 10^7$. Dominant components exhibit improved scaling with the addition of larger determinant constraints while lesser components are largely unaffected. (c) Illustrates the parallel efficiency of the overall ASCI search algorithm with different constraints.

Strong scaling is improved significantly by using larger determinant constraints. Overall, we see a 5-5.5x performance improvement in the strong scaling limit by moving from triplet to hextuple determinant constraints. The reason for this improvement (or conversely why the triplet case exhibits no strong scaling at all) can be seen in Fig. 3 where we present a plot showing the effect of constraint size on the number of ASCI pairs generated per-PE. At the triplet level, there is a single triplet which yields a disproportinate number of pair contributions. As the work can only be distributed at the level of the highest determinant constraint, this single triplet is represents indivisible work. By breaking up the offending triplet into higher constraints, the work is able to be distributed. However, at each constraint level, there always reaches a point where there is a disproportionate amount of work for a few particular constraints. As such, we expect there always to be a strong scaling stagnation for this procedure, but the inclusion of higher constraints (heptuples, octuples, etc) will continue to improve the strong scaling behaviour at larger PE counts. However, due to the relatively low cost of the search relative to the Hamiltonian formation and Davidson iterations (see Sec. III B), we believe this level of optimization to be unnecessary at this time.

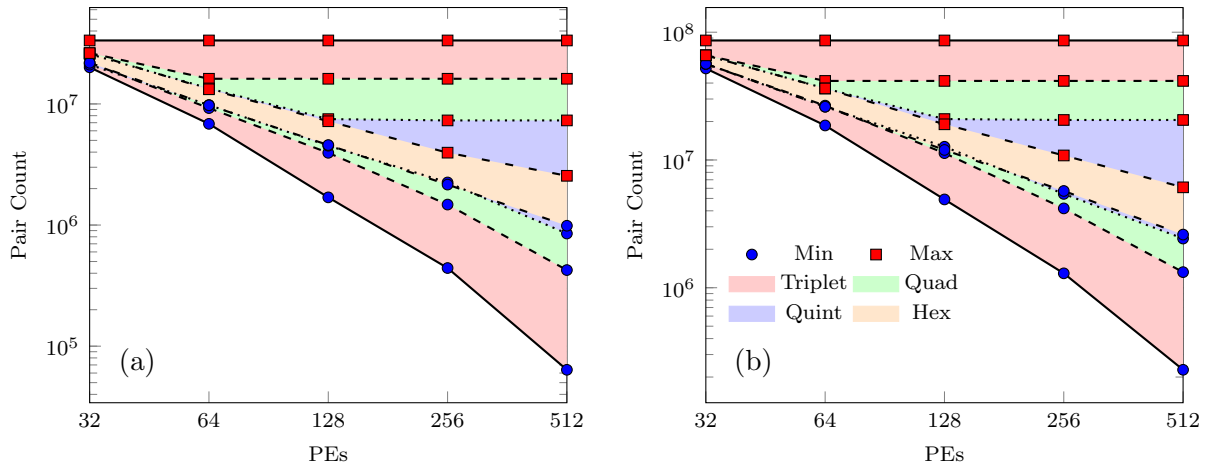

FIG. 3. Constraint pair count statistics for Cr_2 with $N_{tdets} = 10^7$ and (a) $N_{cdets} = 100k$ (b) $N_{cdets} = 300k$. Triplet (3, red shading), quadruplet (4, green shading), quintuplet (5, blue shading), and hextuplets (6, orange scaling) constraints are considered. A smaller shaded volume indicates better strong scaling of the constraint-partitioned ASCI search.

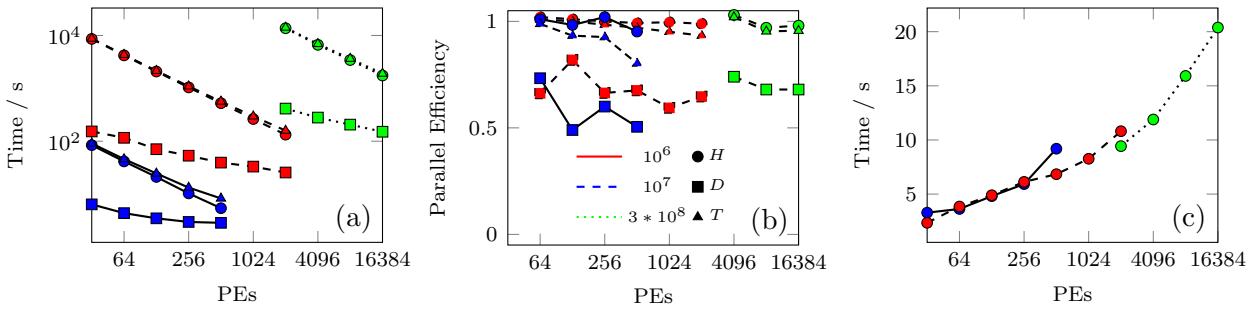

FIG. 4. Strong scaling of $N_{tdets} = 10^6, 10^7$, and $3 * 10^8$ ASCI. (a) Illustrates the strong scaling of the Hamiltonian generation, Davidson iterations and overall application performance for each system while (b) illustrates the corresponding parallel efficiency. (c) Plots the ratio the the largest to the smallest number of non-zero matrix elements over the PEs, which explains the imbalance in the Davidson iterations.

B. Hamiltonian and Eigensolver Performance

Figure 4 illustrates the strong scaling of the sCI Hamiltonian generation and subsequent Davidson iterations for Cr_2 with $N_{cdets} = 300k$ and $N_{tdets} = 10^6, 10^7$ and $3 * 10^8$. It is clear that the cost of Hamiltonian formation dominates the variational ASCI calculation at

all N_{tdets} considered, and is $O(10x)$ the cost of the Davidson iterations and $O(100x)$ the cost of the ASCI search discussed in the previous subsection. The Hamiltonian formation exhibits near linear strong scaling ($> 98\%$ parallel efficiency), whereas the Davidson iterations exhibit between 50%-80% parallel efficiency. The scaling of the former is expected, as each PE performs nearly the same number of determinant distance checks in the double loop method (Sec. II C) and there is no communication between PEs. The scaling of the Davidson iterations can be understood in terms of Fig. 4(c) where the source of the load imbalance can be attributed to the variance of non-zero matrix elements among PE. This is due to the fact that the ordering of the determinants in the ASCI wave function, as produced by the parallel ASCI search, is more or less random, and thus unable to guarantee regular distributions of non-zero matrix elements. This problem is well known in the field of distributed memory sparse linear algebra[56], as both the local work and overall communication volume of the parallel SpMV (6) are affected by irregular distributions of non-zero matrix elements. Improving the communication characteristics of the sCI SpMV will be the subject of future work by the authors. Figure 4 also shows the overall parallel efficiency of the ASCI application for the different problem sizes. As the calculation is dominated by the Hamiltonian construction, we see similar parallel efficiencies for the total application, particularly for the larger problems. Overall, the ASCI calculation maintains >85% parallel efficiency for the smallest problem (10^6) which maintaining >95% parallel efficiency for the largest problem at $3 * 10^8$ variational determinants out to 16,384 PEs. To the best of the authors' knowledge, this is the largest variational ASCI calculation to date.

IV. CONCLUSIONS

In this work, we have proposed and demonstrated a parallel, distributed memory extension for the variational ASCI method. We addressed key concerns for the dominant steps of the ASCI algorithm, including the development of a parallel algorithm for the determinant selection, a parallel algorithm and sparse storage scheme for the sCI Hamiltonian, and a parallel algorithm for the variational eigenvalue calculation via the Davidson method. The parallel search algorithm was enabled by previous work in the context of ASCI-PT2, where the introduction of determinant constraints offered a conventient and robust mechanism to distribute work across PEs. This method was extended to include higher order determi-

nant constraints than were considered for the ASCI-PT2 method and coupled with a load balancer to ensure scaling on large compute resources. We have demonstrated the efficacy and performance of the proposed algorithms on a challenging test system (Cr_2) on up to 16,384 cores. The use of higher-order determinant constraints yielded a 5-5.5x performance improvement in the ASCI search (triplet constraints) in the strong scaling limit. The overall ASCI calculation was demonstrated to maintain 85% - 95% parallel efficiency. In addition we have demonstrated stability and continued convergence of the ASCI method into the regime of hundred-of-millions of determinants, which to the best of the authors' knowledge, is the largest variational ASCI calculation to date. Although the developments for the parallel search algorithm are ASCI specific, the developments pertaining to distributed Hamiltonian construction and Davidson diagonalization are directly applicable to other sCI methods as well. These developments indicate a promising future for massively parallel sCI applications in the years to come.

While these results are promising, there remain open areas for exploration which will be addressed by the authors in future work. In this work, the Hamiltonian construction dominates the overall ASCI application by at least an order of magnitude. Application of more advanced Hamiltonian matrix element techniques[28] for the local sparse matrix construction offer high-potential for acheiving further improvements in the future although it is unclear whether such developed will introduct load imbalance. The interplay between efficient Hamiltonian construction and load imbalance warrants further exploration. In addition, the strong scaling stagnation of the Davidson iterations can be attributed primarily to massive load imbalance in the non-zero element distribution across PEs. In our experience, the existing technologies[84,85] for reordering sparse matrices to minimize communication volume are insufficient for producing balanced partitions for molecular CI calculations. Development of scalable graph reordering techniques for the CI problem would be a particular fruitful area of exploration given massively parallel implementations of sCI. Lastly, the leverage of bit operations both in the search and Hamiltonian construction steps of the ASCI algorithm will be particularly advantageous on current and future GPU architectures and possibly other hardware accelerators to come. As such, the authors will build upon initial results[28] for GPU applications to the ASCI method and develop GPU accelerated extensions of the parallel algorithms proposed in this work.

V. ACKNOWLEDGMENT

DWY and WAdJ were supported from the Center for Scalable Predictive methods for Excitations and Correlated phenomena (SPEC), which is funded by the U.S. Department of Energy (DoE), Office of Science, Office of Basic Energy Sciences, Division of Chemical Sciences, Geosciences and Biosciences as part of the Computational Chemical Sciences (CCS) program at Lawrence Berkeley National Laboratory under FWP 12553. NMT is grateful for support from NASA Ames Research Center. CMZ acknowledges financial support from the European Research Council (ERC), under the European Union's Horizon 2020 research and innovation programme, Grant agreement No. 692670 "FIRSTORM". This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility located at Lawrence Berkeley National Laboratory, operated under Contract No. DE-AC02-05CH11231 using NERSC award BES-ERCAP-M3196. NMT acknowledges funding from the NASA ARMD Transformational Tools and Technology (TTT) Project. Some calculations were performed as part of the XSEDE computational Project No. TG-MCA93S030 on Bridges-2 at the Pittsburgh supercomputer center.

REFERENCES

[1]K. D. Vogiatzis, D. Ma, J. Olsen, L. Gagliardi, and W. A. de Jong, "Pushing configuration-interaction to the limit: Towards massively parallel mcscf calculations," J. Chem. Phys. **147**, 184111 (2017).

[2]C. L. Janssen and I. M. Nielsen, *Parallel Computing in Quantum Chemistry* (2008).

[3]W. A. De Jong, E. Bylaska, N. Govind, C. L. Janssen, K. Kowalski, T. Müller, I. M. Nielsen, H. J. van Dam, V. Veryazov, and R. Lindh, "Utilizing high performance computing for chemistry: parallel computational chemistry," Physical Chemistry Chemical Physics **12**, 6896–6920 (2010).

[4]M. S. Gordon and T. L. Windus, "Editorial: Modern architectures and their impact on electronic structure theory," Chemical Reviews **120**, 9015–9020 (2020).

[5]D. Kothe, S. Lee, and I. Qualters, "Exascale computing in the united states," Computing in Science & Engineering **21**, 17–29 (2019).

[6]F. Alexander, A. Almgren, J. Bell, A. Bhattacharjee, J. Chen, P. Colella, D. Daniel, J. DeSlippe, L. Diachin, E. Draeger, A. Dubey, T. Dunning, T. Evans, I. Foster, M. Francois, T. Germann, M. Gordon, S. Habib, M. Halappanavar, S. Hamilton, W. Hart, Z. (Henry) Huang, A. Hungerford, D. Kasen, P. R. C. Kent, T. Kolev, D. B. Kothe, A. Kronfeld, Y. Luo, P. Mackenzie, D. McCallen, B. Messer, S. Mniszewski, C. Oehmen, A. Perazzo, D. Perez, D. Richards, W. J. Rider, R. Rieben, K. Roche, A. Siegel, M. Sprague, C. Steefel, R. Stevens, M. Syamlal, M. Taylor, J. Turner, J.-L. Vay, A. F. Voter, T. L. Windus, and K. Yelick, "Exascale applications: skin in the game," Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences **378**, 20190056 (2020).

[7]J. A. Calvin, C. Peng, V. Rishi, A. Kumar, and E. F. Valeev, "Many-body quantum chemistry on massively parallel computers," Chemical Reviews **121**, 1203–1231 (2021).

[8]U. Schollwöck, "The density-matrix renormalization group," Rev. Mod. Phys. **77**, 259–315 (2005).

[9]R. Olivares-Amaya, W. Hu, N. Nakatani, S. Sharma, J. Yang, and G. K.-L. Chan, "The ab-initio density matrix renormalization group in practice," J. Chem. Phys. **142**, 034102 (2015).

[10]R. J. Bartlett and M. Musiał, "Coupled-cluster theory in quantum chemistry," Rev. Mod. Phys. **79**, 291–352 (2007).

[11]E. Solomonik, D. Matthews, J. R. Hammond, J. F. Stanton, and J. Demmel, "A massively parallel tensor contraction framework for coupled-cluster computations," Journal of Parallel and Distributed Computing **74**, 3176–3190 (2014), domain-Specific Languages and High-Level Frameworks for High-Performance Computing.

[12]G. H. Booth, A. J. W. Thom, and A. Alavi, "Fermion monte carlo without fixed nodes: A game of life, death, and annihilation in slater determinant space," J. Chem. Phys. **131**, 054106 (2009).

[13]C.-C. Chang, B. M. Rubenstein, and M. A. Morales, "Auxiliary-field-based trial wave functions in quantum monte carlo calculations," Phys. Rev. B **94**, 235144 (2016).

[14]M. Motta and S. Zhang, "Ab initio computations of molecular systems by the auxiliary-field quantum monte carlo method." WIREs Comput Mol Sci , 1–29 (2018).

[15]A. Szabo and N. Ostlund, *Modern Quantum Chemistry* (Dover, 1982).

[16]J. J. Eriksen, T. A. Anderson, J. E. Deustua, K. Ghanem, D. Hait, M. R. Hoffmann, S. Lee, D. S. Levine, I. Magoulas, J. Shen, N. M. Tubman, K. B. Whaley, E. Xu, Y. Yao, N. Zhang, A. Alavi, G. K.-L. Chan, M. Head-Gordon, W. Liu, P. Piecuch, S. Sharma, S. L. Ten-no, C. J. Umrigar, and J. Gauss, "The ground state electronic energy of benzene," The Journal of Physical Chemistry Letters **11**, 8922–8929 (2020), pMID: 33022176, https://doi.org/10.1021/acs.jpclett.0c02621.

[17]F. A. Evangelista, "Adaptive multiconfigurational wave functions," J. Chem. Phys. **140**, 124114 (2014).

[18]N. M. Tubman, J. Lee, T. Y. Takeshita, M. Head-Gordon, and K. B. Whaley, "A deterministic alternative to the full configuration interaction quantum monte carlo method," The Journal of Chemical Physics **145**, 044112 (2016), https://doi.org/10.1063/1.4955109.

[19]A. A. Holmes, N. M. Tubman, and C. J. Umrigar, "Heat-bath configuration interaction: An efficient selected configuration interaction algorithm inspired by heat-bath sampling," J. Chem. Theory Comput. **12**, 3674–3680 (2016), http://dx.doi.org/10.1021/acs.jctc.6b00407.

[20]Y. Garniron, A. Scemama, P.-F. Loos, and M. Caffarel, "Hybrid stochastic-deterministic calculation of the second-order perturbative contribution of multireference perturbation theory," J. Chem. Phys. **147**, 034101 (2017), https://doi.org/10.1063/1.4992127.

[21]P. M. Zimmerman, "Incremental full configuration interaction," The Journal of Chemical Physics **146**, 104102 (2017), https://doi.org/10.1063/1.4977727.

[22]J. P. Coe, "Machine learning configuration interaction," Journal of Chemical Theory and Computation **14**, 5739–5749 (2018), pMID: 30285426, https://doi.org/10.1021/acs.jctc.8b00849.

[23]N. M. Tubman, D. S. Levine, D. Hait, M. Head-Gordon, and K. B. Whaley, "An efficient deterministic perturbation theory for selected configuration interaction methods," arXiv preprint arXiv:1808.02049 (2018).

[24]E. Lesko, M. Ardiansyah, and K. R. Brorsen, "Vibrational adaptive sampling configuration interaction," The Journal of Chemical Physics **151**, 164103 (2019), https://doi.org/10.1063/1.5126510.

[25]J. W. Park, "Second-order orbital optimization with large active spaces using adaptive sampling configuration interaction (asci) and its application to molecular geometry optimization," Journal of Chemical Theory and Computation **17**, 1522–1534 (2021), pMID:

33630610, https://doi.org/10.1021/acs.jctc.0c01292.

[26]C. Mejuto-Zaera, N. M. Tubman, and K. B. Whaley, "Dynamical mean field theory simulations with the adaptive sampling configuration interaction method," Phys. Rev. B **100**, 125165 (2019).

[27]C. Mejuto-Zaera, L. Zepeda-Núñez, M. Lindsey, N. Tubman, B. Whaley, and L. Lin, "Efficient hybridization fitting for dynamical mean-field theory via semi-definite relaxation," Phys. Rev. B **101**, 035143 (2020).

[28]N. M. Tubman, C. D. Freeman, D. S. Levine, D. Hait, M. Head-Gordon, and K. B. Whaley, "Modern approaches to exact diagonalization and selected configuration interaction with the adaptive sampling ci method," Journal of Chemical Theory and Computation **16**, 2139–2159 (2020).

[29]D. S. Levine, D. Hait, N. M. Tubman, S. Lehtola, K. B. Whaley, and M. Head-Gordon, "Casscf with extremely large active spaces using the adaptive sampling configuration interaction method," Journal of Chemical Theory and Computation **16**, 2340–2354 (2020).

[30]D. S. Levine, D. Hait, N. M. Tubman, S. Lehtola, K. B. Whaley, and M. Head-Gordon, "Casscf with extremely large active spaces using the adaptive sampling configuration interaction method," Journal of Chemical Theory and Computation **16**, 2340–2354 (2020), pMID: 32109055, https://doi.org/10.1021/acs.jctc.9b01255.

[31]V. Kremenetski, T. Hogg, S. Hadfield, S. J. Cotton, and N. M. Tubman, "Quantum Alternating Operator Ansatz (QAOA) Phase Diagrams and Applications for Quantum Chemistry," arXiv e-prints , arXiv:2108.13056 (2021), arXiv:2108.13056 [quant-ph].

[32]V. Kremenetski, C. Mejuto-Zaera, S. J. Cotton, and N. M. Tubman, "Simulation of adiabatic quantum computing for molecular ground states," The Journal of Chemical Physics **155**, 234106 (2021), https://doi.org/10.1063/5.0060124.

[33]S. D. Pineda Flores, "Chembot: A machine learning approach to selective configuration interaction," Journal of Chemical Theory and Computation **17**, 4028–4038 (2021), pMID: 34125549, https://doi.org/10.1021/acs.jctc.1c00196.

[34]J. J. Goings, H. Hu, C. Yang, and X. Li, "Reinforcement learning configuration interaction," Journal of Chemical Theory and Computation **17**, 5482–5491 (2021), pMID: 34423637, https://doi.org/10.1021/acs.jctc.1c00010.

[35]A. U. Bhatty and K. R. Brorsen, "An alternative formulation of vibrational heat-bath configuration interaction," Molecular Physics **119**, e1936250 (2021),

https://doi.org/10.1080/00268976.2021.1936250.

[36]E. Epifanovsky, A. T. B. Gilbert, X. Feng, J. Lee, Y. Mao, N. Mardirossian, P. Pokhilko, A. F. White, M. P. Coons, A. L. Dempwolff, Z. Gan, D. Hait, P. R. Horn, L. D. Jacobson, I. Kaliman, J. Kussmann, A. W. Lange, K. U. Lao, D. S. Levine, J. Liu, S. C. McKenzie, A. F. Morrison, K. D. Nanda, F. Plasser, D. R. Rehn, M. L. Vidal, Z.-Q. You, Y. Zhu, B. Alam, B. J. Albrecht, A. Aldossary, E. Alguire, J. H. Andersen, V. Athavale, D. Barton, K. Begam, A. Behn, N. Bellonzi, Y. A. Bernard, E. J. Berquist, H. G. A. Burton, A. Carreras, K. Carter-Fenk, R. Chakraborty, A. D. Chien, K. D. Closser, V. Cofer-Shabica, S. Dasgupta, M. de Wergifosse, J. Deng, M. Diedenhofen, H. Do, S. Ehlert, P.-T. Fang, S. Fatehi, Q. Feng, T. Friedhoff, J. Gayvert, Q. Ge, G. Gidofalvi, M. Goldey, J. Gomes, C. E. González-Espinoza, S. Gulania, A. O. Gunina, M. W. D. Hanson-Heine, P. H. P. Harbach, A. Hauser, M. F. Herbst, M. Hernandez Vera, M. Hodecker, Z. C. Holden, S. Houck, X. Huang, K. Hui, B. C. Huynh, M. Ivanov, A. Jasz, H. Ji, H. Jiang, B. Kaduk, S. Kähler, K. Khistyaev, J. Kim, G. Kis, P. Klunzinger, Z. Koczor-Benda, J. H. Koh, D. Kosenkov, L. Koulias, T. Kowalczyk, C. M. Krauter, K. Kue, A. Kunitsa, T. Kus, I. Ladjanszki, A. Landau, K. V. Lawler, D. Lefrancois, S. Lehtola, R. R. Li, Y.-P. Li, J. Liang, M. Liebenthal, H.-H. Lin, Y.-S. Lin, F. Liu, K.-Y. Liu, M. Loipersberger, A. Luenser, A. Manjanath, P. Manohar, E. Mansoor, S. F. Manzer, S.-P. Mao, A. V. Marenich, T. Markovich, S. Mason, S. A. Maurer, P. F. McLaughlin, M. F. S. J. Menger, J.-M. Mewes, S. A. Mewes, P. Morgante, J. W. Mullinax, K. J. Oosterbaan, G. Paran, A. C. Paul, S. K. Paul, F. Pavosevic, Z. Pei, S. Prager, E. I. Proynov, A. Rak, E. Ramos-Cordoba, B. Rana, A. E. Rask, A. Rettig, R. M. Richard, F. Rob, E. Rossomme, T. Scheele, M. Scheurer, M. Schneider, N. Sergueev, S. M. Sharada, W. Skomorowski, D. W. Small, C. J. Stein, Y.-C. Su, E. J. Sundstrom, Z. Tao, J. Thirman, G. J. Tornai, T. Tsuchimochi, N. M. Tubman, S. P. Veccham, O. Vydrov, J. Wenzel, J. Witte, A. Yamada, K. Yao, S. Yeganeh, S. R. Yost, A. Zech, I. Y. Zhang, X. Zhang, Y. Zhang, D. Zuev, A. Aspuru-Guzik, A. T. Bell, N. A. Besley, K. B. Bravaya, B. R. Brooks, D. Casanova, J.-D. Chai, S. Coriani, C. J. Cramer, G. Cserey, A. E. DePrince, R. A. DiStasio, A. Dreuw, B. D. Dunietz, T. R. Furlani, W. A. Goddard, S. Hammes-Schiffer, T. Head-Gordon, W. J. Hehre, C.-P. Hsu, T.-C. Jagau, Y. Jung, A. Klamt, J. Kong, D. S. Lambrecht, W. Liang, N. J. Mayhall, C. W. McCurdy, J. B. Neaton, C. Ochsenfeld, J. A. Parkhill, R. Peverati, V. A. Rassolov, Y. Shao, L. V. Slipchenko, T. Stauch, R. P. Steele, J. E. Subotnik, A. J. W. Thom,

A. Tkatchenko, D. G. Truhlar, T. Van Voorhis, T. A. Wesolowski, K. B. Whaley, H. L. Woodcock, P. M. Zimmerman, S. Faraji, P. M. W. Gill, M. Head-Gordon, J. M. Herbert, and A. I. Krylov, "Software for the frontiers of quantum chemistry: An overview of developments in the q-chem 5 package," The Journal of Chemical Physics **155**, 084801 (2021), https://doi.org/10.1063/5.0055522.

[37]J. W. Park, "Near-exact casscf-level geometry optimization with a large active space using adaptive sampling configuration interaction self-consistent field corrected with second-order perturbation theory (asci-scf-pt2)," Journal of Chemical Theory and Computation **17**, 4092–4104 (2021), pMID: 34096306, https://doi.org/10.1021/acs.jctc.1c00272.

[38]D.-K. Dang, J. A. Kammeraad, and P. M. Zimmerman, "Advances in parallel heat bath configuration interaction," The Journal of Physical Chemistry A **127**, 400–411 (2023), pMID: 36580361, https://doi.org/10.1021/acs.jpca.2c07949.

[39]J. Li, M. Otten, A. A. Holmes, S. Sharma, and C. J. Umrigar, "Fast semistochastic heat-bath configuration interaction," The Journal of Chemical Physics **149**, 214110 (2018), https://doi.org/10.1063/1.5055390.

[40]C. Mejuto-Zaera, G. Weng, M. Romanova, S. J. Cotton, K. B. Whaley, N. M. Tubman, and V. Vlček, "Are multi-quasiparticle interactions important in molecular ionization?" The Journal of Chemical Physics **154**, 121101 (2021), https://doi.org/10.1063/5.0044060.

[41]B. Herzog, B. Casier, S. Lebègue, and D. Rocca, "Solving the Schrödinger Equation in the Configuration Space with Generative Machine Learning," arXiv e-prints , arXiv:2208.06708 (2022), arXiv:2208.06708 [physics.chem-ph].

[42]J. E. T. Smith, J. Lee, and S. Sharma, "Near-exact nuclear gradients of complete active space self-consistent field wave functions," J. Chem. Phys. **157**, 094104 (2022), arXiv:2201.06514 [physics.chem-ph].

[43]D. B. Chamaki, S. Hadfield, K. Klymko, B. O'Gorman, and N. M. Tubman, "Self-consistent Quantum Iteratively Sparsified Hamiltonian method (SQuISH): A new algorithm for efficient Hamiltonian simulation and compression," arXiv e-prints , arXiv:2211.16522 (2022), arXiv:2211.16522 [quant-ph].

[44]D. Yoffe, A. Natan, and A. Makmal, "A Qubit-Efficient Variational Selected Configuration-Interaction Method," arXiv e-prints , arXiv:2302.06691 (2023), arXiv:2302.06691 [quant-ph].

[45]J. P. Coe, "Analytic gradients for selected configuration interaction," Journal of Chemical Theory and Computation **19**, 874–886 (2023), pMID: 36656261, https://doi.org/10.1021/acs.jctc.2c01062.

[46]X. Wang and S. Sharma, "Relativistic semistochastic heat-bath configuration interaction," arXiv e-prints , arXiv:2301.05794 (2023), arXiv:2301.05794 [physics.chem-ph].

[47]K. Klymko, C. Mejuto-Zaera, S. J. Cotton, F. Wudarski, M. Urbanek, D. Hait, M. Head-Gordon, K. B. Whaley, J. Moussa, N. Wiebe, W. A. de Jong, and N. M. Tubman, "Real-time evolution for ultracompact hamiltonian eigenstates on quantum hardware," PRX Quantum **3**, 020323 (2022).

[48]N. M. Tubman, C. Mejuto-Zaera, J. M. Epstein, D. Hait, D. S. Levine, W. Huggins, Z. Jiang, J. R. McClean, R. Babbush, M. Head-Gordon, and K. B. Whaley, "Postponing the orthogonality catastrophe: efficient state preparation for electronic structure simulations on quantum devices," arXiv e-prints , arXiv:1809.05523 (2018), arXiv:1809.05523 [quant-ph].

[49]K. Kanno, M. Kohda, R. Imai, S. Koh, K. Mitarai, W. Mizukami, and Y. O. Nakagawa, "Quantum-Selected Configuration Interaction: classical diagonalization of Hamiltonians in subspaces selected by quantum computers," arXiv e-prints , arXiv:2302.11320 (2023), arXiv:2302.11320 [quant-ph].

[50]J. W. Mullinax and N. M. Tubman, "Large-scale sparse wavefunction circuit simulator for applications with the variational quantum eigensolver," arXiv e-prints , arXiv:2301.05726 (2023), arXiv:2301.05726 [quant-ph].

[51]M. R. Hirsbrunner, D. Chamaki, J. W. Mullinax, and N. M. Tubman, "Beyond MP2 initialization for unitary coupled cluster quantum circuits," arXiv e-prints , arXiv:2301.05666 (2023), arXiv:2301.05666 [quant-ph].

[52]E. Solomonik and L. V. Kalé, "Highly scalable parallel sorting," in *2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS)* (2010) pp. 1–12.

[53]M. Axtmann, S. Witt, D. Ferizovic, and P. Sanders, "In-place Parallel Super Scalar Samplesort (IPS^{4}o)," ArXiv e-prints (2017), arXiv:1705.02257 [cs.DC].

[54]N. Bell and J. Hoberock, "Thrust: A productivity-oriented library for cuda," in *GPU computing gems Jade edition* (Elsevier, 2011) pp. 359–371.

[55]J. S. Spencer, N. S. Blunt, S. Choi, J. Etrych, M.-A. Filip, W. M. C. Foulkes, R. S. T. Franklin, W. J. Handley, F. D. Malone, V. A. Neufeld, R. Di Remigio, T. W. Rogers,

C. J. C. Scott, J. J. Shepherd, W. A. Vigor, J. Weston, R. Xu, and A. J. W. Thom, "The hande-qmc project: Open-source stochastic quantum chemistry from the ground state up," Journal of Chemical Theory and Computation **15**, 1728–1742 (2019), pMID: 30681844, https://doi.org/10.1021/acs.jctc.8b01217.

[56]M. Axtmann, T. Bingmann, P. Sanders, and C. Schulz, "Practical massively parallel sorting," in *Proceedings of the 27th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '15 (Association for Computing Machinery, New York, NY, USA, 2015) p. 13–23.

[57]M. Blum, R. W. Floyd, V. Pratt, R. L. Rivest, and R. E. Tarjan, "Time bounds for selection," Journal of Computer and System Sciences **7**, 448–461 (1973).

[58]R. W. Floyd and R. L. Rivest, "Expected time bounds for selection," Commun. ACM **18**, 165–172 (1975).

[59]S. G. Akl, "An optimal algorithm for parallel selection," Information Processing Letters **19**, 47–50 (1984).

[60]J. Choi, J. Demmel, I. S. Dhillon, J. J. Dongarra, S. Ostrouchov, A. Petitet, K. Stanley, D. W. Walker, and R. C. Whaley, "Scalapack: A portable linear algebra library for distributed memory computers - design issues and performance," in *Applied Parallel Computing, Computations in Physics, Chemistry and Engineering Science, Second International Workshop, PARA '95, Lyngby, Denmark, August 21-24, 1995, Proceedings*, Lecture Notes in Computer Science, Vol. 1041, edited by J. J. Dongarra, K. Madsen, and J. Wasniewski (Springer, 1995) pp. 95–106.

[61]"Elpa," (2023).

[62]P. Kus, A. Marek, S. S. Koecher, H. H. Kowalski, C. Carbogno, C. Scheurer, K. Reuter, M. Scheffler, and H. Lederer, "Optimizations of the Eigensolvers in the ELPA Library," arXiv e-prints , arXiv:1811.01277 (2018), arXiv:1811.01277 [cs.MS].

[63]E. R. Davidson, "The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices," Journal of Computational Physics **17**, 87–94 (1975).

[64]A. V. Knyazev, "Preconditioned eigensolvers—an oxymoron," Electron. Trans. Numer. Anal **7**, 104–123 (1998).

[65]A. V. Knyazev, "Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method," SIAM journal on scientific computing **23**,

517–541 (2001).

[66]R. W. JK Cullum, *Lanczos algorithms for large symmetric eigenvalue computations: Vol. I: Theory* (SIAM, 2002).

[67]Y. Saad, "Krylov subspace methods on supercomputers," SIAM Journal on Scientific and Statistical Computing **10**, 1200–1232 (1989).

[68]A. Stathopoulos and C. Fischer, "Reducing synchronization on the parallel davidson method for the large, sparse, eigenvalue problem," in *Supercomputing '93:Proceedings of the 1993 ACM/IEEE Conference on Supercomputing* (1993) pp. 172–180.

[69]L. Borges and S. Oliveira, "A parallel davidson-type algorithm for several eigenvalues," Journal of Computational Physics **144**, 727–748 (1998).

[70]M. Nool and A. van der Ploeg, "A parallel jacobi–davidson-type method for solving large generalized eigenvalue problems in magnetohydrodynamics," SIAM Journal on Scientific Computing **22**, 95–112 (2000).

[71]E. Romero and J. E. Roman, "A parallel implementation of the davidson method for generalized eigenproblems," Advances in Parallel Computing **19**, 133–140 (2010).

[72]E. Romero and J. E. Roman, "A parallel implementation of davidson methods for large-scale eigenvalue problems in slepc," **40** (2014), 10.1145/2543696.

[73]J. A. Duersch, M. Shao, C. Yang, and M. Gu, "A robust and efficient implementation of lobpcg," SIAM Journal on Scientific Computing **40**, C655–C676 (2018).

[74]R. Van Beeumen, G. D. Kahanamoku-Meyer, N. Y. Yao, and C. Yang, "A scalable matrix-free iterative eigensolver for studying many-body localization," in *Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region*, HPCAsia2020 (Association for Computing Machinery, New York, NY, USA, 2020) p. 179–187.

[75]R. V. Beeumen, K. Z. Ibrahim, G. D. Kahanamoku–Meyer, N. Y. Yao, and C. Yang, "Enhancing scalability of a matrix-free eigensolver for studying many-body localization," The International Journal of High Performance Computing Applications **36**, 307–319 (2022).

[76]M. Hoemmen, *Communication-avoiding Krylov subspace methods*, Ph.D. thesis (2010).

[77]S. Balay, S. Abhyankar, M. F. Adams, S. Benson, J. Brown, P. Brune, K. Buschelman, E. Constantinescu, L. Dalcin, A. Dener, V. Eijkhout, J. Faibussowitsch, W. D. Gropp, V. Hapla, T. Isaac, P. Jolivet, D. Karpeev, D. Kaushik, M. G. Knepley, F. Kong, S. Kruger, D. A. May, L. C. McInnes, R. T. Mills, L. Mitchell, T. Munson, J. E. Roman, K. Rupp, P. Sanan, J. Sarich, B. F. Smith, S. Zampini, H. Zhang, H. Zhang, and J. Zhang,

"PETSc/TAO users manual," Tech. Rep. ANL-21/39 - Revision 3.18 (Argonne National Laboratory, 2022).

[78]M. Grossman, C. Thiele, M. Araya-Polo, F. Frank, F. O. Alpak, and V. Sarkar, "A survey of sparse matrix-vector multiplication performance on large matrices," arXiv e-prints , arXiv:1608.00636 (2016), arXiv:1608.00636 [cs.PF].

[79]R. H. Bisseling, "190Sparse matrix–vector multiplication," in *Parallel Scientific Computation: A Structured Approach Using BSP* (Oxford University Press, 2020) https://academic.oup.com/book/0/chapter/367275519/chapter-pdf/45164935/oso-9780198788348-chapter-4.pdf.

[80]F. Weigend and R. Ahlrichs, "Balanced basis sets of split valence, triple zeta valence and quadruple zeta valence quality for h to rn: Design and assessment of accuracy," Phys. Chem. Chem. Phys. **7**, 3297 (2005).

[81]H.-J. Werner, P. J. Knowles, G. Knizia, F. R. Manby, and M. Schütz, "Molpro: a general-purpose quantum chemistry program package," WIREs Computational Molecular Science **2**, 242–253 (2012), https://wires.onlinelibrary.wiley.com/doi/pdf/10.1002/wcms.82.

[82]H.-J. Werner, P. J. Knowles, F. R. Manby, J. A. Black, K. Doll, A. Heßelmann, D. Kats, A. Köhn, T. Korona, D. A. Kreplin, Q. Ma, T. F. Miller, A. Mitrushchenkov, K. A. Peterson, I. Polyak, G. Rauhut, and M. Sibaev, "The molpro quantum chemistry package," The Journal of Chemical Physics **152**, 144107 (2020), https://doi.org/10.1063/5.0005081.

[83]R. Olivares-Amaya, W. Hu, N. Nakatani, S. Sharma, J. Yang, and G. K.-L. Chan, "The ab-initio density matrix renormalization group in practice," The Journal of chemical physics **142**, 034102 (2015).

[84]G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," SIAM Journal on Scientific Computing **20**, 359–392 (1998).

[85]E. Cuthill and J. McKee, "Reducing the bandwidth of sparse symmetric matrices," in *Proceedings of the 1969 24th National Conference*, ACM '69 (Association for Computing Machinery, New York, NY, USA, 1969) p. 157–172.

Input: Local part A_I of an array A with $|A| = n$, Element index $k \in [1, n]$

Output: The element a_k (global) such that $|\{a > a_k \,|a \in A\}| = k$

▷ All operations are local unless otherwise specified.

$rank \leftarrow$ PE rank.

$n_I \leftarrow |A_I|$

$N \leftarrow$ Allreduce(n_I) ▷ Collective

while $N \geq n_{min}$ **do**

  $p \leftarrow$ Select a random pivot in $[0, N)$.

  Determine the owner of a_p and broadcast to other PEs ▷ Collective

  $A_I^g, A_I^e, A_I^l \leftarrow$ partition(A^I, a_p)

  $G_I, E_I, L_I \leftarrow |A_I^g|, |A_I^e|, |A_I^l|$

  $\{G, E, L\} \leftarrow$ Allreduce($\{G_I, E_I, L_I\}$) ▷ Collective

  if $k \leq G$ **then**

    $A_I \leftarrow A_I^g$

  else if $k \leq G + E$ **then**

    $a_k \leftarrow a_p$

return a_k

  else

    $A_I \leftarrow A^l$

    $k \leftarrow k - G - E$

  end if

  $n_I \leftarrow |A_I|$

  $N \leftarrow$ Allreduce(n_I) ▷ Collective

end while

$A_{rem} \leftarrow$ Gather(A_I) ▷ Collective (PE 0)

if $rank = 0$ **then**

  $a_k \leftarrow$ SerialQuickselect(A_{rem}) ▷ `std::nth_element`

end if

Broadcast a_k ▷ Collective

return a_k

Algorithm 4. Distributed Memory Quickselect

Input: Matrix-vector product operator, `OP`; preconditioner operator `K`, max Krylov dimension k; initial guess v_0; Residual norm tolerance ϵ.

Output: Eigenpair (E_0, v) approximating the lowest eigenvalue of `OP`.

▷ All vectors are distributed among PEs

$w \leftarrow \texttt{OP}(v_0)$

$h \leftarrow v_0^\dagger w$; $h \leftarrow \text{Allreduce}(h)$; $v \leftarrow w - v_0 h$ ▷ Parallel Gram-Schmidt

$W \leftarrow w$; $V \leftarrow [v_0\ v]$

$i \leftarrow 1$

while $i < k$ **do**

  $W \leftarrow [W\ \texttt{OP}(v)]$

  $H_{loc} \leftarrow W^\dagger V$ ▷ Local all PEs

  $H_{tot} \leftarrow \texttt{Reduce}(H_{loc})$ ▷ Collective to PE 0

  Solve $H_{tot}\tilde{C} = \tilde{C}\tilde{E}$ ▷ Local on PE 0

  Broadcast $(\tilde{E}, C)$ ▷ Collective

  $R \leftarrow (W - \tilde{E}_0 V)\tilde{C}_0$ ▷ Local all PEs

  $r_{loc} \leftarrow R^\dagger R$ ▷ Local all PEs

  $\|r\| \leftarrow \sqrt{\texttt{Allgather}(r_{loc})}$ ▷ Collective

  if $\|r\| \leq \epsilon$ **then**

    return $(E_0, v) \leftarrow (\tilde{E}_0, V\tilde{C}_0)$

  else

    $R \leftarrow \texttt{K}(R)$ ▷ Preconditioned Residual

    $h \leftarrow V^\dagger R$; $h \leftarrow \text{Allreduce}(h)$; $v \leftarrow R - Vh$ ▷ Parallel Gram-Schmidt

    $V \leftarrow [V\ v]$

  end if

  $i \leftarrow i + 1$

end while

Algorithm 5. Parallel Davidson Algorithm

1: **Input:** Matrix H and vector v in block distributed format (Fig. 1). (Optional) Precomputed communication data (Eq. (2))

2: **Output:** $w = Hv$ in conforming row partitioned format

3:

4: **if** Communication data needed **then**

5: Compute communication data for H per Eq. (2)

6: **end if**

7: Post local receives for incoming data from remote PEs ▷ `MPI_Irecv`

8: Pack elements of v required by remote processes ▷ Local

9: Post local sends to satisfy remove receive requests ▷ `MPI_Isend`

10: $w \leftarrow H^d v$ ▷ Local

11: Wait for remote receives to complete

12: $v_{rem} \leftarrow$ Unpack remote data

13: $w \leftarrow w + H^{od} v_{rem}$

Algorithm 6. MPI-Based Non-Blocking Parallel Sparse Matrix-Vector Product (SpMV)