Fast semistochastic heat-bath configuration interaction

Junhao Li,[1,a)] Matthew Otten,[1,b)] Adam A. Holmes,[1,2,c)] Sandeep Sharma,[2,d)]
and C. J. Umrigar[1,e)]
[1]*Laboratory of Atomic and Solid State Physics, Cornell University, Ithaca, New York 14853, USA*
[2]*Department of Chemistry and Biochemistry, University of Colorado Boulder, Boulder, Colorado 80302, USA*

This paper presents in detail our fast semistochastic heat-bath configuration interaction (SHCI) method for solving the many-body Schrödinger equation. We identify and eliminate computational bottlenecks in both the variational and perturbative steps of the SHCI algorithm. We also describe the parallelization and the key data structures in our implementation, such as the distributed hash table. The improved SHCI algorithm enables us to include in our variational wavefunction two orders of magnitude more determinants than has been reported previously with other selected configuration interaction methods. We use our algorithm to calculate an accurate benchmark energy for the chromium dimer with the X2C relativistic Hamiltonian in the cc-pVDZ-DK basis, correlating 28 electrons in 76 spatial orbitals. Our largest calculation uses two billion Slater determinants in the variational space and semistochastically includes perturbative contributions from at least trillions of additional determinants with better than 10^{-5} Ha statistical uncertainty. *Published by AIP Publishing.* https://doi.org/10.1063/1.5055390

I. INTRODUCTION

The choice of quantum chemistry methods requires a trade-off between accuracy and efficiency. Density functional theory (DFT)[1–3] methods with approximate density functionals are popular and efficient, but are often not sufficiently accurate. Coupled cluster with single, double, and perturbative triple excitations [CCSD(T)][4] is very accurate for single reference systems, but not for strongly correlated systems, such as systems with stretched bonds. The density matrix renormalization group (DMRG)[5–12] and full configuration interaction quantum Monte Carlo (FCIQMC)[13–17] are systematically improvable but rapidly get expensive with the number of electrons and the size of the basis set.

The recently developed semistochastic heat-bath configuration interaction (SHCI)[18–23] is another systematically improvable method capable of providing essentially exact energies for small systems. In common with FCIQMC, the computational cost of the method scales exponentially in the number of electrons but with a much smaller exponent than in full configuration interation (FCI). However, SHCI is much faster than FCIQMC. The comparison with DMRG is more involved. While SHCI is much faster than DMRG for small moderately correlated systems, the ratio of costs changes in DMRG's favor as the system size increases and as the correlation strength increases because the methods have different scaling with these parameters. In particular, SHCI scales exponentially with system size with a prefactor that is typically small, but which grows with the strength of the correlation.

DMRG scales exponentially with the $(D − 1)/D$-th power of the system size (where D is the system dimension) with a prefactor that is typically larger, but is not very sensitive to the strength of the correlation.

SHCI is an example of the selected configuration interaction plus perturbation theory (SCI+PT) methods,[24–36] the earliest of which being the configuration interaction by perturbatively selecting iteratively (CIPSI) method[24,26] of Malrieu and collaborators. SCI+PT methods have two stages. In the first stage, a variational wavefunction is constructed iteratively, starting from a determinant that is expected to have a significant amplitude in the final wavefunction, e.g., the Hartree-Fock determinant. Each iteration of the variational stage has three steps: selection of important determinants, construction of the Hamiltonian matrix, and iterative diagonalization of the Hamiltonian matrix. In the second stage, 2nd-order perturbation theory is used to improve upon the variational energy.

The SHCI algorithm has greatly improved the efficiency of both stages. First, as discussed in Sec. II A, it greatly speeds up the determinant selection, and second, as discussed in Sec. II B, it drastically reduces the central processing unit (CPU) cost as well as the memory cost of performing the perturbation step by using a semistochastic algorithm. These two modifications have allowed SHCI to be used for systems as large as hexatriene in an ANO-L-pVDZ basis (32 correlated electrons in 118 orbitals) which has a Hilbert space of 10^{38} determinants.[23] SHCI has also recently been extended to (a) calculate not just the ground state but also the low-lying excited states,[20] (b) perform self-consistent field orbital optimization in very large active spaces,[21] and (c) include relativistic effects including the spin-orbit coupling using "one-step" calculations with two-component Hamiltonians.[22]

Since SHCI has greatly reduced the time required to select determinants, we find, for large systems, that Hamiltonian

a)jl2922@cornell.edu
b)mjo98@cornell.edu
c)aah95@cornell.edu
d)sanshar@gmail.com
e)cyrusumrigar@cornell.edu

149, 214110-1

construction is the most time-consuming step of the variational stage. For around 10^8 variational determinants, it takes two orders of magnitude more time to construct the Hamiltonian matrix than to select the determinants for most molecules. In addition, if a small stochastic error is required, the perturbative stage can be expensive, particularly on computer systems that do not have enough memory. Hence, in this paper, we present an improved SHCI algorithm that greatly speeds up these two steps. For the variational stage, we introduce a fast Hamiltonian construction algorithm that allows us to use two orders of magnitude more determinants in the wavefunction. For the perturbative stage, we introduce the 3-step batch perturbation method that further speeds up the calculation and reduces the memory requirement. We also describe important implementation details of the algorithm, including the key data structures and parallelization.

We organize the paper as follows: In Sec. II, we review the SHCI method. In Sec. III, we introduce our faster Hamiltonian construction algorithm. In Sec. IV, we introduce our 3-step batch perturbation algorithm. In Sec. V, we describe the key data structures in our implementation. In Sec. VI, we describe the parallelization strategy and demonstrate its scalability. In Sec. VII, we apply our improved SHCI to Cr_2. Section VIII concludes the paper.

II. SHCI REVIEW

In this section, we review the semistochastic heat-bath configuration interaction (SHCI) method,[18–20] emphasizing the two important ways it differs from other SCI+PT methods. In the following, we use $\mathcal{V}$ for the set of variational determinants and $\mathcal{P}$ for the set of perturbative determinants, that is, the set of determinants that are connected to the variational determinants by at least one non-zero Hamiltonian matrix element but are not present in $\mathcal{V}$.

A. Variational stage

SHCI starts from an initial determinant and generates the variational wavefunction through an iterative process. At each iteration, the variational wavefunction, Ψ_V, is written as a linear combination of the determinants in the space $\mathcal{V}$,

$$\Psi_V = \sum_{D_i \in \mathcal{V}} c_i |D_i\rangle, \tag{1}$$

and new determinants, D_a, from the space $\mathcal{P}$ that satisfy the criterion

$$\exists\, D_i \in \mathcal{V} \text{ such that } |H_{ai} c_i| \geq \epsilon_1 \tag{2}$$

are added to the $\mathcal{V}$ space, where H_{ai} is the Hamiltonian matrix element between determinants D_a and D_i and ϵ_1 is a user-defined parameter that controls the accuracy of the variational stage.[37] (When $\epsilon_1 = 0$, the method becomes equivalent to FCI.) After adding the new determinants to $\mathcal{V}$, the Hamiltonian matrix is constructed and diagonalized using the diagonally preconditioned Davidson method,[38] to obtain an improved estimate of the lowest eigenvalue, E_V, and eigenvector, Ψ_V. This process is repeated until the change in E_V falls below a certain threshold, e.g., 1 μHa.

Other SCI methods, such as CIPSI,[24,26] use different criteria, usually based on either the first-order perturbative

coefficient of the wavefunction,

$$\left| c_a^{(1)} \right| = \left| \frac{\sum_i H_{ai} c_i}{E_0 - E_a} \right| > \epsilon_1, \tag{3}$$

or the second-order perturbative correction to the energy

$$-\Delta E_2 = -\frac{(\sum_i H_{ai} c_i)^2}{E_0 - E_a} > \epsilon_1. \tag{4}$$

The reason we choose instead the selection criterion in Eq. (2) is that it can be implemented very efficiently without checking the vast majority of the determinants that do not meet the criterion, by taking advantage of the fact that most of the Hamiltonian matrix elements correspond to double excitations, and their values do not depend on the determinants themselves but only on the four orbitals whose occupancies change during the double excitation. Therefore, before performing an HCI run, for each pair of spin-orbitals, the absolute values of the Hamiltonian matrix elements obtained by doubly exciting from that pair of orbitals are computed and stored in decreasing order by magnitude, along with the corresponding pairs of orbitals the electrons would excite to. Then the double excitations that meet the criterion in Eq. (2) can be generated by looping over all pairs of occupied orbitals in the reference determinant and traversing the array of sorted double-excitation matrix elements for each pair. As soon as the cutoff is reached, the loop for that pair of occupied orbitals is exited. Although the criterion in Eq. (2) does not include information from the diagonal elements, the HCI selection criterion is not significantly different from either of the two CIPSI-like criteria because the terms in the numerator of Eq. (3) span many orders of magnitude, so the sum is highly correlated with the largest-magnitude term in the sum in Eq. (3). It was demonstrated in Ref. 18 that the selected determinants give only slightly inferior convergence to those selected using the criterion in Eq. (3). This is greatly outweighed by the improved selection speed. Moreover, one could use the HCI criterion in Eq. (2) with a smaller value of ϵ_1 as a preselection criterion and then select determinants using the criterion in Eq. (4), thereby having the benefit of both a fast selection method and a close to optimal choice of determinants.

B. Perturbative stage

In common with most other SCI+PT methods, the perturbative correction is computed using Epstein-Nesbet perturbation theory.[39,40] The variational wavefunction is used to define the zeroth-order Hamiltonian, H_0, and the perturbation, V,

$$H_0 = \sum_{D_i, D_j \in \mathcal{V}} H_{ij} |D_i\rangle\langle D_j| + \sum_{D_a \notin \mathcal{V}} H_{aa} |D_a\rangle\langle D_a|,$$

$$V = H - H_0. \tag{5}$$

The first-order energy correction is zero, and the second-order energy correction ΔE_2 is

$$\Delta E_2 = \langle \Psi_0 | V | \Psi_1 \rangle = \sum_{D_a \in \mathcal{P}} \frac{\left(\sum_{D_i \in \mathcal{V}} H_{ai} c_i \right)^2}{E_0 - E_a}, \tag{6}$$

where $E_a = H_{aa}$.

It is expensive to evaluate the expression in Eq. (6) because the outer summation includes all determinants in the space $\mathcal{P}$,

and their number is $\mathcal{O}(n^2v^2N_V)$, where N_V is the number of variational determinants, n is the number of electrons, and v is the number of virtual orbitals. For the calculation on Cr_2, described in Sec. VII, $n = 28$, $v = 62$, and $N_V = 2 \times 10^9$, so the number of determinants in $\mathcal{P}$ is huge. The straightforward and time-efficient approach to computing the perturbative correction requires storing the partial sum $\sum_{D_i \in \mathcal{V}} H_{ai}c_i$ for each a, while looping over all the determinants $D_i \in \mathcal{V}$. This creates a severe memory bottleneck.

Various schemes for improving the efficiency have been implemented, including only exciting from a rediagonalized array of the largest-weight determinants[26] and its efficient approximation using diagrammatic perturbation theory.[27] However, this is both more complicated than necessary (requiring a double extrapolation with respect to the two variational spaces to reach the FCI limit) and is more computationally expensive than necessary since even the largest weight determinants have many connections that make only small contributions to the energy. The SHCI algorithm instead uses two other strategies to reduce both the computational time and the storage requirement.

First, SHCI screens the sum[18] using a second threshold, ϵ_2 (where $\epsilon_2 < \epsilon_1$), as the criterion for selecting perturbative determinants $\mathcal{P}$,

$$\Delta E_2(\epsilon_2) = \sum_{D_a \in \mathcal{P}} \frac{\left(\sum_{D_i \in \mathcal{V}}^{(\epsilon_2)} H_{ai}c_i\right)^2}{E_V - H_{aa}}, \qquad (7)$$

where $\sum^{(\epsilon_2)}$ indicates that only terms in the sum for which $|H_{ai}c_i| \geq \epsilon_2$ are included. Similar to the variational stage, we find the connected determinants efficiently with precomputed arrays of double excitations sorted by the magnitude of their Hamiltonian matrix elements.[18] Note that the vast number of terms that do not meet this criterion are *never evaluated*.

Even with this screening, the simultaneous storage of all terms indexed by a in Eq. (7) can exceed computer memory when ϵ_2 is chosen small enough to obtain essentially the exact perturbation energy. The second innovation in the calculation of the SHCI perturbative correction is to overcome this memory bottleneck by evaluating this perturbative correction semistochastically.[19] The most important contributions are evaluated deterministically, and the rest are sampled stochastically. The total perturbative correction is

$$\Delta E_2(\epsilon_2) = \left[\Delta E_2^s(\epsilon_2) - \Delta E_2^s\left(\epsilon_2^d\right)\right] + \Delta E_2^d\left(\epsilon_2^d\right), \qquad (8)$$

where ΔE_2^d is the deterministic perturbative correction obtained by using a larger threshold $\epsilon_2^d \geq \epsilon_2$ in Eq. (7). ΔE_2^s is the stochastic perturbative correction from randomly selected samples of the variational determinants and is given by

$$\Delta E_2^s(\epsilon_2) = \frac{1}{N_d(N_d - 1)} \left\langle \sum_{D_a \in \mathcal{P}} \left[\left(\sum_{D_i \in \mathcal{V}}^{N_d^{\text{uniq}},(\epsilon_2)} \frac{w_i c_i H_{ai}}{p_i} \right)^2 \right. \right.$$

$$\left. \left. + \sum_{D_i \in \mathcal{V}}^{N_d^{\text{uniq}},(\epsilon_2)} \left(\frac{w_i(N_d - 1)}{p_i} - \frac{w_i^2}{p_i^2} \right) c_i^2 H_{ai}^2 \right] \frac{1}{E_0 - E_a} \right\rangle, \qquad (9)$$

where N_d is the number of variational determinants per sample and N_d^{uniq} is the number of different determinants in a sample. p_i and w_i are the probability of selecting determinant D_i and the number of copies of that determinant in a sample, respectively. The N_d determinants are sampled from the discrete probability distribution

$$p_i = \frac{|c_i|}{\sum_j^{N_V} |c_j|}, \qquad (10)$$

using the Alias method,[41,42] which allows samples to be drawn in $\mathcal{O}(1)$ time. [The more commonly used heatbath method requires $\mathcal{O}(\log(n))$ time to do a binary search of an array of cumulative probabilities.] $\Delta E_2^s[\epsilon_2]$ and $\Delta E_2^s[\epsilon_2^d]$ are calculated using the same set of samples, and thus, there is significant cancellation of stochastic error. Furthermore, because these two energies are calculated simultaneously, the additional cost of performing this calculation, compared to a purely stochastic summation, is very small. Clearly, in the limit that $\epsilon_2^d = \epsilon_2$, the entire perturbative calculation becomes deterministic.

The perturbative stage of the SHCI algorithm has the interesting feature that it achieves super-linear speedup with the number of computer nodes used. There are two reasons for this, both having to do with the increase in the total computer memory. First, a larger fraction of the perturbative energy can be computed deterministically, using a smaller value of ϵ_2^d in Eq. (8). Second, a larger value of N_d in Eq. (9) can be used. For a given total number of samples, the statistical error is smaller for a small number of large samples, than for a large number of small samples, because the number of sampled contributions to the energy correction is a quadratic function of the number of sampled variational determinants. For example, N_s samples, each of size N_d, will have $N_s N_d^2$ contributions to the energy, whereas $N_s/2$ samples, each of size $2N_d$, will have $2N_s N_d^2$ contributions. Consequently, this too contributes to a super-linear speedup.

C. Other features of SHCI

We note that although SHCI has a stochastic component, it has the advantages compared to quantum Monte Carlo algorithms that there is no sign problem and that each sample is independent. Another feature of the method is that if the calculation is done for various values of the variational threshold ϵ_1, a plot of the total energy (variational plus perturbative correction) plotted versus the perturbative correction yields a smooth curve that can be used to assess the convergence and extrapolate to the FCI limit, $\Delta E = 0$.[20] We typically use a quadratic fit, with the points weighted by $(\Delta E)^{-2}$.[23]

As is typical in many quantum chemistry methods, we note that the convergence of both the variational energy and the total (variational plus perturbative) energy depends on the choice of orbitals. Natural orbitals, calculated within HCI, are typically a better choice than Hartree Fock orbitals, and optimized orbitals[21] are a yet better choice. For systems with more than a few atoms, split-localized optimized orbitals lead to yet better convergence.[23]

We describe, in Secs. III and IV, improvements we have made to the variational and the perturbative stages of the SHCI

algorithm, which speed up the calculations by an order of magnitude or more for large systems.

III. FAST HAMILTONIAN CONSTRUCTION

The Hamiltonian matrix is stored in upper-triangular sparse matrix form. At each variational iteration, we have a set of old determinants and a set of new determinants. We have already calculated the Hamiltonian matrix for the old determinants and need to calculate the old-new and the new-new matrix elements.

The SHCI algorithm greatly speeds up the step of finding the important determinants, and one can very quickly generate hundreds of millions or more. With these many determinants, the construction of the Hamiltonian matrix is expensive. Most of the matrix elements are zero, but finding the non-zero Hamiltonian elements quickly is challenging because the determinants in the variational wavefunction do not exhibit any pattern. (Efficient construction of the Hamiltonian matrix of the same size in FCI is much more straightforward than that in SCI.) There are two straightforward "brute force" approaches to building the Hamiltonian matrix: (a) looping over all pairs of determinants to find those pairs that are related by single or double excitations and (b) generating all connections of each determinant in $\mathcal{V}$ and searching for the connections in the sorted array of variational determinants. When the number of determinants is not very large, the former is more efficient. Both of these are much too expensive for the very large number of variational determinants that we use.

The original SHCI algorithm introduced auxiliary arrays[19] to speed up the Hamiltonian construction, but it still spends considerable time on elements that are zero. In our improved SHCI algorithm, we use a larger number of auxiliary arrays to further reduce the time. All the relevant data structures are shown in Table I. Some of these are appended to at each variational iteration because they contain information about all the variational determinants currently included in the wavefunction, whereas others are constructed from scratch since part of their information content pertains to only the new determinants.

The auxiliary arrays are constructed by looping over just the new determinants. First, each new α encountered is appended to array $\vec{\alpha}$ and hash map $i_\alpha(\alpha)$. Also, each new determinant is appended to the arrays $\vec{i}_{D\alpha}(i_\alpha)$ and $\vec{i}_{\beta\alpha}(i_\alpha)$. In order to speed up the generation of the Hamiltonian matrix (described later), these are sorted by i_D when the number of new determinants is much smaller than the number of old determinants and by i_β otherwise. Then, the hash map $\vec{i}_\alpha(\alpha^{(-1)})$ is constructed, and finally, the array $\vec{i}_{\alpha\alpha}(j_\alpha)$. The purpose of $\vec{i}_\alpha(\alpha^{(-1)})$ is simply to speed up the construction of $\vec{i}_{\alpha\alpha}(j_\alpha)$. Note that if two α strings are a single excitation apart, they will be simultaneously present under one and only one key of the hash map $\vec{i}_\alpha(\alpha^{(-1)})$.

Then, we update the Hamiltonian matrix using these auxiliary arrays and a loop over all the determinants. Algorithms 1 and 2 describe the algorithm using pseudocode. The Hamiltonian matrix elements are nonzero only for determinants that are at most two excitations apart, namely, diagonal elements, same-spin single excitations, same-spin double excitations,

TABLE I. The notation for the data structures in our current algorithm for efficiently constructing the Hamiltonian matrix. Analogous data structures with the alpha and beta roles reversed are also used. The text gives details of how they are constructed efficiently.

Notation	Description
$\vec{D}$	The array of determinants in $\mathcal{V}$, in the order they were generated.
$\vec{\alpha}$	The array of all alpha strings, without repeats that are present in at least one determinant in $\mathcal{V}$, in the order they were generated.
$i_\alpha(\alpha)$	Hash map that takes an alpha string, α, and returns its index, i_α, in $\vec{\alpha}$.
$\vec{i}_{D\alpha}(i_\alpha)$	The array of determinant indices in $\vec{D}$ such that the alpha strings of those determinants have index i_α in $\vec{\alpha}$. The elements of $\vec{i}_{D\alpha}(i_\alpha)$ are sorted either by their values or by the indices of their beta strings in $\vec{\beta}$. (See text for details.)
$\vec{i}_{\beta\alpha}(i_\alpha)$	The array of all beta string indices in β that appear with α_{i_α} in a determinant. It is sorted so that the elements of $\vec{i}_{D\alpha}(i_\alpha)$ and $\vec{i}_{\beta\alpha}(i_\alpha)$ are always in correspondence.
$\vec{i}_\alpha(\alpha^{(-1)})$	Hash map that takes an alpha string with one less electron, $\alpha^{(-1)}$, and returns an array of indices of alpha strings in $\vec{\alpha}$ that can give $\alpha^{(-1)}$ upon removing an electron. These are generated only for the α's present in the new determinants.
$\vec{i}_{\alpha\alpha}(j_\alpha)$	The array of indices of alpha strings in the array $\vec{\alpha}$, connected by a single excitation to the j^{th}_α alpha string of array $\vec{\alpha}$, sorted in ascending order. These are generated only for the j_α's present in the new determinants.
$i_D, j_D, \dots$	Indices of $\vec{D}$.
$i_\alpha, j_\alpha, \dots$	Indices of $\vec{\alpha}$.
$\alpha(D_i)$	Alpha string of determinant D_i.

and opposite-spin double excitations. For finding the same-spin connections, we use a method closely related to that in Ref. 33. Finding the opposite-spin connections is more computationally expensive and our algorithm speeds this up significantly.

Algorithm 1. Hamiltonian matrix update for determinants connected by single or double alpha excitations. The algorithm for single or double beta excitations is very similar.

---
for D_i in $\vec{D}$ **do**
   Use hash map $i_\beta(\beta)$ to find i_β, the index of $\beta(D_i)$
   s is the index of the first new determinant with $s > i$.
   for j in $\vec{i}_{D\beta}(i_\beta)$ **do**
      if $j \geq s$ and D_i, D_j are connected **then**
         Compute and add H_{ij} to the Hamiltonian
      end if
   end for
end for

Algorithm 2. Hamiltonian matrix update for determinants connected by an opposite-spin double excitation.

for D_i in $\vec{D}$ **do**
    Use hash maps $i_\alpha(\alpha)$ and $i_\beta(\beta)$ to find i_α, i_β,
        the indices of $\alpha(D_i)$ and $\beta(D_i)$.
    s is the index of the first new determinant with $s > i$.
    for k_α in $\vec{i}_{\alpha\alpha}(i_\alpha)$ **do**
        if number of new determinants is small **then**
            for $j \geq s$ in $\vec{i}_{D\alpha}(k_\alpha)$ (reverse loop) **do**
                if $i_\beta(\beta(D_j)) \in \vec{i}_{\beta\beta}(i_\beta)$ (binary search) **then**
                    Compute and add H_{ij} to the Hamiltonian
                end if
            end for
        else
            Find the intersection $\vec{j}_\beta$ of sorted arrays
            $\vec{i}_{\beta\alpha}(k_\alpha)$ and $\vec{i}_{\beta\beta}(i_\beta)$ in $\mathcal{O}(n)$ time.
            Since $\vec{i}_{\beta\alpha}(k_\alpha)$ and $\vec{i}_{D\alpha}(k_\alpha)$ are in
            1-To-1 correspondence, this provides the
            corresponding determinants $\vec{j}_D$
            for j in $\vec{j}_D$ **do**
                if $j \geq s$ **then**
                    Compute and add H_{ij} to the Hamiltonian
                end if
            end for
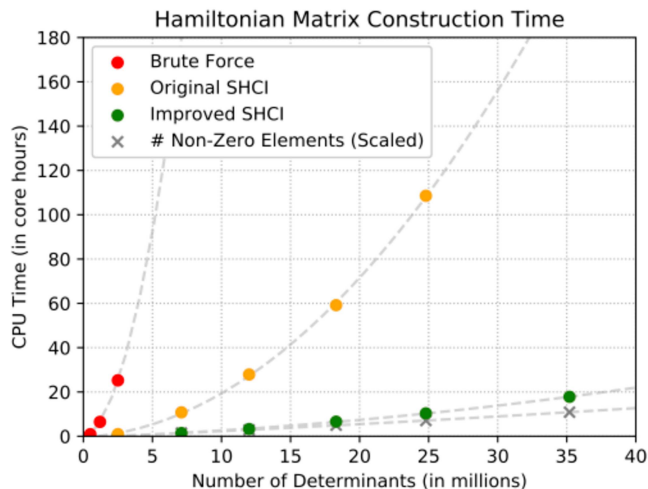   **end if**
    end for
end for

FIG. 1. Hamiltonian matrix construction time for a copper atom in a cc-pVTZ basis. Hamiltonian construction is the performance bottleneck in the variational stage. Hamiltonian construction in our improved SHCI algorithm is an order of magnitude faster than that in our original SHCI algorithm and several orders of magnitude faster than the faster of the two brute force approaches (loop over each pair of determinants). Also shown is the number of nonzero elements in the Hamiltonian, scaled so that the first point coincides with the first point of the improved SHCI CPU time.

A. Same-spin excitations

For determinants connected by single or double alpha excitations to a given determinant D_i, the beta strings must be the same as $\beta(D_i)$. Hence, we simply loop over the determinants in the $\vec{i}_{D\beta}(i_\beta)$ array and check if the alpha strings are related by a single or a double excitation, and if they are, we compute that Hamiltonian matrix element. Similarly, we can find the single and double beta excitations by looping over the determinants in the $\vec{i}_{D\alpha}(i_\alpha)$ array.

B. Opposite-spin excitations

For the opposite-spin double excitations, we first loop over all k_α in the $\vec{i}_{\alpha\alpha}(i_\alpha)$ array, i.e., the indices of $\vec{\alpha}$ connected by single excitations to $\alpha(D_i)$. The determinants that have alpha string k_α are in $\vec{i}_{D\alpha}(k_\alpha)$, but since only some of these have beta strings are single excitations of $\beta(D_i)$, we need to filter $\vec{i}_{D\alpha}(k_\alpha)$ to find the connected determinants. This is done in two different ways as described in Algorithm 2 depending on the number of new determinants. When the number of new determinants is less than 20% of the total number of determinants (e.g., in the later iterations of a given ϵ_1), $\vec{i}_{D\alpha}(i_\alpha)$ and $\vec{i}_{\beta\alpha}(i_\alpha)$ are sorted by i_D; otherwise, they are sorted by i_β. The remaining determinants after filtering are the determinants connected to the given determinant through opposite-spin double excitations. Each connection is visited only once during this process, which was not the case in the original SHCI method.

In Fig. 1, we use a copper atom[43,62] with a pseudopotential and the cc-pVTZ basis to compare the improved SHCI algorithm to the original SHCI algorithm and to the brute force algorithm where we loop over each pair of determinants. The improved algorithm is about an order of magnitude faster than

the original SHCI for medium size calculations. For large calculations, e.g., the Cr_2 calculation described in Sec. VII where we use billions of variational determinants, the speedup is even greater.

IV. 3-STEP PERTURBATION THEORY

As described in Sec. II, our original SHCI algorithm[19] solves the memory bottleneck problem of SCI+PT methods by introducing a semistochastic algorithm for computing the perturbative correction to the energy. In Sec. II, we have emphasized that the statistical error can be dramatically reduced by decreasing the value of ϵ_2^d in Eq. (8) and by increasing the size of the stochastic samples, N_d, in Eq. (9). However, decreasing ϵ_2^d or increasing N_d can quickly lead to very large memory requirements, making the calculations impractical even on large computers. In this situation, one is left with no choice but to run relatively inefficient calculations with larger ϵ_2^d and smaller N_d. For example, in Table II, rows 3 and 4 show a comparison of the total CPU time of the perturbation stage for a copper atom in a cc-pVTZ basis on a machine with large memory versus on a machine with small memory. When we decrease the memory by a factor of four, the total CPU time of the original SHCI algorithm increases by almost a factor of 8.

For a given target error, assuming that we have infinite computer memory, there is an optimal choice of ϵ_2^d and N_d for reaching that target error using the least computer time. Our improved algorithm is designed to have an efficiency that depends only weakly on the available computer memory. It is always more efficient than the original algorithm, especially when running on computers with small memory, in which case the gain in efficiency can be orders of magnitude. To achieve that, we replace the original 2-step SHCI algorithm with a 3-step algorithm. In each of the three steps, the perturbative determinants are divided into batches using a hash function,[44,45] and the energy correction is computed either

TABLE II. Computational cost of perturbative correction for a copper atom in a cc-pVTZ basis. The variational space has 19×10^6 determinants for $\epsilon_1 = 5 \times 10^{-5}$ Ha, and the perturbative space has 35×10^9 determinants for $\epsilon_2 = 10^{-7}$ Ha. HCI uses the deterministic perturbation of Ref. 18. SHCI uses the 2-step semistochastic perturbation algorithm of Ref. 19. Improved SHCI introduces the 3-step batch perturbation that significantly improves the efficiency of SHCI, especially for memory constrained cases. The timings for the 32 GB machine are obtained by running on the same 128 GB large memory machine but intentionally tuning the parameters so that the memory usage is kept below 32 GB throughout the run. We also provide the timing to reach a 1.8 μHa uncertainty to illustrate that our statistical error goes down much faster than $1/\sqrt{T}$ since we use smaller $\epsilon_2^{\mathrm{dtm}}$ and $\epsilon_2^{\mathrm{psto}}$ values for smaller target errors.

Method	Memory	CPU time (core hours)	Error (μHa)
HCI (deterministic)	3 TB	145.0	0
Original SHCI	32 GB	116.6	10
	128 GB	14.5	10
Improved SHCI	32 GB	4.2	9
	128 GB	3.7	9
	128 GB	5.9	1.8

by adding, in succession, the contribution from each batch or by estimating their sum by evaluating only a subset of these batches. The hash function maps a determinant to a 64-bit integer h. A batch contains all the determinants that satisfy $h \bmod n = i$, where i is the batch index and n is the number of batches. We use a high-quality hash function which ensures a highly uniform mapping, so each batch has about the same number of determinants, i.e., the fluctuations in the number of determinants in the various batches are the square root of the average number of determinants in each batch. The contributions of the various batches fluctuate both because the contributions of the perturbative determinants within a batch fluctuate and the number of perturbative determinants in a batch fluctuates. For both contributions, the ratio of the fluctuation to the expected value is $\sim \sqrt{N}/N \to 0$ for large N, where N is the average number of determinants in a batch.

In brief, our improved SHCI algorithm has the following 3 steps:

1. A deterministic step with cutoff $\epsilon_2^{\mathrm{dtm}}(< \epsilon_1)$, wherein all the variational determinants are used and all the perturbative batches are summed over.
2. A "pseudo-stochastic" step, with cutoff $\epsilon_2^{\mathrm{psto}}(< \epsilon_2^{\mathrm{dtm}})$, wherein all the variational determinants are used, and typically only a small fraction of the perturbative batches need be summed over to achieve an error much smaller than the target error.
3. A stochastic step, with cutoff $\epsilon_2(< \epsilon_2^{\mathrm{psto}})$, wherein a few stochastic samples of variational determinants, each consisting of N_d determinants, is sampled using Eq. (10), and only one of the perturbative batches is randomly selected per variational sample.

The total perturbative correction is

$$\Delta E_2(\epsilon_2) = \left[\Delta E_2^{\mathrm{sto}}(\epsilon_2) - \Delta E_2^{\mathrm{sto}}\left(\epsilon_2^{\mathrm{psto}}\right) \right]$$
$$+ \left[\Delta E_2^{\mathrm{psto}}\left(\epsilon_2^{\mathrm{psto}}\right) - \Delta E_2^{\mathrm{psto}}\left(\epsilon_2^{\mathrm{dtm}}\right) \right]$$
$$+ \Delta E_2^{\mathrm{dtm}}\left(\epsilon_2^{\mathrm{dtm}}\right). \quad (11)$$

The choice of these parameters depends on the system and the desired statistical error, but reasonable choices for a target error around 10^{-5} Ha are $\epsilon_2^{\mathrm{dtm}} = 2 \times 10^{-6}$ Ha, $\epsilon_2^{\mathrm{psto}} = 10^{-7}$ Ha, and $\epsilon_2 = \epsilon_1/10^6$. Of course, if $\epsilon_1 \leq \epsilon_2^{\mathrm{dtm}}$, the deterministic step is skipped. We next describe each of the 3 steps in detail.

The first step is a deterministic step similar to the original SHCI's deterministic step, except that when there is not enough memory to afford the chosen $\epsilon_2^{\mathrm{dtm}}$, we divide the perturbative space into batches according to the hash value of the perturbative determinants and evaluate their contributions batch by batch. The total deterministic correction is simply the sum of the corrections from all the batches

$$\Delta E_2\left(\epsilon_2^{\mathrm{dtm}}\right) = \sum_B \sum_{\substack{D_a \in \mathcal{P} \\ h(D_a) \in B}} \frac{\left(\sum_{D_i \in \mathcal{V}}^{(\epsilon_2^{\mathrm{dtm}})} H_{ai} c_i \right)^2}{E_V - H_{aa}}, \quad (12)$$

where $h(D)$ is the hash function and B is the hash value space for a batch. This method solves the memory bottleneck in a different way than the original SHCI algorithm. We could do the full calculation in this way, i.e., use a very small value for $\epsilon_2^{\mathrm{dtm}}$ and a large number of batches, but it is much more efficient to only evaluate the large contributions here and leave the huge number of small contributions to the later stochastic steps.

The second step is a pseudo-stochastic step. It is similar to the deterministic step, except for the following differences: (a) we use an $\epsilon_2^{\mathrm{psto}}$ much smaller than $\epsilon_2^{\mathrm{dtm}}$ as the selection criterion, (b) we divide the perturbative space into as many batches as is needed in order for one batch to fit in memory, with the constraint that there are at least 16 batches, and (c) we use the corrections from the perturbative determinants in a small subset of the batches (often one is enough) to estimate the total correction from all the perturbative determinants, as well as its standard error. Looping over batches, for each batch, we calculate the correction from each unique perturbative determinant in that batch. We accumulate the number of unique determinants, the sum, and the sum of squares of the corrections from these determinants. At the end of each batch iteration, we calculate the mean and standard deviation of the corrections from all the evaluated perturbative determinants and use these to estimate the total correction from all the perturbative determinants. Note that the standard deviation of the total correction is the standard deviation of the sum of only the unevaluated determinants. If we process all the batches, the pseudo-stochastic step becomes deterministic and has zero standard deviation. When the standard deviation of the total correction is smaller than 40% of the target error, we exit the loop over batches. However, a single batch is often sufficient to reach a statistical error below that threshold, for the smallest ϵ_1 values that we typically use.

The third step is a stochastic step that is similar to the stochastic step of the original SHCI algorithm, except that instead of keeping all the perturbative determinants that satisfy the ϵ_2 criterion, we keep only one randomly selected

batch out of several. The available computer memory constrains the number of perturbative determinants, and one can obtain the same number sampling a certain number of variational determinants and all the perturbative determinants that satisfy the ϵ_2 criterion (the original SHCI algorithm) or by using a larger number of variational determinants and selecting just one batch of the perturbative determinants. The latter allows us to use much larger variational samples. Using larger variational samples is advantageous because we find that the additional fluctuations due to sampling the perturbative determinants are much smaller than the reduction in the fluctuations due to having larger variational samples. Typically, we use $\epsilon_2 = 10^{-6}\epsilon_1$. Since the smallest ϵ_1 that we use is typically around 10^{-5} Ha, this value is in fact much smaller than is needed to ensure that the perturbative correction is fully converged. For a statistical error of 10^{-5} Ha, 128 batches are usually a good choice to start with. The size of the variational sample is chosen so that a single perturbative batch fits in the available memory. We use a minimum of 10 samples in our stochastic step in order to get a meaningful estimate of the uncertainty. On large memory machines, we often achieve a much smaller statistical error than the target with 10 samples. In that case, we can decrease the size of the variational sample in later runs for similar systems.

In Table II, the last four rows compare the original SHCI to the improved version with 3-step batch perturbation. In the memory-constrained case, the improved SHCI runs more than an order of magnitude faster than the original SHCI. Even when memory is abundant, the improved SHCI is still a few times faster.

The main reasons that the improved SHCI is much faster are as follows: (1) It computes a larger fraction of the perturbative correction in the deterministic step. (2) A small fraction of the batches in the pseudo-stochastic step is usually sufficient to give an accurate estimate of the total correction. (3) It uses much larger samples of variational determinants in the stochastic step.

We now comment on a couple of aspects of our algorithm that may not be obvious:

(1) The value of the perturbative correction depends only on ϵ_2 and not on ϵ_2^{dtm} and ϵ_2^{psto}. The latter two quantities affect only the efficiency of the calculation. By using batches in the stochastic step, we can use a much smaller ϵ_2^{sto} and thereby include almost the entire perturbative space. In our calculations, we usually set $\epsilon_2^{sto} = 10^{-6}\epsilon_1$, which is much smaller than is possible using our previous 2-step perturbation method and much smaller than necessary to keep the systematic error within the target statistical error.

(2) In the pseudo-stochastic step, we estimate the fluctuations of the unevaluated perturbative determinants from the fluctuation of the evaluated perturbative determinants. This relies on having a sufficiently uniform hash function. Note that since we are using all the variational determinants in this step, the fluctuations come just from the perturbative determinants. By contrast, in the stochastic step, the fluctuations come both from the choice of variational determinants and the choice of

batches. In that case, one cannot simply use the standard deviation of the corrections from the evaluated perturbative determinants to estimate the standard deviation of the total correction. So, in the stochastic step, we use a minimum of 10 samples, calculate the correction from each of these samples, and use the standard deviation of these sample corrections to estimate the standard deviation of the total correction.

(3) In the stochastic step, the fluctuation between batches of perturbative determinants is much smaller than the fluctuation between samples of variational determinants. The reason for this is that there are many more perturbative determinants in a batch (each making only a small contribution) than there are variational determinants in a sample. Furthermore, the variational determinants vary greatly in importance. This is why we use importance sampling as described by Eq. (10) when selecting variational determinants and why we precede the stochastic step with the deterministic and pseudo-stochastic steps, but even with these improvements, the fluctuations from the choice of variational samples are much larger than the fluctuation from the choice of batches. Hence, we use only one randomly selected batch of perturbative determinants (typically out of 128 batches) per variational sample.

(4) The use of batches carries a small computational overhead of having to regenerate the perturbative determinants for each batch. Using our method, generating determinants is sufficiently fast that the increase in computational cost would be substantial only if this is done many times. If we employed a purely deterministic algorithm, the number of batches would be very large, but with our 3-step semistochastic algorithm, the number of batches actually computed is sufficiently small in each of the three steps that there is never a large computational overhead.

Finally, we comment on two other algorithms that have been recently proposed for calculating the perturbative correction. First, another very efficient semistochastic algorithm has been proposed by Garniron *et al.*[34] However, that algorithm has, for each perturbative determinant, a loop over the variational determinants to find those that are connected. For the very large number of variational determinants that we employ here (up to 2×10^9), this is impractical. To avoid confusion, we should mention that the reason that their energy for Cr_2 is very different from ours is that they used a nonrelativistic Hamiltonian. Second, another algorithm that uses batches of perturbative determinants to overcome the memory bottleneck has been proposed very recently.[36] It is an efficient deterministic algorithm for memory constrained environments, but for a reasonable statistical error tolerance, e.g., 10^{-5} Ha, a semistochastic approach is usually much faster, as we can see from Table II. Also, in our Cr_2 calculation, we stochastically estimate the perturbative correction from at least trillions of perturbative determinants, for $\epsilon_2 = 3 \times 10^{-12}$ Ha, which probably involves quadrillions of contributions ($n^2v^2N_V = 9 \times 10^{15}$), which is infeasible with a deterministic algorithm.

V. KEY DATA STRUCTURES

In this section, we discuss three key data structures used to store the determinants, the distributed Hamiltonian matrix and the distributed partial sums in the perturbative stage of the calculation.

A. Determinants

We use two different representations of determinants. For storing and accessing determinants locally in memory, we use arrays of bit-packed 64-bit unsigned integers. Each bit represents a spin-orbital. The n-th orbital is represented by the (n mod 64)-th bit of the (n/64)-th integer, where "/" means integer (Euclidean) division and the counting starts from zero. (n mod 64) can be implemented as (n & 63), and (n/64) can be implemented as (n >> 6), where "&" is the bitwise "and" and ">>" is the bitwise right shift. Both operations cost only one clock cycle on modern CPUs.

For transferring the determinants to other nodes or saving them to disk, we use base-128 variable-length integers (VarInts)[46] to compress the 64-bit integers. VarInts take only a few bit operations to compute and reduce the memory footprint by up to 87.5% for small integers, which reduces the network traffic and the size of the wavefunction files considerably, especially for large basis sets.

B. Hamiltonian matrix

We store only the upper triangle of the Hamiltonian matrix. The rows are distributed to each node in a round-robin fashion: the first row goes to the first node, the second row goes to the second node, and when we reach the end of the node array, we loop back and start from the first node again. Each row is a sparse vector, represented by two arrays, one stores the indices of the nonzero elements and the other stores the values.

During the matrix-vector multiplication, each node will apply its own portion of the Hamiltonian to the vector to get a partial resulting vector. The partial results are then merged together using a binomial tree reduction. The work on each node is distributed to the cores with dynamic load balancing. To save space, we store only one copy of the partial resulting vector on each node and each thread updates that vector with hardware atomic operations. In addition, we cache the diagonal of the matrix on each node to speed up the Davidson diagonalization.[38]

C. Partial sums

In the perturbative stage, we loop over the variational determinants $\{D_i\}$ to compute the partial sum $\sum_i H_{ai} c_i$ for each perturbative determinant D_a. The map from D_a to $\sum_i H_{ai} c_i$ is stored in a distributed hash table.[47] This choice is dictated by the enormous number of perturbative determinants we employ. The time complexity of inserting one element into the hash table is $\mathcal{O}(1)$, while for a sorted array, it is $\mathcal{O}(\log(n))$. For large calculations, the prefactor from using hash tables is small compared to the $\log(n)$ cost from using a sorted array.

The distributed hash table is based on lock-free[48] open-addressing[49] linear-probing[50] concurrent hash tables[51] specifically designed for intensive commutative insertion and update operations. The linear-probing technique for conflict resolution has better efficiency than separate chaining during parallel insertion, and the lock-free implementation allows all the threads to almost always operate at their full speed.

On each node, we have n of these concurrent hash tables, where n is the number of nodes. One of them stores the entries belonging to that node, and the other $(n - 1)$ tables store the entries belonging to other nodes' pending synchronization. Each concurrent hash table is implemented as lots of segments (at least four times the number of hardware threads), and each segment can be modified by only one thread at a time. When a thread wants to insert or update a (key, value) pair, it first checks whether the segment that the key belongs to is being used by other threads. If the segment is being used, the thread will insert or update the entry to a thread-local hash table, which will be merged to the main table later periodically. We can do this because the insertion and the update operations of the partial sums are commutative. Hence, each insertion and update is guaranteed to finish within $\mathcal{O}(1)$ time without getting blocked, even for perturbative determinants with lots of connections to the reference determinants. The inter-node synchronization runs periodically so that most of the perturbative determinants will have only one copy during the entire run on the entire cluster, except for those with lots of connections to the reference determinants.

VI. PARALLELIZATION

All the critical parts of SHCI are parallelized with MPI+OpenMP. This section describes the parallelization and the scalability of each part.

When finding the connected determinants, performing the matrix-vector multiplication during the diagonalization, and constructing the Hamiltonian matrix from the auxiliary arrays, we use the round-robin scheme to distribute the load across the nodes and use dynamic load balancing for all the cores on the same node.

We parallelize the construction of the α-single, $\vec{i}_{\alpha\alpha}(j_\alpha)$, and β-single, $\vec{i}_{\beta\beta}(j_\beta)$, arrays on each node, which is the most time-consuming part of constructing the auxiliary arrays. For each entry of $\vec{i}_{\alpha\alpha}$ and $\vec{i}_{\beta\beta}$, we initialize a lock to ensure exclusive modification. We loop over all the $\vec{i}_\alpha(\alpha^{(-1)})$ arrays, and for each (i_α, j_α) pair (which are one excitation away) inside a particular $\vec{i}_\alpha(\alpha^{(-1)})$ array, we lock and append j_α to $\vec{i}_{\alpha\alpha}(i_\alpha)$, and we lock and append i_α to $\vec{i}_{\alpha\alpha}(j_\alpha)$. When both i_α and j_α occur only in the new determinants, the smaller of the two does both appends.

In Figs. 2 and 3, we demonstrate the parallel scalability of our SHCI implementation when applied to a copper atom in a cc-pVTZ basis. We use up to 16 nodes, and each node has 6 cores.

For the variational part, our implementation scales almost linearly up to 4 nodes. At 16 nodes, we have 75% parallel efficiency.

For the perturbative stage, two major factors determine the speedup. One is the additional communication associated with shuffling perturbative determinants across the nodes, which increases with the number of nodes. The other is the speedup

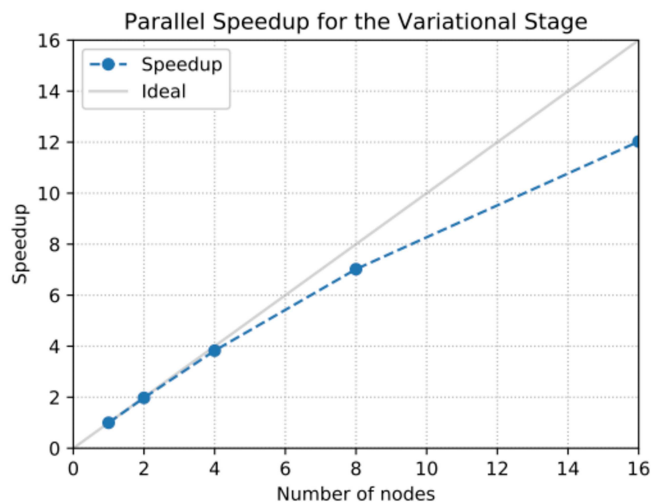## Parallel Speedup for the Variational Stage

FIG. 2. Parallel speedup of the variational stage for a copper atom in a cc-pVTZ basis. There is almost perfect scaling for up to 4 nodes and 75% parallel efficiency at 16 nodes.

from having more cores. We can see from Fig. 3 that from 1 to 4 nodes, the first factor dominates and there is significant deviation from ideal speedup. Starting from 8 nodes, we have to shuffle almost all the perturbative determinants from the spawning node to the storage node that each determinant belongs to, so there is little change in the first factor and the second factor starts to dominate, pushing the speedup curve upward and producing almost perfect scaling. Note that in the original SHCI algorithm,[19] there is a superlinear speedup from using more nodes because many stochastic samples are needed when running in a memory-constrained environment. Here we have solved this problem with the 3-step batch perturbation,

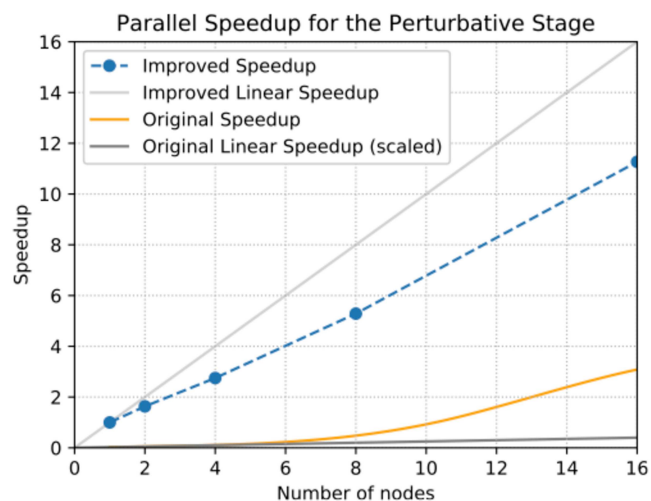## Parallel Speedup for the Perturbative Stage

FIG. 3. Parallel speedup for improved SHCI compared to the original SHCI for the perturbative stage of the calculation for a copper atom in a cc-pVTZ basis. From 1 node to 4 nodes, we see a significant deviation from linear speedup due to the additional communication from shuffling the perturbative determinants across nodes. Starting from 8 nodes, the number of shuffles approaches a constant and we can see an almost linear speedup from using more processors. The estimate of the speedup of the original SHCI is based on the assumption that the total memory of 10 nodes is enough to support the optimal choice of ϵ_2^{dtm} and N_d, in Eqs. (8) and (9). The "original" curves are scaled to reflect the relative speed of the original SHCI algorithm to that of the improved algorithm.

for which the number of stochastic samples is almost always 10. (We require a minimum of 10 samples in order to have a reasonable estimate of the stochastic error.) Consequently, on memory constrained environments, we achieve a few orders of magnitude speedup.

VII. RESULTS

In this section, we apply SHCI to the chromium dimer, which is a challenging strongly correlated system that has been studied using a variety of methods.[52–56,63] We will publish the potential energy surface in a separate publication; here instead, our goal is just to use it as a test case for the improved SHCI method. We use a relativistic exact two-component (X2C) Hamiltonian, the cc-pVDZ-DK basis, and we correlate the valence and the semi-core electrons. This gives an active space of (28e, 76o) and a Hilbert space of 5×10^{29} determinants, which is far beyond the reach of FCI. We show how we obtain an accurate estimate of the FCI energy in this large active space with our improved SHCI algorithm.

We use PySCF[57] to generate the molecular orbital integrals for orbitals that minimize the HCI variational energy for $\epsilon_1 = 2 \times 10^{-4}$ Ha, using the method of Ref. 21. We perform SHCI with several ϵ_1 values from 5×10^{-5} to 3×10^{-6} Ha. The Hamiltonian matrix is constructed only once. We use very small values of $\epsilon_2 = 10^{-6}\epsilon_1$ to ensure that the perturbative correction is exceedingly well converged and choose the target error for the stochastic perturbation energy to be 10^{-5} Ha.

The improved SHCI is fast enough that we can use over two billion variational determinants and stochastically include the contributions of at least trillions of perturbative determinants. The largest variational calculation, where we iteratively find and diagonalize 2×10^9 determinants for $\epsilon_1 = 3.0 \times 10^{-6}$ Ha, takes only one day on 8 nodes, each of which has 4 Intel Xeon E7-8870 v4 CPUs. The corresponding perturbative calculation takes only 6 h using only one of these nodes. During that perturbative calculation, we skip the deterministic step and perform a pseudo-stochastic step with $\epsilon_2^{psto} = 1 \times 10^{-7}$ Ha and a stochastic step with $\epsilon_2 = 3 \times 10^{-12}$ Ha. We skip the deterministic step here because $\epsilon_1 = 3 \times 10^{-6}$ is already close to our default ϵ_2^{dtm} of 2×10^{-6}, so skipping this will not affect the efficiency of subsequent steps much. The pseudo-stochastic step uses 25 batches, each of which has about 8.9×10^9 determinants. We evaluate only one of them, from which we obtain an estimate of the total correction for all the 25 batches (223×10^9 determinants) to be $-0.011\,681(1)$ Ha. Since the estimated error is already much smaller than our target error, we skip the remaining 24 batches. The pseudo-stochastic step takes 1.6 h. The stochastic step uses 128 batches and 6×10^6 variational determinants in each sample, which results in about 3.7×10^9 determinants per batch. We use 10 samples and obtain the additional correction from $\epsilon_2 = 3.0 \times 10^{-12}$ Ha to be $-0.001\,203(6)$ Ha. The combined uncertainty of the entire semistochastic perturbation stage is 6 μHa. It is hard to estimate how many determinants are stochastically included for $\epsilon_2 = 3 \times 10^{-12}$ Ha, so we estimate a lower bound with $\epsilon_2^{psto} = 1.4 \times 10^{-8}$ Ha and obtain 1.8 trillion unique perturbative determinants. Hence, with $\epsilon_2 = 3 \times 10^{-12}$ Ha (the value we are actually using), we

TABLE III. Results for Cr_2 at r = 1.68 Å in the cc-pVDZ-DK basis. The active space is (28e, 76o). N_V is the number of variational determinants. $\epsilon_2 = 10^{-6}\epsilon_1$. We use weighted quadratic extrapolation, shown in Fig. 4, to obtain the FCI limit corresponding to $\Delta E = 0$.

ϵ_1 (Ha)	N_V	E_{var} (Ha)	E_{total} (Ha)
5.0×10^{-5}	24M	−2099.863 816	−2099.909 741(7)
3.0×10^{-5}	53M	−2099.875 327	−2099.912 356(7)
2.0×10^{-5}	102M	−2099.883 027	−2099.914 132(8)
1.0×10^{-5}	309M	−2099.893 761	−2099.916 595(1)
7.0×10^{-6}	539M	−2099.898 165	−2099.917 540(1)
5.0×10^{-6}	911M	−2099.901 781	−2099.918 306(3)
3.0×10^{-6}	2.00B	−2099.906 322	−2099.919 205(6)
0.0 (extrap.)	...		−2099.922 4(6)

stochastically estimate contributions from at least trillions of unique perturbative determinants and obtain better than 10^{-5} Ha statistical uncertainty in 6 h using only one node.

These large calculations enable us to obtain an estimate of the FCI energy with sub-millihartree uncertainty in this large active space. Table III reports the results.

We extrapolate our results using a weighted quadratic fit and obtain for the ground state energy, −2099.9224 Ha as $\Delta E \to 0$. The weight of each point is $(E_{var} - E_{tot})^{-2}$. Figure 4 shows the computed energies and the extrapolation. We also perform a weighted linear fit and use the difference of the extrapolated values from the quadratic and the linear fits (0.6 mHa) as the uncertainty. In summary, the estimated FCI energy of Cr_2 in the cc-pVDZ-DK basis with 28 correlated electrons and the relativistic X2C Hamiltonian is −2099.9224(6) Ha.

We compare our result with DMRG and p-DMRG, which are the only essentially exact methods that have been applied to this large active space of the chromium dimer. The DMRG calculations use up to bond dimension $M = 16\,000$ and obtain extrapolated energies of −2099.9195(27) Ha (default
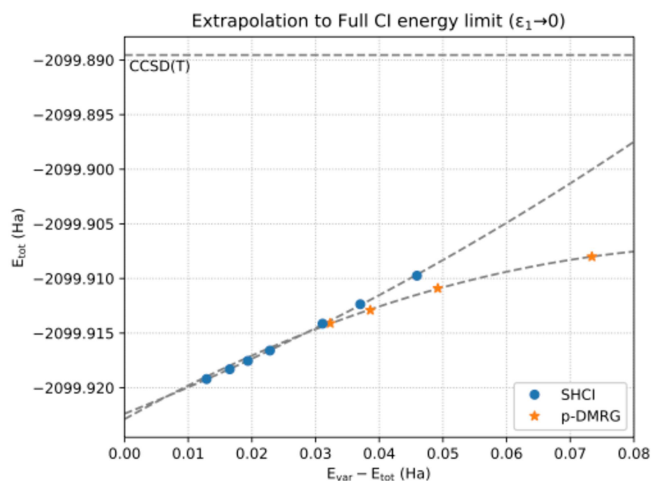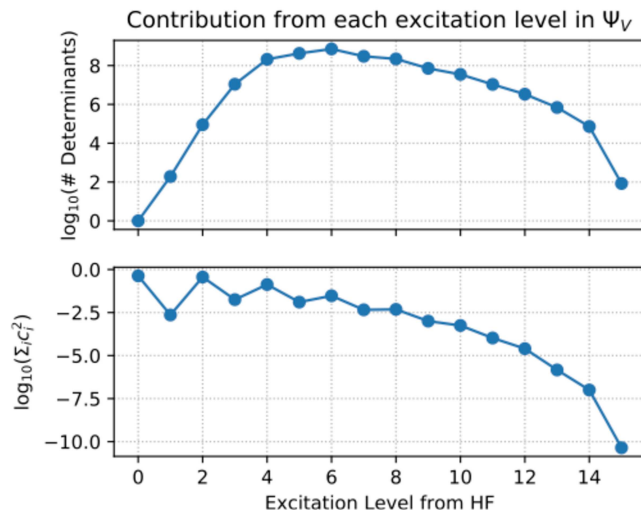

FIG. 5. Contribution from each excitation level to the variational wavefunction for Cr_2 with 2×10^9 determinants. Determinants with up to 15 excitations are present in the variational wavefunction.

schedule) and −2099.9192(24) (reverse schedule).[12] These two values are similar to our most accurate data point but higher than our extrapolated result by 3 mH, which is about the estimated error of the DMRG results. The p-DMRG calculations use up to $M = 4000$, and extrapolated energy obtained from a linear fit is −2099.9201 Ha.[12] If instead we perform a weighted quadratic fit (shown in Fig. 4), the extrapolated energy is −2099.9225 Ha, in perhaps fortuitously good agreement with our result of −2099.9224(6) Ha. However, the extrapolation uncertainty is larger than the SHCI extrapolation uncertainty. By contrast, the CCSD(T) energy is considerably higher.

One of the merits of selected-CI methods is the ability to include all excitations, regardless of the excitation level. To see the contribution from each excitation level, we plot the number of selected determinants and the $\sum_i |c_i|^2$ versus excitation level in Fig. 5. Determinants with excitation levels up to 15 excitations are present in the variational wavefunction even though we are using optimized orbitals. (Using Hartree-Fock orbitals, we expect that determinants with even higher excitation levels will be present.) This implies that truncating the CI expansion at the double, triple, or quadruple excitation levels (which is the most that is usually done in systematic CI expansions), will give poor energies for such strongly correlated systems.

VIII. CONCLUSION

In this paper, we introduced our fast semistochastic heat-bath configuration interaction algorithm, an efficient and essentially exact algorithm, for estimating the FCI energy. We introduced a new Hamiltonian generation algorithm and a 3-step batch perturbation algorithm to overcome the bottlenecks in the original SHCI algorithm. We also presented the key data structures and parallelization strategy, which are also crucial to the performance. These improvements allowed us to use 2×10^9 variational determinants, which is more than one order of magnitude larger than the 9×10^7 determinants used in our earlier SHCI calculation[23] and two orders of magnitude larger than the largest variational space of 2×10^7 determinants employed to date in any other selected CI method.[34]

FIG. 4. Weighted quadratic extrapolation of the Cr_2 ground state energy. The weight of each point is $(E_{var} - E_{tot})^{-2}$. The extrapolated energy is −2099.9224(6), where the uncertainty comes from the difference between linear extrapolation and quadratic extrapolation. The p-DMRG extrapolation and the CCSD(T) value are also shown.

Future extensions of the method include going to yet larger variational spaces using a *direct* method,[58,59] wherein the Hamiltonian matrix is recalculated at each Davidson iteration and therefore need not be stored. Although this increases the computational cost, the increase is not overwhelming because of the use of the auxiliary arrays introduced in this paper. Other extensions include increasing the range of applicability of the method to larger systems by combining SHCI with range-separated density functional theory[60] and the use of SHCI as an impurity solver in embedding theories. Recently, a selected coupled cluster method has been developed.[61] Although the current version has only been used with small basis sets, it is possible that with further development, this will become a highly competitive method, especially for weakly correlated systems.

Possible applications of the SHCI method include providing benchmark energies for a variety of organic molecules, as well as for transition metal atoms, dimers, and monoxides, and calibration or training data for large scale methods, e.g., to calibrate interatomic potentials for molecular dynamics and exchange-correlation functionals for density functional theory and to train machine learning based quantum chemistry solvers. Calculations on the homogeneous electron gas are also underway.

ACKNOWLEDGMENTS

[1] R. G. Parr and W. Yang, *Density-Functional Theory of Atoms and Molecules* (Oxford University Press, New York, 1989).
[2] R. M. Dreizler and E. K. U. Gross, *Density Functional Theory* (Springer-Verlag, Berlin, 1990).
[3] W. Kohn, Rev. Mod. Phys. **71**, 1253 (1999).
[4] K. Raghavachari, G. W. Trucks, J. A. Pople, and M. Head-Gordon, Chem. Phys. Lett. **157**, 479 (1989).
[5] S. R. White, Phys. Rev. B **48**, 10345 (1993).
[6] S. R. White and R. L. Martin, J. Chem. Phys. **110**, 4127 (1999).
[7] G. K.-L. Chan and M. Head-Gordon, J. Chem. Phys. **116**, 4462 (2002).
[8] G. K.-L. Chan and S. Sharma, Annu. Rev. Phys. Chem. **62**, 465 (2011).
[9] S. Sharma and G. K.-L. Chan, J. Chem. Phys. **136**, 124121 (2012).
[10] R. Olivares-Amaya, W. Hu, N. Nakatani, S. Sharma, J. Yang, and G. K.-L. Chan, J. Chem. Phys. **142**, 034102 (2015).
[11] U. Schollwöck, Rev. Mod. Phys. **77**, 259 (2005).
[12] S. Guo, Z. Li, and G. K.-L. Chan, J. Chem. Theory Comput. **14**, 4063 (2018).
[13] G. H. Booth, A. J. Thom, and A. Alavi, J. Chem. Phys. **131**, 054106 (2009).
[14] D. Cleland, G. H. Booth, and A. Alavi, J. Chem. Phys. **132**, 041103 (2010).
[15] F. R. Petruzielo, A. A. Holmes, H. J. Changlani, M. P. Nightingale, and C. J. Umrigar, Phys. Rev. Lett. **109**, 230201 (2012).
[16] G. H. Booth, A. Grüneis, G. Kresse, and A. Alavi, Nature **493**, 365 (2013).
[17] A. A. Holmes, H. J. Changlani, and C. J. Umrigar, J. Chem. Theory Comput. **12**, 1561 (2016).
[18] A. A. Holmes, N. M. Tubman, and C. J. Umrigar, J. Chem. Theory Comput. **12**, 3674 (2016).
[19] S. Sharma, A. A. Holmes, G. Jeanmairet, A. Alavi, and C. J. Umrigar, J. Chem. Theory Comput. **13**, 1595 (2017).
[20] A. A. Holmes, C. J. Umrigar, and S. Sharma, J. Chem. Phys. **147**, 164111 (2017).
[21] J. E. Smith, B. Mussard, A. A. Holmes, and S. Sharma, J. Chem. Theory Comput. **13**, 5468 (2017).
[22] B. Mussard and S. Sharma, J. Chem. Theory Comput. **14**, 154 (2017).
[23] A. D. Chien, A. A. Holmes, M. Otten, C. J. Umrigar, and S. Sharma, J. Phys. Chem. A **122**, 2714 (2018).
[24] B. Huron, J. Malrieu, and P. Rancurel, J. Chem. Phys. **58**, 5745 (1973).
[25] R. J. Buenker and S. D. Peyerimhoff, Theor. Chim. Acta **35**, 33 (1974).
[26] S. Evangelisti, J.-P. Daudey, and J.-P. Malrieu, Chem. Phys. **75**, 91 (1983).
[27] R. Cimiraglia and M. Persico, J. Comput. Chem. **8**, 39 (1987).
[28] R. J. Harrison, J. Chem. Phys. **94**, 5021 (1991).
[29] L. Bytautas and K. Ruedenberg, Chem. Phys. **356**, 64 (2009).
[30] T. P. Kelly, A. Perera, R. J. Bartlett, and J. C. Greer, J. Chem. Phys. **140**, 084114 (2014).
[31] J. Coe, P. Murphy, and M. Paterson, Chem. Phys. Lett. **604**, 46 (2014).
[32] F. A. Evangelista, J. Chem. Phys. **140**, 124114 (2014).
[33] A. Scemama, T. Applencourt, E. Giner, and M. Caffarel, J. Comput. Chem. **37**, 1866 (2016).
[34] Y. Garniron, A. Scemama, P.-F. Loos, and M. Caffarel, J. Chem. Phys. **147**, 034101 (2017).
[35] P.-F. Loos, A. Scemama, A. Blondel, Y. Garniron, M. Caffarel, and D. Jacquemin, J. Chem. Theory Comput. **14**, 4360–4379 (2018).
[36] N. M. Tubman, D. S. Levine, D. Hait, M. Head-Gordon, and K. B. Whaley, e-print arXiv:1808.02049.
[37] Since the absolute values of c_i for the most important determinants tend to go down as more determinants are included in the wavefunction, a somewhat better selection of determinants is obtained by using a larger value of ϵ_1 in the initial iterations.
[38] E. R. Davidson, Comput. Phys. Commun. **53**, 49 (1989).
[39] P. S. Epstein, Phys. Rev. **28**, 695 (1926).
[40] R. K. Nesbet, Proc. R. Soc. London, Ser. A. **230**, 312 (1955).
[41] A. J. Walker, ACM Trans. Math. Software **3**, 253 (1977).
[42] R. A. Kronmal and A. V. Peterson, Jr., Am. Stat. **33**, 214 (1979).
[43] We employ the Trail-Needs pseudopotential,[62] but the conclusions of this paper would be the same for an all-electron calculation.
[44] B. Jenkins, Dr. Dobbs J. **22**, 107 (1997).
[45] See https://www.boost.org/doc/libs for Boost C++ Libraries (2012).
[46] S. Stuart and R. Fernando, https://tools.ietf.org/html/draft-rfernando-protocol-buffers-00 (2012).
[47] See https://en.wikipedia.org/wiki/Distributed_hash_table for Distributed hash table; accessed 12 October 2018.
[48] See https://en.wikipedia.org/wiki/Non-blocking_algorithm for Non-blocking algorithm; accessed 12 October 2018.
[49] See https://en.wikipedia.org/wiki/Open_addressing for Open addressing; accessed 12 October 2018.
[50] See https://en.wikipedia.org/wiki/Linear_probing for Linear probing; accessed 12 October 2018.
[51] Z. Chen, X. He, J. Sun, H. Chen, and L. He, Future Gener. Comput. Syst. **82**, 127 (2018).
[52] G. E. Scuseria, J. Chem. Phys. **94**, 442 (1991).
[53] W. Purwanto, S. Zhang, and H. Krakauer, J. Chem. Phys. **142**, 064302 (2015).
[54] D. Ma, G. L. Manni, J. Olsen, and L. Gagliardi, J. Chem. Theory Comput. **12**, 3208 (2016).
[55] S. Vancoillie, P. A. Malmqvist, and V. Veryazov, J. Chem. Theory Comput. **12**, 1647 (2016).
[56] S. Guo, M. A. Watson, W. Hu, Q. Sun, and G. K.-L. Chan, J. Chem. Theory Comput. **12**, 1583 (2016).
[57] Q. Sun, T. C. Berkelbach, N. S. Blunt, G. H. Booth, S. Guo, Z. Li, J. Liu, J. McClain, S. Sharma, S. Wouters, and G. K.-L. Chan, Wiley Interdiscip. Rev.: Comput. Mol. Sci. **8**, e1340 (2018).
[58] P. Knowles and N. Handy, Chem. Phys. Lett. **111**, 315 (1984).
[59] J. Ivanic and K. Ruedenberg, Theor. Chem. Acc. **106**, 339 (2001).
[60] A. Savin, in *Recent Developments of Modern Density Functional Theory*, edited by J. M. Seminario (Elsevier, Amsterdam, 1996), pp. 327–357.
[61] E. Xu, M. Uejima, and S. L. Ten-no, Phys. Rev. Lett. **121**, 113001 (2018).
[62] J. R. Trail and R. J. Needs, J. Chem. Phys. **142**, 064110 (2015).
[63] Y. Kurashige and T. Yanai, J. Chem. Phys. **135**, 094104 (2011).