Multi-determinant Compression Algorithm for Quantum Monte Carlo

Gihan L Weerasinghe, Pablo López Ríos, Richard J Needs

TCM group, Cavendish Laboratory, University of Cambridge.

Multi-determinant expansions

Multi-determinant expansions are a tool to construct **very accurate** wave functions, and are frequently used in quantum Chemistry methods.

Slater determinants constitute a **basis** for anti-symmetric functions in $\mathscr{R}^{3N}$, thus an expansion in Slater determinants converges to the **exact** wave function of the system as the number of determinants tends to ∞.

The multi-determinant expansion

Multi-determinant expansion

$$\Psi_{\mathrm{MD}}(\mathbf{R}) = \sum_{k=1}^{N_s} c_k \Phi_k^{\uparrow}(\mathbf{R}_{\uparrow}) \Phi_k^{\downarrow}(\mathbf{R}_{\downarrow})$$

- $\Phi_k^{\uparrow}(\mathbf{R}_{\uparrow}) = \det\left[\phi_{a_{i,k}^{\uparrow}}(\mathbf{r}_j^{\uparrow})\right]$
- $\Phi_k^{\downarrow}(\mathbf{R}_{\downarrow}) = \det\left[\phi_{a_{i,k}^{\downarrow}}(\mathbf{r}_j^{\downarrow})\right]$
- $a_{i,k}^{\sigma}$ is an orbital-selecting index

Cost and limitations

Multi-determinant expansions have the drawback that the number of determinants N_s required to obtain a wave function of a given accuracy **grows very rapidly** with system size N.

- This is why methods such as CI typically scale as N^7
- Restricts multi-determinant wave fuctions to **small systems**

The cost of running a given number of steps in a QMC calculation is proportional to N_s.

Methods to **speed up** the evaluation of multi-determinant wave functions **extend the range** of system sizes where this tool can be used.

Notation

For convenience, we use a compact notation to represent determinants:

Notation for determinants

$$[\phi_{a_1}, \phi_{a_2}, \ldots, \phi_{a_n}] \equiv \begin{vmatrix} \phi_{a_1}(\mathbf{r}_1) & \phi_{a_2}(\mathbf{r}_1) & \cdots & \phi_{a_n}(\mathbf{r}_1) \\ \phi_{a_1}(\mathbf{r}_2) & \phi_{a_2}(\mathbf{r}_2) & \cdots & \phi_{a_n}(\mathbf{r}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{a_1}(\mathbf{r}_n) & \phi_{a_2}(\mathbf{r}_n) & \cdots & \phi_{a_n}(\mathbf{r}_n) \end{vmatrix}$$

De-duplicating determinants

Multi-determinant wave functions are commonly written as a sum of **Configuration State Functions** (CSFs), which are themselves a sum of determinant products obeying a certain symmetry.

Each determinant product may appear in **more than one** CSF, and hence a multi-determinant expansion may contain **multiple instances** of a given determinant product. We combine these into a single term:

De-duplication example

$$c_1\Phi^{\uparrow}\Phi^{\downarrow} + c_2\Phi^{\uparrow}\Phi^{\downarrow} = (c_1 + c_2)\Phi^{\uparrow}\Phi^{\downarrow} = c_1'\Phi^{\uparrow}\Phi^{\downarrow}$$

This operation may involve more than two terms. The de-duplication process is quick and simple, and independent from what follows.

Combining determinants

The basic operation at the core of our compression algorithm is:

Basic combination example, simplified

$$[\phi_{a_1}, \phi_{a_3}, \ldots, \phi_{a_4}] + [\phi_{a_2}, \phi_{a_3}, \ldots, \phi_{a_4}] =$$
$$[\phi_{a_1} + \phi_{a_2}, \phi_{a_3}, \ldots, \phi_{a_4}] =$$
$$[\tilde{\phi}_{\tilde{a}_1}, \phi_{a_3}, \ldots, \phi_{a_4}]$$

That is, we can "compress" two determinants **which differ by a single orbital** into **one** determinant by defining a **new orbital** which is a linear combination of two of the orbitals in the original pool of orbitals.

Combining determinants

In practice, we need to consider the **other-spin determinant**, which must be **equal** in both terms, and the (de-duplicated) expansion coefficients:

Basic combination example

$$\begin{aligned} c'_1[\phi_{a_1},\phi_{a_3},\ldots,\phi_{a_4}]\Phi^{\downarrow}+c'_2[\phi_{a_2},\phi_{a_3},\ldots,\phi_{a_4}]\Phi^{\downarrow} &= \\ [c'_1\phi_{a_1}+c'_2\phi_{a_2},\phi_{a_3},\ldots,\phi_{a_4}]\Phi^{\downarrow} &= \\ \tilde{c}_1[\tilde{\phi}_{\tilde{a}_1},\phi_{a_3},\ldots,\phi_{a_4}]\Phi^{\downarrow} & \end{aligned}$$

This operation may involve more than two terms.

Recursive combination

In some cases determinants which have already been compressed can be **compressed again**:

Recursive combination example

$$
\begin{aligned}
[\phi_{a_1},\phi_{a_3}]+[\phi_{a_2},\phi_{a_3}]+[\phi_{a_1},\phi_{a_4}]+[\phi_{a_2},\phi_{a_4}] &= \\
[\phi_{a_1}+\phi_{a_2},\phi_{a_3}]+[\phi_{a_1}+\phi_{a_2},\phi_{a_4}] &= \\
[\phi_{a_1}+\phi_{a_2},\phi_{a_3}+\phi_{a_4}]
\end{aligned}
$$

This operation may involve several terms.

Generalization

A compressed multi-determinant expansion may be expressed as

General form of compressed expansion

$$\Psi_{\rm MD}(\mathbf{R}) = \sum_{k=1}^{N_c} \tilde{c}_k \det\left[\tilde{\phi}_{\tilde{a}_{ik}^{\uparrow}}(\mathbf{r}_j^{\uparrow})\right] \det\left[\tilde{\phi}_{\tilde{a}_{ik}^{\downarrow}}(\mathbf{r}_j^{\downarrow})\right]$$

$$\tilde{c}_k = \pm c'_{\nu_{k1}} \prod_{p=2}^{P_k} \frac{c'_{\nu_{kp}}}{c'_{\delta_{kp}}}$$

$$\tilde{\phi}_a(\mathbf{r}) = \sum_{x=1}^{X_a} \pm \prod_{q=1}^{Q_{ax}} \frac{c'_{n_{axq}}}{c'_{d_{axq}}} \phi_{\mu_{ax}}(\mathbf{r})$$

Properties of the expansion

From the form of the expansion it can be seen that:

- The compressed expansion is a **regular multi-determinant expansion** and can be evaluated with **the usual algorithms**.
- The evaluation of the expansion is **reduced** by a factor of N_c/N_s, although there is the **additional step** of calculating the set of compressed orbitals from the original orbitals.
- The compressed expansion coefficients can be **easily recalculated** if the original coefficients change, i.e., during optimization.
  NB, we optimize the original expansion coefficients to be able to **keep the constraints** on the parameters, e.g., proportionality between coefficients in the same CSF.

Selecting operations

It may be possible to combine a given term in an expansion with more than one other term:

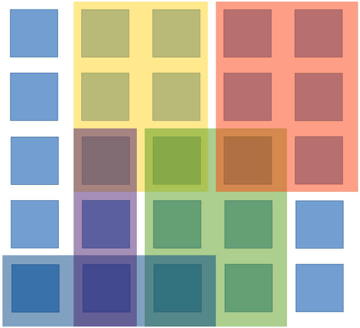
Multiple choices

$$\begin{aligned}
[\phi_{a_1},\phi_{a_3}]+[\phi_{a_2},\phi_{a_3}]+[\phi_{a_1},\phi_{a_4}] &= \\
[\phi_{a_1}+\phi_{a_2},\phi_{a_3}]+[\phi_{a_1},\phi_{a_4}] &= \\
[\phi_{a_1},\phi_{a_3}+\phi_{a_4}]+[\phi_{a_2},\phi_{a_3}] &
\end{aligned}$$

Choosing one grouping over another gives a different compressed expansion. One needs to choose the groupings so as to **minimize** the number of determinants in the compressed expansion N_c.

Set covering problem

This can be reformulated as a set-covering problem: find the least number of subsets of a set that cover the entire set.



The greedy algorithm

The set-covering problem can be approximately solved in polynomial time using the "greedy algorithm".

The greedy algorithm

Pick the subsets from largest to smallest until the set is covered.

The greedy algorithm gives very acceptable results in practice, but it does not give the optimal solution to the set-covering problem in general.

Linear programs

The set-covering problem can be **solved exactly** using a **binary linear program**.

A binary linear program is an optimization problem where an **objective function** of binary **unknowns** is to be optimized subject to a set of linear **constraints**.

In the case of compression:

- the unknowns determine whether a subset is **picked or not**
- the constraints make sure we pick each term exactly **once**
- the objective function is the **number of terms** in the compressed expansion.

Potential issues with the cost of compression

Linear programs are in principle **NP-hard**. However in practice it is possible to solve **many** of them **quickly**.

The construction of the **complete set** of operations that can be performed on the multi-determinant expansion is potentially **very costly**. This can be easily **avoided**, except in the presence of **recursively-compressible terms** where the full set of recursive operations need to be explicitly listed.

It is possible to deal approximately with recursively-compressible terms by applying the compression algorithm to the compressed expansion **iteratively** until not further gains are obtained.

Limitations

We do not take into account the ability to do something like:

Using a term multiple times

$$[\phi_{a_1},\phi_{a_2}]+2[\phi_{a_1},\phi_{a_3}]+[\phi_{a_2},\phi_{a_3}]+[\phi_{a_1},\phi_{a_4}]+2[\phi_{a_2},\phi_{a_4}]+[\phi_{a_3},\phi_{a_4}]=$$
$$[\phi_{a_1},\phi_{a_2}+\phi_{a_3}]+[\phi_{a_1}+\phi_{a_2},\phi_{a_3}]+[\phi_{a_1}+\phi_{a_2},\phi_{a_4}]+[\phi_{a_2}+\phi_{a_3},\phi_{a_4}]=$$
$$[\phi_{a_1}-\phi_{a_4},\phi_{a_2}+\phi_{a_3}]+[\phi_{a_1}+\phi_{a_2},\phi_{a_3}+\phi_{a_4}]$$

where the second and fifth terms have been used twice. This type of operation would complicate matters quite a bit.

Implementation

We provide a GPLv3 implementation of the compression algorithm,

Our implementation

`http://www.tcm.phy.cam.ac.uk/~pl275/DetCompress`

This implementation offers three levels of compression:

- "Quick": greedy algorithm, iterative recursion
- "Good": linear program, iterative recursion
- "Best": linear program with full recursion

Less-costly levels are available in case the exponential cost of the potentially problematic parts of the algorithm manifests itself.

Results: number of determinants

	N_{CSF}	Original	De-dupe	"Quick"	"Good"	"Best"
Be_2	61	200	200	100	97	97
N	50	1271	764	324	324	324
O	100	3386	1271	535	534	534
Li	500	8140	5824	1226	1210	1210
B	500	14057	5703	530	529	529
Be	500	14212	10600	2218	2177	2163
Ne	400	22827	16260	1844	1805	1805
F	600	57456	17174	2801	2749	2747

The reduction in the number of determinants is very significant.
All three levels offer very similar performance.

Results: number of orbitals

	N_{CSF}	Original	De-dupe	"Quick"	"Good"	"Best"
Be_2	61	12	12	60	72	73
N	50	51	51	191	188	188
O	100	53	53	365	361	361
Li	500	105	105	1023	1036	1046
B	500	105	105	629	629	631
Be	500	105	105	1174	1188	1198
Ne	400	105	105	1182	1191	1191
F	600	105	105	2553	2622	2627

The increase in the number of orbitals is likewise very significant.

Results: compression CPU time

	N_{CSF}	De-dupe	"Quick"	"Good"	"Best"
Be_2	61	0.001(1)	0.011(1)	0.022(1)	0.023(1)
N	50	0.012(1)	0.038(2)	0.044(1)	0.054(1)
O	100	0.058(1)	0.125(1)	0.144(1)	0.143(1)
Li	500	0.332(6)	1.270(2)	1.430(3)	3.144(3)
B	500	0.79(1)	1.802(2)	1.884(7)	3.73(1)
Be	500	1.18(1)	4.63(1)	5.17(2)	7.82(3)
Ne	400	4.56(3)	14.79(8)	15.04(6)	18.77(4)
F	600	11.43(1)	22.3(2)	22.5(2)	24.18(4)

The three levels take very similar CPU times.

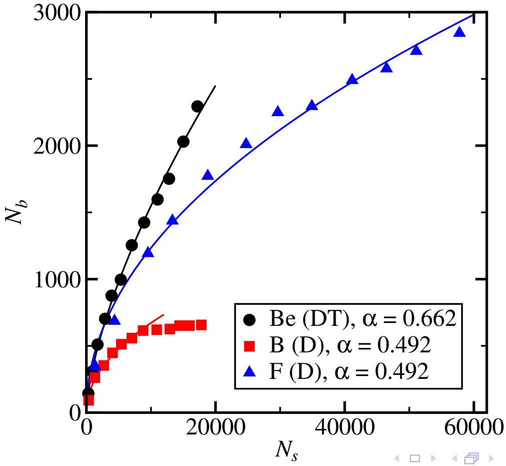
Results: QMC CPU time

	N_s/N_b	T_s/T_b
Be_2	2.06	1.885(3)
N	3.92	3.718(9)
O	6.34	6.09(1)
Li	6.73	6.50(2)
B	26.57	25.23(4)
Be	6.57	6.48(1)
Ne	12.65	13.17(2)
F	20.92	21.77(5)

CPU times closely mirror compression ratios. Computation of compressed orbitals has an insignificant impact in our tests.

Results: scaling of QMC CPU time

The dependence of the cost on N_s appears to become sub-linear when compression is used:



Summary

Compression method for reducing the cost of evaluating multi-determinant wave functions.

- Leaves evaluation algorithms unchanged.
- Leaves original coefficients optimizable.
- Achieves compression ratios of 2 – 25 in tests.
- Compression ratio directly translates to QMC speed-up.
- Sub-linear scaling of cost of QMC with expansion size.
- Future work: combine with accelerated evaluation methods.
- Implementation: `http://www.tcm.phy.cam.ac.uk/~pl275/DetCompress`
- Paper: PRE **89**, 023304 (2014)

End

End