Novel Algorithms for Computing Correlation Functions of Large Nuclei

Nabil Humphrey

Special Research Centre for the Subatomic Structure of Matter (CSSM)
University of Adelaide

July 25, 2021

Collaborators:
Ross Young (CSSM), James Zanotti (CSSM), Phiala Shanahan (MIT CTP), Will Detmold (MIT CTP), Artur Avkhadiev (MIT CTP)

1. Introduction
2. Factor Trees

Nuclear Structure Through Lattice QCD

- This work seeks to enable more detailed lattice probes into the nuclear structure of (relatively) large nuclei
- Key Challenges:
    1. Signal-to-Noise scaling: errors generally scale poorly with quark number
    2. Identifying physically relevant states: achieving good overlap with the ground-state becomes increasingly difficult for many-hadron systems
    3. **Numerical correlator evaluation**:
        - **Wick contractions scale factorially in quark number**
        - **Index set scales exponentially in quark number**
        - **Floating point errors interact poorly with delicate cancellations**
- Key Resources:
    1. Discrete permutation symmetry within interpolating operators – both term-wise and factor-wise
    2. Common subexpressions
    3. Highly iterated computational workflow: it's worth putting in upfront resources to save on compute overall

Algorithm Anthology

- Hadron Blocks [e.g. Detmold, Orginos (2013) or Doi, Endres (2013)]
- Index Lists [Doi, Endres (2013)]
  - For an expensive tensor $C^{pn}_{\alpha'\beta'}(\xi'_1, \ldots, \xi'_6)$, pre-compute which subset of the index set has non-vanishing contribution to the correlator
- Recursive Formulation [e.g. for mesons: Detmold, Savage (2010)]
- Determinant Formulation [Detmold, Orginos (2013)]
  - Map fermion anti-symmetry onto matrix determinant anti-symmetry, then use LU factorisation to attain $\mathcal{O}(n^3)$ scaling rather than $\mathcal{O}(n!)$ scaling – only Wick Contraction scaling, not index set scaling
- Tensor Expression Canonicalisation (using Vertex-Labelled Graph Canonicalisation)
  - Map the tensor network of a tensor expression to a vertex-labelled graph such that graph isomorphism corresponds to tensor expression equivalence
- **Factor Trees**
- Tensor E-graphs
  - Simultaneously explore all possible common tensor subexpressions and extract the set that minimises a cost function

Interpolating Operators

- Begin with standard baryon operators, e.g.:

$$
\begin{aligned}
p^{\alpha} &= \epsilon_{abc}(u_a^T(C\gamma_5)d_b)u_c^{\alpha} \\
n^{\alpha} &= \epsilon_{abc}(u_a^T(C\gamma_5)d_b)d_c^{\alpha} \\
p_{\pm}^{\alpha} &= \epsilon_{abc}(u_a^T(C\gamma_5 P_{\pm})d_b)u_c^{\alpha} \\
n_{\pm}^{\alpha} &= \epsilon_{abc}(u_a^T(C\gamma_5 P_{\pm})d_b)d_c^{\alpha}
\end{aligned}
$$

- Construct candidate multi-baryon operators, e.g.:

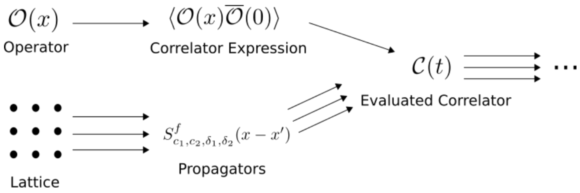
Deuteron I: $D_I(x) = n^T(x)(C\gamma_5)p(x)$

Deuteron II: $D_{II}(x) = \frac{1}{\sqrt{2}}\left[n^T(x)(C\gamma_5)p(x) - p^T(x)(C\gamma_5)n(x)\right]$

Helium-3 I: ${}^3He_I^j(x) = p_-^T(x)(C\gamma_5)n_+(x)p_+^j(x)$

Helium-3 II:
$$
\begin{aligned}
{}^3He_{II}^j(x) = \frac{1}{\sqrt{6}}\Big[&p_-^T(x)(C\gamma_5)n_+(x)p_+^j(x) - p_+^T(x)(C\gamma_5)n_+(x)p_-^j(x) \\
&+n_+^T(x)(C\gamma_5)p_+(x)p_-^j(x) - n_+^T(x)(C\gamma_5)p_-(x)p_+^j(x) \\
&+p_+^T(x)(C\gamma_5)p_-(x)n_+^j(x) - p_-^T(x)(C\gamma_5)p_+(x)n_+^j(x)\Big]
\end{aligned}
$$

Computational Workflow



for each e.g. time slice t:

- Construct $C(t)$ from e.g. proton: $p_\alpha(x) = \epsilon_{abc} u^a_\sigma(x)(C\gamma_5)_{\sigma\rho} d^b_\rho(x) u^c_\alpha(x)$

$$C(t) = \left\langle \sum_{\vec{x}} p_\alpha(x)\overline{p}_{\alpha'}(0) \right\rangle$$

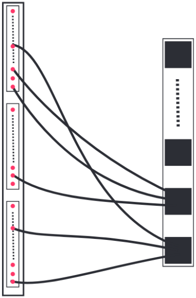
$$= \sum_{\vec{x}} \epsilon_{abc}\epsilon_{a'b'c'}(C\gamma_5)_{\sigma\rho}(C\gamma_5)_{\sigma'\rho'} \left\langle u^a_\sigma(x) d^b_\rho(x) u^c_\alpha(x) \overline{u}^{a'}_{\sigma'}(0)\overline{d}^{b'}_{\rho'}(0)\overline{u}^{c'}_{\alpha'}(0) \right\rangle$$

$$= \sum_{\vec{x}} \epsilon_{abc}\epsilon_{a'b'c'}(C\gamma_5)_{\sigma\rho}(C\gamma_5)_{\sigma'\rho'} \left[-S^{u;aa'}_{\sigma\sigma'}(x) S^{u;cc'}_{\alpha\alpha'}(x) + S^{u;ac'}_{\sigma\alpha'}(x) S^{u;ca'}_{\alpha\sigma'}(x) \right] S^{d;bb'}_{\rho\rho'}(x)$$

- Evaluate $C(t)$ for the corresponding time slice of $S^{f;c_1,c_2}_{\delta_1,\delta_2}(x-x')$

Hadron Blocks

- Construct a tensor corresponding to a set of quarks created at the source and annihilated as a momentum-projected hadron at the sink
- Block expressions are re-used in the course of evaluating the multi-hadron correlator
- The factorial number of Wick Contractions is suppressed by a factor of 2^A for A baryons



$$f^P_\alpha(x', a', \beta', b', \gamma', c', \alpha') := \left\langle \sum_{\vec{x}'} p_\alpha(x')\bar{u}^{a'}_{\beta'}(x)\bar{d}^{b'}_{\gamma'}(x)\bar{u}^{c'}_{\alpha'}(x) \right\rangle$$

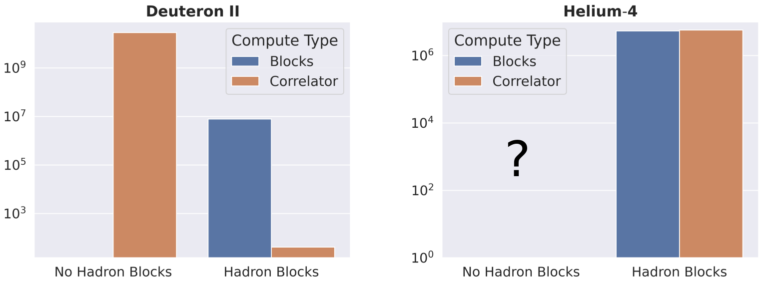
$\mathcal{O}(x) \longrightarrow \langle \mathcal{O}(x)\overline{\mathcal{O}}(0)\rangle \longrightarrow \langle \mathcal{O}(x)\overline{\mathcal{O}}_B(0)\rangle \longrightarrow \mathcal{C}(t) \Longrightarrow \cdots$

Operator | Correlator Expression | Sink Block Correlator Expression | Evaluated Correlator

Lattice $\Longrightarrow S^f_{c_1,c_2,\delta_1,\delta_2}(x - x') \Longrightarrow f^{P/N}_{\alpha'}(x, a, \beta, b, \gamma, c, \alpha) \Longrightarrow \mathcal{C}(t)$

Lattice | Propagators | Hadron Blocks

Hadron Blocks Benchmark

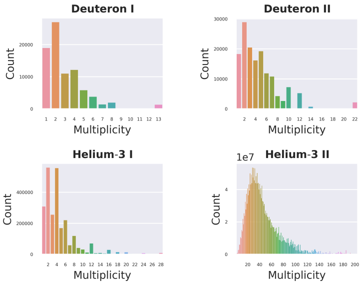


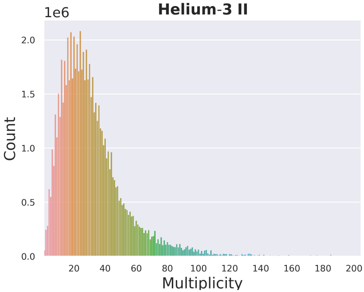
- Hadron Blocks give a clear performance improvement even for light nuclei
- Hadron block cost dominates for the Deuteron, but remains constant* compared to the exponentially growing correlator cost
- Benchmark Details: Measuring wall-clock time in milliseconds on a single core of an Intel Xeon Scalable Cascade Lake processor; Lattice Volume 64^3; Linked to Chroma for Propagator computation; Helium-4 Operator as in [Yamazaki (2010)]
  *Hadron Block cost differs between relativistic (e.g. Deuteron II) and non-relativistic (e.g. Helium-4) forms.

1. Introduction
2. Factor Trees

Multiplicity Histograms

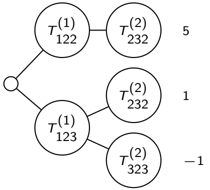
- Take correlators $\epsilon_{abc} \dots \Gamma_{\alpha\beta} \dots f_\alpha^{P/N}(\dots) \dots$: sum over all internal indices $a, b, c, \dots \alpha, \beta, \dots$, leaving strings of $f_\alpha^{P/N}$-like terms
- Sort, group, and assign multiplicities to identically equal terms
- The vast majority of terms have multiplicity > 1, with some terms having multiplicity $\gg 1$
- Exploit this property by computing each degenerate term once and multiply by the multiplicity-adjusted coefficient





Factor Trees

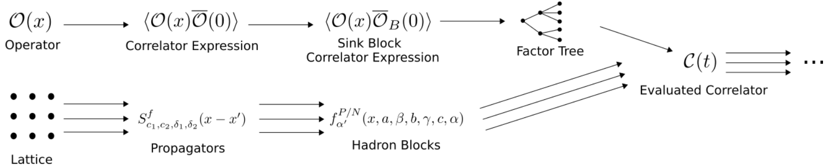
- Storing the full set of terms carries an infeasible memory footprint
- We have the property that the set of factors (e.g. $\{f_\alpha^{P/N}\}$) is small compared to the set of terms, so we can expect a high degree of factorisation



Abstract Factor Tree

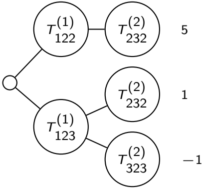
$$\longleftrightarrow \quad E = 5\, T^{(1)}_{122} T^{(2)}_{232} + T^{(1)}_{123} T^{(2)}_{232} - T^{(1)}_{123} T^{(2)}_{323}$$

$$= 5\, T^{(1)}_{122} T^{(2)}_{232} + T^{(1)}_{123} \left(T^{(2)}_{232} - T^{(2)}_{323} \right)$$



Abstract Factor Trees → Linearised Factor Trees

- Abstract Factor Trees have each node pointing to it's children nodes. This is highly inefficient:
    - For a *saturated expression* (term set dominates the factor set), the number of children for each node is $\mathcal{O}(\text{num factors}) \Rightarrow$ the dominant storage cost is to store the links.
    - Cache performance is terrible
- Linearised Factor Trees don't store links: only the factors and the number of children



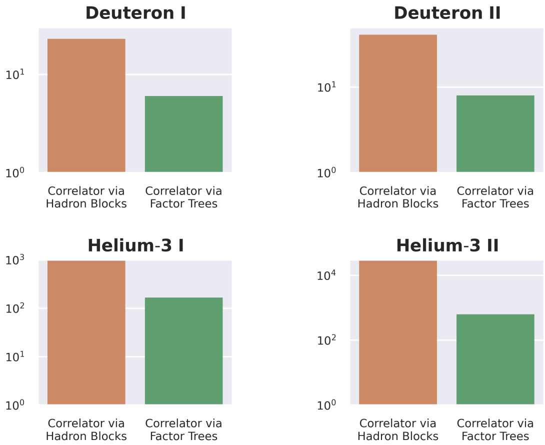
Abstract Factor Tree

$\longleftrightarrow$

$$\begin{aligned} \textit{factors} &= \left[\circ,\ T^{(1)}_{122},\ T^{(2)}_{232},\ T^{(1)}_{123},\ T^{(2)}_{232},\ T^{(2)}_{323}\right] \\ \textit{children} &= [2,\ 1,\ 0,\ 2,\ 0,\ 0] \\ \textit{coefficients} &= [5,\ 1,\ -1] \end{aligned}$$

Linearised Factor Tree

Correlator Evaluation Benchmarks



Benchmark Details: Measuring wall-clock time in milliseconds on a single core of an Intel Xeon Scalable Cascade Lake processor; Lattice Volume 64^3; Linked to Chroma for Propagator computation

Outlook

- Hadron blocks have excellent performance compared to a naive implementation
- Factor trees present promising performance improvements for correlators of light nuclei
- The hadron block cost dominates the correlator cost for light nuclei, but beyond about Helium-4 the correlator cost becomes the relevant cost to optimise
- There are memory constraints in scaling factor trees beyond light nuclei, motivating a hybrid factor tree / tensor e-graph approach for larger systems

3 Backup

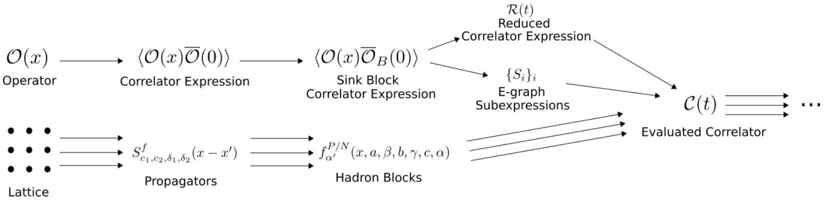
Tensor E-graphs

E-graphs (Equivalence Graphs) [Willsey, et al. (2021)]: Perform all re-writes simultaneously and extract the optimal re-write path as the argmin of some cost function.

Re-writes

Given a tensor expression, split into two pieces, canonicalise, and then collect common subexperssions.

$$\begin{aligned}(C\gamma_5)_{\alpha\beta}(C\gamma_5)_{\gamma\delta}T_{\alpha\beta}T_{\gamma\delta} &\to (C\gamma_5)_{\alpha\beta}T_{\alpha\beta}\times(C\gamma_5)_{\gamma\delta}T_{\gamma\delta}\\ &\to (C\gamma_5)_{\alpha\beta}T_{\alpha\beta}\times(C\gamma_5)_{\alpha\beta}T_{\alpha\beta}\\ &\to B^2 \qquad \text{where } B=(C\gamma_5)_{\alpha\beta}T_{\alpha\beta}\end{aligned}$$



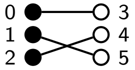
Vertex-Labelled Graph Canonicalisation

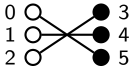
Definition: *Graph Canonicalisation Map*

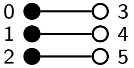
Given two graphs,

$$G_1, G_2 \in \boldsymbol{G}(V) := \{\text{labelled (simple) graphs with vertex set } V \text{ with } |V| =: n\}$$

And given two *ordered vertex colourings* $\pi_1, \pi_2 \in \mathbf{\Pi}(V) := \{(V_1, ..., V_r) \mid \dot{\bigcup}_j V_j = V\}$ related by $\pi_2 = \pi_1^{\gamma}$ for some $\gamma \in S_n$, then a *Graph Canonicalisation Map* $C : \boldsymbol{G}(V) \times \mathbf{\Pi}(V) \to \boldsymbol{G}(V)$ satisfies $C(G_1, \pi_1) = C(G_2, \pi_2)$ if and only if $\exists \delta \in S_n$ such that $G_2 = G_1^{\delta}$ and $\pi_2 = \pi_1^{\delta}$.

$G_1 =$  , $\pi_1 = (\{0, 1, 2\}, \{3, 4, 5\})$

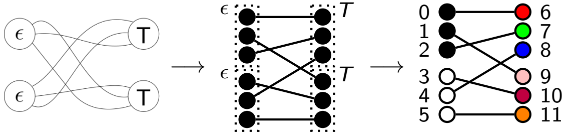
$G_2 =$  , $\pi_2 = (\{3, 4, 5\}, \{0, 1, 2\})$

$C(G_1, \pi_1) =$  $= C(G_2, \pi_2)$,

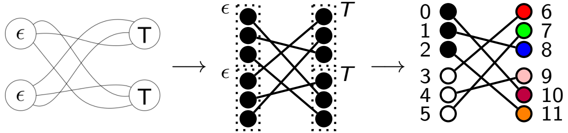
$$\gamma = (0\ 3)(1\ 4)(2\ 5) \in S_6$$
$$\delta = (0\ 4)(1\ 3)(2\ 5) \in S_6$$

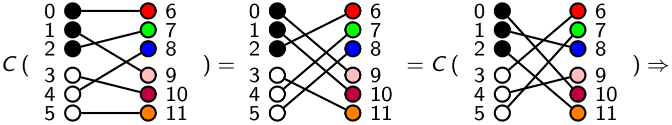
Tensor Network Formulation

$\epsilon_{ijk}\epsilon_{lmn}T_{ikm}T_{jln} \longrightarrow$



$\epsilon_{ijk}\epsilon_{lmn}T_{lnj}T_{mik} \longrightarrow$



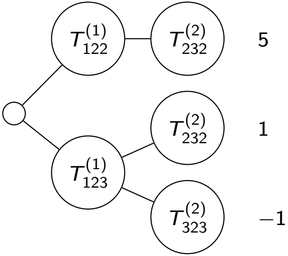


$$\epsilon_{ijk}\epsilon_{lmn}T_{ikm}T_{jln} = \epsilon_{ijk}\epsilon_{lmn}T_{lnj}T_{mik}$$

Abstract Factor Trees

Definition: *Abstract Factor Tree*

Given a tensor expression sum $E = \sum_a c_a E_a$, a corresponding *Abstract Factor Tree* τ of depth m is a rooted tree on the vertex set of tensor elements together with a non-zero real number for each leaf such that each root-to-leaf path corresponds to a set of terms in the tensor expression sum. Moreover, the *sum* of all root-to-leaf paths is equal to the original tensor expression sum.

$\tau =$ 

$$\longleftrightarrow \quad E = 5\ T^{(1)}_{122} T^{(2)}_{232} + T^{(1)}_{123} \left(T^{(2)}_{232} - T^{(2)}_{323} \right)$$

Linearised Tree Evaluation

Algorithm 1: Linearised Factor Tree Evaluation

```
Input: tree{ factors, children, coeffs }
Output: sum := 0
Data: values
index := [ ]
position := [ tree→factors[0] ]
cumulativeProd := [ values[ position[0] ] ]
leafIdx := 0
while True do
    nextPos := position[-1] + 1
    if tree→children[ position[-1] ] == 0 then
        sum += tree→coeffs[leafIdx] × cumulativeProd[-1]
        leafIdx += 1
        success := false
        while ! index→empty() do
            index[-1] += 1
            if index[-1] < tree→children[ position[-2] ] then
                success = true
                position[-1] = nextPos
                cumulativeProd[-1] = cumulativeProd[-2] × values[ tree→factors[nextPos] ]
                break
            else
                position = position[:-1]
                index = index[:-1]
                cumulativeProd = cumulativeProd[:-1]
                termValues = termValues[:-1]
        if ! success then
            done
    else
        position.append( nextPos )
        index.append( 0 )
        cumulativeProd.append( cumulativeProd[-1] × values[ tree→factors[nextPos] ] )
```