Automating Many-Body QM

Ed Valeev
Department of Chemistry
Virginia Tech

ESNT Workshop “Automated Tools for Many-Body Theory”
June 8, 2023
Saclay, France

Talk Synopsis

need to raise the level of abstraction to enable many-body QM

symbolic techniques are a component of the needed many-body QM technology stack

particularly needed for supporting tensor compressed/factorized methods (e.g., PNO)

ideas are old, let's learn from the pioneers and make this sustainable

Outline

- Motivation: Richness of Tensor Algebras in Quantum Mechanics
- Technology Roadmap for Many-Body QM
- SeQuant
  - Overview
  - Key innovations
- Ongoing/Future work

Tensors

Tensors in Quantum Mechanics

Tensor structures arise naturally, and with great(est) variety

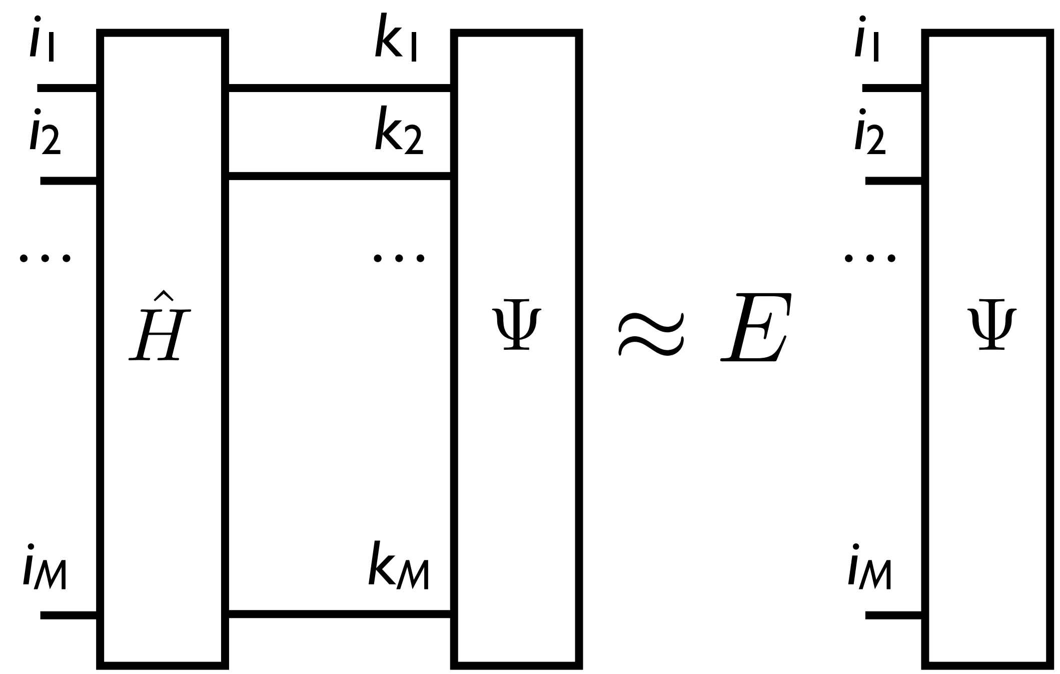
key objects are *fields* i.e. functions of space(time) coordinates	➧	tensor *meshes* and other PDE technologies
n-particle QM is *polylinear* state = tensor of order O(n)	➧	*high*-order tensors
states are *data-sparse* at least all(?) states we care about	➧	tensor *networks* of *many* kinds built out of *block/rank-sparse* tensors

To make sense of this and understand the relevant problem scales let's start with a few pictures

QM States and Their Properties/Changes Are Tensors

N-Body Schrödinger Equation = Tensor Eigenvalue Problem

$$\hat{H}\Psi = E\Psi \quad \Longrightarrow$$



Important simplifications

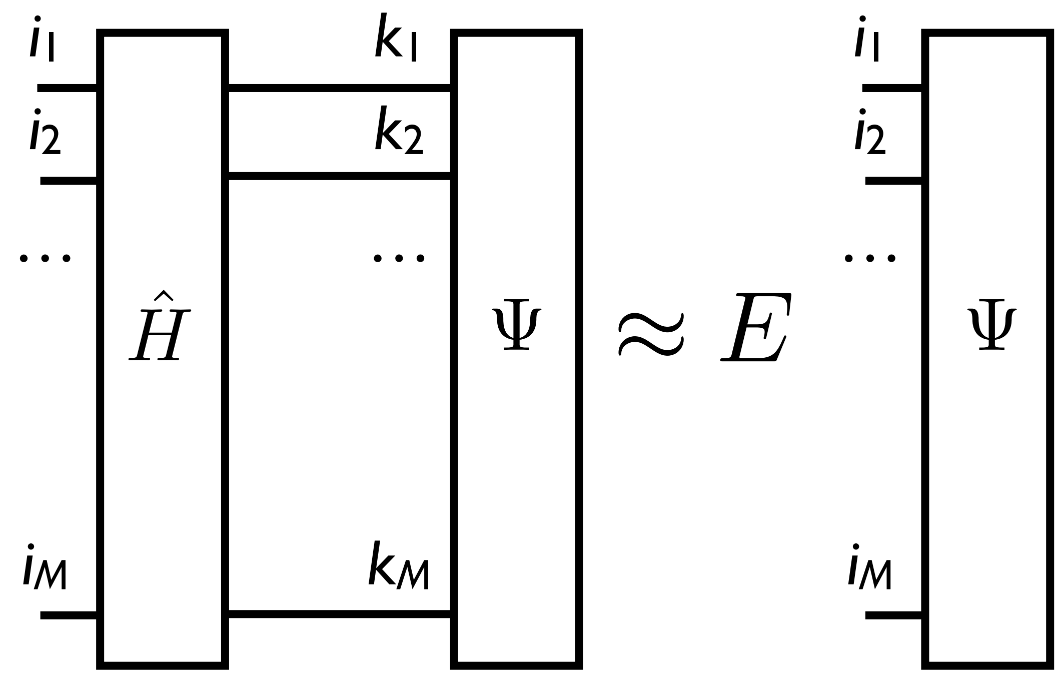
1. *H* is very sparse: up to 2 output indices can differ from the input indices for 2-body *H*

2. For most important states **Ψ** is also sparse in good basis

Properties of Quantum States Are Also Tensors

N-Body Schrödinger Equation = Tensor Eigenvalue Problem

$$\hat{H}\Psi = E\Psi \quad \Longrightarrow$$



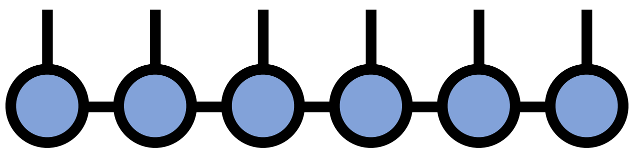
Example: N_2 molecule

N=14 M=60 (cc-pVTZ basis)

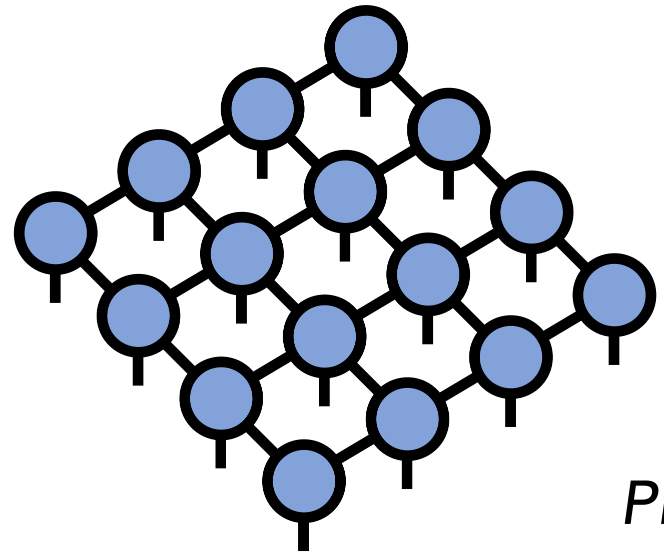
size(**Ψ**) = 1.5 x 10^{17} but only <10^9 elements are significant!

element sparsity can be useful, but essential to exploit general data sparsity in H and **Ψ**

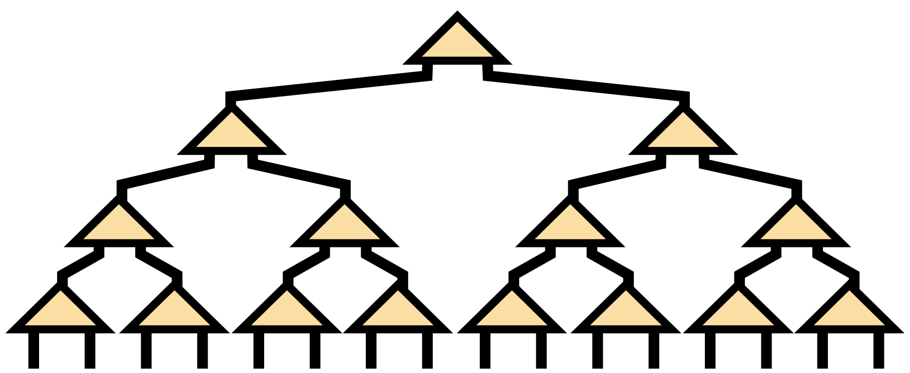
Reducing Complexity v1: Tensor Networks



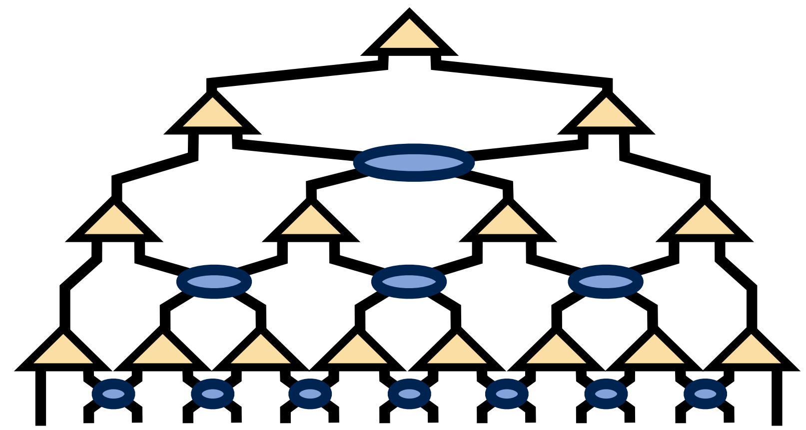
matrix product state (MPS), or tensor-train



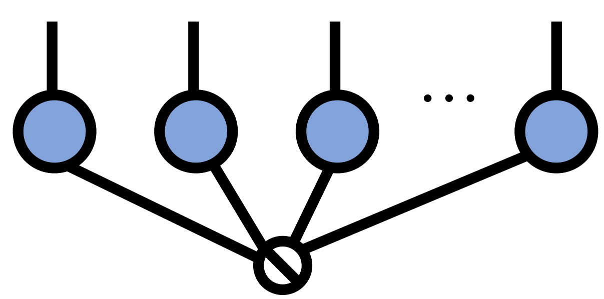
PEPS



tree tensor network (TTN)



MERA



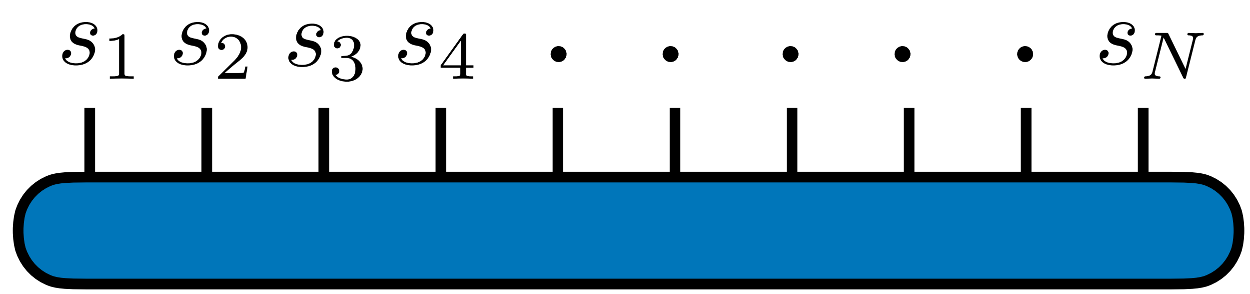
CP

physics, sometimes chemistry

Reducing Complexity v2: Cumulant/Perturbative Expansion

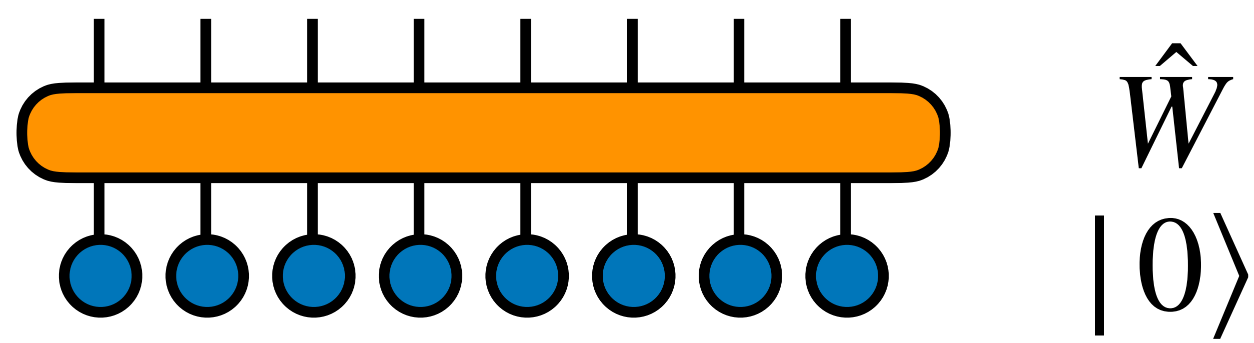
instead of encoding joint probability amplitudes

$$|\Psi\rangle \equiv$$



encode differences in probability amplitudes relative to simple (usually, uncorrelated) state

$$|\Psi\rangle \approx \hat{W}|0\rangle$$



efficient if *W* limited to a sum of few-body terms (e.g, 2-body in CCSD)

tensor ≈ sum of tensor networks

chemistry and physics

Cumulant/Perturbative Expansion: Coupled-Cluster

$$|\Psi\rangle = \exp(\hat{T}_2 + \hat{T}_3 + \ldots + \hat{T}_N) \;\times\; \widetilde{|0\rangle}$$

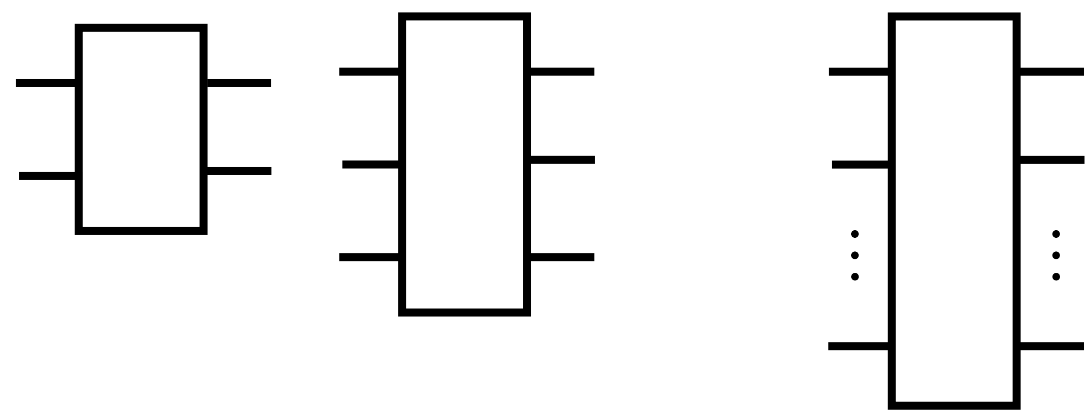
N correlated particles

2-body correlator

3-body correlator

N-body correlator

N independent particles



$\mathcal{O}(N^4)$ parameters

$\mathcal{O}(N^6)$ parameters

$\mathcal{O}(N^{2N})$ parameters

Roadmap for Automation

What Is Automation?

What Is Automation?

International Society of Automation

Standards Certification Training

About ISA / What is Automation?

What is Automation?

The dictionary defines *automation* as "the technique of making an apparatus, a process, or a system operate automatically."

We define automation as "the creation and application of technology to monitor and control the production and delivery of products and services."

isa.org

What Is Automation?

International Society of Automation

Standards Certification Training

About ISA / What is Automation?

What is Automation?

The dictionary defines *automation* as "the technique of making an apparatus, a process, or a system operate automatically."

We define automation as "the creation and application of technology to monitor and control the production and delivery of products and services."

isa.org

What Is Automation?

ISA International Society of Automation

Standards Certification Training

About ISA / What is Automation?

What is Automation?

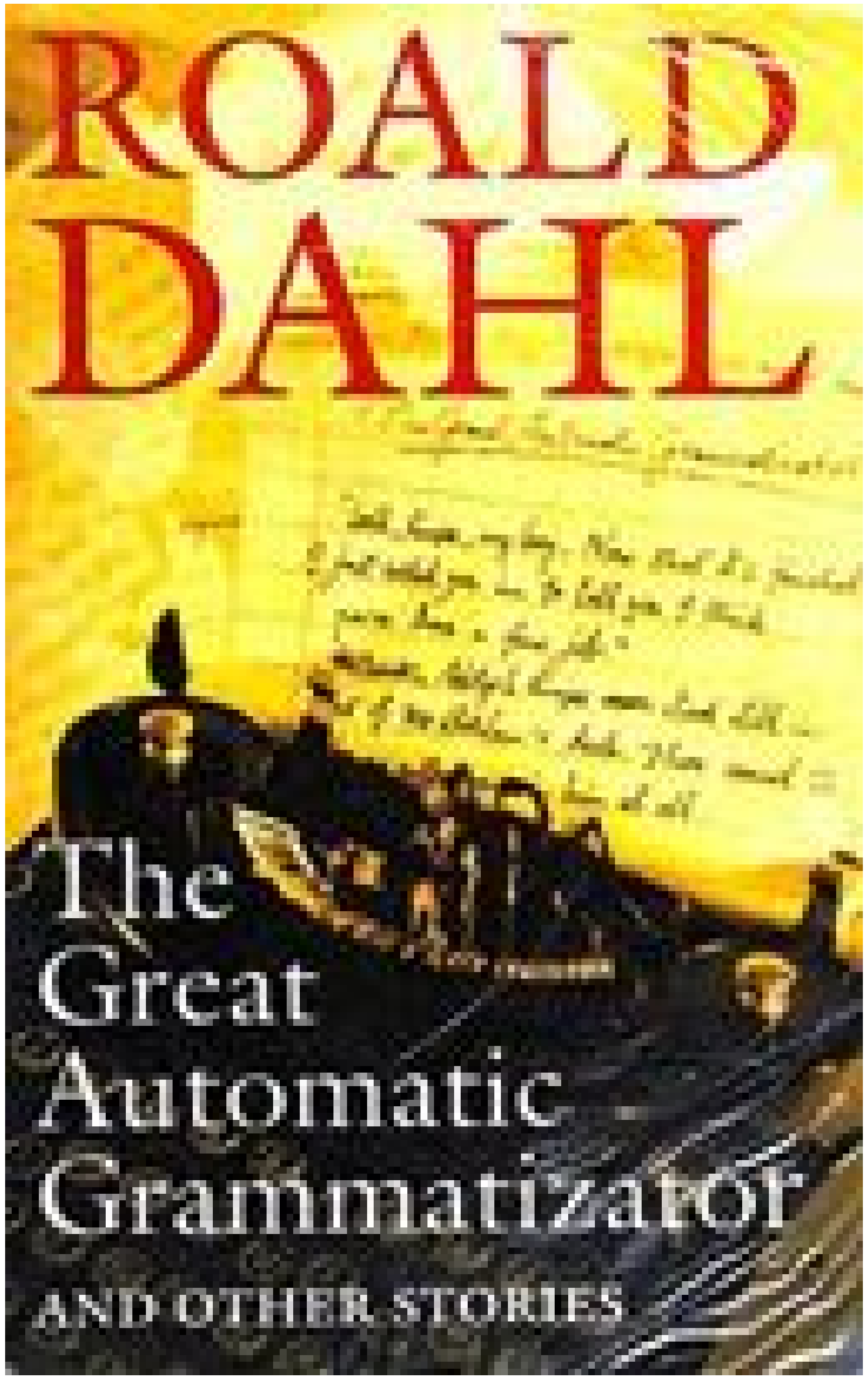
The dictionary defines *automation* as "the technique of making an apparatus, a process, or a system operate automatically."

We define automation as "the creation and application of technology to monitor and control the production and delivery of products and services."

funding agency: product is manuscript

isa.org

Clearly, not a new idea …



wicked funny

oddly relevant

What Is Automation?

- Here: replacing tedious human work by machine work
  - Formal manipulation
  - Algorithm design/optimization
  - Code transformation
  - Code generation
  - Performance Analysis/Optimization
  - Graphics/table generation …
- Not everything can or should be automated
- Automation needs to be controlled, hence understandable
- Automation needs to be high-quality

Purposes of Automation

- Correctness for new (and old) methods
- Future-proofing
- Optimization: execution speed, resource use

automation = use of technology to *enable* many-body QM simulation

Wishlist for Automation

automation = use of technology to *enable* many-body QM simulation

- High level of abstraction: algebraic or graphical
- Ability to lower level of abstraction gradually (operator algebra/TN -> tensor algebra -> tensor data structures + algorithms -> generic IR
- Non-monolithic
- Deployable to large machines
- Open source

Again, This Has Been Known and Realized

Theor Chim Acta (1991) 79: 1–42

Theoretica
Chimica Acta

The automated solution of second quantization equations with applications to the coupled cluster approach*

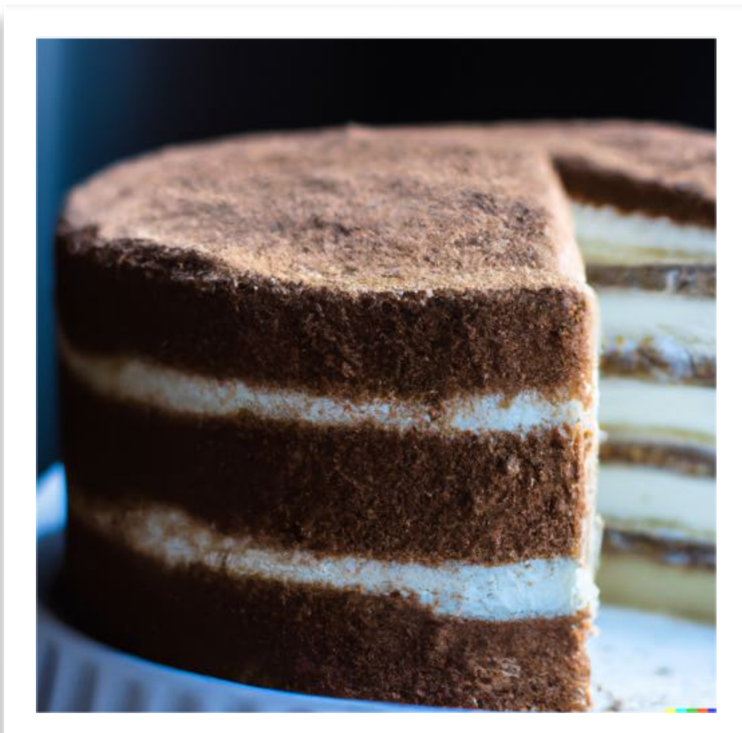
Curtis L. Janssen and Henry F. Schaefer III

Center for Computational Quantum Chemistry, University of Georgia, Athens, GA 30602, USA

Received August 14, 1990; received in revised form/Accepted September 26, 1990

Technology
~~Automation~~ Vision

many-body physics runtime stack



solvers

operator algebra / TN “compiler” as embedded DSL / library

tensor algebra engine

QM operator evaluation

Technology ~~Automation~~ Vision

many-body physics runtime stack

solvers

operator algebra / TN "compiler" as embedded DSL / library

ValeevGroup/SeQuant

tensor algebra engine

ValeevGroup/TiledArray

QM operator evaluation

ValeevGroup/Libint{,X} **m-a-d-n-e-s-s/madness**

Technology ~~Automation~~ Vision

many-body physics runtime stack

solvers — ValeevGroup/mpqc

operator algebra / TN "compiler" as embedded DSL / library

ValeevGroup/SeQuant

tensor algebra engine

ValeevGroup/TiledArray

QM operator evaluation

ValeevGroup/Libint{,X} m-a-d-n-e-s-s/madness

Reusable Layers Highlight 1: Fast Hybrid All-Electron Kohn-Sham

arXiv > physics > arXiv:2303.14280

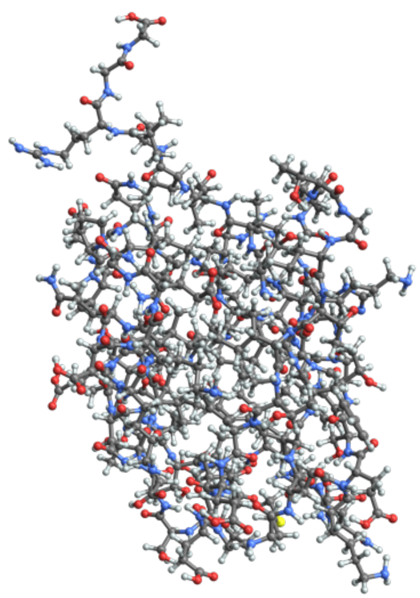
Help | Advanced

Physics > Computational Physics

[Submitted on 24 Mar 2023]

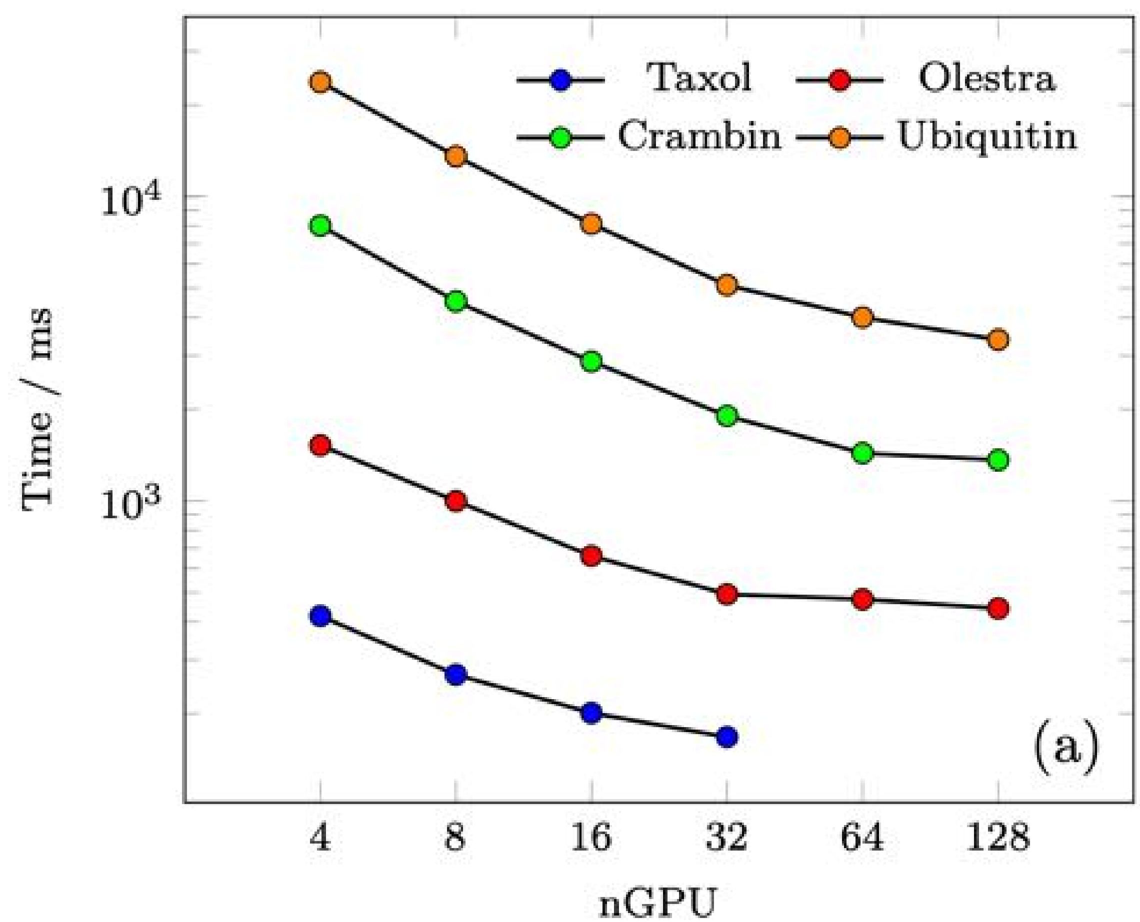
Distributed Memory, GPU Accelerated Fock Construction for Hybrid, Gaussian Basis Density Functional Theory

David B. Williams-Young, Andrey Asadchev, Doru Thom Popovici, David Clark, Johnathan Waldrop, Theresa Windus, Edward F. Valeev, Wibe A. de Jong



ubiquitin (1k+ atoms, 10k+ bf)

GPU-accelerated hybrid Fock build



Reusable Layers Highlight 1: Fast Hybrid All-Electron Kohn-Sham

arXiv > physics > arXiv:2303.14280

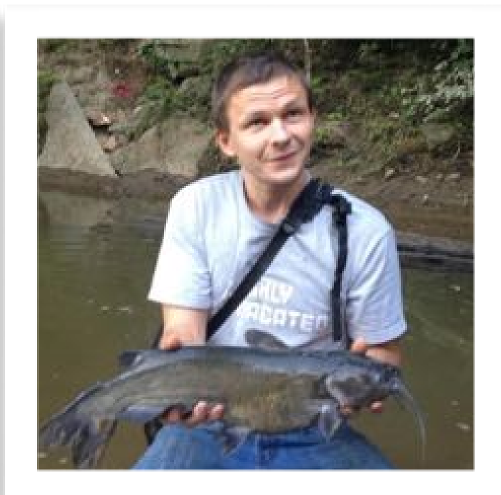
Help | Advanced

Physics > Computational Physics

[Submitted on 24 Mar 2023]

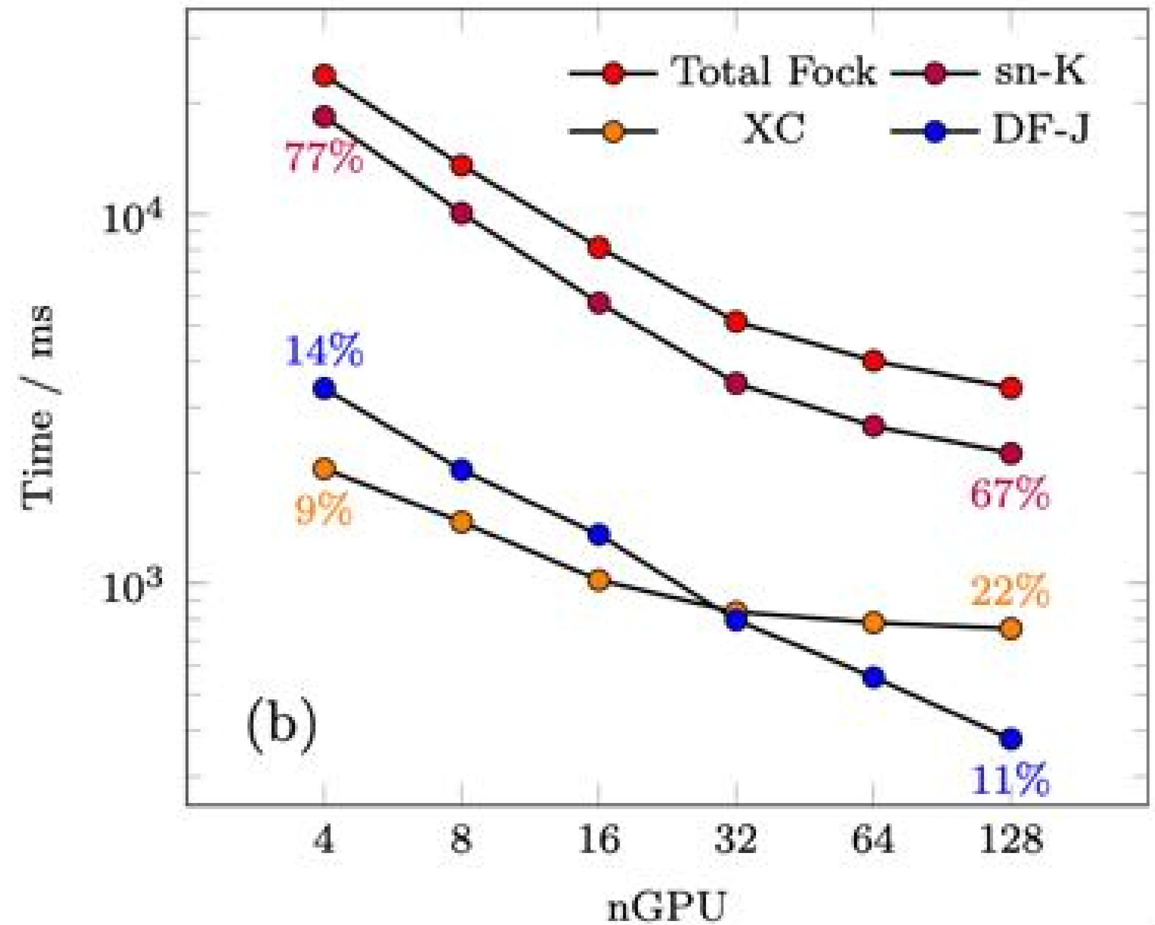
Distributed Memory, GPU Accelerated Fock Construction for Hybrid, Gaussian Basis Density Functional Theory

David B. Williams-Young, Andrey Asadchev, Doru Thom Popovici, Davi



Dr. Andrey Asadchev

ValeevGroup/LibintX



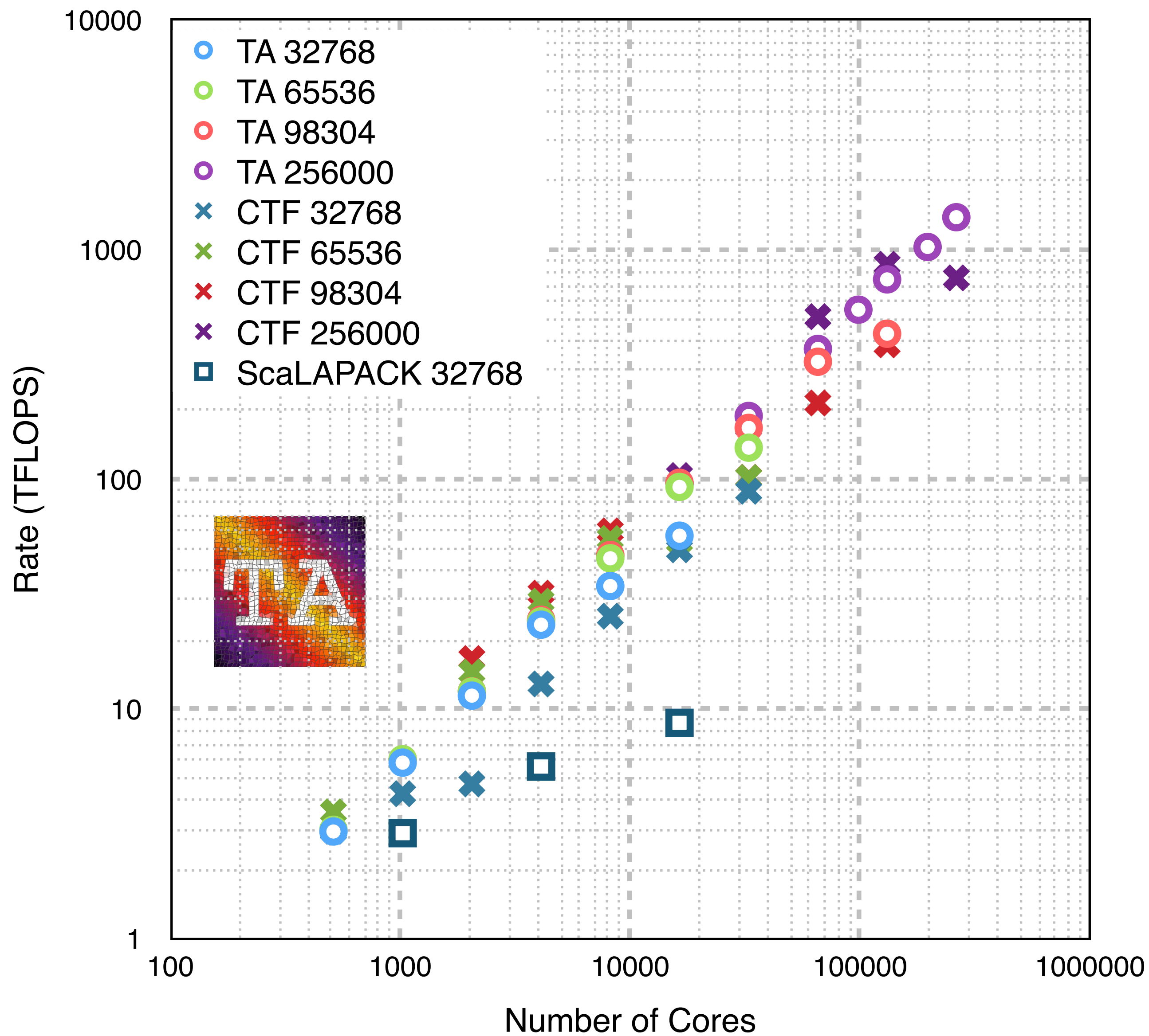
N.B. Open-source parallel GPU-accelerated J-engine … soon fast high-L 4-center AO integrals

Reusable Layers Highlight 2: TiledArray Framework

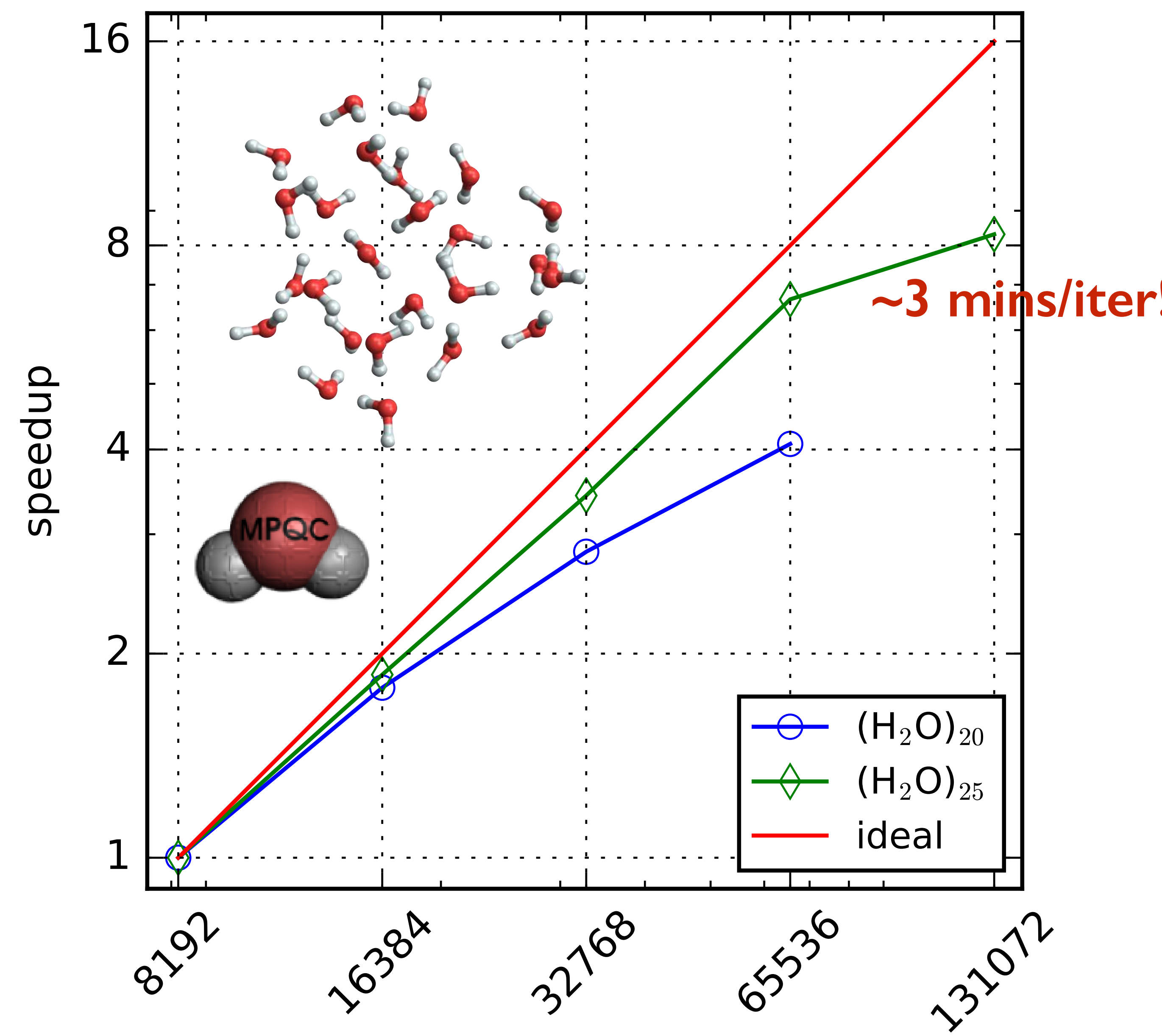
- **covers many domains**: dense and block-sparse arrays/tensors
- **general purpose**: no domain concepts, applicable to chemistry/physics/engineering
- **for users**: high-level post-einsum DSL
- **for developers**: powerful STL-like abstractions
- **scalable**: intra- and inter-node, >10^5 cores
- h**igh-performance**: CUDA, other backends in progress
- **free and open source**: GPL
- **open development**: central repo on Github

ValeevGroup/TiledArray

Our Tensor Framework TiledArray Allows Us To Simulate Electrons on Largest Machines



dense matrix multiplication benchmark



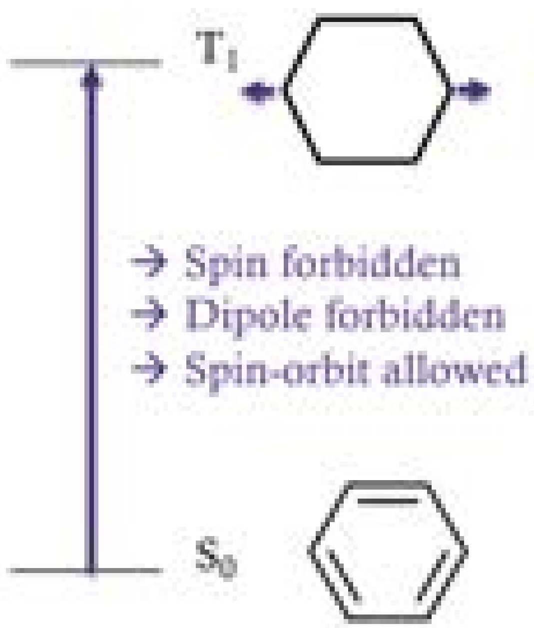
production CCSD-F12 solver
IBM BG/Q "Mira"

U.S. DEPARTMENT OF ENERGY INCITE LEADERSHIP COMPUTING

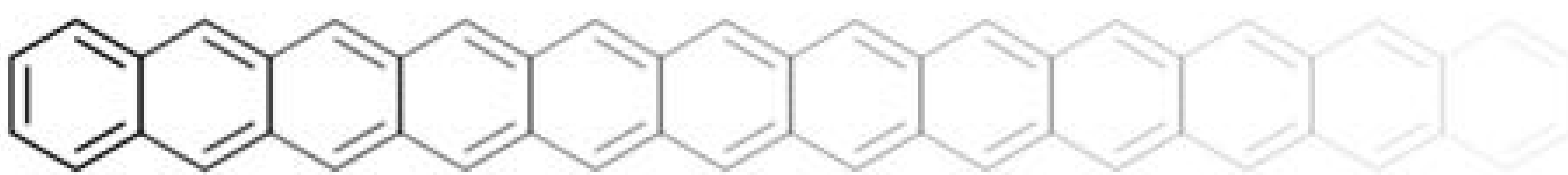
Reusable Layers Highlight 2: TiledArray Framework

others are starting to use it too, e.g.

Phosphorescent lifetimes of the $S_0 \rightarrow T_1$ transition



	Benzene	Naphthalene	Anthracene
Predicted Energy	3.25 eV	2.26 eV	1.53 eV
Exp. Energy	3.66 eV	2.63 eV	1.83 eV
Oscillator Strength	3.6e-10	2.89e-10	2.07e-10
Predicted Lifetime	6 s	15 s	48 s
Exp. Lifetime	10 s	18 s	60 s



relativistic ground and excited state massively-parallel CCSD in ChronusQ

SciDAC collaboration with Eugene DePrince (FSU), Xiaosong Li (UW) and Chao Yang (LBL)

TiledArray Arithmetic: DSL

Math

$$R_{iajb} = G_{iajb} + F_{ac}T_{icjb} + F_{bc}T_{iajc} - F_{ik}T_{kajb} - F_{jk}T_{iakb}$$
$$E = (G_{iajb} + R_{iajb})(2T_{iajb} - T_{ibja})$$

C++

```
TArrayD R(world, ovov);

R("i,a,j,b") = G("i,a,j,b") + Fv("a,c") * T("i,c,j,b") +
                Fv("b,c") * T("i,a,j,c") - Fo("i,k") * T("k,a,j,b") -
                Fo("j,k") * T("i,a,k,b");

double energy =
     (G("i,a,j,b") + R("i,a,j,b")).dot(2 * T("i,a,j,b") - T("i,b,j,a"));
```

TiledArray Arithmetic: DSL

Math

$$
\begin{aligned}
R_{iajb} =& G_{iajb} + F_{ac} T_{icjb} + F_{bc} T_{iajc} - F_{ik} T_{kajb} - F_{jk} T_{iakb} \\
E =& (G_{iajb} + R_{iajb})(2T_{iajb} - T_{ibja})
\end{aligned}
$$

C++

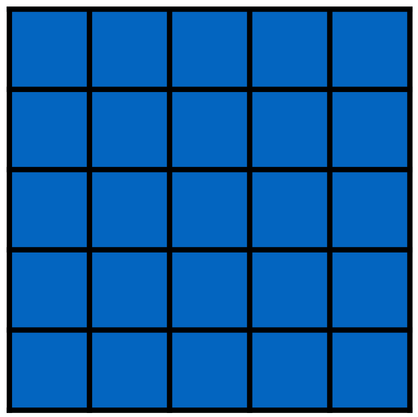
```
TArrayD R(world, ovov);

R("i,a,j,b") = G("i,a,j,b") + Fv("a,c") * T("i,c,j,b") +
                Fv("b,c") * T("i,a,j,c") - Fo("i,k") * T("k,a,j,b") -
                Fo("j,k") * T("i,a,k,b");

double energy =
     (G("i,a,j,b") + R("i,a,j,b")).dot(2 * T("i,a,j,b") - T("i,b,j,a"));
```

***deeply* customizable:**

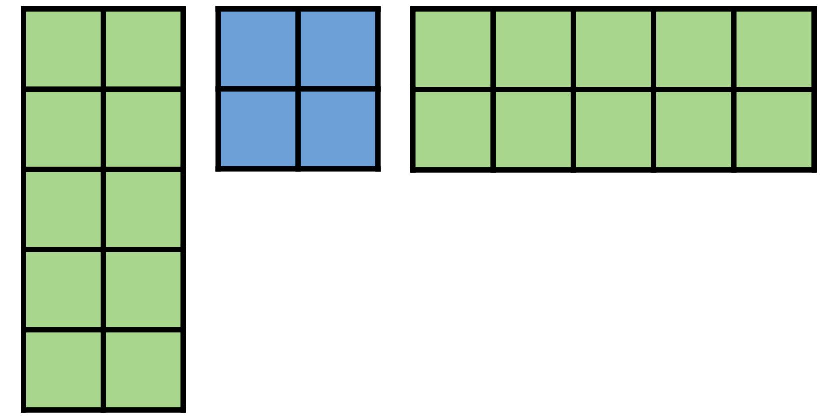
```
TArrayD = DistArray<Tensor<double>, DensePolicy>
```

can do lazily-evaluated tiles, tensors of tensors, sparse tensors, etc.

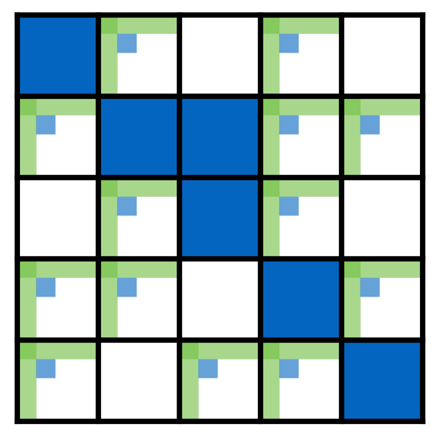
TiledArray Supports General Data Sparsity



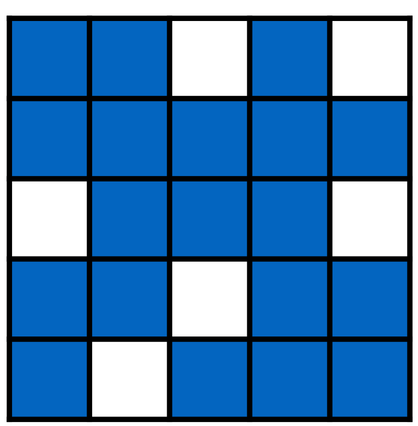
dense



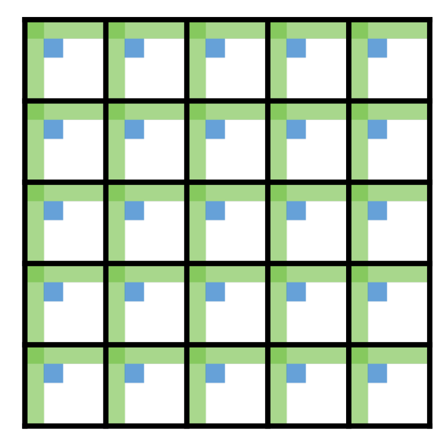
rank sparse



Clustered Low Rank



element/block sparse



block-rank sparse

Lewis, Calvin, Valeev, *J. Chem. Theor. Comp.* **12**, 5868 (2016).

SeQuant

SeQuant Synopsis

fast symbolic algebra of tensors

interpreter using parallel data-sparse tensor engine TiledArray

reusable open-source C++ library

old ideas in modern form, with some twists

SeQuant: **Se**cond **Quant**ization Algebra System

- SeQuant v1 (2002-now)
  - Implemented in Mathematica
  - Original objective to support R12 methods development (incl. CC-R12)
  - Extended by Martin Torheyden and Chong Peng (2007-2010) to support extended WT w.r.t. multi-determinant vacuum
  - Sufficient for CCSD, MR-F12, etc.
  - Slow, imperfect expression reduction, no factorization, etc.
  - Publicly available at github.com/ValeevGroup/SeQuant
- SeQuant2 (2018-now)
  - Implemented in C++17
  - Online symbolic manipulation now possible
  - Can interpret expressions using external tensor backend
  - github.com/ValeevGroup/SeQuant2

SeQuant2: Getting Started

- TL;DR

```
$ git clone https://github.com/ValeevGroup/SeQuant2
$ cmake -S SeQuant2 -B SeQuant2/build -DCMAKE_PREFIX_PATH="path-to-boost;path-to-eigen3"
$ cmake --build SeQuant2/build --target check
```

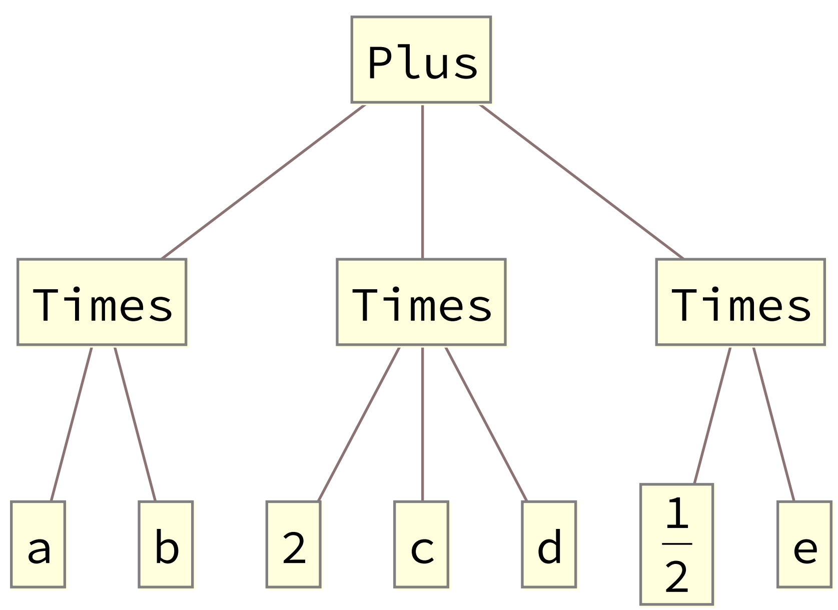
- Prereqs:
  - Boost 1.67+ (can't use 1.70, 1.77, 1.78)
  - Range-v3 (can autobuild)
  - (Unless BOOST_TESTING=OFF): Eigen3

SeQuant2: Core

- Core of SeQuant2 provides basic support for representing and manipulating *expressions*
  - SeQuant1 leveraged Mathematica for this job (hence its choice)
- Expressions are traditionally represented as trees/graphs

In[10]:= `TreeForm[a*b+c*2*d+(1/2)*e]`

Out[10]//TreeForm=



- But actual representation is some recursive data structure

In[14]:= `FullForm[a*b+c*2*d+(1/2)*e]`

Out[14]//FullForm=

```
Plus[Times[a, b], Times[2, c, d], Times[Rational[1, 2], e]]
```

SeQuant2: Core

- Expr = node on an expression tree
- Every type of expression (Constant, Product, Sum, etc.) must derive from Expr
  - Since expressions are polymorphic they should be stored on heap and held via a shared_ptr; use shortcuts ExprPtr ≡ shared_ptr<Expr> and ex<Type>(…) ≡ static_pointer_cast<Expr>(make_shared<Type>(…))
- Expr is a polymorphic *range* of pointers (ExprPtr) to subexpressions

```
auto prod = ex<Constant>(1) * ex<Constant>(2);
for(auto& factor: *(prod)) {
  std::wcout << "factor = " << to_wolfram(factor) << std::endl;
}
```

```
factor = 1.000000
factor = 2.000000
```

- Iteration over the tree trivially implemented by iterating over subexprs

SeQuant2: Core

- The `Expr` range is *mutable* for many expressions, this makes it possible to transform expressions

```
auto prod = ex<Constant>(1) * ex<Constant>(2);
std::wcout << "Old prod = " << to_wolfram(prod) << std::endl;
for(auto& factor: *(prod)) {
  factor = ex<Constant>(factor->as<Constant>().value() * 2.);
}
std::wcout << "New prod = " << to_wolfram(prod) << std::endl;
```

⇒

```
Old prod = Times[1.000000,2.000000]
New prod = Times[2.000000,4.000000]
```

- To make it easier dealing with polymorphic ranges, type ID of an `Expr` can be examined at runtime via `Expr::is<Type>` and it can be cast to `Type` via `Expr::as<Type>`(this does not use RTTI! See `Expr::get_type_id()`)

```
auto x = ex<Constant>(1) * (ex<Constant>(2) + ex<Constant>(3));
for(auto& factor: *(x)) {
  std::wcout << "factor is constant? “
             << (factor->is<Constant>() ? “true” : “false”) << std::endl;
}
```

⇒

```
factor is Constant? true
factor is Constant? false
```

SeQuant2: Core

- Most operations on expressions involve traversing the tree and invoking a function on (*visiting*) every node (optionally only invoking it on leaves only).

```
auto x = ex<Constant>(1) * (ex<Constant>(2) + ex<Constant>(3));
x->visit([](const ExprPtr& ex){
  std::wcout << to_wolfram(ex)
             << (ex->is<Constant>() ? " is" : " is not”)
             << " a Constant" << std::endl;
});
```

```
1.000000 is a Constant
2.000000 is a Constant
3.000000 is a Constant
Plus[2.000000,3.000000] is not a Constant
Times[1.000000,Plus[2.000000,3.000000]] is not a Constant
```

- e.g. `to_wolfram(ExprPtr)` ~~can be~~ is implemented by a visitor

SeQuant2: Core

- Most operations on expressions involve traversing the tree and invoking a function on (*visiting*) every node (optionally only invoking it on leaves only).
- Visitor can change its argument arbitrarily! This is another way expressions can be transformed.

```
auto x = ex<Constant>(1) * (ex<Constant>(2) + ex<Constant>(3));
std::wcout << “Old expr = " << to_wolfram(x) << std::endl;
x->visit([](ExprPtr& subx){
  if (subx->is<Constant>())
    subx = ex<Constant>(2);
});
std::wcout << “New expr = " << to_wolfram(x) << std::endl;
```

⇩

```
Old expr = Times[1.000000,Plus[2.000000,3.000000]]
New expr = Times[2.000000,Plus[2.000000,2.000000]]
```

SeQuant2: Core

- `Tensor` represents abstract tensorial quantities of finite order
  - Covariant and contravariant modes in Einstein notation are referred to as *bra* and *ket* modes (in the sense of Dirac notation for matrix elements of operators)
  - `Tensor` modes are represented by `Index` objects, composed of a label and an `IndexSpace`
  - `IndexSpace` represents a vector space; it has a type (`Type`) and quantum number attributes (`QuantumNumbers`).

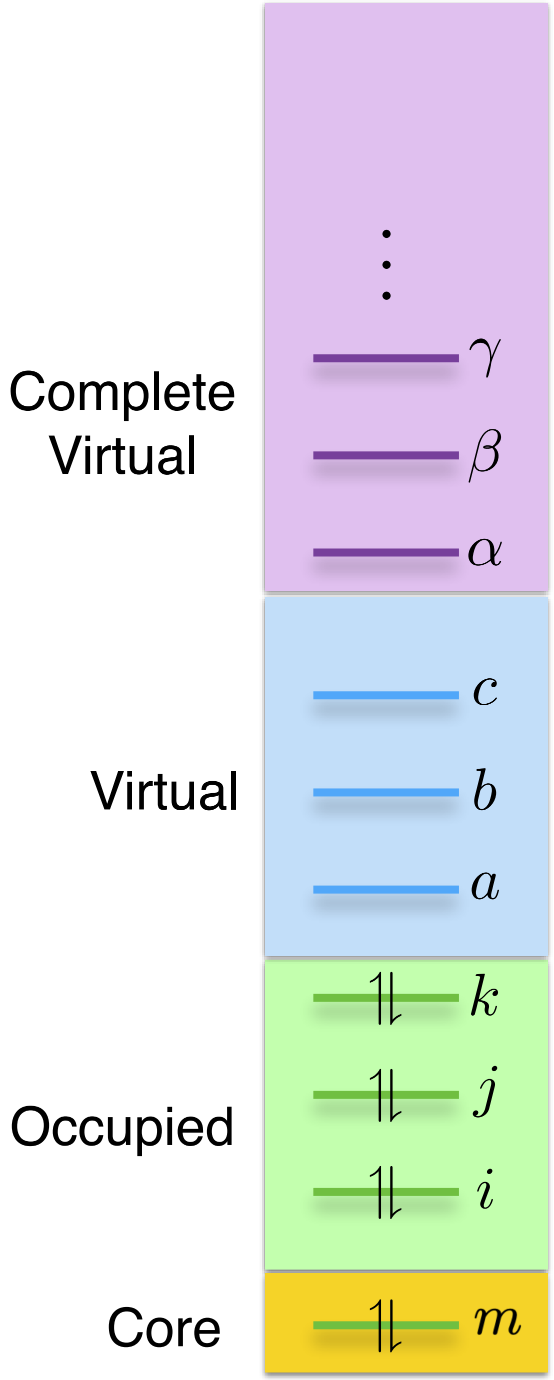
SeQuant DSL

- Index Type

i_1
```
Index i1(L"i_1");
```
a_1
```
Index a1(L"a_1");
```
p_1^α
```
auto p1A = Index(L”p⁺_1", IndexSpace::alpha);
```
p_1^β
```
auto p1B = Index(L"p⁻_1", IndexSpace::beta);
```
$a_3^{i_1 i_2}$
```
Index a3(L”a_3", IndexSpace::active_occupied,
         {i1, i2});
```

- Tensor Type

$F_{i_1}^{i_2}$
```
Tensor(L"F", {L"i_1"}, {L"i_2"});
```
$\bar{g}_{i_1 i_2}^{i_3 i_4}$
```
Tensor(L"g", {Index{L"i_1"}, Index{L"i_2"}},
       {Index{L”i_3"}, Index{L"i_4"}},
       Symmetry::antisymm);
```



SeQuant DSL

- Operators

$$\tilde{a}^{i_1 i_2}_{a_1 a_2}$$

```
auto nop1 = FNOperator({L"i_1", L"i_2"}, {L"a_1", L"a_2"},
                       Vacuum::SingleProduct);
```

$$\tilde{a}^{i_1 i_2}_{\quad a_2}$$

```
auto nop2 = FNOperator({L"i_1", L"i_2"}, {L"a_2"},
                       Vacuum::SingleProduct);
```

- Operator sequence

$$\tilde{a}^{i_1 i_2}_{a_1 a_2} \tilde{a}^{i_1 i_2}_{\quad a_2}$$

```
auto nopseq = FNOperatorSeq({nop1, nop2});
```

- Wick's theorem

$$\tilde{a}^{i_1 i_2}_{a_1 a_2} \tilde{a}^{a_3 a_4}_{i_3 i_4}$$

```
auto wick = FWickTheorem{opseq};
```

$$s^{i_4}_{i_1} s^{i_3}_{i_2} s^{a_3}_{a_2} s^{a_4}_{a_1} - s^{i_4}_{i_1} s^{i_3}_{i_2} s^{a_4}_{a_2} s^{a_3}_{a_1} - s^{i_3}_{i_1} s^{i_4}_{i_2} s^{a_3}_{a_2} s^{a_4}_{a_1} + s^{i_3}_{i_1} s^{i_4}_{i_2} s^{a_4}_{a_2} s^{a_3}_{a_1}$$

SeQuant DSL

- Expression transcription: operator algebra

$$F^{i_2}_{i_1} \tilde{a}^{i_1}_{i_2}$$

```
auto h1 = ex<Tensor>(L"F",{L"i_1"},{L"i_2"}) *
          ex<FNOperator>({L"i_1"},{L"i_2"});
```

- Expression transcription: tensor products

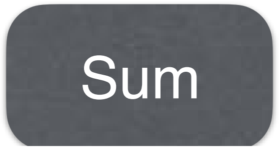
$$\frac{1}{8} \bar{g}^{a_4 a_5}_{i_4 i_5} \bar{t}^{i_1 i_2}_{a_1 a_4} \bar{t}^{i_3 i_4 i_5}_{a_2 a_3 a_5}$$

```
auto input = ex<Constant>(1./8) *
             ex<Tensor>(L"g",{L"i_4", L"i_5"},
                        {L"a_4", L"a_5"}, Symmetry::antisymm) *
             ex<Tensor>(L"t",{L"a_1", L"a_4"},
                        {L"i_1", L"i_2"}, Symmetry::antisymm) *
             ex<Tensor>(L"t",{L"a_2",L"a_3", L"a_5"},
                        {L"i_3", L"i_4",L"i_5"}, Symmetry::antisymm);
```

SeQuant DSL

Expressions are represented as trees

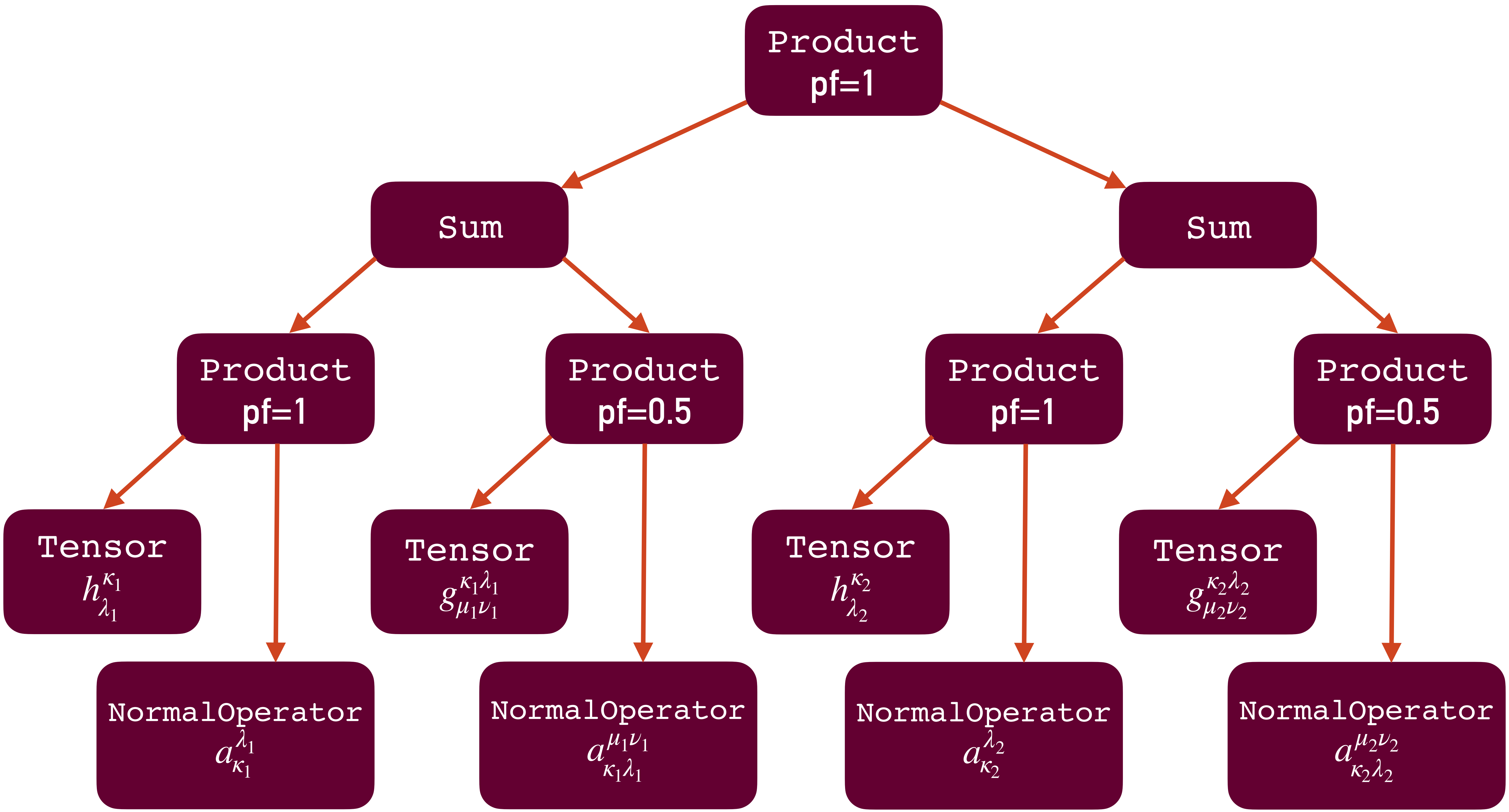
$$\frac{1}{4} A^{a_1 a_2}_{i_1 i_2} \bar{g}^{i_1 i_2}_{a_1 a_2} + \frac{1}{2} A^{a_1 a_2}_{i_1 i_2} f^{i_1}_{i_3} \bar{t}^{i_2 i_3}_{a_1 a_2} + \dots$$



math: $$\hat{H}^2 \equiv \left(h^{\kappa}_{\lambda} a^{\lambda}_{\kappa} + \frac{1}{2} g^{\kappa\lambda}_{\mu\nu} a^{\mu\nu}_{\kappa\lambda} \right)^2 = \left(h^{\kappa_1}_{\lambda_1} a^{\lambda_1}_{\kappa_1} + \frac{1}{2} g^{\kappa_1\lambda_1}_{\mu_1\nu_1} a^{\mu_1\nu_1}_{\kappa_1\lambda_1} \right) \left(h^{\kappa_2}_{\lambda_2} a^{\lambda_2}_{\kappa_2} + \frac{1}{2} g^{\kappa_2\lambda_2}_{\mu_2\nu_2} a^{\mu_2\nu_2}_{\kappa_2\lambda_2} \right)$$

SeQuant:

```
#include <SeQuant/domain/mbpt/sr/sr.hpp>
using namespace sequant::mbpt::sr::so;
auto H2 = H() * H();
```



SeQuant DSL

- Custom expression elements

```
namespace sequant::mbpt {
  template <typename QuantumNumbers, Statistics S>
  class Operator : public Expr {
      // ...
  };

  namespace sr {
    template <Statistics S>
    class Operator : public mbpt::Operator<QuantumNumberSet<2>> {
        // ...
    };
  }
}
```

SeQuant DSL

- compose CC equations at high level

```
using namespace SeQuant::mbpt;

// 1. construct hbar(op) in canonical form
auto hbar = op::H();
auto H_Tk = hbar;
for (int64_t k = 1; k <= 4; ++k) {
  H_Tk = simplify(ex<Constant>(rational{1, k}) * H_Tk * op::T(N));
  hbar += H_Tk;
}

for(auto p : range(0,P)) {
  // 2.a screen by ex level

  // 2.b multiply by A(P)
  auto A_hbar = simplify(op::A(p) * hbar_p);

  // 2.c compute vacuum average
  auto R_p = op::vac_av(A_hbar);
  simplify(R_p);
}
```

SeQuant2: Key Algorithms

- Tensor network automorphizer
- Fast(er) Wick engine
- Spin tracing
- Expression optimization
- Expression evaluation

SeQuant2: Tensor network automorphizer

- Maps tensor network onto (symmetry-preserving) colored graph and determines its automorphism group
- Used to manipulate tensor networks
  - e.g. find topologically-equivalent parts
- Used to manipulate expressions involving TNs
  - e.g. recognizing equivalent terms produced by the Wick engine

Example: CCSD

$$\langle 0 | \hat{L}_2 (\hat{W}\hat{T}_2^2)_c | 0 \rangle = (-\frac{1}{4} \times A^{a_1a_2}_{i_1i_2} g^{a_3a_4}_{i_3i_4} t^{i_1i_2}_{a_1a_3} t^{i_3i_4}_{a_2a_4} +$$

equivalent

$$A^{a_1a_2}_{i_1i_2} g^{a_3a_4}_{i_3i_4} t^{i_1i_3}_{a_1a_3} t^{i_2i_4}_{a_2a_4} +$$

$$-\frac{1}{4} \times A^{a_1a_2}_{i_1i_2} g^{a_3a_4}_{i_3i_4} t^{i_3i_4}_{a_1a_3} t^{i_1i_2}_{a_2a_4} +$$

$$\frac{1}{8} \times A^{a_1a_2}_{i_1i_2} g^{a_3a_4}_{i_3i_4} t^{i_3i_4}_{a_1a_2} t^{i_1i_2}_{a_3a_4} +$$

$$-\frac{1}{2} \times A^{a_1a_2}_{i_1i_2} g^{a_3a_4}_{i_3i_4} t^{i_1i_3}_{a_1a_2} t^{i_2i_4}_{a_3a_4})$$

Example: CCSDT

$$\langle 0 | \hat{L}_3(\hat{W}\hat{T}_2\hat{T}_3)_c | 0 \rangle = (\frac{1}{48} \times A^{a_1a_2a_3}_{i_1i_2i_3} g^{a_4a_5}_{i_4i_5} t^{i_1i_2}_{a_4a_5} t^{i_3i_4i_5}_{a_1a_2a_3} +$$

$$\frac{1}{24} \times A^{a_1a_2a_3}_{i_1i_2i_3} g^{a_4a_5}_{i_4i_5} t^{i_1i_4}_{a_4a_5} t^{i_2i_3i_5}_{a_1a_2a_3} +$$

$$\frac{1}{24} \times A^{a_1a_2a_3}_{i_1i_2i_3} g^{a_4a_5}_{i_4i_5} t^{i_4i_5}_{a_1a_4} t^{i_1i_2i_3}_{a_2a_3a_5} +$$

$$\frac{1}{4} \times A^{a_1a_2a_3}_{i_1i_2i_3} g^{a_4a_5}_{i_4i_5} t^{i_1i_4}_{a_1a_4} t^{i_2i_3i_5}_{a_2a_3a_5} +$$

$$\frac{1}{8} \times A^{a_1a_2a_3}_{i_1i_2i_3} g^{a_4a_5}_{i_4i_5} t^{i_1i_2}_{a_1a_4} t^{i_3i_4i_5}_{a_2a_3a_5} +$$

$$\frac{1}{8} \times A^{a_1a_2a_3}_{i_1i_2i_3} g^{a_4a_5}_{i_4i_5} t^{i_1i_4}_{a_1a_2} t^{i_2i_3i_5}_{a_3a_4a_5} +$$

$$\frac{1}{48} \times A^{a_1a_2a_3}_{i_1i_2i_3} g^{a_4a_5}_{i_4i_5} t^{i_4i_5}_{a_1a_2} t^{i_1i_2i_3}_{a_3a_4a_5})$$

Example: PNO CCSD

$$\langle 0|\hat{L}_2(\hat{W}\hat{T}_2^2)_c|0\rangle = (\frac{1}{8}\times A^{a_1^{i_1i_2}a_2^{i_1i_2}}_{i_1i_2} g^{a_3^{i_1i_2}a_4^{i_1i_2}}_{i_3i_4} t^{i_3i_4}_{a_5^{i_3i_4}a_6^{i_3i_4}} t^{i_1i_2}_{a_3^{i_1i_2}a_4^{i_1i_2}} S^{a_5^{i_3i_4}}_{a_1^{i_1i_2}} S^{a_6^{i_3i_4}}_{a_2^{i_1i_2}} +$$

$$-\frac{1}{2}\times A^{a_1^{i_1i_2}a_2^{i_1i_2}}_{i_1i_2} g^{a_3^{i_1i_3}a_4^{i_1i_3}}_{i_3i_4} t^{i_1i_3}_{a_3^{i_1i_3}a_4^{i_1i_3}} t^{i_2i_4}_{a_5^{i_2i_4}a_6^{i_2i_4}} S^{a_5^{i_2i_4}}_{a_1^{i_1i_2}} S^{a_6^{i_2i_4}}_{a_2^{i_1i_2}} +$$

equivalent

$$\frac{1}{2}\times A^{a_1^{i_1i_2}a_2^{i_1i_2}}_{i_1i_2} g^{a_3^{i_1i_3}a_4^{i_2i_4}}_{i_3i_4} t^{i_1i_3}_{a_3^{i_1i_3}a_5^{i_1i_3}} t^{i_2i_4}_{a_4^{i_2i_4}a_6^{i_2i_4}} S^{a_5^{i_1i_3}}_{a_1^{i_1i_2}} S^{a_6^{i_2i_4}}_{a_2^{i_1i_2}} +$$

$$-\frac{1}{2}\times A^{a_1^{i_1i_2}a_2^{i_1i_2}}_{i_1i_2} g^{a_3^{i_1i_3}a_4^{i_2i_4}}_{i_3i_4} t^{i_1i_3}_{a_3^{i_1i_3}a_5^{i_1i_3}} t^{i_2i_4}_{a_4^{i_2i_4}a_6^{i_2i_4}} S^{a_6^{i_2i_4}}_{a_1^{i_1i_2}} S^{a_5^{i_1i_3}}_{a_2^{i_1i_2}} +$$

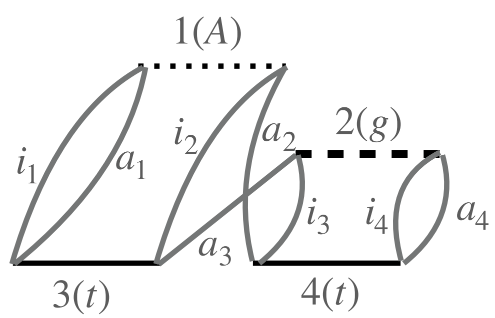
$$-\frac{1}{2}\times A^{a_1^{i_1i_2}a_2^{i_1i_2}}_{i_1i_2} g^{a_3^{i_3i_4}a_4^{i_1i_2}}_{i_3i_4} t^{i_3i_4}_{a_3^{i_3i_4}a_5^{i_3i_4}} t^{i_1i_2}_{a_1^{i_1i_2}a_4^{i_1i_2}} S^{a_5^{i_3i_4}}_{a_2^{i_1i_2}})$$

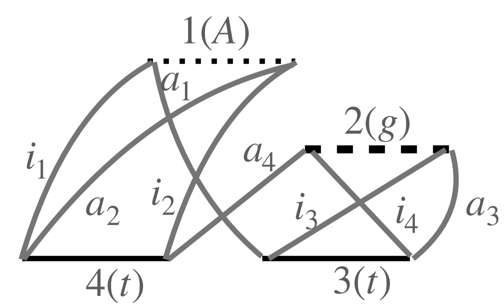
need *complete* canonization

Example: tensor network canonization

$$-\frac{1}{4} \times A_{i_1 i_2}^{a_1 a_2} g_{i_3 i_4}^{a_3 a_4} t_{a_1 a_3}^{i_1 i_2} t_{a_2 a_4}^{i_3 i_4}$$

$$-\frac{1}{4} \times A_{i_1 i_2}^{a_1 a_2} g_{i_3 i_4}^{a_3 a_4} t_{a_1 a_3}^{i_3 i_4} t_{a_2 a_4}^{i_1 i_2}$$





how to determine if 2 diagrams are equivalent?

this is a *graph isomorphism* problem

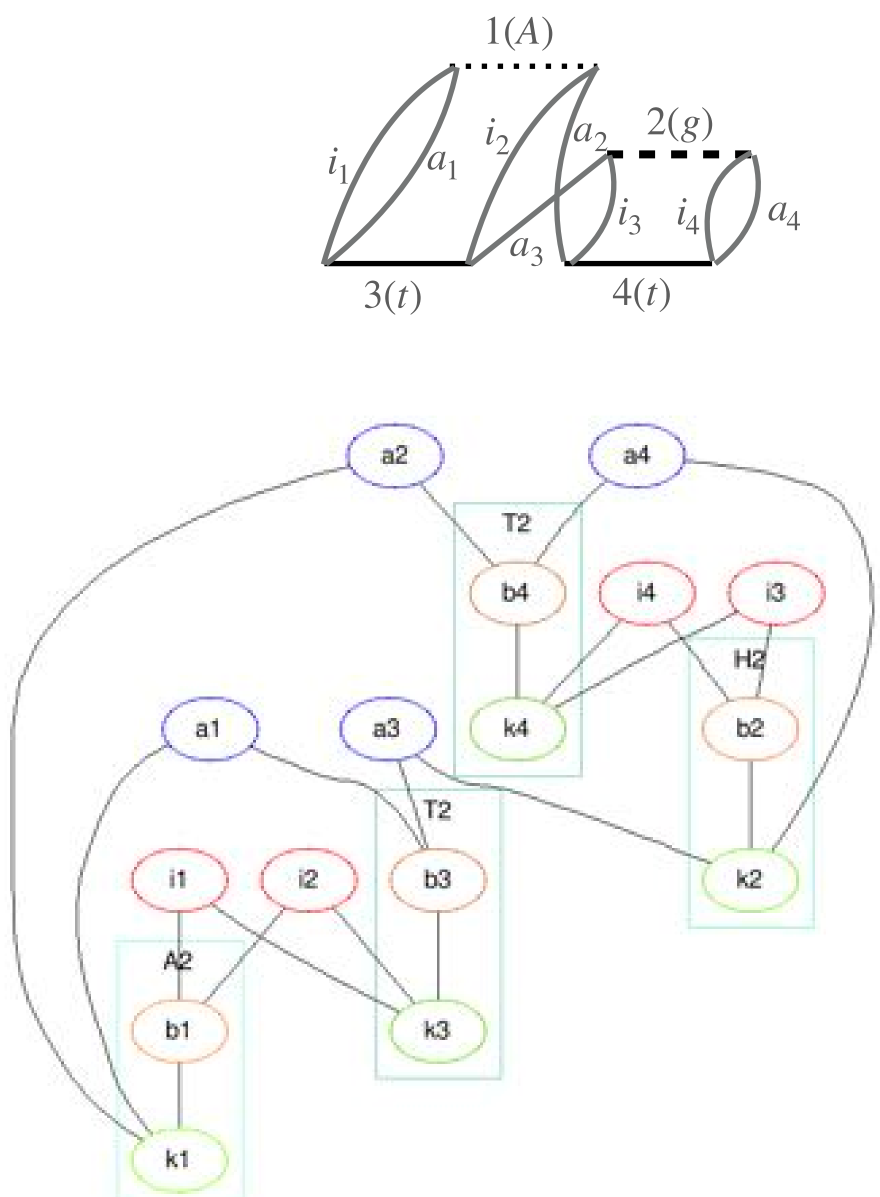
Tensor network canonization

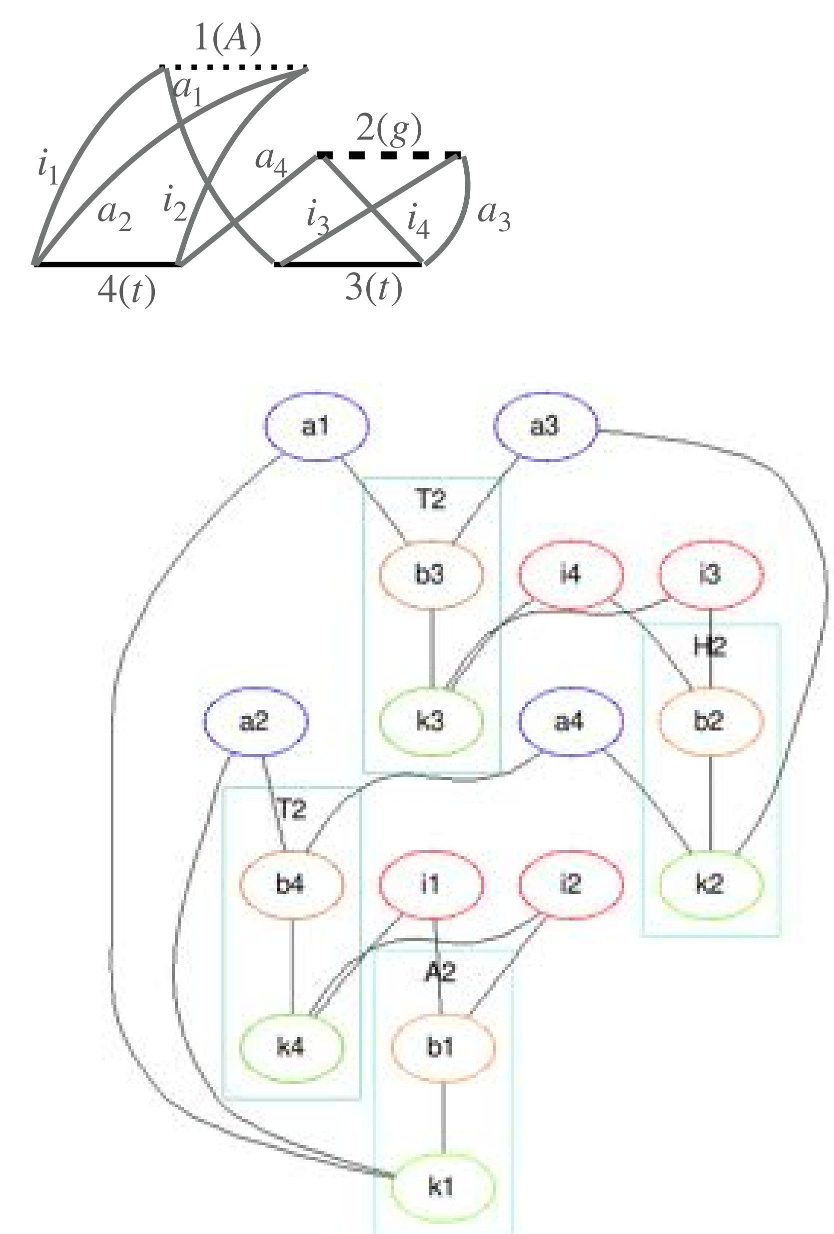
represent diagram as a colored graph whose structure and vertex colors reflect the symmetries of the diagram …

diagram	graph
line	vertex (color = line type)
operator	linked bra and ket vertices (color = operator type)

Tensor network canonization

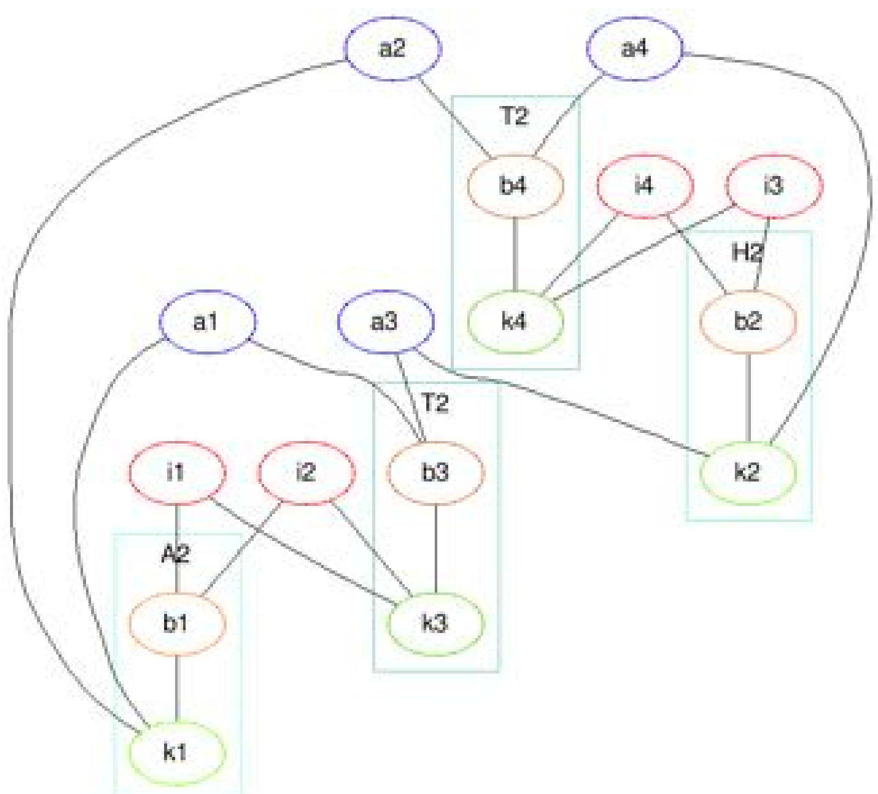
represent diagram as a colored graph whose structure and vertex colors reflect the symmetries of the diagram …

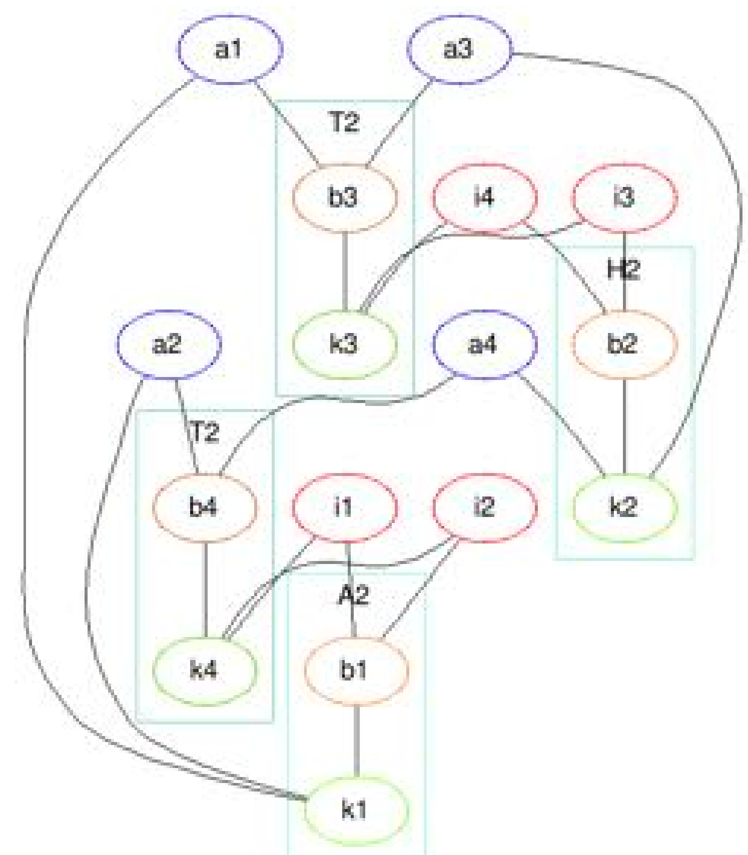




Tensor network canonization

… determine **canonical** order of its vertices …



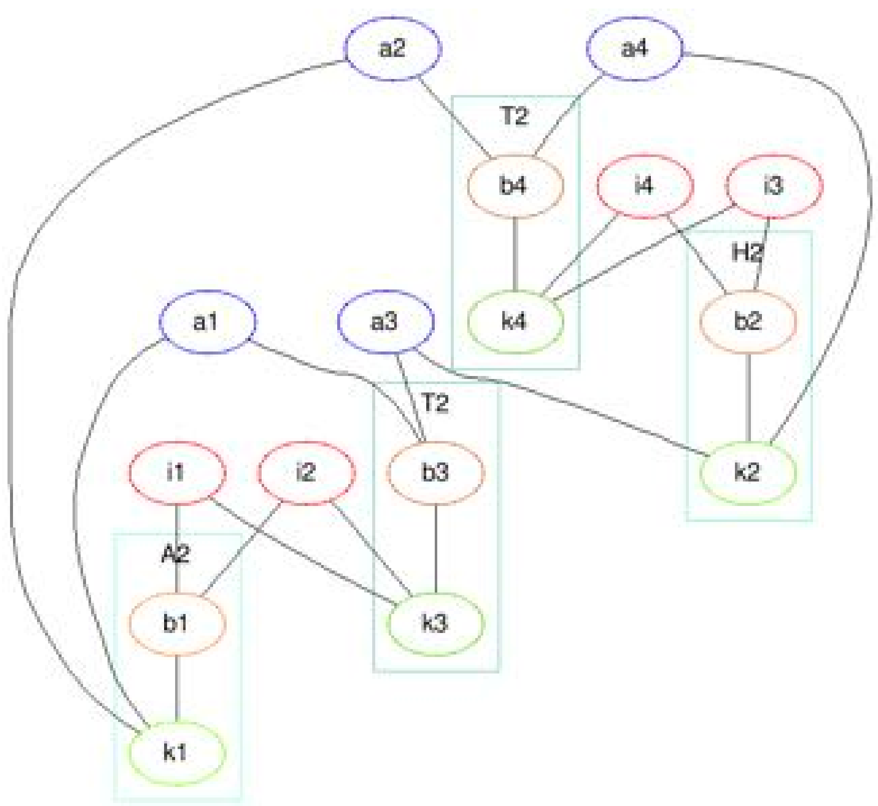


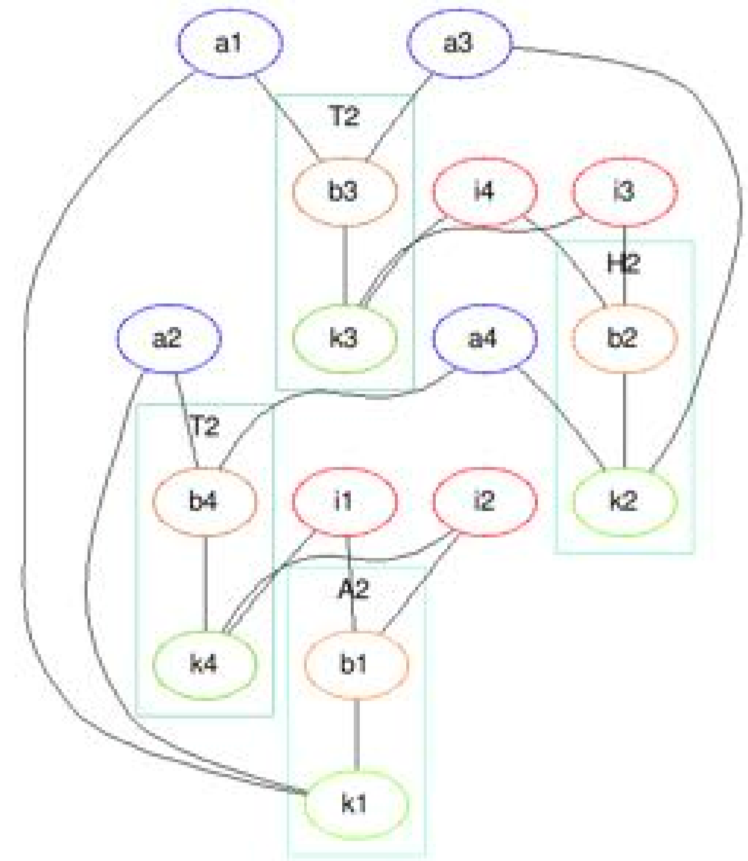

```
b1=>0 k1=>1 b2=>2 k2=>3 b3=>4 k3=>6 b4=>5 k4=>7
i1=>8 i2=>9 i3=>10 i4=>11 a1=>13 a2=>12 a3=>15 a4=>14
```

```
b1=>0 k1=>1 b2=>2 k2=>3 b3=>6 k3=>4 b4=>7 k4=>5
i1=>8 i2=>9 i3=>10 i4=>11 a1=>12 a2=>13 a3=>14 a4=>15
```

Tensor network canonization

… compare the vertices in canonical order …





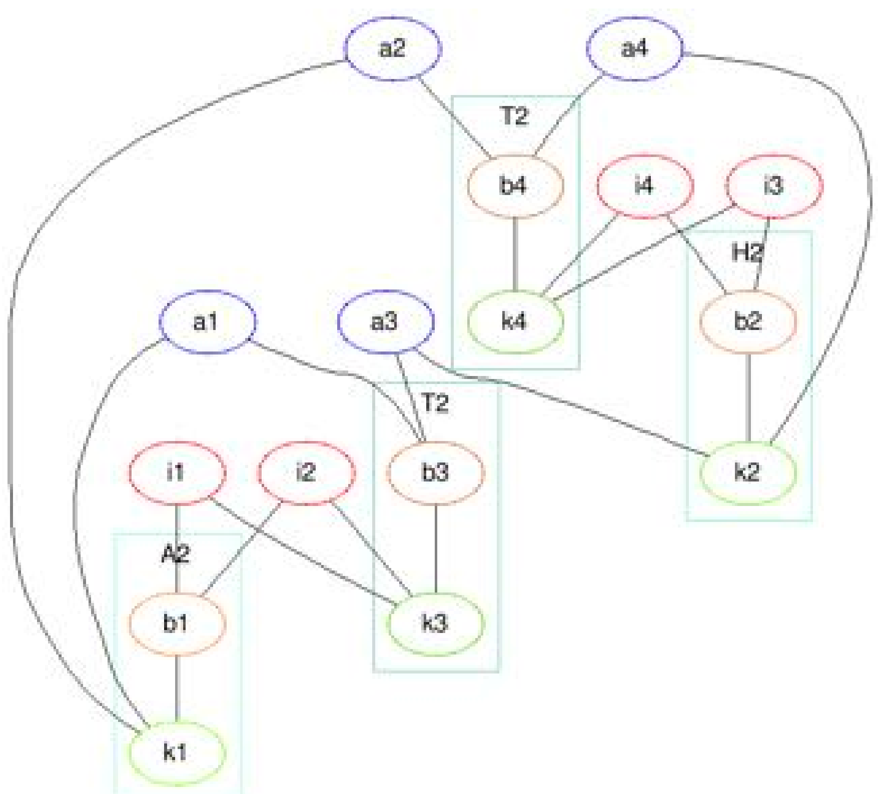

```
 b1=>0 k1=>1 b2=>2 k2=>3 b3=>4 k3=>6 b4=>5 k4=>7
i1=>8 i2=>9 i3=>10 i4=>11 a1=>13 a2=>12 a3=>15 a4=>14
```

```
 b1=>0 k1=>1 b2=>2 k2=>3 b3=>6 k3=>4 b4=>7 k4=>5
i1=>8 i2=>9 i3=>10 i4=>11 a1=>12 a2=>13 a3=>14 a4=>15
```

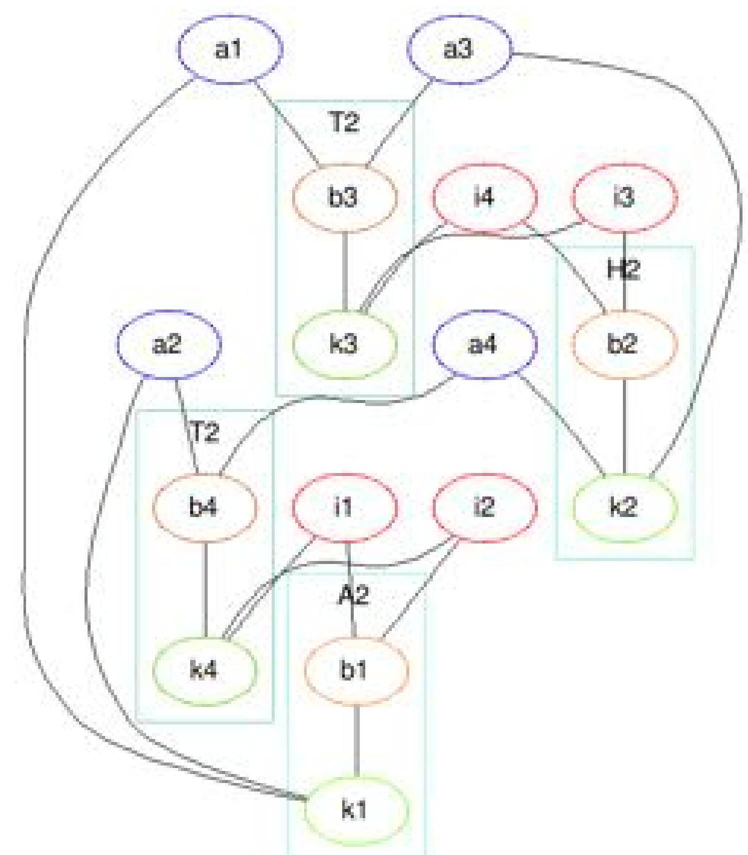
```
b1=>b1 k1=>k1 b2=>b2 k2=>k2 b3=>b4 k3=>k4 b4=>b3 k4=>k3
i1=>i1 i2=>i2 i3=>i3 i4=>i4 a1=>a2 a2=>a1 a3=>a4 a4=>a3
```

Tensor network canonization

… and combine the diagrams



$$a1 \leftrightarrow a2$$
$$a3 \leftrightarrow a4$$
$$b3 \leftrightarrow b4$$
$$k3 \leftrightarrow k4$$



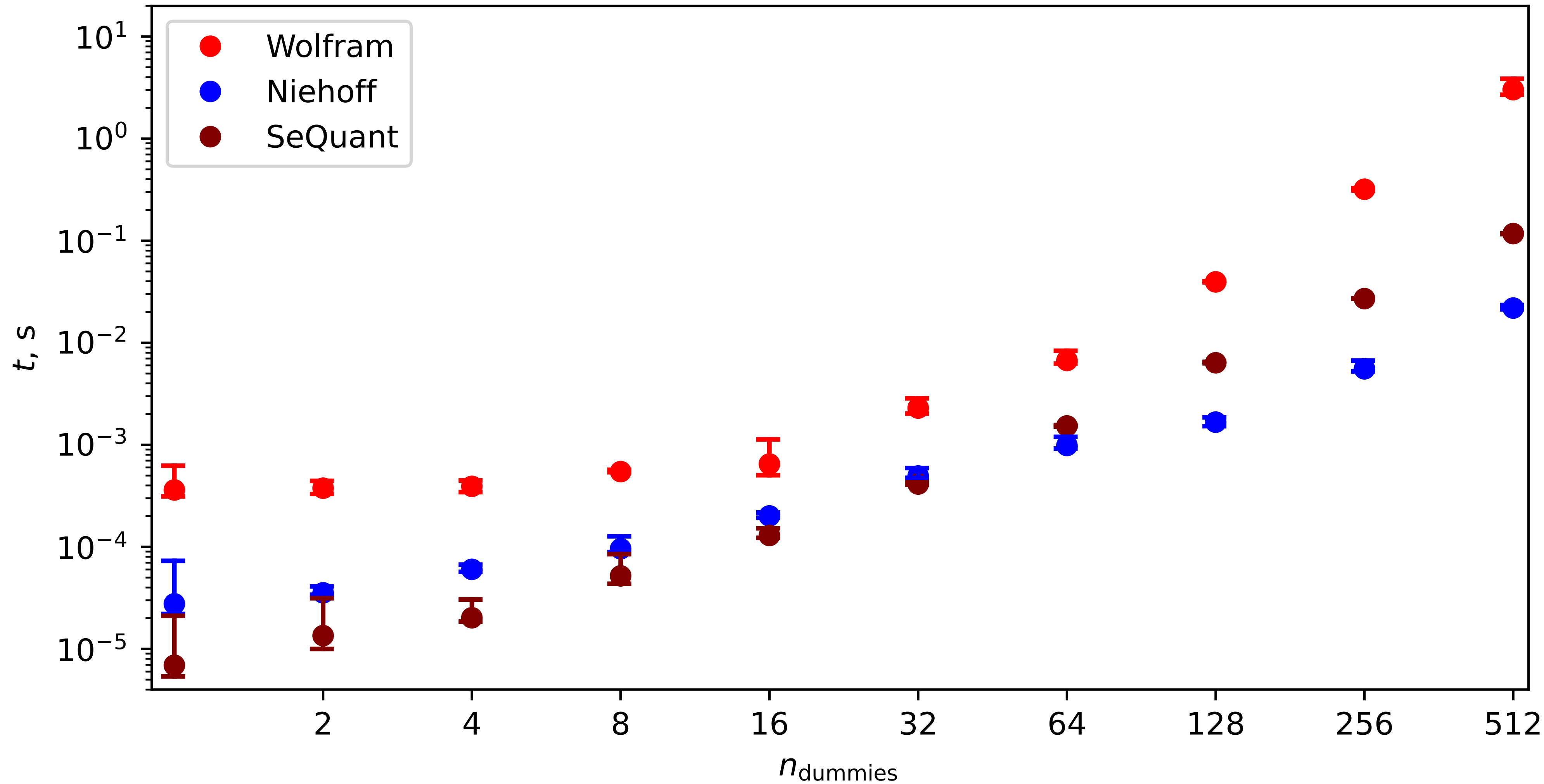
$$-\frac{1}{4} \times A^{a_1 a_2}_{i_1 i_2} g^{a_3 a_4}_{i_3 i_4} t^{i_1 i_2}_{a_1 a_3} t^{i_3 i_4}_{a_2 a_4} - \frac{1}{4} \times A^{a_1 a_2}_{i_1 i_2} g^{a_3 a_4}_{i_3 i_4} t^{i_3 i_4}_{a_1 a_3} t^{i_1 i_2}_{a_2 a_4} = -\frac{1}{2} \times A^{a_1 a_2}_{i_1 i_2} g^{a_3 a_4}_{i_3 i_4} t^{i_1 i_2}_{a_1 a_3} t^{i_3 i_4}_{a_2 a_4}$$

still need extra structure to support protoindices and non-symmetric tensors …

Tensor network canonization: colored graph vs others

canonicalizing $D_{i_1 \ldots i_N} U^{\pi[i_1 \ldots i_N]}$

asymmetric D and U

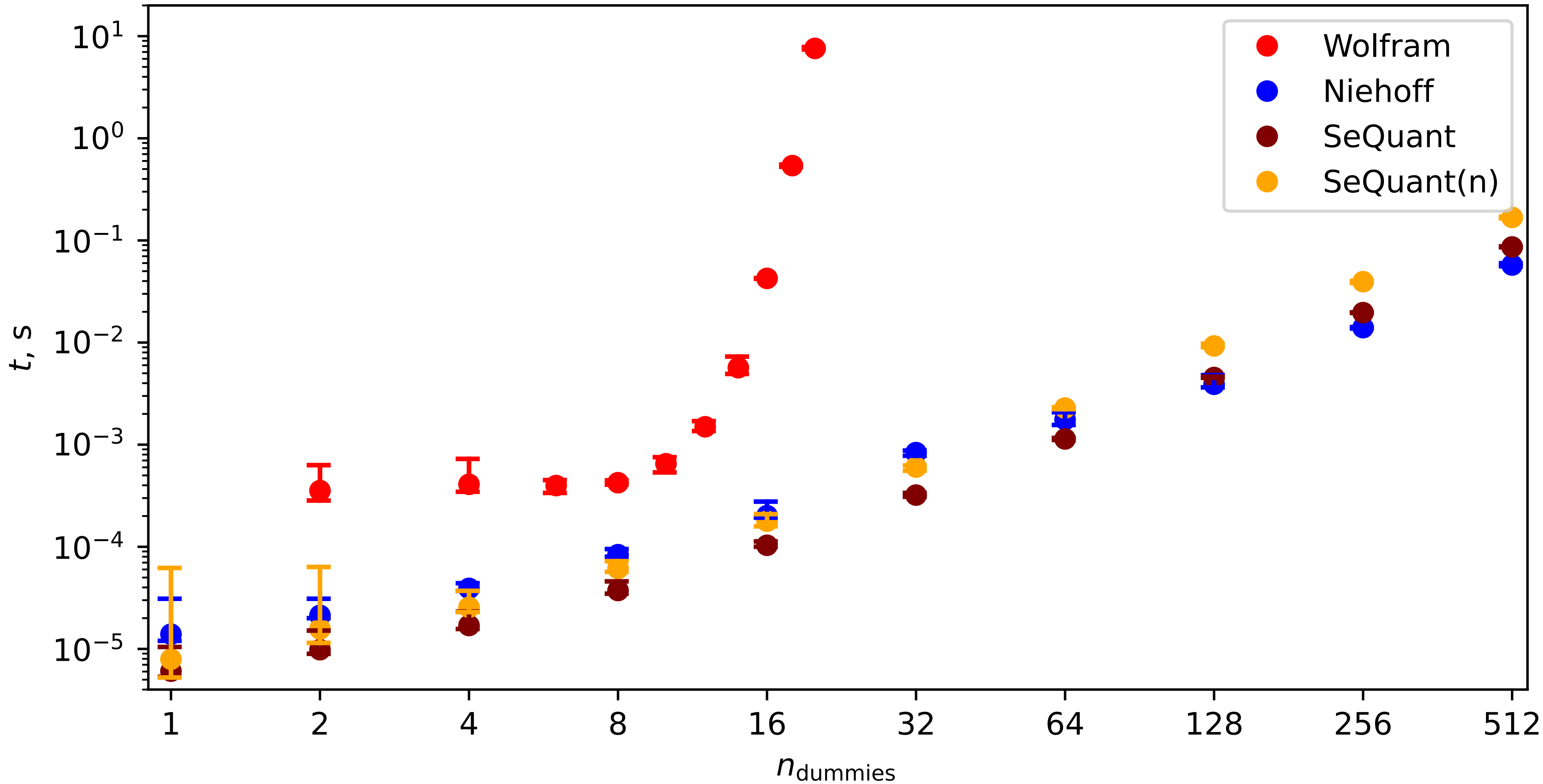


group-theoretic Butler-Portugal algorithm ("Wolfram") is near optimal for asymmetric tensors

Tensor network canonization: colored graph vs others

canonicalizing $D_{i_1 \ldots i_N} U^{\pi[i_1 \ldots i_N]}$

totally-symmetric D, asymmetric U



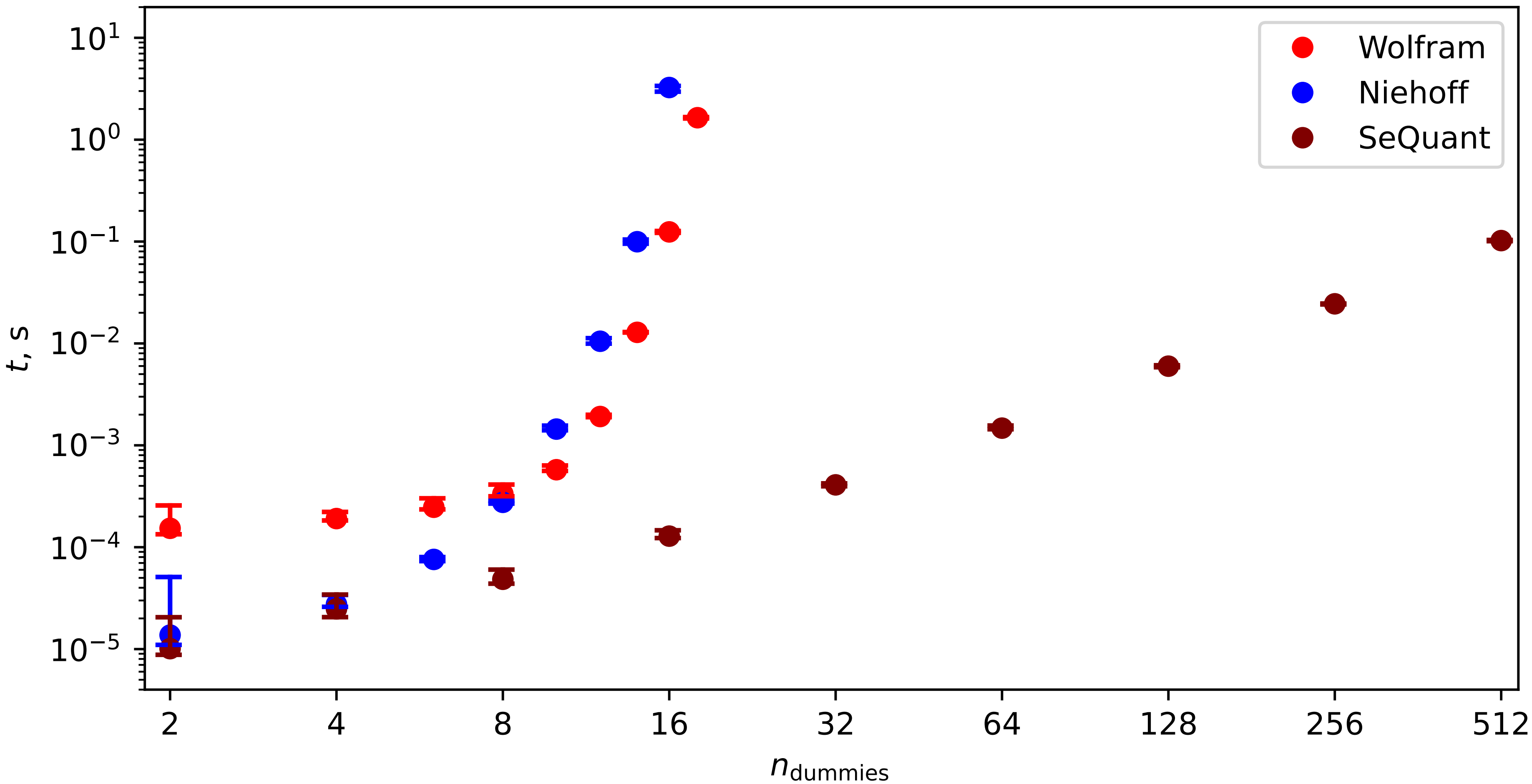
graph-theoretic extension of Butler-Portugal ("Niehoff") can handle some symmetric cases

Niehoff, Comp. Phys. Comm. 228, 123-145 (2018).

Tensor network canonization: colored graph vs others

canonicalizing $D_{i_1 i_2} D_{i_3 i_4} \ldots D_{i_{N-1} i_N} U^{\pi[i_1 \ldots i_N]}$

asymmetric D **and** U



our group-theoretic algorithm is fast for general tensor networks

SeQuant2: Fast(er) Wick Engine

- Thread-level concurrency (C++ threads or std::execution::par)
- Avoid equivalent contractions by topological info generated by tensor network automorphizer

Wick Theorem Optimization

- Further optimizations are possible if vacuum expectation values are wanted:
  - By tracking quasiparticle numbers (esp. easy if normal operators are pure (quasi)particle creators/annihilators e.g. in single-reference coupled-cluster)
  - By using topological symmetry of the expression* (this is similar to diagram-based approaches)
    - Topologically-equivalent ops in normal operators, e.g. $\langle 0 | \hat{T}_3^\dagger \hat{T}_3 | 0 \rangle$

$$\tilde{a}^{i_1 i_2 i_3}_{a_1 a_2 a_3} \tilde{a}^{a_4 a_5 a_6}_{i_4 i_5 i_6} \rightarrow \times 3 \quad \tilde{a}^{i_1 i_2 i_3}_{a_1 a_2 a_3} \tilde{a}^{a_4 a_5 a_6}_{i_4 i_5 i_6} \rightarrow \times 0 \quad \tilde{a}^{i_1 i_2 i_3}_{a_1 a_2 a_3} \tilde{a}^{a_4 a_5 a_6}_{i_4 i_5 i_6} \rightarrow \times 2$$

    - Topologically-equivalent normal operators in product, e.g. $\langle 0 | \hat{W} \hat{T}_1^2 | 0 \rangle$

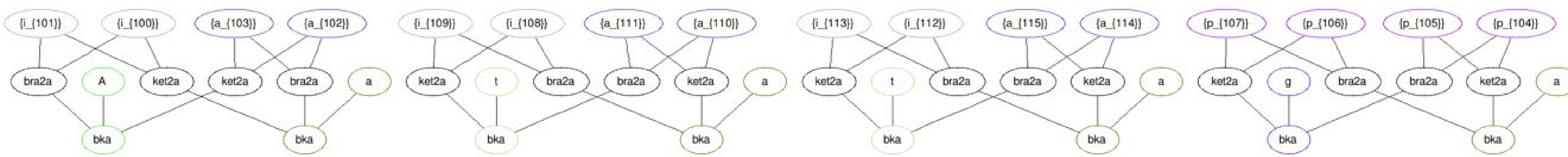
$$\tilde{a}^{p_1 p_2}_{p_3 p_4} \tilde{a}^{a_4}_{i_4} \tilde{a}^{a_5}_{i_5} \rightarrow \times 2 \quad \tilde{a}^{p_1 p_2}_{p_3 p_4} \tilde{a}^{a_4}_{i_4} \tilde{a}^{a_5}_{i_5} \rightarrow \times 0 \quad \tilde{a}^{p_1 p_2}_{p_3 p_4} \tilde{a}^{a_4}_{i_4} \tilde{a}^{a_5}_{i_5} \rightarrow \times 1 \quad \tilde{a}^{p_1 p_2}_{p_3 p_4} \tilde{a}^{a_4}_{i_4} \tilde{a}^{a_5}_{i_5} \rightarrow \times 1$$

e.g. see Xiao, Wang, Zhu, Comp. Phys. Comm. 184, 1966 (2013)

Wick Theorem Optimization: Details

- Topological symmetry of the input expression is determined by colored graph mapping and analysis similar to that used previously for canonization
  - Step 1: input expression is mapped to colored graph

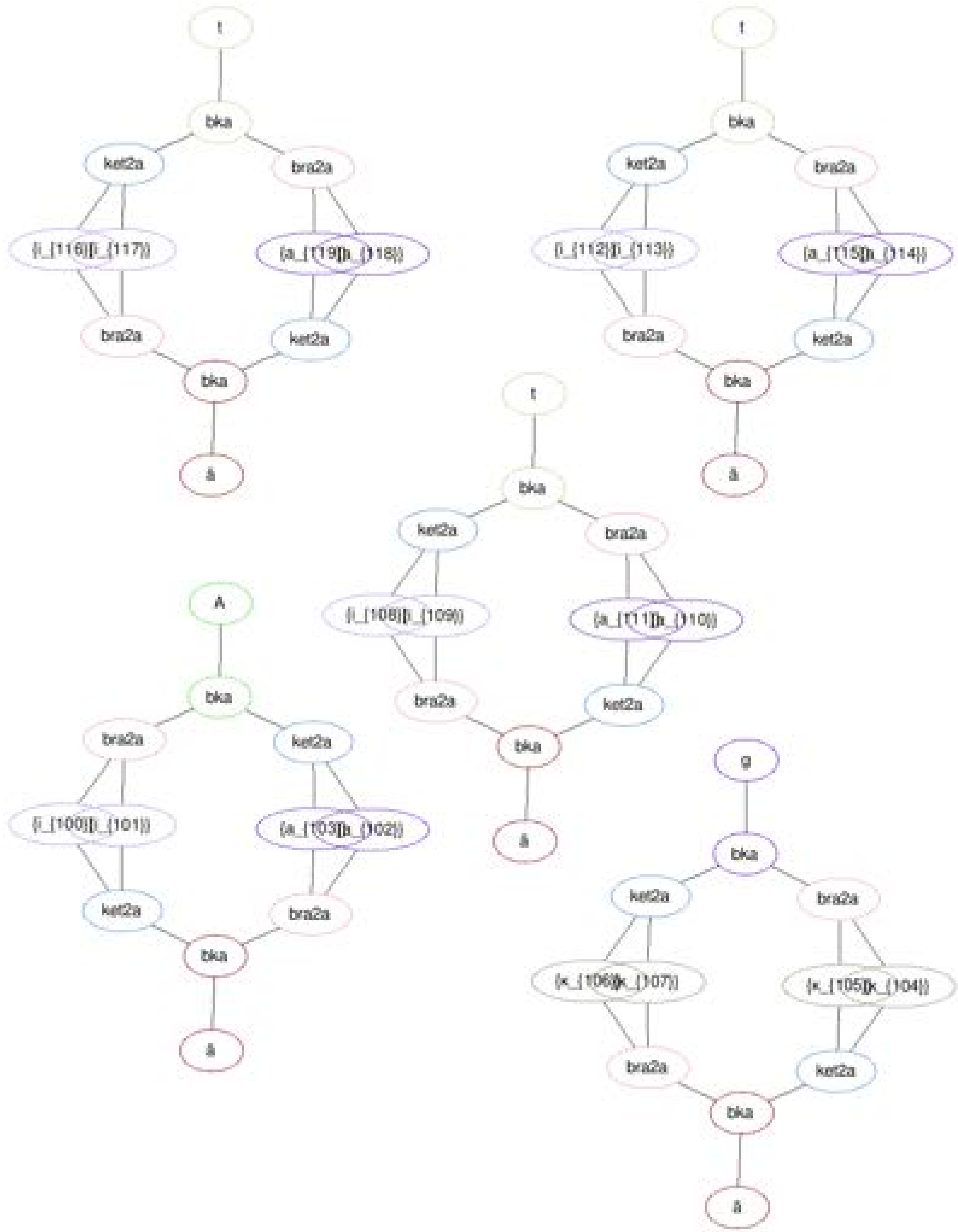
$$\langle 0|\hat{A}_2\hat{W}\hat{T}_2\hat{T}_2|0\rangle \equiv \frac{1}{256} \times A^{a_{102}a_{103}}_{i_{100}i_{101}}\tilde{a}^{i_{100}i_{101}}_{a_{102}a_{103}} \times g^{p_{106}p_{107}}_{p_{104}p_{105}}\tilde{a}^{p_{104}p_{105}}_{p_{106}p_{107}} \times t^{i_{108}i_{109}}_{a_{110}a_{111}}\tilde{a}^{a_{110}a_{111}}_{i_{108}i_{109}} \times t^{i_{112}i_{113}}_{a_{114}a_{115}}\tilde{a}^{a_{114}a_{115}}_{i_{112}i_{113}}$$



  - Step 2: compute the automorphism group of the graph
  - Step 3: test topological equivalence of operators and indices

Wick Theorem Optimization: Details

$$\langle 0|\hat{A}_2\hat{W}\hat{T}_2^3|0\rangle \equiv \frac{1}{1024}\times A^{a_{102}a_{103}}_{i_{100}i_{101}}\tilde{a}^{i_{100}i_{101}}_{a_{102}a_{103}}\times g^{\kappa_{106}\kappa_{107}}_{\kappa_{104}\kappa_{105}}\tilde{a}^{\kappa_{104}\kappa_{105}}_{\kappa_{106}\kappa_{107}}\times t^{i_{108}i_{109}}_{a_{110}a_{111}}\tilde{a}^{a_{110}a_{111}}_{i_{108}i_{109}}\times t^{i_{112}i_{113}}_{a_{114}a_{115}}\tilde{a}^{a_{114}a_{115}}_{i_{112}i_{113}}\times t^{i_{116}i_{117}}_{a_{118}a_{119}}\tilde{a}^{a_{118}a_{119}}_{i_{116}i_{117}}$$



Automorphism Group Generators:

- ({κ_{106}},{κ_{107}})
- ({κ_{104}},{κ_{105}})
- ({a_{102}},{a_{103}})
- ({i_{100}},{i_{101}})
- ({a_{110}},{a_{111}})
- ({i_{108}},{i_{109}})
- ({a_{114}},{a_{115}})
- ({a_{118}},{a_{119}})
- ({i_{112}},{i_{113}})
- ({i_{116}},{i_{117}})
- ({a_{114}},{a_{118}})({a_{115}},{a_{119}})({i_{112}},{i_{116}})({i_{113}},{i_{117}})(t,t)(bra2a,bra2a)(ket2a,ket2a)(bka,bka)(ã,ã)(bra2a,bra2a)(ket2a,ket2a)(bka,bka)
- ({a_{110}},{a_{114}})({a_{111}},{a_{115}})({i_{108}},{i_{112}})({i_{109}},{i_{113}})(t,t)(bra2a,bra2a)(ket2a,ket2a)(bka,bka)(ã,ã)(bra2a,bra2a)(ket2a,ket2a)(bka,bka)

Example: SR CC timings (seconds)

	Screen*/Topology				Simple H**
	F/F	T/F	F/T	T/T	
SD	2.4	0.14	0.04	0.02	**0.02**
+T	6540	6.0	1.8	0.14	**0.07**
+Q	-	1210	115	1.35	**0.43**
+P	-	-	-	16.0	**3.5**
+H	-	-	-	-	**43**

Intel Core i7 7820HQ/ Apple Clang 10 / Hoard 3.13 malloc
* Screen operator products by possible excitation level
** Nonredundant BSH expansion of CC Hbar, i.e. combine H T1 T2 and H T2 T1

further screening improvements are straightforward …

SeQuant2: Optimization

- Evaluation of individual tensor networks uses exhaustive or heuristic search to determine optimized evaluation order

$$X = (AB)C = A(BC)$$

- Evaluation of sets (e.g., sums) of tensor networks uses CSE

$$X = AB + (AB)C = Y + YC; \quad Y = AB$$

- Fusion is also possible, not yet deployed in production

$$X = AB + AC = A(B + C)$$

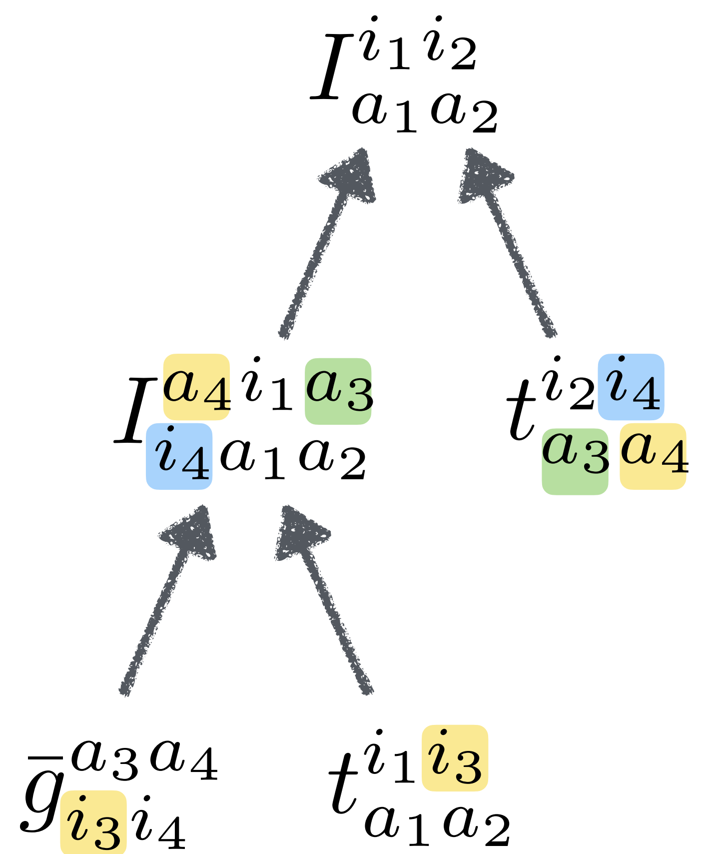
SeQuant2: Single Term Optimization / TN Binarization

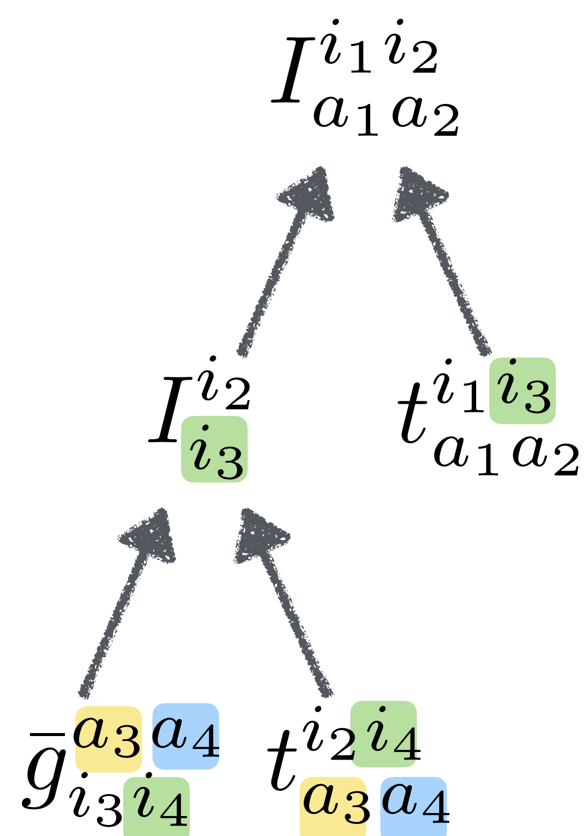
$$\left(\bar{g}^{a_3 a_4}_{i_3 i_4} t^{i_1 i_3}_{a_1 a_2}\right) t^{i_2 i_4}_{a_3 a_4}$$

$$\left(\bar{g}^{a_3 a_4}_{i_3 i_4} t^{i_2 i_4}_{a_3 a_4}\right) t^{i_1 i_3}_{a_1 a_2}$$

O^3V^4

O^3V^4

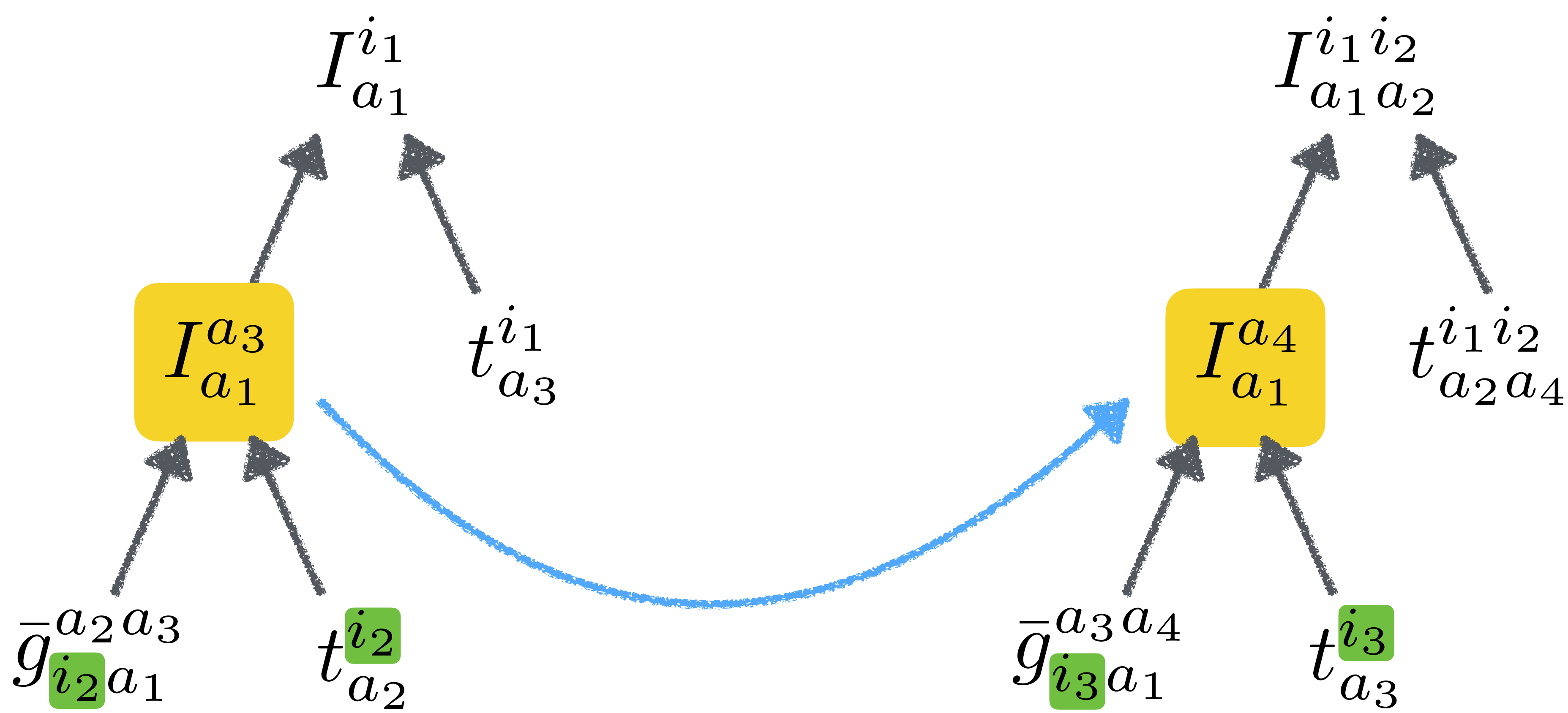




O^3V^2

O^3V^2

SeQuant2: TN Set CSE



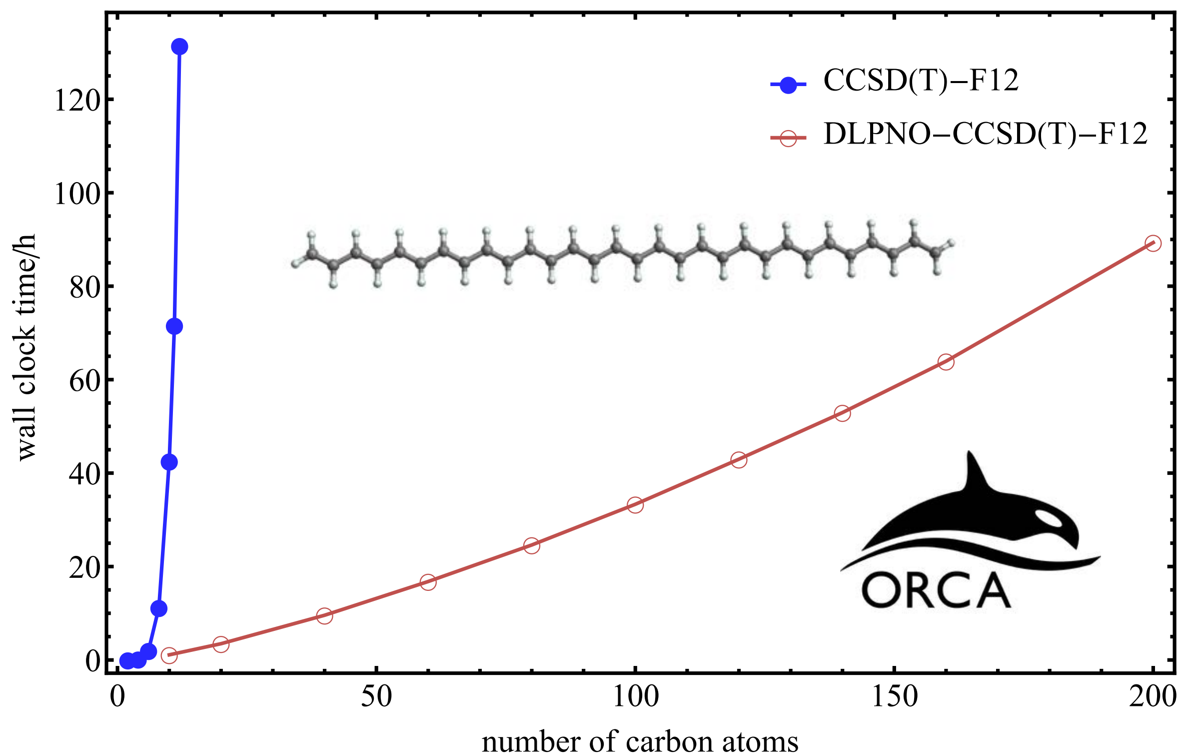
SeQuant2: Evaluation

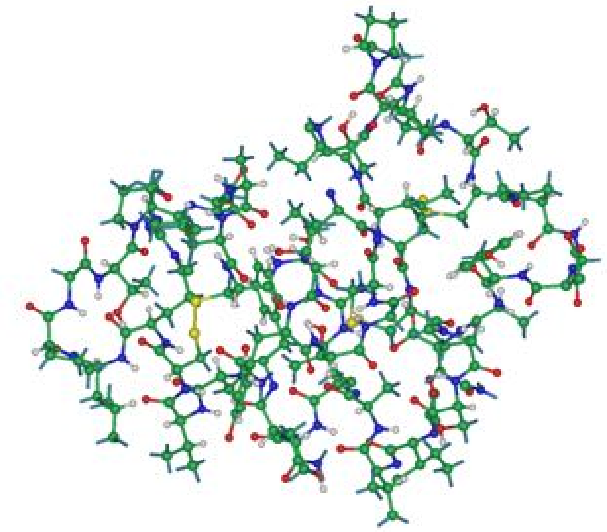
- Uses TiledArray (BTAS for reference testing; can add others)
- Rudimentary resource management
- Limited to conventional expressions (protoindex-free)
  - Work underway to extend to protoindex-containing expressions
    - Requires more functionality in TA

DATA-SPARSE TENSOR ALGEBRA

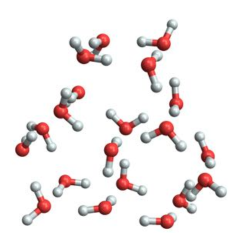
Better: Tensors in CC Networks are Data-Sparse!

conventional/dense CCSD(T) = O(N^7)   data-sparse CCSD(T) = O(N)





DLPNO-CC-F12 can be done on entire proteins now!



$(H_2O)_{20}$
60 atoms
960 basis functions

Formulation	# of cores (nodes)	t(min)	Dissociation Energy (kcal/mol)
dense	32768 (2048)	94.1[a]	185.61
sparse	16 (1)	77.7[b]	184.75

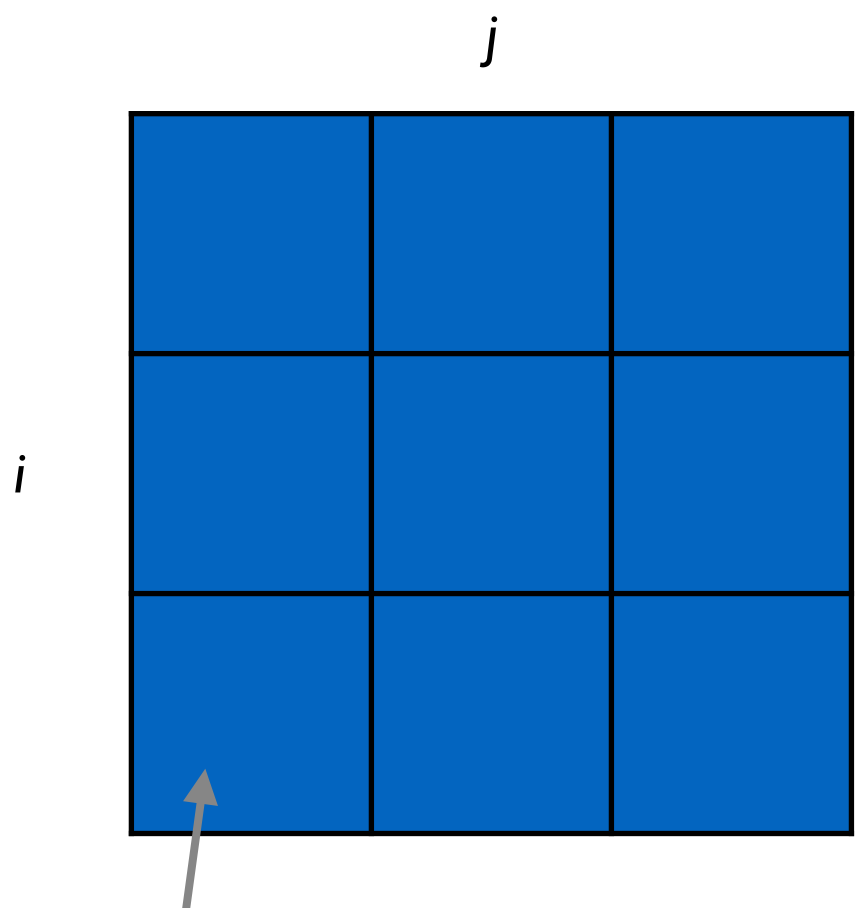
MPQC
ORCA

Pavosevic, Peng, Pinski, Riplinger, Neese, Valeev, J. Chem. Phys. (2017)

Data Sparsity Patterns Suggest New Tensor Formats

dense T2 tensor

order-4 tensor of scalars

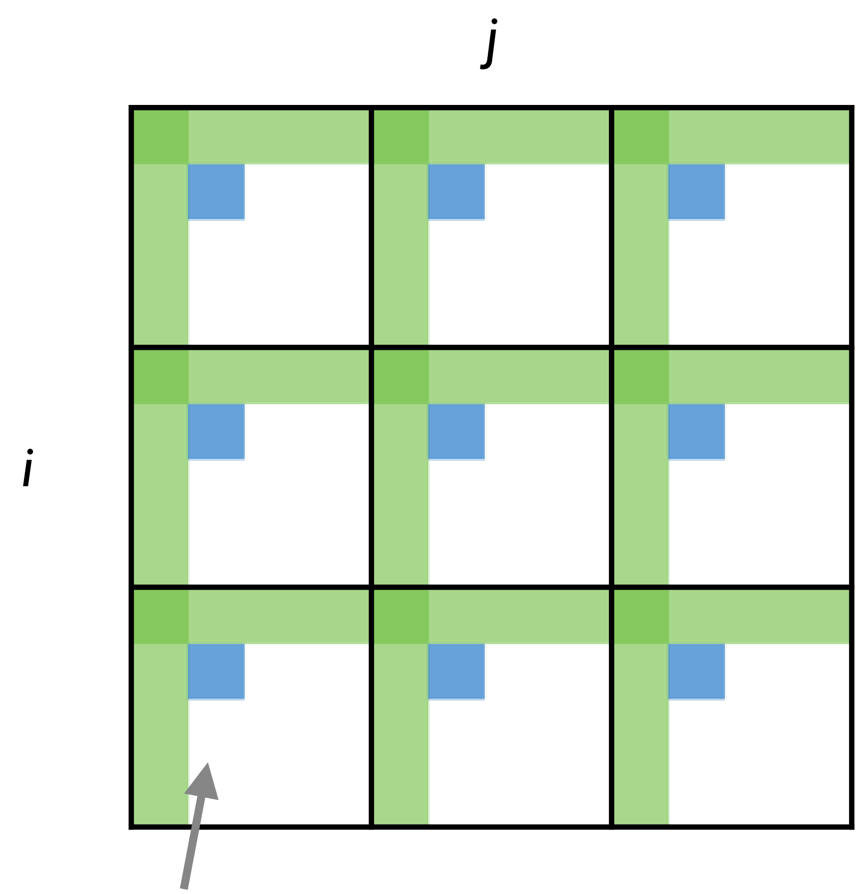


$$\mathbf{T}^{3,1} \equiv \{t_{ab}^{31}\}$$

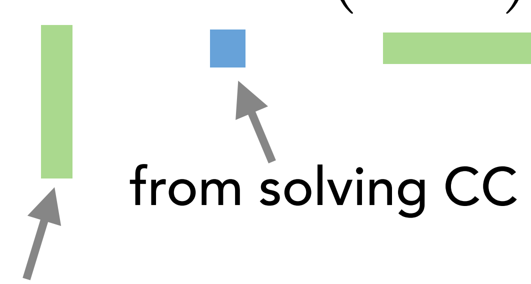
block-rank compressed (PNO) T2 tensor

Edmiston (1960s), Meyer (1970s), Neese (2000s)

order-2 tensor of order-2 tensors



$$\mathbf{T}^{3,1} \approx \mathbf{U}^{3,1}\ \mathbf{t}^{3,1}\ \left(\mathbf{V}^{3,1}\right)^{\dagger}$$



from solving CC eqn in subspace

fixed subspace from crude guess

Conventional vs PNO CC Methods

MP2 $R^{ij}_{ab} = G^{ij}_{ab} + F^c_a T^{ij}_{cb} + F^c_b T^{ij}_{ac} - F^i_k T^{kj}_{ab} - F^j_k T^{ik}_{ab}$

- familiar algebra of dense/block-sparse tensors with independent dimensions
- covariant algebra = affine loop nests, familiar optimization problems
- reduces to dense linear algebra kernels with high FLOP/MOP ratio

PNO MP2

$$R^{ij}_{a_{ij}b_{ij}} = G^{ij}_{a_{ij}b_{ij}} + F^{c_{ij}}_{a_{ij}} T^{ij}_{c_{ij}b_{ij}} + F^{c_{ij}}_{b_{ij}} T^{ij}_{a_{ij}c_{ij}}$$
$$- F^i_k T^{kj}_{a_{kj}b_{kj}} S^{a_{kj}}_{a_{ij}} S^{b_{kj}}_{b_{ij}} - F^j_k T^{ik}_{a_{ik}b_{ik}} S^{a_{ik}}_{a_{ij}} S^{b_{ik}}_{b_{ij}}$$

- algebra of block-sparse/element-sparse tensors with dependent dimensions
- noncovariant algebra = nonaffine loop nests
- sparsity patterns are *bootstrapped* iteratively, controlled by user-controlled thresholding
- reduces to dense linear algebra kernels with low FLOP/MOP ratio
- implementation complexity is largely driven by the sparsity computation/metadata manipulation

Complex Noncovariant Tensor Networks in PNO CC Methods

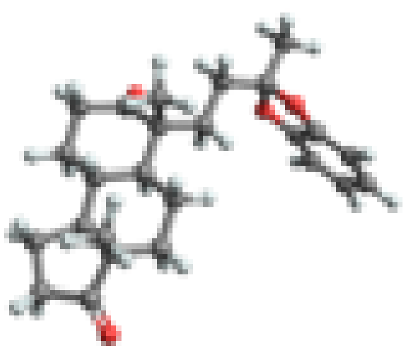
PNO MP2 $$G^{ij}_{a_{ij}b_{ij}} = C^{\tilde{\mu}_{ij}}_{a_{ij}} V^{iX_{ij}}_{\tilde{\mu}_{ij}} (\mathbf{V}^{-1})_{X_{ij}Y_{ij}} V^{jY_{ij}}_{\tilde{\nu}_{ij}} C^{\tilde{\nu}_{ij}}_{b_{ij}}$$

- high-level tensor notation doesn't capture complete details without undue complexity, hence need a mix of high-level DSL-like composition and imperative programming
- richer variety of data structures and algorithms, e.g. the above is implemented as a single 5-way contraction rather than a sequence of binary contractions

```
for i in [0,#occ):
  for all j in pair list of i:
    if i!=j:
      obtain unique parts of V(i,mu_i,X_i), V(i,mu_j,X_i), V(i,mu_i,X_j), V(i,mu_j,X_j) and
        merge into V(i,mu_ij,X_ij)
      obtain unique parts of V(j,mu_i,X_i), V(j,mu_j,X_i), V(j,mu_i,X_j), V(j,mu_j,X_j) and
        merge into V(j,mu_ij,X_ij)
      obtain unique parts of V(X_i,Y_i), V(X_i,Y_j), V(X_j,Y_i), V(X_j,Y_j), and
        merge into V(X_ij,Y_ij)
      concatenate OSVs C(a_i,mu_i) and C(a_j,mu_j) into C(a_ij,mu_ij)
      W(i,a_ij,X_ij) = C(a_ij,mu_ij) * V(i,mu_ij,X_ij)
      W(j,a_ij,X_ij) = C(a_ij,mu_ij) * V(i,mu_ij,X_ij)
      Vinv(X_ij,Y_ij) = inverse(V(X_ij,Y_ij))
      f(j,a_ij,X_ij) = W(j,a_ij,X_ij) * Vinv(X_ij,Y_ij)
      g(i,j,a_ij,b_ij) = W(i,a_ij,X_ij) * f(j,a_ij,X_ij)
    else: // i==j
      ..
```

composed as tasks generated by imperative code over local and distributed data

Manually-Written PNO MP2 Implementation in MPQC



- 31% of electron pairs screened out
- average pair domain = 729 bf
- average # of PNOs = 91
- recovered 99.7 % of DF-MP2 correlation energy

Performance (sec) vs. state-of-the-art PNO-MP2 in Molpro*

*Werner, et al, DOI 10.1021/ct500725e

Step	Molpro	MPQC
orbital domains (OSV)	43	54
integral transform	161	139
PNO generation	37	91
LMP2 solver	54	21
Total	265	341

20 cores x 2.8 GHz AVX

competitive performance with leading code already, but more optimizations possible

Progress: Automating PNO-CCSD implementation

$$\langle 0|\hat{A}_2\bar{H}|0\rangle = \frac{1}{8}\times A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} g^{\alpha_3^{i_1i_2}\alpha_4^{i_1i_2}}_{i_3i_4} t^{i_3}_{\alpha_5^{i_3}} t^{i_4}_{\alpha_6^{i_4}} t^{i_1i_2}_{\alpha_3^{i_1i_2}\alpha_4^{i_1i_2}} S^{\alpha_5^{i_3}}_{\alpha_1^{i_1i_2}} S^{\alpha_6^{i_4}}_{\alpha_2^{i_1i_2}} + -1\times A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} g^{i_1\alpha_3^{i_2i_4}}_{i_3i_4} t^{i_3}_{\alpha_4^{i_3}} t^{i_2i_4}_{\alpha_3^{i_2i_4}\alpha_5^{i_2i_4}} S^{\alpha_5^{i_2i_4}}_{\alpha_1^{i_1i_2}} S^{\alpha_4^{i_3}}_{\alpha_2^{i_1i_2}} + \frac{1}{16}\times A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} g^{\alpha_3^{i_1i_2}\alpha_4^{i_1i_2}}_{i_3i_4} t^{i_3i_4}_{\alpha_5^{i_3i_4}\alpha_6^{i_3i_4}} t^{i_1i_2}_{\alpha_3^{i_1i_2}\alpha_4^{i_1i_2}} S^{\alpha_5^{i_3i_4}}_{\alpha_1^{i_1i_2}} S^{\alpha_6^{i_3i_4}}_{\alpha_2^{i_1i_2}} +$$

$$\frac{1}{8}\times A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} g^{\alpha_3^{i_1i_2}\alpha_4^{i_1i_2}}_{i_3i_4} t^{i_3}_{\alpha_5^{i_3}} t^{i_4}_{\alpha_6^{i_4}} t^{i_1i_2}_{\alpha_3^{i_1i_2}\alpha_4^{i_1i_2}} S^{\alpha_5^{i_3}}_{\alpha_1^{i_1i_2}} S^{\alpha_6^{i_4}}_{\alpha_2^{i_1i_2}} + -1\times A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} g^{i_1\alpha_3^{i_2i_4}}_{i_3i_4} t^{i_3}_{\alpha_4^{i_3}} t^{i_2i_4}_{\alpha_3^{i_2i_4}\alpha_5^{i_2i_4}} S^{\alpha_5^{i_2i_4}}_{\alpha_1^{i_1i_2}} S^{\alpha_4^{i_3}}_{\alpha_2^{i_1i_2}} + \frac{1}{16}\times A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} g^{\alpha_3^{i_1i_2}\alpha_4^{i_1i_2}}_{i_3i_4} t^{i_3i_4}_{\alpha_5^{i_3i_4}\alpha_6^{i_3i_4}} t^{i_1i_2}_{\alpha_3^{i_1i_2}\alpha_4^{i_1i_2}} S^{\alpha_5^{i_3i_4}}_{\alpha_1^{i_1i_2}} S^{\alpha_6^{i_3i_4}}_{\alpha_2^{i_1i_2}} +$$

$$\frac{1}{2}\times A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} g^{\alpha_3^{i_4}\alpha_4^{i_1i_2}}_{i_3i_4} t^{i_3}_{\alpha_5^{i_3}} t^{i_4}_{\alpha_3^{i_4}} t^{i_1i_2}_{\alpha_1^{i_1i_2}\alpha_4^{i_1i_2}} S^{\alpha_5^{i_3}}_{\alpha_2^{i_1i_2}} + -\frac{1}{2}\times A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} f^{\alpha_3^{i_1i_2}}_{i_3} t^{i_3}_{\alpha_4^{i_3}} t^{i_1i_2}_{\alpha_1^{i_1i_2}\alpha_3^{i_1i_2}} S^{\alpha_4^{i_3}}_{\alpha_2^{i_1i_2}} + -\frac{1}{2}\times A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} g^{\alpha_3^{i_3}\alpha_4^{i_1i_2}}_{i_3\alpha_1^{i_1i_2}} t^{i_3}_{\alpha_3^{i_3}} t^{i_1i_2}_{\alpha_2^{i_1i_2}\alpha_4^{i_1i_2}} +$$

$$\frac{1}{8}\times A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} g^{\alpha_3^{i_1}\alpha_4^{i_2}}_{i_3i_4} t^{i_1}_{\alpha_3^{i_1}} t^{i_2}_{\alpha_4^{i_2}} t^{i_3i_4}_{\alpha_5^{i_3i_4}\alpha_6^{i_3i_4}} S^{\alpha_5^{i_3i_4}}_{\alpha_1^{i_1i_2}} S^{\alpha_6^{i_3i_4}}_{\alpha_2^{i_1i_2}} + \frac{1}{4}\times A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} g^{\alpha_3^{i_1}\alpha_4^{i_2}}_{i_3i_4} t^{i_3}_{\alpha_5^{i_3}} t^{i_4}_{\alpha_6^{i_4}} t^{i_1}_{\alpha_3^{i_1}} t^{i_2}_{\alpha_4^{i_2}} S^{\alpha_5^{i_3}}_{\alpha_1^{i_1i_2}} S^{\alpha_6^{i_4}}_{\alpha_2^{i_1i_2}} + A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} g^{i_1\alpha_3^{i_2i_3}}_{i_3\alpha_1^{i_1i_2}} t^{i_2i_3}_{\alpha_3^{i_2i_3}\alpha_4^{i_2i_3}} S^{\alpha_4^{i_2i_3}}_{\alpha_2^{i_1i_2}} +$$

$$-\frac{1}{2}\times A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} f^{\alpha_3^{i_1i_2}}_{\alpha_1^{i_1i_2}} t^{i_1i_2}_{\alpha_2^{i_1i_2}\alpha_3^{i_1i_2}} + \frac{1}{2}\times A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} f^{i_1}_{i_3} t^{i_2i_3}_{\alpha_3^{i_2i_3}\alpha_4^{i_2i_3}} S^{\alpha_3^{i_2i_3}}_{\alpha_1^{i_1i_2}} S^{\alpha_4^{i_2i_3}}_{\alpha_2^{i_1i_2}} + A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} g^{i_1\alpha_3^{i_2}}_{i_3\alpha_1^{i_1i_2}} t^{i_3}_{\alpha_4^{i_3}} t^{i_2}_{\alpha_3^{i_2}} S^{\alpha_4^{i_3}}_{\alpha_2^{i_1i_2}} +$$

$$\frac{1}{4}\times A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} g^{i_1i_2}_{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}} + \frac{1}{2}\times A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} g^{i_1\alpha_3^{i_2}}_{i_3i_4} t^{i_3}_{\alpha_4^{i_3}} t^{i_4}_{\alpha_5^{i_4}} t^{i_2}_{\alpha_3^{i_2}} S^{\alpha_4^{i_3}}_{\alpha_1^{i_1i_2}} S^{\alpha_5^{i_4}}_{\alpha_2^{i_1i_2}} + \frac{1}{2}\times A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} f^{\alpha_3^{i_1}}_{i_3} t^{i_1}_{\alpha_3^{i_1}} t^{i_2i_3}_{\alpha_4^{i_2i_3}\alpha_5^{i_2i_3}} S^{\alpha_4^{i_2i_3}}_{\alpha_1^{i_1i_2}} S^{\alpha_5^{i_2i_3}}_{\alpha_2^{i_1i_2}} +$$

$$\frac{1}{4}\times A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} g^{i_1i_2}_{i_3i_4} t^{i_3}_{\alpha_3^{i_3}} t^{i_4}_{\alpha_4^{i_4}} S^{\alpha_3^{i_3}}_{\alpha_1^{i_1i_2}} S^{\alpha_4^{i_4}}_{\alpha_2^{i_1i_2}} + -\frac{1}{2}\times A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} g^{\alpha_3^{i_1}\alpha_4^{i_3}}_{i_3i_4} t^{i_1}_{\alpha_3^{i_1}} t^{i_3}_{\alpha_4^{i_3}} t^{i_2i_4}_{\alpha_5^{i_2i_4}\alpha_6^{i_2i_4}} S^{\alpha_5^{i_2i_4}}_{\alpha_1^{i_1i_2}} S^{\alpha_6^{i_2i_4}}_{\alpha_2^{i_1i_2}} + \frac{1}{2}\times A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} g^{\alpha_3^{i_1i_3}\alpha_4^{i_2i_4}}_{i_3i_4} t^{i_1i_3}_{\alpha_3^{i_1i_3}\alpha_5^{i_1i_3}} t^{i_2i_4}_{\alpha_4^{i_2i_4}\alpha_6^{i_2i_4}} S^{\alpha_5^{i_1i_3}}_{\alpha_1^{i_1i_2}} S^{\alpha_6^{i_2i_4}}_{\alpha_2^{i_1i_2}} +$$

$$\frac{1}{2}\times A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} g^{i_1\alpha_3^{i_2}}_{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}} t^{i_2}_{\alpha_3^{i_2}} + -\frac{1}{2}\times A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} g^{i_1\alpha_3^{i_3}}_{i_3i_4} t^{i_3}_{\alpha_3^{i_3}} t^{i_2i_4}_{\alpha_4^{i_2i_4}\alpha_5^{i_2i_4}} S^{\alpha_4^{i_2i_4}}_{\alpha_1^{i_1i_2}} S^{\alpha_5^{i_2i_4}}_{\alpha_2^{i_1i_2}} + -\frac{1}{4}\times A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} g^{\alpha_3^{i_3i_4}\alpha_4^{i_1i_2}}_{i_3i_4} t^{i_3i_4}_{\alpha_3^{i_3i_4}\alpha_5^{i_3i_4}} t^{i_1i_2}_{\alpha_1^{i_1i_2}\alpha_4^{i_1i_2}} S^{\alpha_5^{i_3i_4}}_{\alpha_2^{i_1i_2}} +$$

$$\frac{1}{4}\times A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} g^{\alpha_3^{i_1i_2}\alpha_4^{i_1i_2}}_{i_3\alpha_1^{i_1i_2}} t^{i_3}_{\alpha_5^{i_3}} t^{i_1i_2}_{\alpha_3^{i_1i_2}\alpha_4^{i_1i_2}} S^{\alpha_5^{i_3}}_{\alpha_2^{i_1i_2}} + -1\times A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} g^{\alpha_3^{i_1}\alpha_4^{i_2i_4}}_{i_3i_4} t^{i_3}_{\alpha_5^{i_3}} t^{i_1}_{\alpha_3^{i_1}} t^{i_2i_4}_{\alpha_4^{i_2i_4}\alpha_6^{i_2i_4}} S^{\alpha_6^{i_2i_4}}_{\alpha_1^{i_1i_2}} S^{\alpha_5^{i_3}}_{\alpha_2^{i_1i_2}} + \frac{1}{4}\times A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} g^{\alpha_3^{i_1}\alpha_4^{i_2}}_{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}} t^{i_1}_{\alpha_3^{i_1}} t^{i_2}_{\alpha_4^{i_2}} +$$

$$\frac{1}{4}\times A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} g^{i_1\alpha_3^{i_2}}_{i_3i_4} t^{i_2}_{\alpha_3^{i_2}} t^{i_3i_4}_{\alpha_4^{i_3i_4}\alpha_5^{i_3i_4}} S^{\alpha_4^{i_3i_4}}_{\alpha_1^{i_1i_2}} S^{\alpha_5^{i_3i_4}}_{\alpha_2^{i_1i_2}} + A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} g^{\alpha_3^{i_1}\alpha_4^{i_2i_3}}_{i_3\alpha_1^{i_1i_2}} t^{i_1}_{\alpha_3^{i_1}} t^{i_2i_3}_{\alpha_4^{i_2i_3}\alpha_5^{i_2i_3}} S^{\alpha_5^{i_2i_3}}_{\alpha_2^{i_1i_2}} + \frac{1}{2}\times A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} g^{i_1i_2}_{i_3\alpha_1^{i_1i_2}} t^{i_3}_{\alpha_3^{i_3}} S^{\alpha_3^{i_3}}_{\alpha_2^{i_1i_2}} +$$

$$\frac{1}{2}\times A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} g^{\alpha_3^{i_1}\alpha_4^{i_2}}_{i_3\alpha_1^{i_1i_2}} t^{i_3}_{\alpha_5^{i_3}} t^{i_1}_{\alpha_3^{i_1}} t^{i_2}_{\alpha_4^{i_2}} S^{\alpha_5^{i_3}}_{\alpha_2^{i_1i_2}} + \frac{1}{8}\times A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} g^{i_1i_2}_{i_3i_4} t^{i_3i_4}_{\alpha_3^{i_3i_4}\alpha_4^{i_3i_4}} S^{\alpha_3^{i_3i_4}}_{\alpha_1^{i_1i_2}} S^{\alpha_4^{i_3i_4}}_{\alpha_2^{i_1i_2}} + -\frac{1}{4}\times A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} g^{\alpha_3^{i_2i_4}\alpha_4^{i_2i_4}}_{i_3i_4} t^{i_1i_3}_{\alpha_5^{i_1i_3}\alpha_6^{i_1i_3}} t^{i_2i_4}_{\alpha_3^{i_2i_4}\alpha_4^{i_2i_4}} S^{\alpha_5^{i_1i_3}}_{\alpha_1^{i_1i_2}} S^{\alpha_6^{i_1i_3}}_{\alpha_2^{i_1i_2}} +$$

$$\frac{1}{8}\times A^{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}}_{i_1i_2} g^{\alpha_3^{i_1i_2}\alpha_4^{i_1i_2}}_{\alpha_1^{i_1i_2}\alpha_2^{i_1i_2}} t^{i_1i_2}_{\alpha_3^{i_1i_2}\alpha_4^{i_1i_2}}$$

Talk Synopsis

need to raise the level of abstraction to enable many-body QM

symbolic techniques are a component of the needed many-body QM technology stack

particularly needed for supporting tensor compressed/factorized methods (e.g., PNO)

ideas are old, let's learn from the pioneers and make this sustainable

Acknowledgement

- Dr.Andrei Asadchev
- Dr. Nakul Teke (spin-tracing)
- Sam Slattery
- Bimal Gaudel (evaluation)
- Conner Masteran
- Samuel Powell
- Jaden Yon
- Ashawini Thakur
- Kshitij Surjuse
- Ajay Melecamburath

1931347

DE-SC0019374

postdoctoral positions available in methods and software development for electronic structure