Tensor network optimizations

University of Stuttgart
Institute for Theoretical Chemistry

Project type
Internship / B.Sc. / M.Sc.

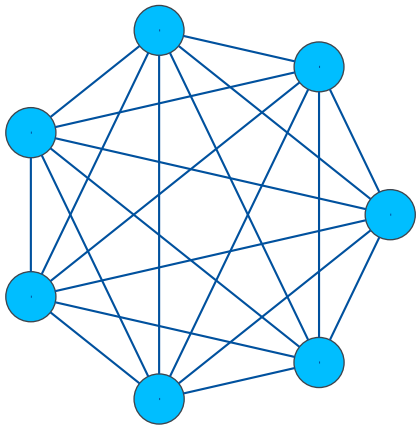
Supervisor:
Robert Adam*

Examiner:
Prof. Dr. Andreas Köhn

*adam@theochem.uni-stuttgart.de

Keywords: Symbolic Algebra ◇ Tensor Networks ◇ Optimization

Topic



Your topic will be the investigation and/or development of algorithms for the symbolic rewriting (simplification) of equations with the intent to minimize the required floating point operations in the numeric evaluation of these equations. This is complicated by the fact that these equations contain products of tensor elements (so-called tensor *networks*) as they arise from *tensor contractions*.

This problem can either be tackled as a general (very high-dimensional) optimization problem for which viable algorithms include

- Heuristic 2-step algorithm (Hartono *et al.* [1])
- Genetic algorithms (Engels-Putzka & Hanrath [2])

Alternatively, the problem can be recast as a shortest path problem on an exponentially large graph. In this setting the following algorithms or related problems could be relevant:

- Dijkstra's / A* algorithm
- Bellman-Ford algorithm
- Combinatorial optimization, e.g. the Generalized Traveling Salesman Problem (Pop *et al.* [3])

Finally, it might be possible to train and apply some form of machine learning to this optimization problem, either in the graph formulation (Peng *et al.* [4]) or in the original setting of the problem.

Prerequisite for this topic is that you are *proficient* in at least one programming language (e.g. Python) that you will use to implement and explore the different optimization strategies.

Motivation

Now to the question how the above relates to (Theoretical) Chemistry. The implementation of quantum chemical methods can often be boiled down to implementing a series of tensor contractions (generalization of matrix multiplication to higher dimensions). The cost of performing a simulation using such a method is then often dominated by how expensive it is to evaluate these tensor contractions on a computer.

The numeric cost of computing the result of tensor contractions is typically measured in so-called floating-point operations (FLOPs). Therefore, a significant part of making a given method's implementation

efficient consists in trying to minimize the amount of FLOPs needed in the first place. To this end, there exist two approaches:

- Minimize the FLOP count for the individual contractions
- Identifying intermediates that are used in multiple contractions so that they have to be computed only once

If $\mathbf{v}$ is a vector and $\mathbf{M}$ is a matrix, then the first point can be illustrated with the example

$$\mathbf{M}\mathbf{v}\mathbf{v}^T = (\mathbf{M}\mathbf{v})\mathbf{v}^T = \mathbf{M}(\mathbf{v}\mathbf{v}^T)$$

Due to associativity of matrix multiplications, all of the above expressions are equivalent in the sense that they all compute the same result. However, if we first compute the matrix-vector product (which results in a vector), we have to do a subsequent vector-vector multiplication. On the contrary, if we first perform the vector-vector product (leading to a scalar), we are left only with multiplying the matrix by a scalar. Since the latter requires less operations than the matrix-vector multiplication, choosing the last option allows to compute the same result with fewer FLOPs. In the case of tensors, the FLOP savings can easily be multiple orders of magnitude.

The identification of intermediates is the classic process of factoring out common factors across a sum of products. For instance in

$$ABC + BCD + CDE = C(AB + BD + DE) = BC(A + D) + CDE$$

it is clear that by factoring out, certain products have to be computed only once, instead of multiple times. However, the choice of which factors to factor out is not always unique (C vs. BC in the above example) and choosing to factor out in a given way may prevent the use of the optimal order in which a given individual product can be evaluated. For instance, if the FLOP-minimization for the product ABC yields $(AC)B$, then neither BC nor C can be factored out in the way shown above. In fact, forgoing the possibility of factoring out a subexpression may still lead to an overall minimal FLOP count, simply because the FLOP savings for any particular product can be so tremendous.

This shows that the two approaches to FLOP minimization are in fact tightly coupled to each other and in order to have any chance at finding an optimal combination of factored-out subexpressions and optimized evaluation orders for each product, both strategies have to be combined into a single optimization algorithm.

The big challenge with such a global optimization algorithm is that the space of possibilities which it (in principle) has to explore is extremely large ($> 10^{100}$). Therefore, approximate algorithms have to be used, which only find reasonable good solutions instead of having a guarantee of finding the best solution possible.

References

1. Hartono, A. *et al. Identifying Cost-Effective Common Subexpressions to Reduce Operation Count in Tensor Contraction Evaluations* en. in *Computational Science – ICCS 2006* (eds Alexandrov, V. N., van Albada, G. D., Sloot, P. M. A. & Dongarra, J.) (Springer, Berlin, Heidelberg, 2006), 267–275. doi:10.1007/11758501_39.
2. Engels-Putzka, A. & Hanrath, M. A fully simultaneously optimizing genetic approach to the highly excited coupled-cluster factorization problem. *J. Chem. Phys.* **134,** 124106. doi:10.1063/1.3561739 (2011).
3. Pop, P. C., Cosma, O., Sabo, C. & Sitar, C. P. A comprehensive survey on the generalized traveling salesman problem. *Eur. J. Oper. Res.* **314,** 819–835. doi:10.1016/j.ejor.2023.07.022 (2024).
4. Peng, Y., Choi, B. & Xu, J. Graph Learning for Combinatorial Optimization: A Survey of State-of-the-Art. en. *Data Sci. Eng.* **6,** 119–141. doi:10.1007/s41019-021-00155-3 (2021).