Fast Summation Algorithms and Tensor Network Methods for Scientific Applications

Xuanzhao Gao

2025-1-10

Hong Kong University of Science and Technology

Outline

A Fast Spectral Sum-of-Gaussians Method for Coulomb Interaction

Joint work with Shidong Jiang, Jiuyang Liang, Zhenli Xu, and Qi Zhou

arXiv:2412.04595

Quasi-2D charged systems

Quasi-2D systems (Mazars, 2011) are at the macroscopic scale in xy, but microscopic in z, so that are always modeled as doubly periodic in numerical simulations.

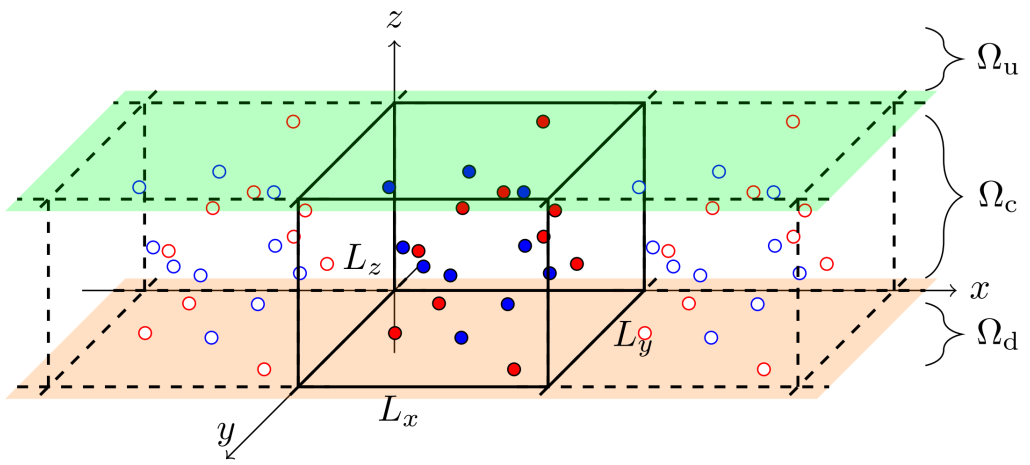


Figure 1: Illustration of a quasi-2D charged system.

Coulomb interaction plays a key role in nature, leading to effect such as ion transportation and self-assembly (Barros & Luijten, 2014).

However, the Coulomb interaction decays as r^{-1} in 3D, so that it is long ranged and singular at $r = 0$, which make such simulation computationally expensive.

Background

Algorithms for Q2D charged systems

Methods have been developed to accelerate the Coulomb interaction in Q2D systems.

The very first method is the Ewald2D (Parry, 1975) method based on the Ewald splitting of the Coulomb kernel. It is accurate but with $O(N^2)$ complexity.

To reduce the complexity, most methods rely on the following three strategies:

- **Fourier spectral method** (Lindbo & Tornberg, 2011; 2012; Nestler et al., 2015; Shamshirgar & Tornberg, 2017; Shamshirgar et al., 2021; Maxian et al., 2021): based on Ewald splitting and fast Fourier transform (FFT), with $O(N \log N)$ complexity.
- **Fast multipole methods** (Greengard, 1987; Greengard & Rokhlin, 1987; Berman & Greengard, 1994; Yan & Shelley, 2018; Liang et al., 2020): accelerated by hierarchical low-rank compression, adaptive and with $O(N)$ complexity.
- **Random batch Ewald** (Jin et al., 2021; Liang et al., 2022; Gan et al., 2024a; Gan et al., 2024b): based on Ewald splitting and random batch sampling, stochastic and with $O(N)$ complexity, efficient parallelization.

Background

Algorithms for Q2D charged systems

For doubly periodic systems, one major challenge is the large prefactor in $O(N)$ or $O(N \log N)$ compared to 3D-PBC solvers (Mazars, 2011), especially when the system is strongly confined in the z direction, i.e., $L_z \ll L_x, L_y$.

- For the FFT based methods, **huge zero-padding** is required.
- For the FMM based methods, **more near field contributions** is needed.

Some recently developed methods offer potential solutions to this challenge, including:

- Anisotropic truncation kernel method (Greengard et al., 2018)
- Periodic FMM (Pei, Askham, Greengard & Jiang, 2023)
- Dual-space multilevel kernel-splitting method (Jiang & Greengard, 2024)

However, these methods have not yet been extended to handle quasi-2D systems.

The algorithm

The sum-of-Gaussians approximation

In our work, we use the bilateral series approximation (Beylkin & Monzón, 2010) of the Coulomb kernel, where

$$\frac{1}{r} \approx \frac{2\log b}{\sqrt{2\pi\sigma^2}} \sum_{l=-\infty}^{\infty} \frac{1}{b^l} e^{-\frac{r^2}{\left(\sqrt{2}b^l\sigma\right)^2}}, \text{ with } \mathcal{E}_r < 2\sqrt{2}e^{-\frac{\pi^2}{2\log b}}, r > 0$$

Based on the u-series decomposition (Predescu et al., 2020), we further split the potential into three parts:

$$\frac{1}{r} \approx \underbrace{\left(\frac{1}{r} - \sum_{l=0}^{M} w_l e^{-\frac{r^2}{s_l^2}}\right) \mathbb{1}_{r<r_c}}_{\text{near-field}} + \underbrace{\sum_{l=0}^{m} w_l e^{-\frac{r^2}{s_l^2}}}_{\text{mid-range}} + \underbrace{\sum_{l=m+1}^{M} w_l e^{-\frac{r^2}{s_l^2}}}_{\text{long-range}}$$

The weight of the narrowest Gaussian is modified to be

$$w_0 = \omega \frac{2\log b}{\sqrt{2\pi\sigma^2}}$$

to enforce the C^0 and C^1 continuity of the near-field potential at $r = r_c$, which is important for MD simulations (Shamshirgar et al., 2019).

The algorithm

Splitting the far-field potential

Selecting m so that $s_m < \eta L_z < s_{m+1}$, where η is $O(1)$ constant.

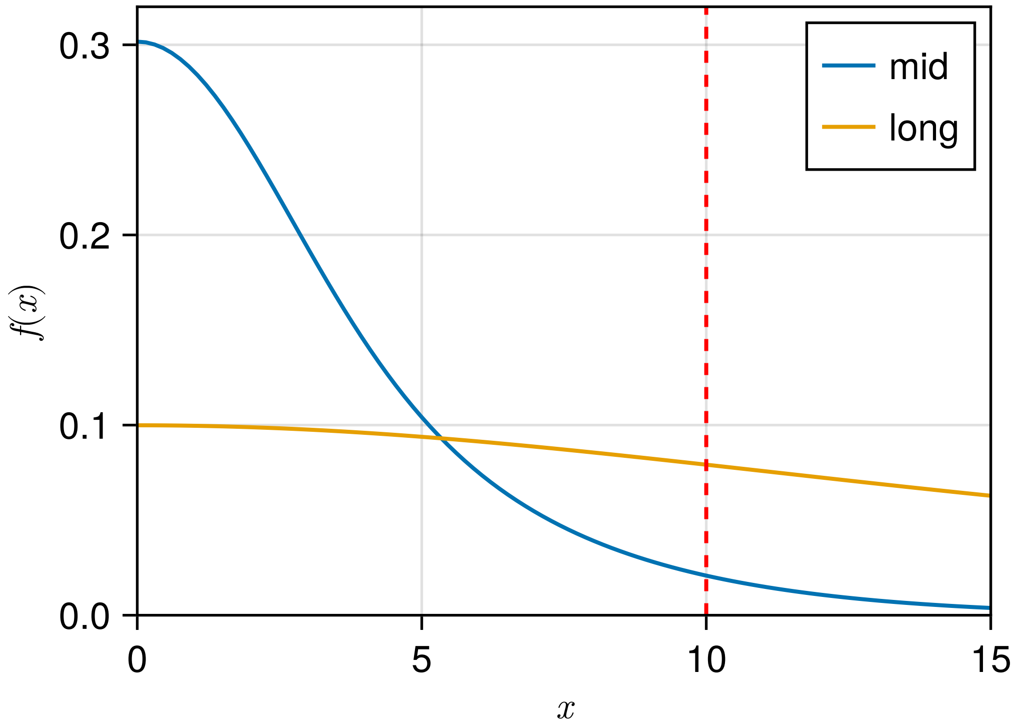


Figure 2: Mid-range part and long-range part of the potential, $L_z = 10$, $\eta \approx 0.6$.

The algorithm

Mid-range potential

$$\Phi^{l}_{\mathrm{mid}}(\vec{r}) = \sum_{\vec{n}} \sum_{j=1}^{N} q_j w_l e^{-\frac{\left(\vec{r}-\vec{r}_j+\vec{n}\circ\vec{L}\right)^2}{s_l^2}}, \quad s_l < \eta L_z$$

The mid-range potential is computed by a standard Fourier spectral solver[1] with **little zero padding** ($\lambda_z < 2$ for double precision). **No need of the kernel truncation** in the free direction due to the smoothness and separability of the Gaussian.

$$\begin{tikzcd}
\boxed{\text{Gridding}} \arrow[r] & \boxed{\text{FFT}} \arrow[r] & \boxed{\text{Scaling}} \arrow[r] & \boxed{\text{IFFT}} \arrow[r] & \boxed{\text{Gathering}}
\end{tikzcd}$$

The process is similar to type-1 and type-2 NUFFT in 3D (Barnett et al., 2019). **No upsampling** is needed in gridding and gathering steps since the Fourier transform of the Gaussian decays quickly and it compensates the loss of accuracy in calculating the Fourier transform of the data.

[1] https://github.com/HPMolSim/ChebParticleMesh.jl

The algorithm

Long-range potential

$$\Phi^{l}_{\text{long}}(\vec{r}) = \sum_{\vec{n}} \sum_{j=1}^{N} q_j w_l e^{-\frac{\left(\vec{r}-\vec{r}_j+\vec{n}\circ\vec{L}\right)^2}{s_l^2}}, \quad s_l > \eta L_z$$

The long-range potential is computed by a Fourier-Chebyshev solver.

The extremely smooth long-range Gaussians are interpolated on the Chebyshev proxy points in z, similar to that of the periodic FMM (Pei, Askham, Greengard & Jiang, 2023), and only **$O(1)$ number of Chebyshev points are required**.

Then 2D NUFFT like steps can be used to evaluate the potential on a tensor-product grid, where upsampling is also not needed.

$$\begin{tikzcd}\boxed{\text{Chebyshev interpolation}} \arrow[r] & \boxed{\text{2D NUFFT like steps}} \arrow[r] & \boxed{\text{Evaluating polynomial}}\end{tikzcd}$$

In cubic systems, $L_x \sim L_y \sim L_z$, $O(1)$ Fourier modes in xy and $O(1)$ Chebyshev points in z, no need for NUFFT.

In strongly confined systems, $s_0 > \eta L_z$, only long range potential is needed.

The algorithm

Complexity

Using DFT for long-range potential, the complexity is

$$O(\underbrace{4\pi r_c^3 \rho_r N}_{\text{near-field}} + \underbrace{\mathcal{P}_x \mathcal{P}_y \mathcal{P}_z N + \frac{\lambda_z \left(1 + \frac{\delta}{L_z}\right)}{r_c^3 \rho_r} N \log N}_{\text{mid-range}} + \underbrace{\frac{P L_x L_y}{\eta^2 L_z^2} N}_{\text{long-range}})$$

where $\mathcal{P}_x, \mathcal{P}_y, \mathcal{P}_z$ are the window supports, λ_z is the padding ratio, δ is the extended length of the box in the free direction to accommodate the support of the window function, P is the number of Chebyshev points. By taking $r_c \sim O(1)$ and assume $L_z \sim O\left(\sqrt{L_x L_y}\right)$, the complexity is $O(N \log N)$.

Using 2D-NUFFT for long-range potential, the complexity is

$$O(\underbrace{4\pi r_c^3 \rho_r N}_{\text{near-field}} + \underbrace{\mathcal{P}_x \mathcal{P}_y \mathcal{P}_z N + \frac{\lambda_z \left(1 + \frac{\delta}{L_z}\right)}{r_c^3 \rho_r} N \log N}_{\text{mid-range}} + \underbrace{\mathcal{P}_x \mathcal{P}_y P N + \frac{P}{\rho_r L_z^3 \eta^2} N \log N}_{\text{long-range}})$$

which is needed when $L_z \ll L_x, L_y$, the total complexity is also $O(N \log N)$.

Numerical results

The method[1] is benchmarked on the following systems:

- Cubic systems with fixed aspect ratio equals to 1.
- Strongly confined systems with fixed L_z, aspect ratio up to $10^{3.5}$

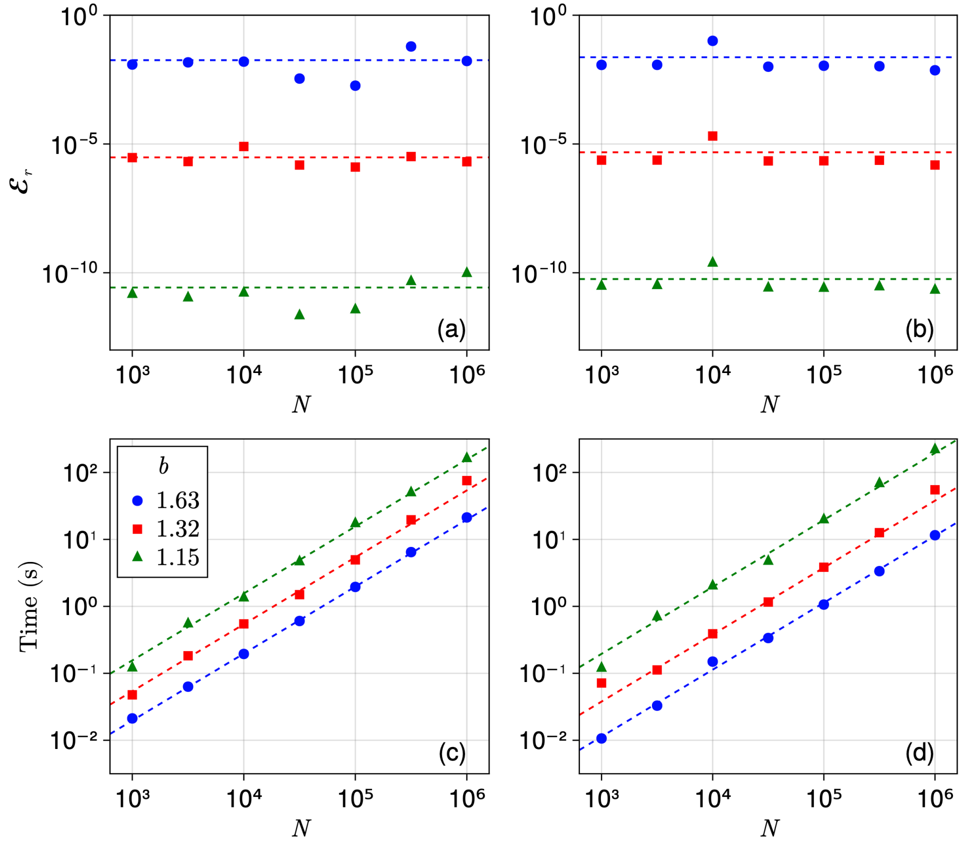


Figure 3: Error and time cost for the SOG method in the (a,c) cubic and (b,d) strongly confined systems.

[1] https://github.com/HPMolSim/FastSpecSoG.jl

Summary

A fast and accurate solver for Q2D charged systems is developed based on the sum-of-Gaussian approximation of the Coulomb kernel and the kernel splitting technique. The method can be regarded as a 2-level DMK method (Jiang & Greengard, 2024).

The solver **addresses challenges arising from singularities and strong confinement**, and has the following advantages:

- spectrally accurate with rigorous error analysis (Liang et al., 2023)
- need little/no zero-padding for systems that are confined in a rectangular box of high aspect ratio
- no need for upsampling in the gridding and gathering steps
- all calculations are carried out in the fundamental cell itself
- easy to be implemented and parallelized for large-scale MD simulations

Currently, the major shortcoming of this method is its non-adaptive nature, and has a complexity of $O(N \log N)$ rather than $O(N)$.

Automated Discovery of the Optimal Branching Rules

Joint work with Yi-Jia Wang, Pan Zhang, and Jin-Guo Liu

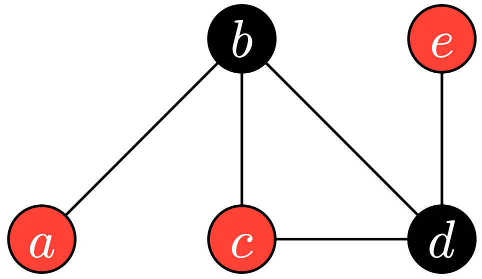
arXiv:2412.07685

Background

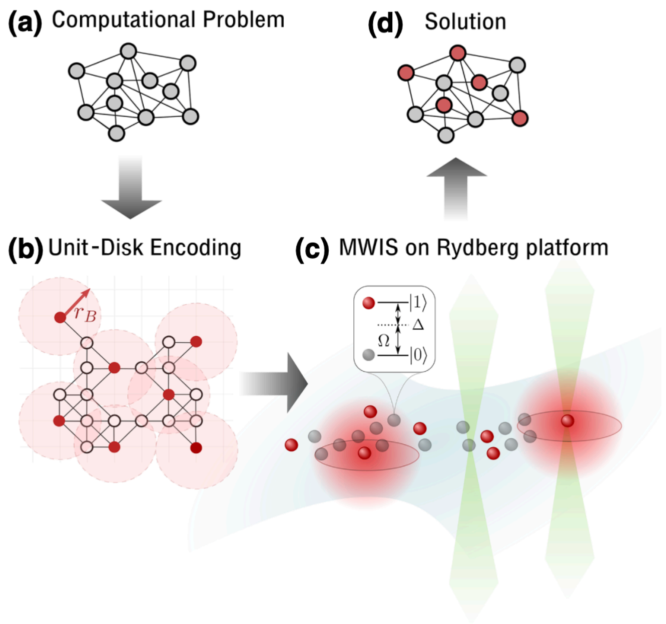
The maximum independent set (MIS) problem

One of the first batch of 21 NP-hard problems proved by (Karp, 1972).

An independent set is a set of vertices in a graph, no two of which are adjacent.



$G = (V, E)$, MIS = $\{a, c, e\}$, size $\alpha(G) = 3$



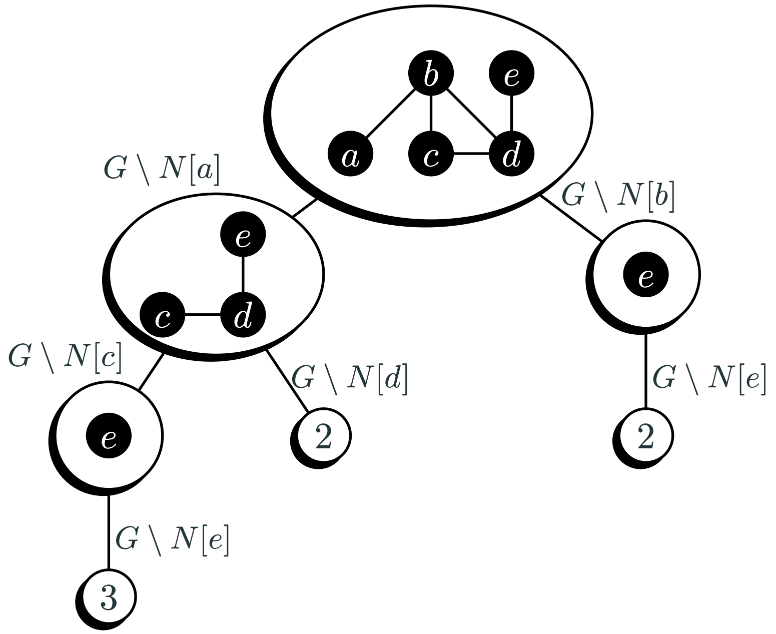
MIS problem has an exponential large solution space, since it is NP-hard, no polynomial-time algorithm is known to solve it exactly. It caught much attention since the Rydberg atom systems realize spin models that naturally MIS problem (Nguyen et al., 2023).

Background

Branching algorithm

The branching algorithm (Fomin & Kaski, 2013) explores the solution space using a tree-like structure, relying on **predesigned rules**.

Complexity of a branching algorithm is always described as $O(\gamma^n)$ where γ is the branching factor and n is the size of the problem.

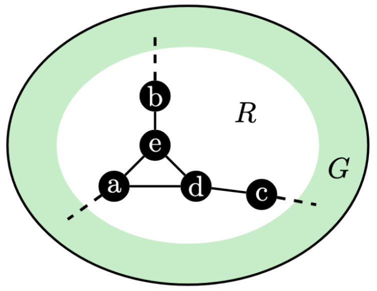


Year	Running times	References	Notes
1977	$O^*(1.2600^n)$	(Tarjan & Trojanowski, 1977)	
1986	$O^*(1.2346^n)$	(Jian, 1986)	
1986	$O^*(1.2109^n)$	(Robson, 1986)	
1999	$O^*(1.0823^m)$	(Beigel, 1999)	num of edges
2001	$O^*(1.1893^n)$	(Robson, 2001)	
2003	$O^*(1.1254^n)$ for 3-MIS	(Chen et al., 2003)	
2005	$O^*(1.1034^n)$ for 3-MIS	(Xiao et al., 2005)	
2006	$O^*(1.2210^n)$	(Fomin et al., 2006)	
2006	$O^*(1.1225^n)$ for 3-MIS	(Fomin & Høie, 2006)	
2006	$O^*(1.1120^n)$ for 3-MIS	(Fürer, 2006)	
2006	$O^*(1.1034^n)$ for 3-MIS	(Razgon, 2006)	
2008	$O^*(1.0977^n)$ for 3-MIS	(Bourgeois et al., 2008)	
2009	$O^*(1.0919^n)$ for 3-MIS	(Xiao, 2009)	
2009	$O^*(1.2132^n)$	(Kneis et al., 2009)	
2013	$O^*(1.0836^n)$ for 3-MIS	(Xiao & Nagamochi, 2013)	SOTA
2016	$O^*(1.2210^n)$	(Akiba & Iwata, 2016)	PACE winner
2017	$O^*(1.1996^n)$	(Xiao & Nagamochi, 2017)	SOTA

The algorithm

Tensor networks for the MIS problem

Tensor networks can be used to extract the local information of the sub-graph (Gao et al., 2024a; Liu et al., 2023).



$$\xrightarrow{\hspace{3cm}}$$

Contract the local tensor network

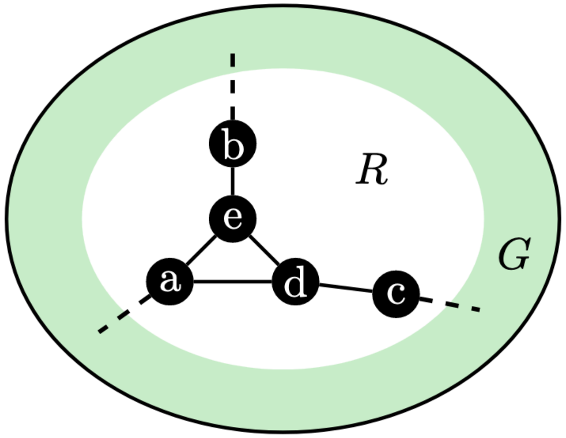
Boundary configuration: s_{abc}	Possible assignments: S_{abcde}
000	00001, 00010
001	00101
010	01010
111	11100

However, a pure tensor network method does not work well for non-geometric graphs. Its complexity on 3-regular graphs is about $O(1.1224^n)$, far from the SOTA ($O^*(1.0836^n)$).

The algorithm

The optimal branching algorithm

We use the tensor network to extract the local information, and then automatically search the optimal branching rules[1] (Gao et al., 2024b).



$\boldsymbol{s}_{abcde}$	clauses in $\mathcal{D}$
$S_{000} = \{00001, 00010\}$	$\neg a \wedge \neg b \wedge \neg d \wedge e$
$S_{001} = \{00101\}$	
$S_{010} = \{01010\}$	$\neg a \wedge b \wedge \neg c \wedge d \wedge \neg e$
$S_{111} = \{11100\}$	$a \wedge b \wedge c \wedge \neg d \wedge \neg e$

Naive branching

4 branches, each fix 5 variables

$$\gamma^n = 4 \times \gamma^{n-5}$$

$$\gamma \approx 1.3195$$

Optimal branching

3 branches, fix [4, 5, 5] variables

$$\gamma^n = \gamma^{n-4} + 2 \times \gamma^{n-5}$$

$$\gamma \approx 1.2671$$

Key point: Find the **correct pattern**!

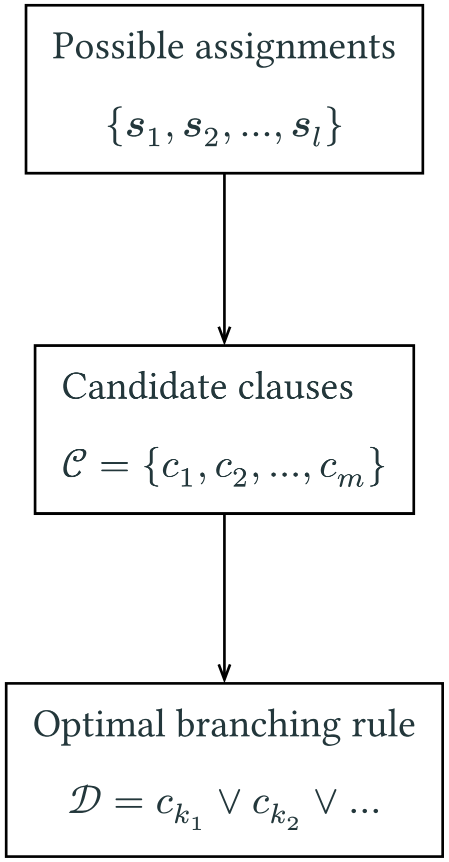
[1] https://github.com/OptimalBranching/OptimalBranching.jl

The algorithm

Finding the optimal branching rule

Bruteforce search? $\to$ Given l assignments, possible rules: $O\left(2^{2^l}\right)$.

The process of finding the optimal branching rules is as the following:



Possible assignments
$\{s_1, s_2, \ldots, s_l\}$
Candidate clauses
$\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$
Optimal branching rule
$\mathcal{D} = c_{k_1} \lor c_{k_2} \lor \ldots$

The first step is very direct forward, we generate all possible combination of the assignments (the candidate clauses).

The second step is formulated as a **set covering problem**, which can be solved by mixed integer programming solvers (Achterberg, 2009).

$$\min_{\gamma, \boldsymbol{x}} \gamma \quad \text{s.t.} \quad \sum_{i=1}^{m} \gamma^{-\Delta\rho(c_i)} x_i = 1,$$

$$\bigcup_{\substack{i=1,\ldots,m \\ x_i=1}} J_i = \{1, 2, \ldots, l\}, \ \to \text{valid branching rule}$$

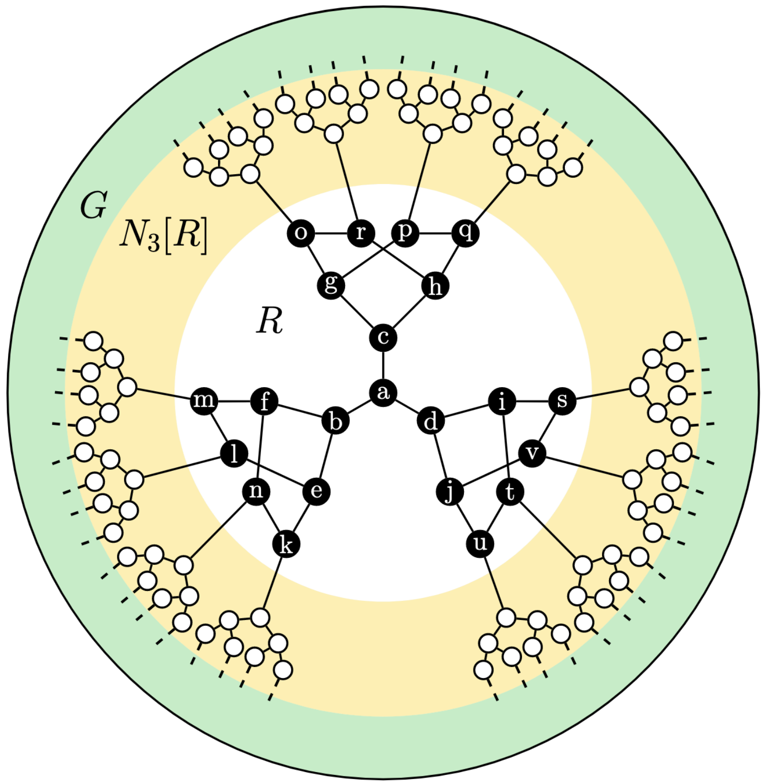
$$x_i \in \{0, 1\} \ \to \text{a clause is selected or not}$$

where $\Delta\rho(c_i)$ is the size reduced by the clause c_i of the problem.

Numerical results

A bottleneck case

A bottle neck case has been reported in Xiao's work (Xiao & Nagamochi, 2013), with a branching factor of 1.0836.



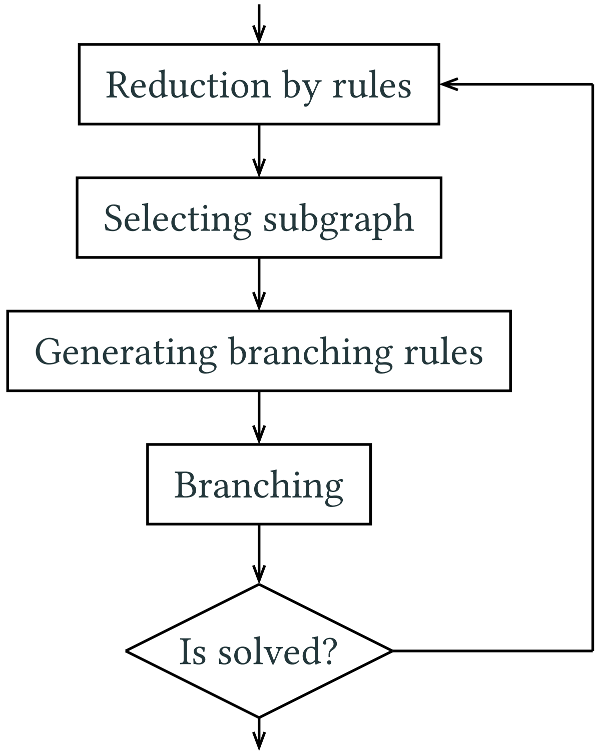
- 71 possible assignments, 15782 candidate clauses.
- 4 branches, size of the problem reduced by branches: $[10, 16, 26, 26]$, with

$$\gamma = \mathbf{1.0817} < \mathbf{1.0836}$$

which indicates our method can find better branching rules than the predesigned rules.

Numerical results

Benchmark on random graphs



The resulting methods are denoted as **ob** and **ob+xiao** and the average branching factor is shown in the table.

	ob	ob+ xiao	xiao2013	akiba2015	akiba2015+ xiao&packing
3RR	1.0457	**1.0441**	**1.0487**	-	-
ER	1.0011	1.0002	-	1.0044	1.0001
KSG	1.0116	1.0022	-	1.0313	1.0019
Grid	1.0012	1.0009	-	1.0294	1.0007

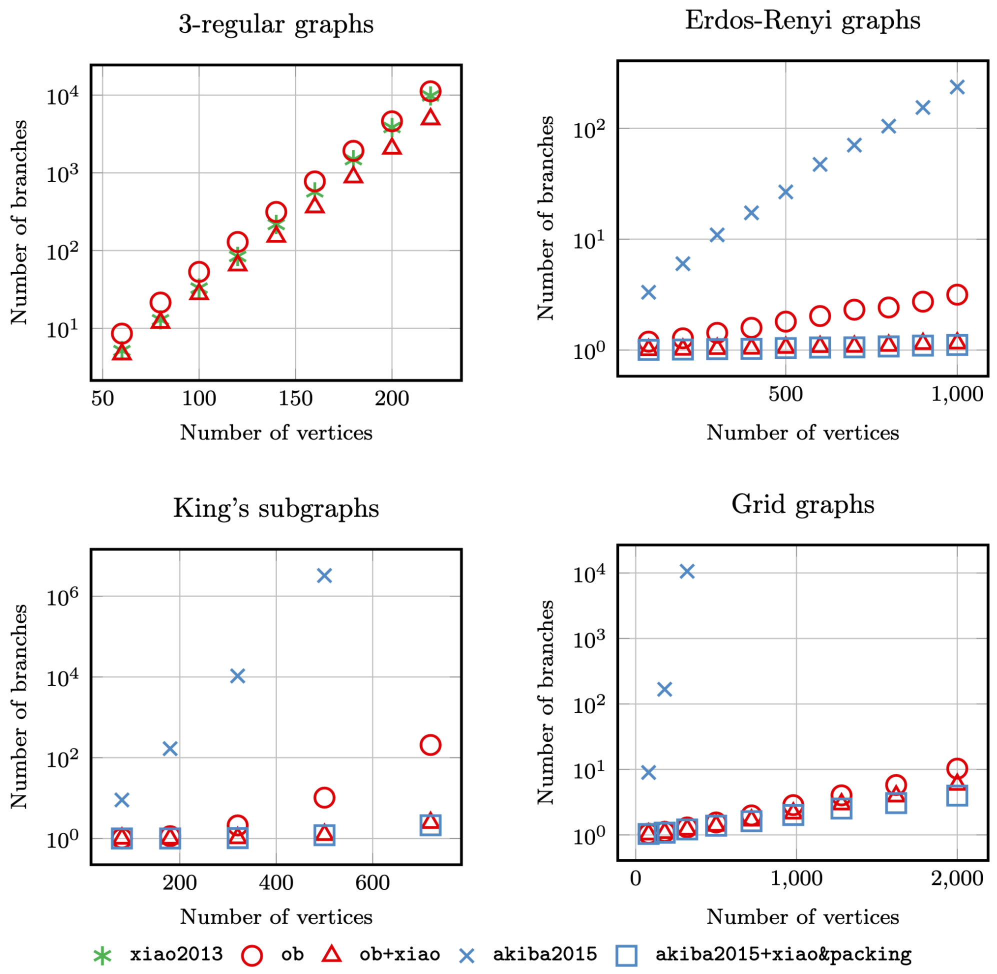
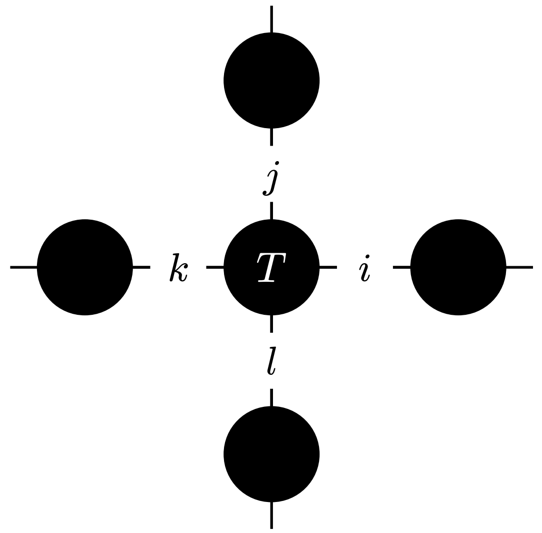


Figure 4: Average number of branches generated by different branching algorithms on 1000 random graphs.

Potential application

Sparse Tensor Networks Contraction

The optimal branching algorithm can be applied to contract the sparse tensor networks.

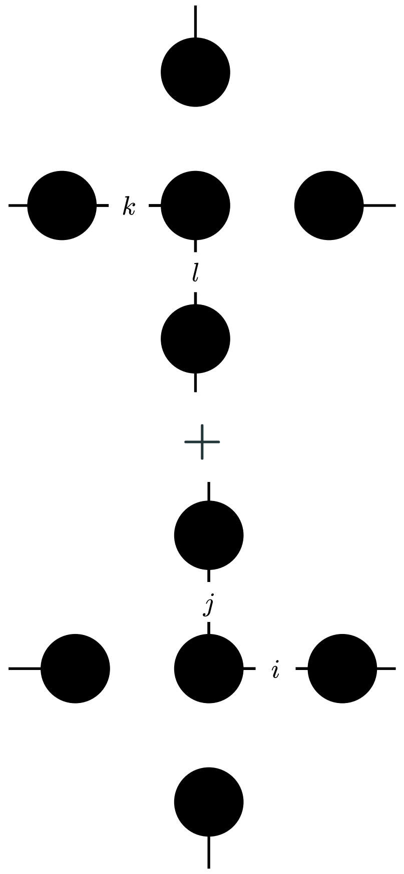


$\rightarrow$

T

$ijkl$	value
1101	0.1
1110	0.2
1100	0.3
1000	0.4
0100	0.5

$\rightarrow$



Such sparsity is common in many problems, including **probabilistic inference**, **combinatorial optimization**, and **quantum circuit simulations** (Markov & Shi, 2008).

Summary

A new method to automatically discover the optimal branching rules is proposed, by combining the tensor network method and the branching algorithm.

Advantages:

- generate the branching rules automatically without human effort
- fully utlize the information of the sub-graph
- the sub-graph can be selected flexibly
- can be applied to different problems, not only the MIS problem

Disadvantages:

- solving the rule can be computationally expensive
- cannot capture the rules need graph rewriting

Summary and Outlook

Publications

Fast Summation Algorithms

- Z. Gan, **X. Gao**, J. Liang, and Z. Xu, Fast algorithm for quasi-2D Coulomb systems. *Journal of Computational Physics* 113733, (2025).
- **X. Gao**, S. Jiang, J. Liang, Z. Xu, and Q. Zhou, A fast spectral sum-of-Gaussians method for electrostatic summation in quasi-2D systems, Arxiv:2412.04595 (2024)
- Z. Gan, **X. Gao**, J. Liang, and Z. Xu, Random batch Ewald method for dielectrically confined Coulomb systems, Arxiv:2405.06333 (2024)
- **X. Gao** and Z. Gan, Broken symmetries in quasi-2D charged systems via negative dielectric confinement, *The Journal of Chemical Physics* 161, (2024)

Tensor Network Algorithms

- **X. Gao**, Y.-J. Wang, P. Zhang, and J.-G. Liu, Automated discovery of branching rules with optimal complexity for the maximum independent set problem, Arxiv:2412.07685 (2024)
- **X. Gao**, X. Li, and J. Liu, Programming guide for solving constraint satisfaction problems with tensor networks, Arxiv:2501.00227 (2024)
- M. Roa-Villescas, **X. Gao**, S. Stuijk, H. Corporaal, and J.-G. Liu, Probabilistic inference in the era of tensor networks and differential programming, *Physical Review Research* 6, 33261 (2024)

Software packages

My packages

- **ChebParticleMesh.jl**[1]: Toolkits for particle mesh methods (type-1 and type-2 NUFFT).
- **CuTropicalGEMM.jl**[2]: Custom GPU kernel for tropical matrix multiplication.
- **TreeWidthSolver.jl**[3]: Solving the treewidth problem (supported by GSoC 2024).
- **FastSpecSoG.jl**[4]: Implementation of the fast spectral SOG method.
- **EwaldSummations.jl**[5]: Various Ewald summation methods with parallelization.
- **OptimalBranching.jl**[6]: Implementation of the optimal branching algorithm.

Contributions to popular packages

- **OMEinsum.jl**[7] (185 stars) and its backend **OMEinsumContractionOrders.jl**[8]: Optimizing the tensor network contraction order and contracting the tensor network.

[1] https://github.com/HPMolSim/ChebParticleMesh.jl

[2] https://github.com/TensorBFS/CuTropicalGEMM.jl

[3] https://github.com/ArrogantGao/TreeWidthSolver.jl

[4] https://github.com/HPMolSim/FastSpecSoG.jl

[5] https://github.com/HPMolSim/EwaldSummations.jl

[6] https://github.com/OptimalBranching/OptimalBranching.jl

[7] https://github.com/under-Peter/OMEinsum.jl

[8] https://github.com/TensorBFS/OMEinsumContractionOrders.jl

Future Research Plans

Fast Summation Algorithms

- Extending our work to fully adaptive case, other kernels and other periodic systems based on the DMK framework
- GPU acceleration for the fast algorithms.

Tensor Network Algorithms

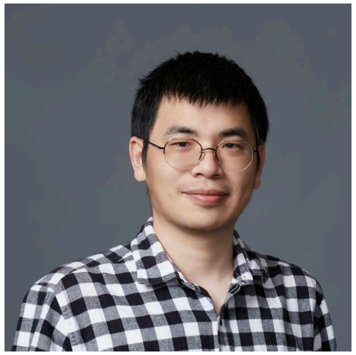
- Branching based sparse tensor network contraction.
- More flexible quantum many-body ansatz.

Acknowledgements

Prof. Zecheng Gan
HKUST(GZ)

Prof. Zhenli Xu
SJTU

Prof. Shidong Jiang
CCM


Prof. Jinguo Liu,
HKUST(GZ)

Prof. Pan Zhang,
ITP, CAS

Also thank Jiuyang Liang (SJTU & CCM), Qi Zhou (SJTU), Martin Roa-Villescas (TU/e), and Yi-Jia Wang (ITP, CAS) for collaboration.

Thank you for your attention!

Appendix

The U-series and its derivative

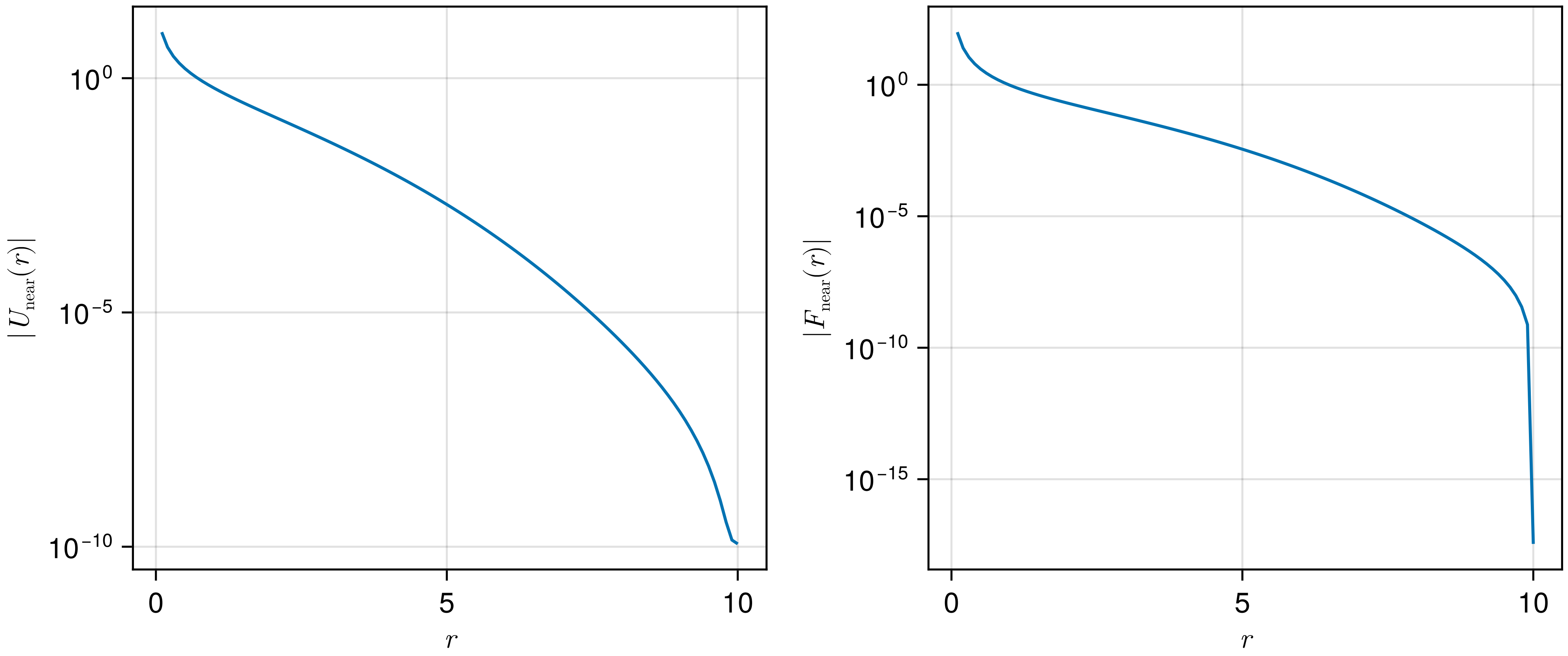


Figure 5: The U-series and its derivative, $r_c = 10.0$.

SOG parameters

b	r_0	ω	Energy		Force	
			Error	M	Error	M
2	1.9892536839080267	0.9944464927622323	3.12×10^{-2}	16	9.93×10^{-3}	11
1.62976708826776469	2.7520026668023417	1.0078069793438068	2.33×10^{-3}	31	6.21×10^{-4}	16
1.48783512395703226	3.7554672283554990	0.9919117057598183	2.29×10^{-4}	46	7.98×10^{-5}	26
1.32070036405934420	4.3914554711638349	1.0018891411481198	1.18×10^{-6}	76	5.76×10^{-7}	41
1.21812525709410644	5.6355288151271085	1.0009014615603334	7.14×10^{-10}	166	5.14×10^{-10}	71
1.14878150173321925	7.2956245490719404	1.0000368348358225	1.30×10^{-15}	271	1.98×10^{-14}	116

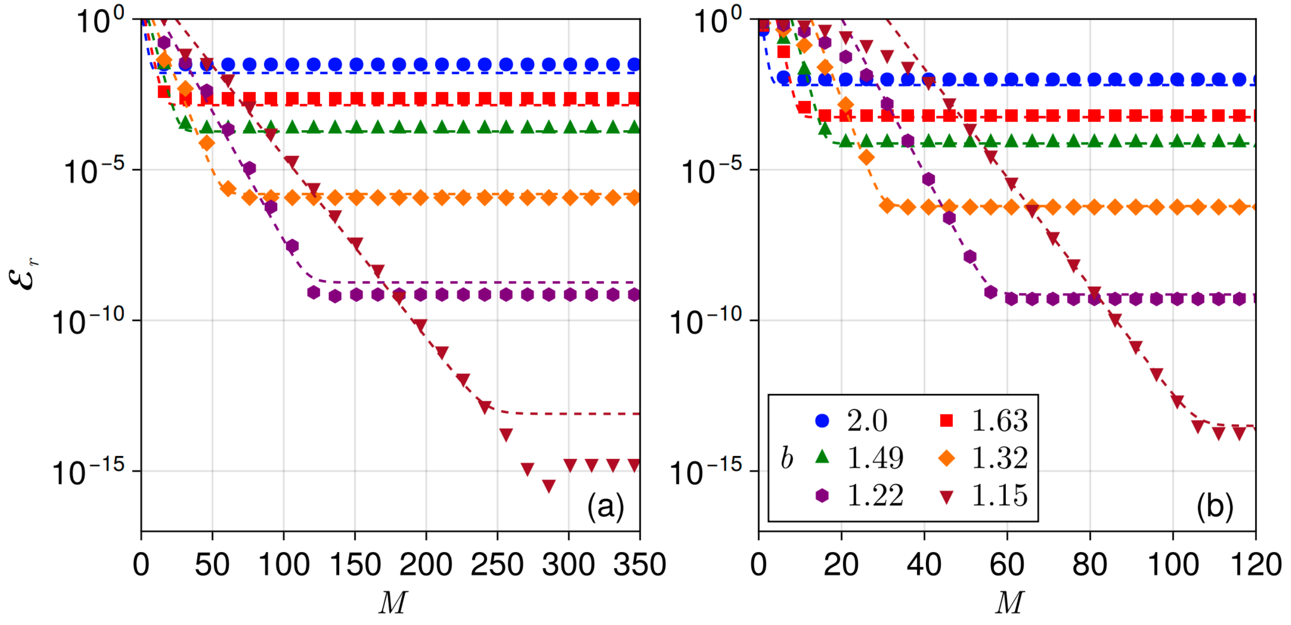


Figure 6: U-series parameters and the error

Window function

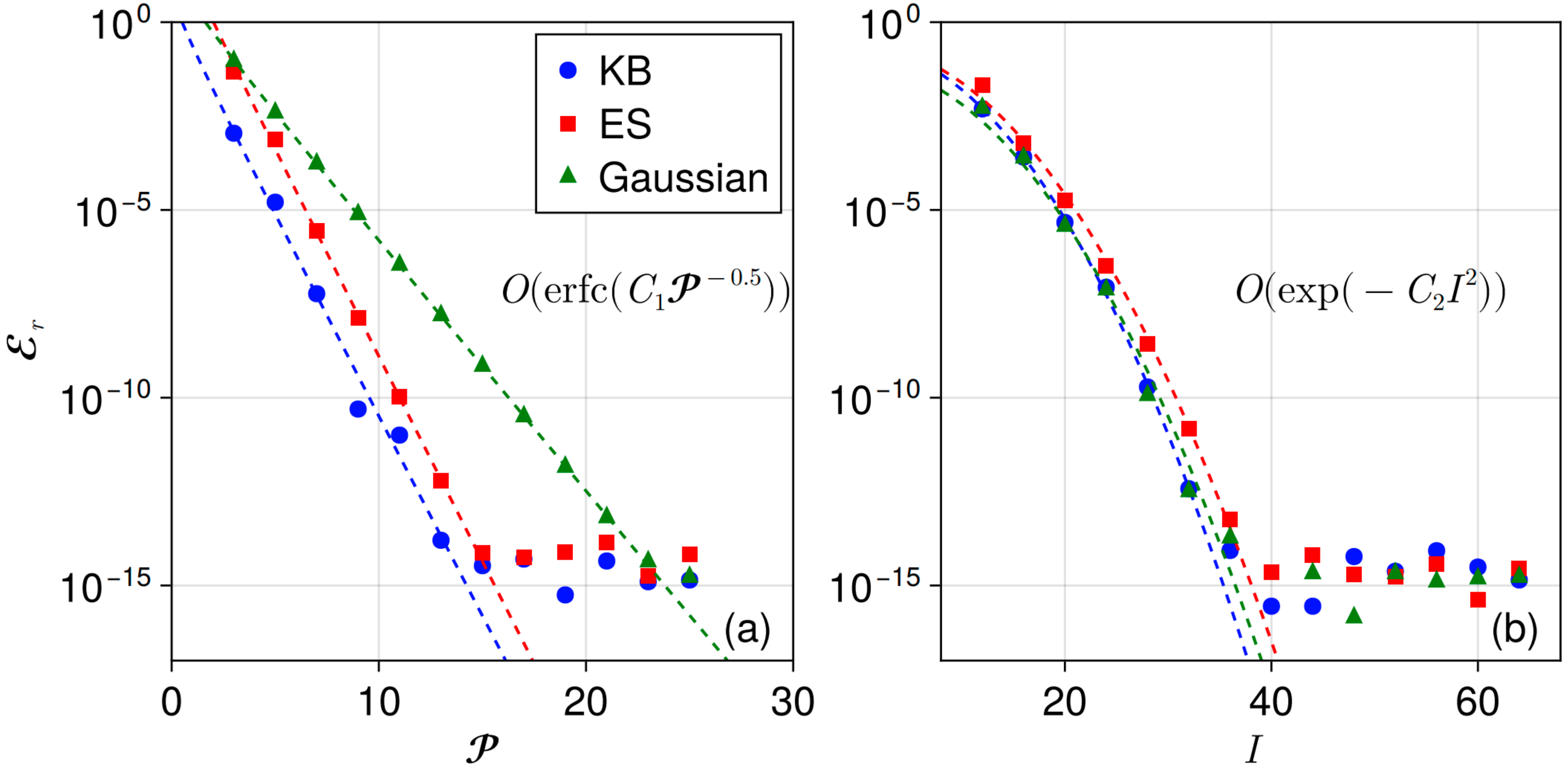


Figure 7: Different window functions.

Zero-padding

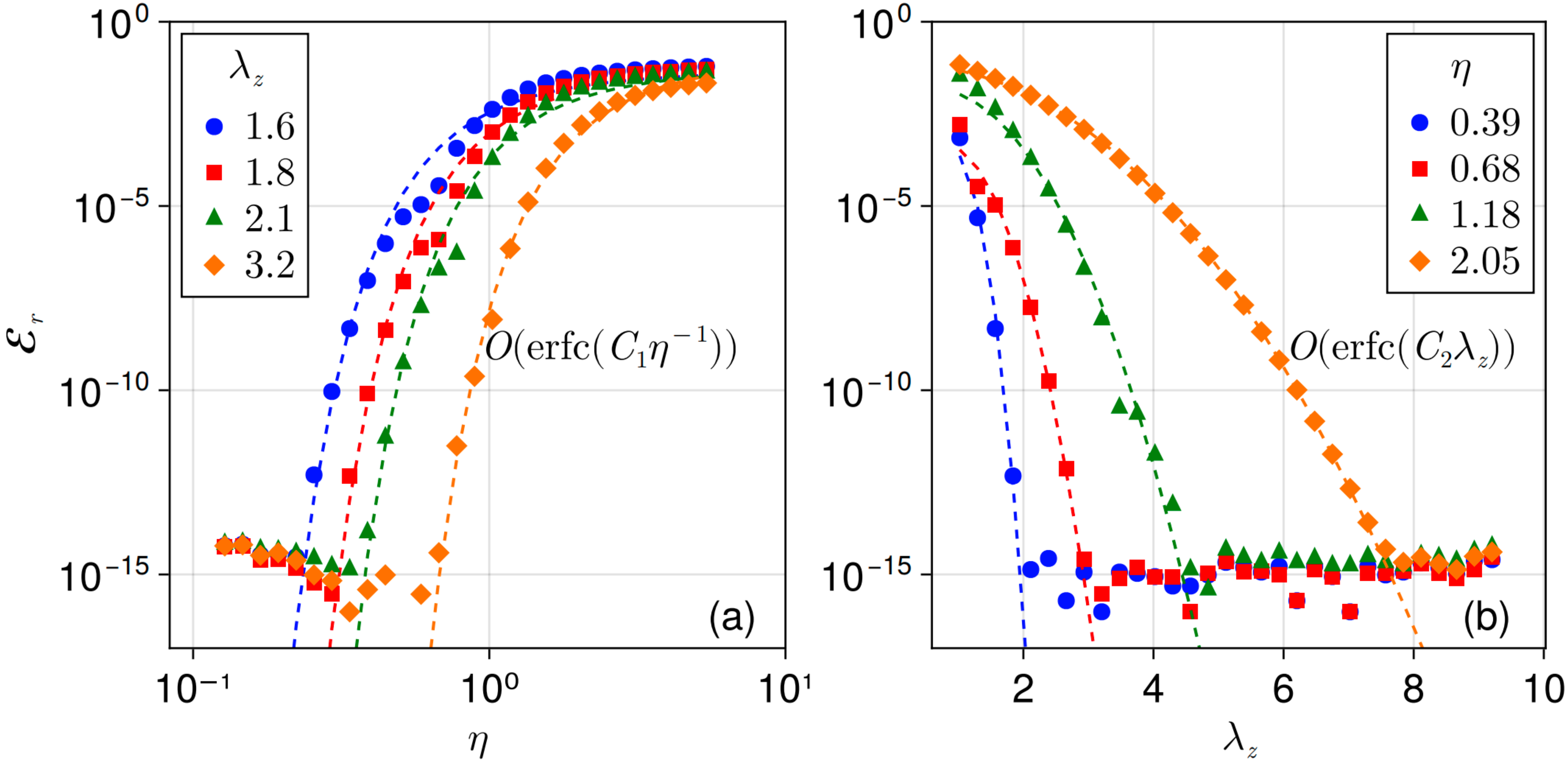


Figure 8: Zero-padding.

Chebyshev interpolation

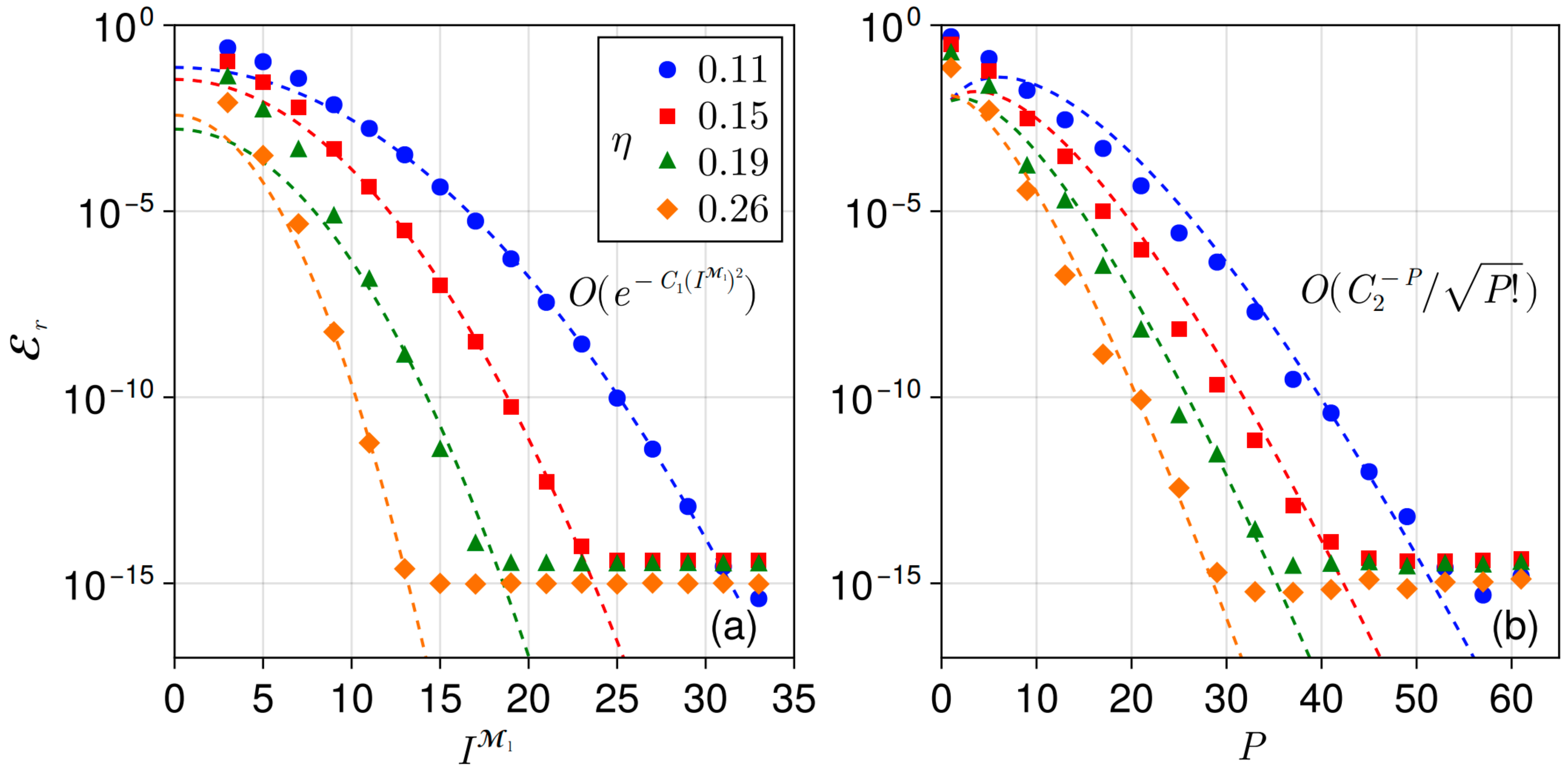


Figure 9: Accuracy of the Fourier-Chebyshev solver.

Strongly confined systems

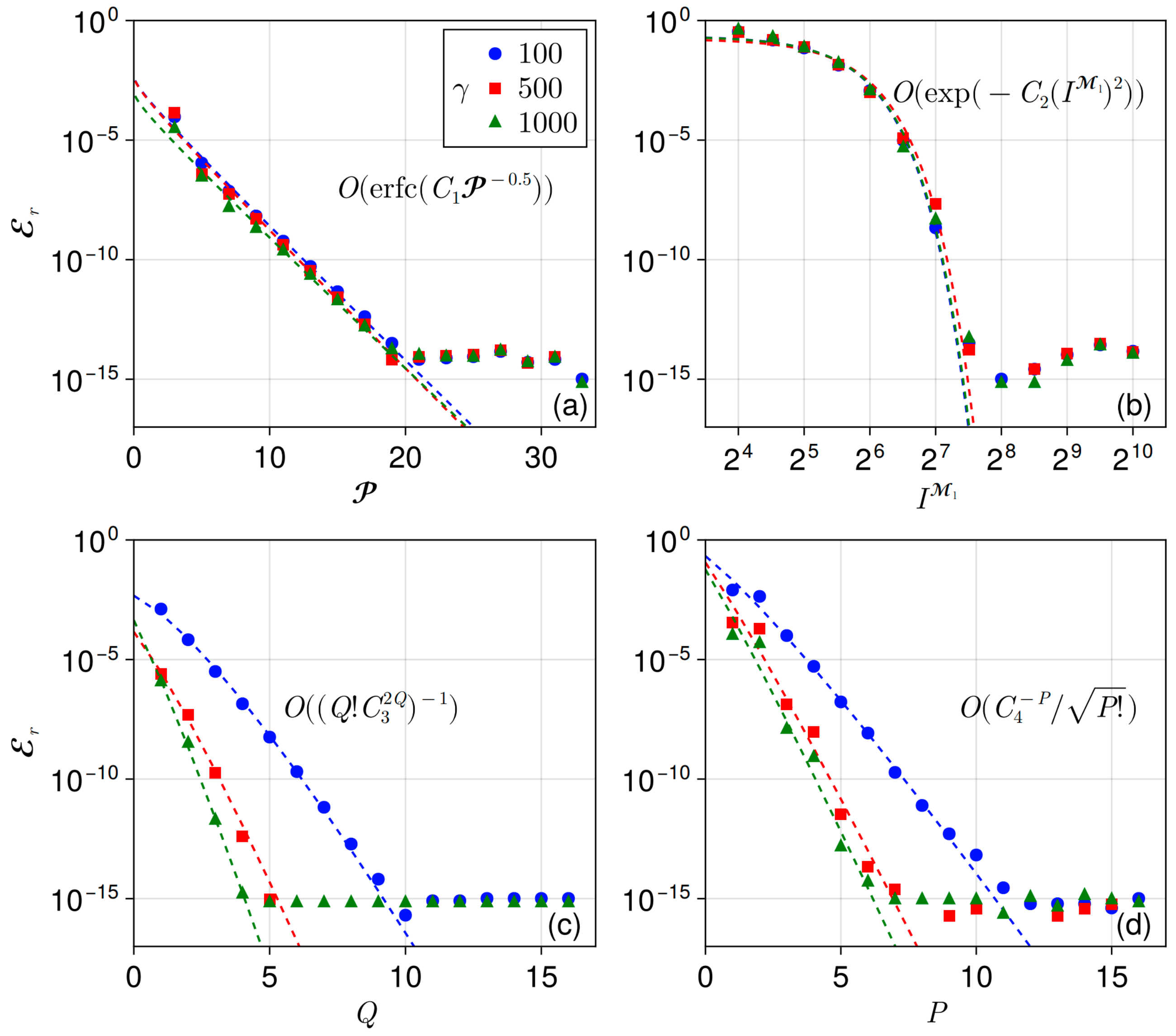


Figure 10: Strongly confined systems.

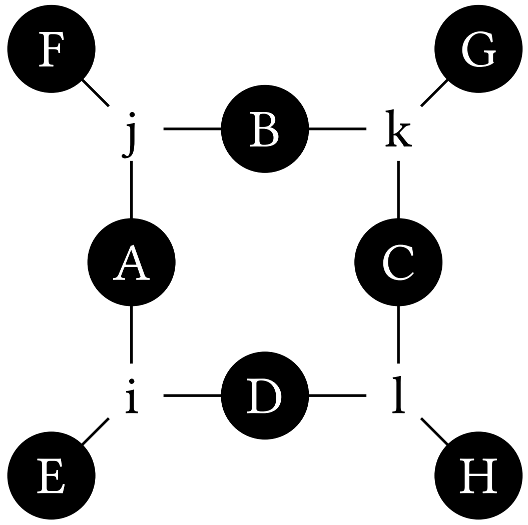
Einsum notation

Tensor networks can be represented as the so called Einsum notation:

$$Y_{i_y\cdots} = \sum_{i \notin \{i_y\cdots\}} A_{i_a\cdots} B_{i_b\cdots} C_{i_c\cdots} \cdots$$

It also has a hyper-graph representation, where each node is a tensor and each edge is an index:

$$s = \sum_{i,j,k,l} A_{ij} B_{jk} C_{kl} D_{li} E_i F_j G_k H_l \quad \longrightarrow$$



Contraction order

A contraction order can be represented as a rooted (binary) tree:

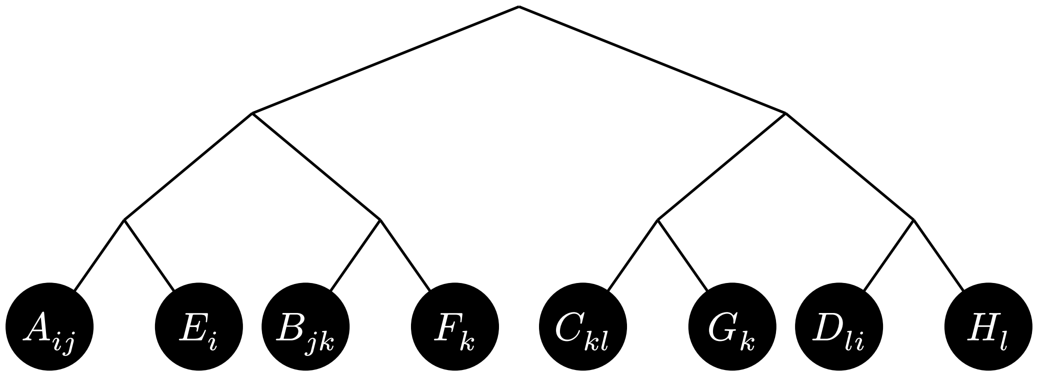


Figure 11: An example of binary contraction tree.

Different contraction orders can lead to different complexities, the order with the minimum complexity is called the optimal contraction order.

Optimizing the contraction order

Finding the optimal contraction order is a NP-hard[1] problem!

In the past few years, tools have been developed to optimize the contraction order:

- OMEinsumContractionOrder.jl[2] in Julia
- Cotengra[3] in Python

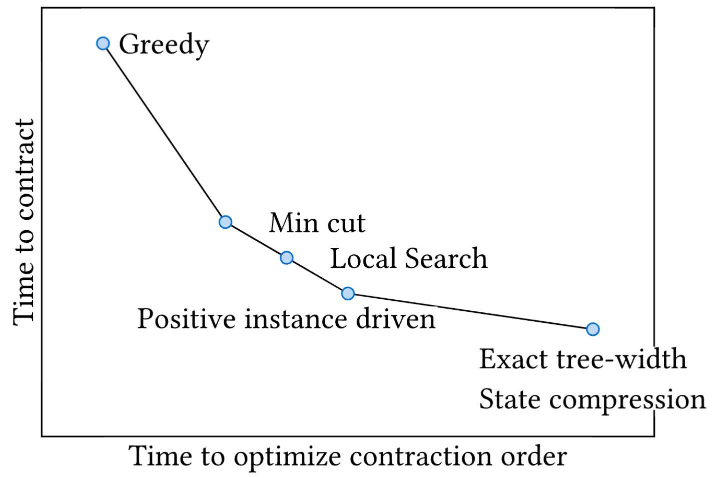


Figure 12: Comparison of different contraction orders.

[1] I.L. Markov, Y. Shi, SIAM J. Comput. 38, 963–981 (2008).

[2] https://github.com/TensorBFS/OMEinsumContractionOrder.jl

[3] https://github.com/jcmgray/cotengra

Tropical Tensor Network

In tropical semiring, the multiplication and addition are defined as:

$$a \otimes b = a + b$$
$$a \oplus b = \max(a, b)$$

By replacing the matrix multiplication with the tropical matrix multiplication, we get the tropical TN, where the contraction results:

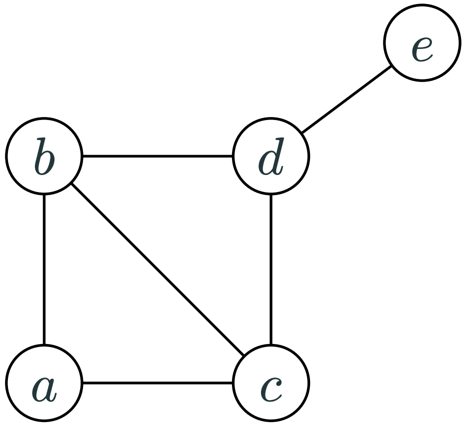
$$T_{i_y \dots} = \max_{i \notin \{i_y \dots\}} \left(A_{i_a \dots} + B_{i_b \dots} + C_{i_c \dots} + \dots \right)$$

which is the maximum of the sum of the elements among all the possible assignments.

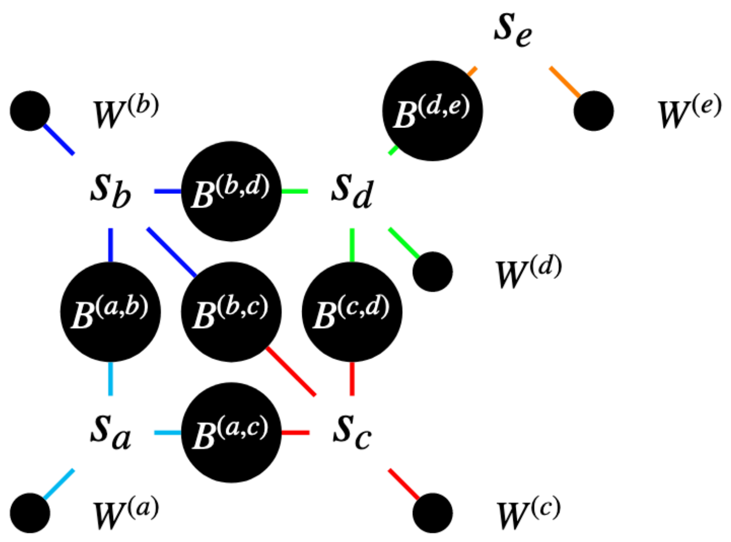
Very useful in combinatorial optimization problems and ground state search.

Tensor Network for Maximum Independent Set Problem

A tropical TN can be used to solve the MIS problem, a simple example is shown below:



$\rightarrow$



with

$$B = \begin{pmatrix} 0 & 0 \\ 0 & -\infty \end{pmatrix}, \text{ and } \quad W = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

under a tropical semiring.

Solving the set covering problem via Mixed Integer Linear Programming

In the original set covering problem, the function to be optimized is not linear. We solve it iteratively.

Fixed the branching complexity γ, find a solution to x that satisfies

$$\min_x \sum_{i=1}^{|\mathcal{C}|} \gamma^{-\Delta\rho(c_i)} x_i, \text{ s.t. } \bigcup_{\substack{i=1,\dots,\,|\mathcal{D}|,\\ x_i=1}} J_i = \{1, 2, \dots, |\mathcal{S}_R|\}$$

It corresponds to the following WMSC problem:

$$\begin{cases} \text{Alphabet: } \{1, 2, \dots, |\mathcal{S}_R|\} \\ \text{Sets: } \left\{J_1, J_2, \dots, J_{|\mathcal{C}|}\right\} \\ \text{Weights: } i \mapsto \gamma^{-\Delta\rho(c_i)} \end{cases}$$

After each iteration, the branching complexity γ is updated, coverge in a few iterations.

Solving the set covering problem via Mixed Integer Linear Programming

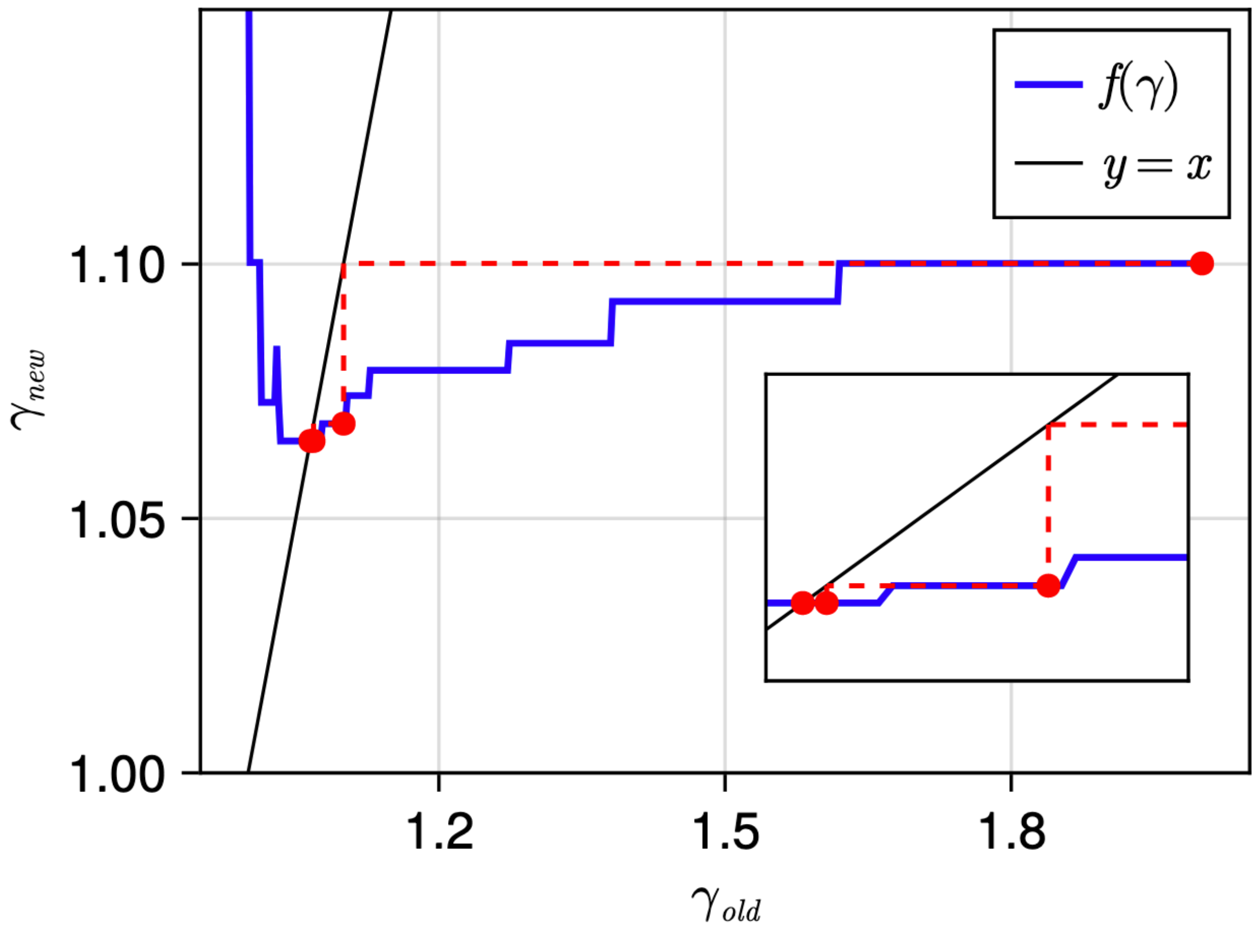


Figure 13: An example of solving the set covering problem via iterative MIP.

The branching algorithm used in benchmark

Reduction \ Branching	Optimal Branching (this work)	Xiao 2013	Akiba 2015
d1/d2 reduction	ob	-	akiba2015
d1/d2 reduction Xiao's rules	ob+xiao	xiao2013	-
d1/d2 reduction Xiao's rules packing rule	-	-	akiba2015+xiao&packing

Benchmarks for the worst-case complexity

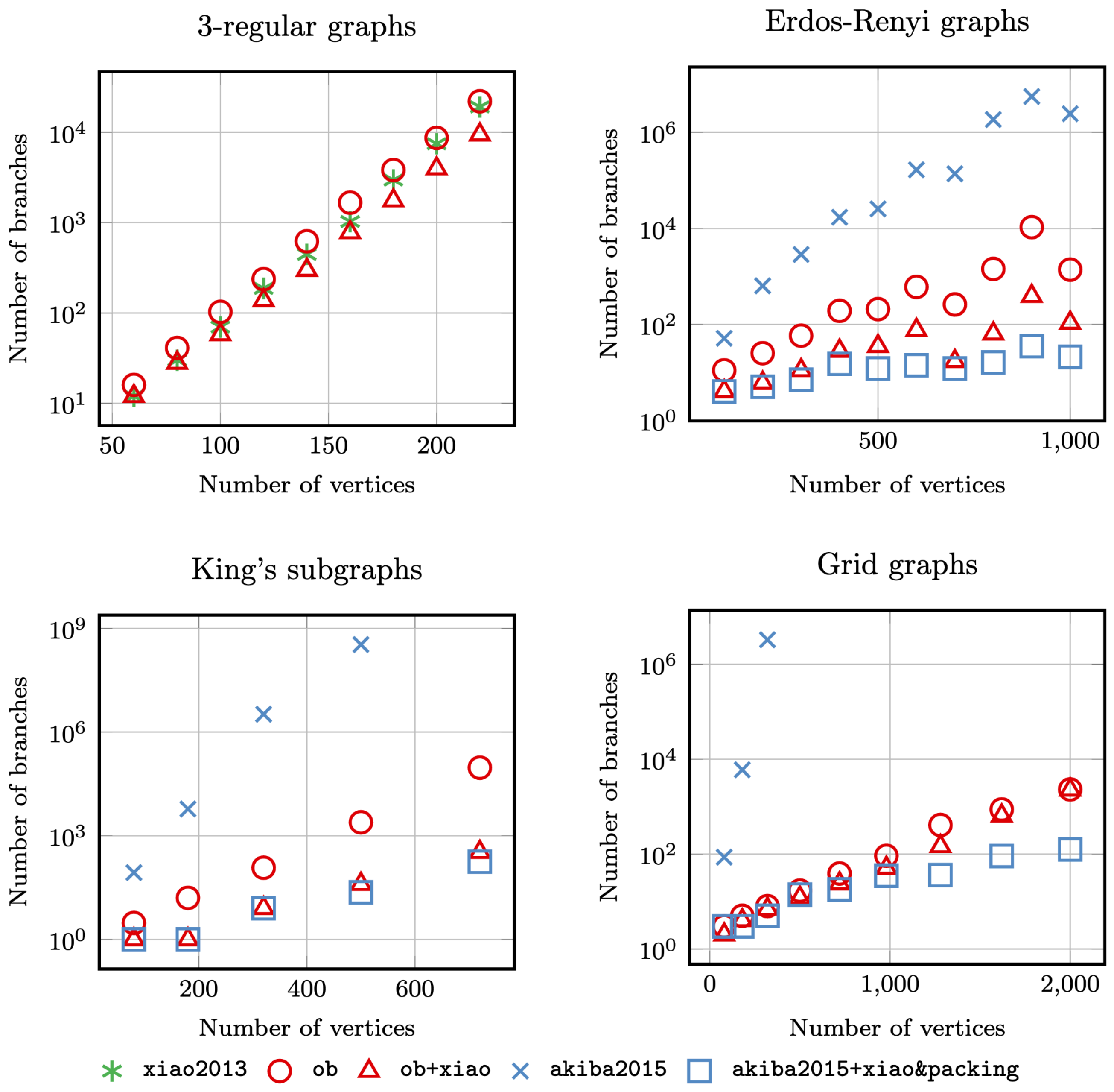


Figure 14: The worst-case complexity of the proposed method on random graphs.

References

Achterberg, T. (2009). SCIP: Solving constraint integer programs. *Math. Program. Comput.*, *1*(1), 1–41. https://doi.org/10.1007/s12532-008-0001-1

Akiba, T., & Iwata, Y. (2016). Branch-and-reduce exponential/FPT algorithms in practice: A case study of vertex cover. *Theoretical Computer Science*, *609*, 211–225. https://doi.org/10.1016/j.tcs.2015.09.023

Barnett, A. H., Magland, J., & Klinteberg, L. af. (2019). A parallel nonuniform fast Fourier transform library based on an ``Exponential of semicircle'' kernel. *SIAM J. Sci. Comput.*, *41*(5), C479–C504.

Barros, K., & Luijten, E. (2014). Dielectric Effects in the Self-Assembly of Binary Colloidal Aggregates. *Phys. Rev. Lett.*, *113*(1), 17801.

Beigel, R. (1999). Finding maximum independent sets in sparse and general graphs. *SODA*, *99*, 856–857.

Beylkin, G., & Monzón, L. (2010). Approximation by exponential sums revisited. *Applied and Computational Harmonic Analysis*, *28*(2), 131–149.

Bourgeois, N., Escoffier, B., & Paschos, V. T. (2008). An O*(1.0977 n) exact algorithm for max independent set in sparse graphs. *Parameterized and Exact Computation: Third*

References

International Workshop, IWPEC 2008, Victoria, Canada, May 14-16, 2008. Proceedings 3, 55–65.

Chen, J., Kanj, I. A., & Xia, G. (2003). Labeled search trees and amortized analysis: improved upper bounds for NP-hard problems. *Algorithms and Computation: 14th International Symposium, ISAAC 2003, Kyoto, Japan, December 15-17, 2003. Proceedings 14*, 148–157.

Fomin, F. V., & Høie, K. (2006). Pathwidth of cubic graphs and exact algorithms. *Information Processing Letters*, *97*(5), 191–196.

Fomin, F. V., & Kaski, P. (2013). Exact exponential algorithms. *Communications of the ACM*, *56*(3), 80–88.

Fomin, F. V., Grandoni, F., & Kratsch, D. (2006). Measure and conquer: A simple O (20. 288 n) independent set algorithm. *SODA*, *6*, 18–25.

Fürer, M. (2006). A faster algorithm for finding maximum independent sets in sparse graphs. *LATIN 2006: Theoretical Informatics: 7th Latin American Symposium, Valdivia, Chile, March 20-24, 2006. Proceedings 7*, 491–501.

Gao, X., Li, X., & Liu, J. (2024a,). *Programming guide for solving constraint satisfaction problems with tensor networks*. https://arxiv.org/abs/2501.00227

References

Gao, X., Wang, Y.-J., Zhang, P., & Liu, J.-G. (2024b, December). *Automated Discovery of Branching Rules with Optimal Complexity for the Maximum Independent Set Problem* (Issue arXiv:2412.07685). arXiv. https://doi.org/10.48550/arXiv.2412.07685

Greengard, L., Jiang, S., & Zhang, Y. (2018). The anisotropic truncated kernel method for convolution with free-space Green's functions. *SIAM J. Sci. Comput.*, *40*(6), A3733–A3754.

Jian, T. (1986). An $O(2^{0.304n})$ algorithm for solving maximum independent set problem. *IEEE Transactions on Computers*, *100*(9), 847–851.

Jiang, S., & Greengard, L. (2024). A Dual-space Multilevel Kernel-splitting Framework for Discrete and Continuous Convolution. *Accepted by Commun. Pure Appl. Math. (2024), Also Available at \url{Https://arxiv.org/pdf/2308.00292.pdf}*, *0*, .

Karp, R. M. (1972). *Reducibility among combinatorial problems*. Springer.

Kneis, J., Langer, A., & Rossmanith, P. (2009,). A fine-grained analysis of a simple independent set algorithm. *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (2009)*.

Liang, J., Xu, Z., & Zhou, Q. (2023,). *Error estimate of the u-series method for molecular dynamics simulations*. https://arxiv.org/abs/2305.05369

References

Liu, J.-G., Gao, X., Cain, M., Lukin, M. D., & Wang, S.-T. (2023). Computing solution space properties of combinatorial optimization problems via generic tensor networks. *SIAM Journal on Scientific Computing*, *45*(3), A1239–A1270.

Markov, I. L., & Shi, Y. (2008). Simulating Quantum Computation by Contracting Tensor Networks. *SIAM Journal on Computing*, *38*(3), 963–981. https://doi.org/10.1137/050644756

Mazars, M. (2011). Long ranged interactions in computer simulations and for quasi-2D systems. *Phys. Rep.*, *500*(2–3), 43–116.

Nguyen, M.-T., Liu, J.-G., Wurtz, J., Lukin, M. D., Wang, S.-T., & Pichler, H. (2023). Quantum Optimization with Arbitrary Connectivity Using Rydberg Atom Arrays. *PRX Quantum*, *4*(1), 10316. https://doi.org/10.1103/PRXQuantum.4.010316

Parry, D. E. (1975). The electrostatic potential in the surface region of an ionic crystal. *Surf. Sci.*, *49*(2), 433–440. https://doi.org/10.1016/0039-6028(75)90362-3

Predescu, C., Lerer, A. K., Lippert, R. A., Towles, B., Grossman, J., Dirks, R. M., & Shaw, D. E. (2020). The u-series: A separable decomposition for electrostatics computation with improved accuracy. *J. Chem. Phys.*, *152*(8), 84113.

References

Razgon, I. (2006). *A Faster Solving of the Maximum In-dependent Set Problem for Graphs with Maximal Degree 3.*

Robson, J. M. (2001). *Finding a maximum independent set in time O (2n/4).*

Robson, J. M. (1986). Algorithms for maximum independent sets. *Journal of Algorithms, 7*(3), 425–440.

Shamshirgar, D. S., Yokota, R., Tornberg, A.-K., & Hess, B. (2019). Regularizing the fast multipole method for use in molecular simulation. *The Journal of Chemical Physics, 151*(23), 234113.

Tarjan, R. E., & Trojanowski, A. E. (1977). Finding a maximum independent set. *SIAM Journal on Computing, 6*(3), 537–546.

Xiao, M.-Y., Chen, J.-E., & Han, X.-L. (2005). Improvement on vertex cover and independent set problem for low-degree graphs. *Jisuanji Xuebao(Chin. J. Comput.), 28*(2), 153–160.

Xiao, M. (2009, April). *New Branching Rules: Improvements on Independent Set and Vertex Cover in Sparse Graphs* (Issue arXiv:904.2712). arXiv. https://doi.org/10.48550/arXiv.0904.2712

References

Xiao, M., & Nagamochi, H. (2013). Confining sets and avoiding bottleneck cases: A simple maximum independent set algorithm in degree-3 graphs. *Theoretical Computer Science*, *469*, 92–104. https://www.sciencedirect.com/science/article/pii/S0304397512008729

Xiao, M., & Nagamochi, H. (2017). Exact Algorithms for Maximum Independent Set. *Information and Computation*, *255*, 126–146. https://doi.org/10.1016/j.ic.2017.06.001