Quantum algorithms in “modern” combinatorial optimization

Simon Apers

(CNRS, IRIF)

Journée Francilienne de Recherche Opérationnelle
ENSIIE, February 2022

"Modern" combinatorial optimization

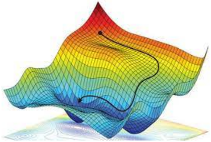
combine tools from

“Modern” combinatorial optimization

combine tools from

continuous optimization

- interior point methods
- (accelerated) gradient descent
- second-order methods

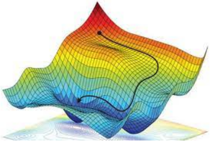


“Modern” combinatorial optimization

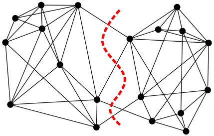
combine tools from

continuous optimization and discrete optimization

- interior point methods
- (accelerated) gradient descent
- second-order methods



- graph sparsification
- combinatorial preconditioning
- random walks/electric networks

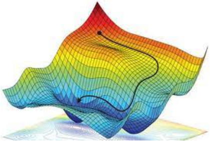


“Modern” combinatorial optimization

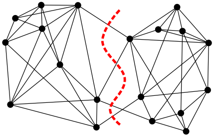
combine tools from

continuous optimization and discrete optimization

- interior point methods
- (accelerated) gradient descent
- second-order methods



- graph sparsification
- combinatorial preconditioning
- random walks/electric networks



e.g., best maximum flow algorithms

use IPMs…
with fast Laplacian solvers …

…based on graph sparsifiers

This talk

quantum algorithms catch up:

This talk

quantum algorithms catch up:

1) cut sparsification and cut problems (‘90-‘00)

This talk

quantum algorithms catch up:

1) cut sparsification and cut problems (‘90-‘00)
2) spectral sparsification and Laplacian solving (‘00-‘10)

This talk

quantum algorithms catch up:

1) cut sparsification and cut problems (‘90-‘00)
2) spectral sparsification and Laplacian solving (‘00-‘10)
3) matrix scaling and second-order methods (‘10-‘20)

Quantum model

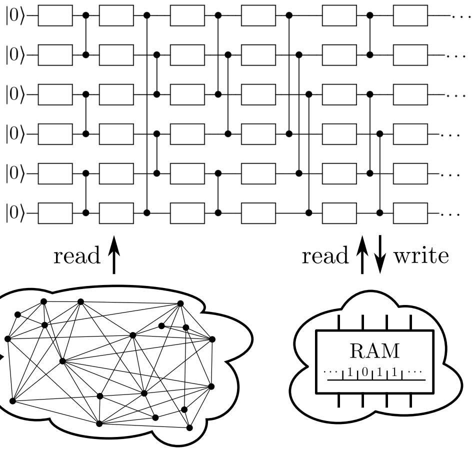
quantum time complexity

= total number of elementary gates, queries to input*,
QRAM operations

Quantum model

quantum time complexity

= total number of elementary gates, queries to input*, QRAM operations



*typically, adjacency matrix of input graph

Quantum primitives

key routine = Grover search

Quantum primitives

key routine = Grover search

consider N elements, k of which are “marked”
Grover search finds marked element in time $O(\sqrt{N/k})$

Quantum primitives

key routine = Grover search

consider N elements, k of which are "marked"
Grover search finds marked element in time $O(\sqrt{N/k})$

variations:

quantum approximate counting:
approximate k to error ϵ in time $O(\epsilon^{-1}\sqrt{N/k})$

Quantum primitives

key routine = Grover search

consider N elements, k of which are "marked"
Grover search finds marked element in time $O(\sqrt{N/k})$

variations:

quantum approximate counting:
approximate k to error ϵ in time $O(\epsilon^{-1}\sqrt{N/k})$

amplitude amplification, amplitude estimation
generalize to "marked" subspaces

Disclaimer

Caveats:

Disclaimer

Caveats:

- (subquadratic) polynomial speedup

Disclaimer

Caveats:

- (subquadratic) polynomial speedup
- QRAM requirements

Disclaimer

Caveats:

- (subquadratic) polynomial speedup
- QRAM requirements

Motivation:

Disclaimer

Caveats:

- (subquadratic) polynomial speedup
- QRAM requirements

Motivation:

- long-term applications, better understanding of quantum computing

Disclaimer

Caveats:

- (subquadratic) polynomial speedup
- QRAM requirements

Motivation:

- long-term applications, better understanding of quantum computing
- new insights in classical algorithms
  (similar to streaming, dynamic, distributed, . . . settings)

Disclaimer

Caveats:

- (subquadratic) polynomial speedup
- QRAM requirements

Motivation:

- long-term applications, better understanding of quantum computing
- new insights in classical algorithms
  (similar to streaming, dynamic, distributed, . . . settings)
- “true” complexity of a problem?
  (e.g., complexity of matrix multiplication)

Quantum algorithms for

1) cut sparsification and cut problems (‘90-‘00)

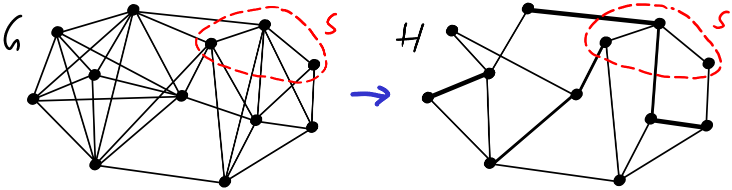
2) spectral sparsification and Laplacian solving (‘00-‘10)

3) matrix scaling and second-order methods (‘10-‘20)

Cut sparsification

ϵ-cut sparsifier H of G is sparse subgraph such that cuts in H approximate cuts in G:

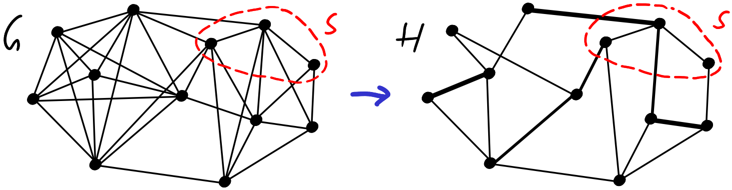
$$\mathrm{val}_H(S) = \sum_{x \in S, y \notin S} w_H(x,y) = (1 \pm \epsilon)\, \mathrm{val}_G(S), \quad \forall S \subset V.$$



Cut sparsification

ϵ-cut sparsifier H of G is sparse subgraph
such that cuts in H approximate cuts in G:

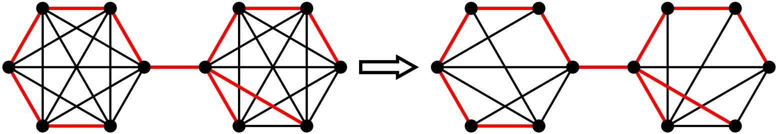
$$\mathrm{val}_H(S) = \sum_{x \in S, y \notin S} w_H(x,y) = (1 \pm \epsilon)\,\mathrm{val}_G(S), \quad \forall S \subset V.$$



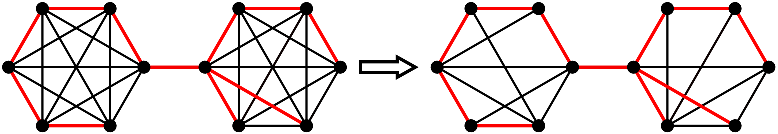
[Benczúr-Karger '96]: exists ϵ-cut sparsifier with $\widetilde{O}(n/\epsilon^2)$ edges

= building block of many classical algorithms for graph problems

Algorithm for cut sparsification

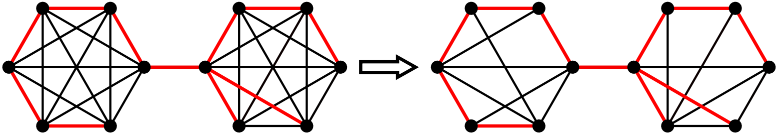


Algorithm for cut sparsification



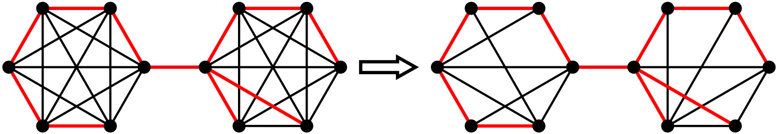
1. construct $\widetilde{O}(1/\epsilon^2)$ minimum spanning trees, keep these edges

Algorithm for cut sparsification



1. construct $\widetilde{O}(1/\epsilon^2)$ minimum spanning trees, keep these edges
2. $\forall$ remaining edge: keep with probability $1/2$ and double its weight

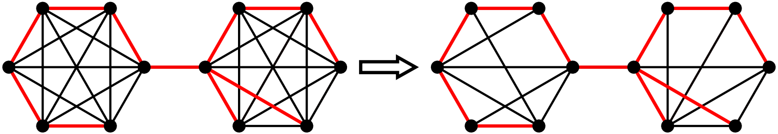
Algorithm for cut sparsification



1. construct $\widetilde{O}(1/\epsilon^2)$ minimum spanning trees, keep these edges
2. $\forall$ remaining edge: keep with probability $1/2$ and double its weight

= ϵ-cut sparsifier with $\approx m/2 + \widetilde{O}(n/\epsilon^2)$ edges (w.h.p.)

Algorithm for cut sparsification

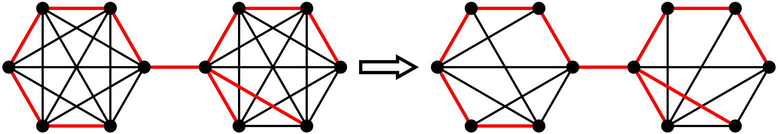


1. construct $\widetilde{O}(1/\epsilon^2)$ minimum spanning trees, keep these edges
2. $\forall$ remaining edge: keep with probability $1/2$ and double its weight

= ϵ-cut sparsifier with $\approx m/2 + \widetilde{O}(n/\epsilon^2)$ edges (w.h.p.)

$\rightarrow$ repeat $O(\log n)$ times = ϵ-cut sparsifier with $\widetilde{O}(n/\epsilon^2)$ edges

Algorithm for cut sparsification



1. construct $\widetilde{O}(1/\epsilon^2)$ minimum spanning trees, keep these edges
2. $\forall$ remaining edge: keep with probability $1/2$ and double its weight

= ϵ-cut sparsifier with $\approx m/2 + \widetilde{O}(n/\epsilon^2)$ edges (w.h.p.)

$\rightarrow$ repeat $O(\log n)$ times = ϵ-cut sparsifier with $\widetilde{O}(n/\epsilon^2)$ edges

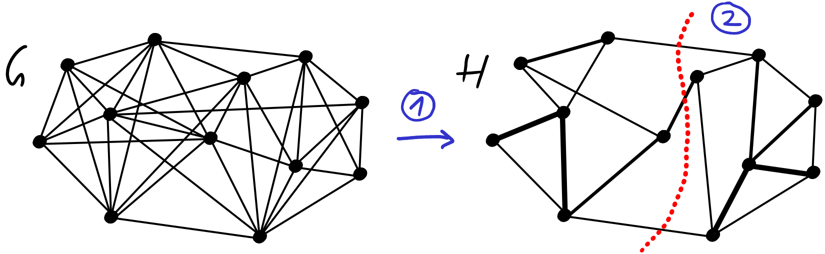
using quantum algorithm for MST* + more work:

Theorem (A-de Wolf '20)

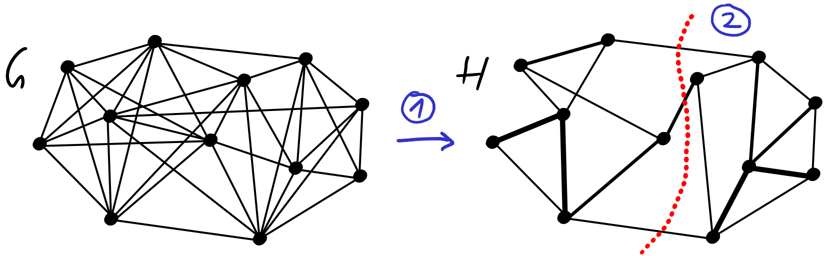
There is a quantum algorithm for constructing an ϵ-cut sparsifier H in time $\widetilde{O}(n^{3/2}/\epsilon)$, which is optimal.

*[Dürr-Heiligman-Høyer-Mhalla '08]

Quantum speedup for cut problems

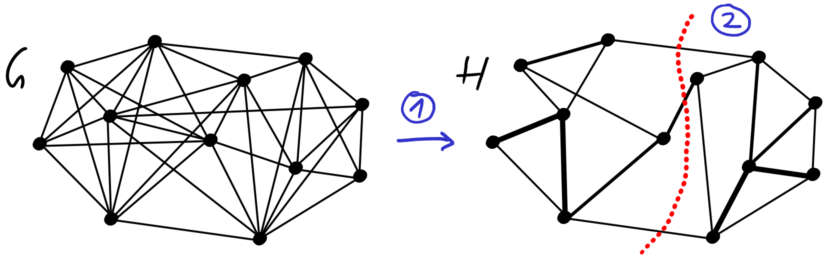


Quantum speedup for cut problems



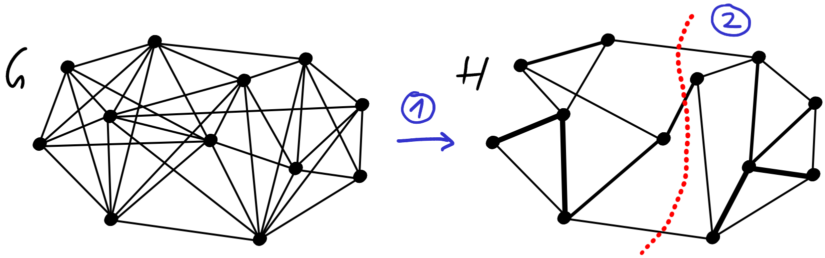
1. construct ϵ-sparsifier H

Quantum speedup for cut problems



1. construct ϵ-sparsifier H
2. Classically solve cut problem in H

Quantum speedup for cut problems



1. construct ϵ-sparsifier H
2. Classically solve cut problem in H

= approximate minimum cut, maximum cut, sparsest cut, . . . in time

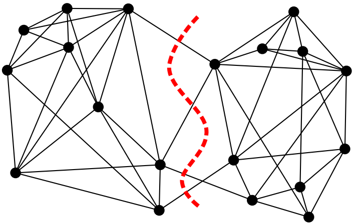
$$\widetilde{O}(n^{3/2}/\epsilon)$$

versus $\Omega(n^2)$ classically

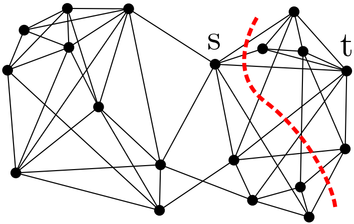
Quantum speedup for cut problems

With more work (on simple graphs):

exact minimum cut
in time $\widetilde{O}(n^{3/2})$, optimal
[Apers-Lee '20]



exact minimum s-t cut
in time $\widetilde{O}(n^{11/6})$, suboptimal?
[Apers-Auza-Lee '21]



versus $\Omega(n^2)$ classically

Quantum algorithms for

1) cut sparsification and cut problems (‘90-‘00)

2) spectral sparsification and Laplacian solving (‘00-‘10)

3) matrix scaling and second-order methods (‘10-‘20)

Spectral sparsification

spectral sparsifier H is sparse subgraph of G such that "quadratic forms" in Laplacian L_H approximate those in L_G:

$$x^T L_H x = \sum_{i<j} w_H(i,j)(x_i - x_j)^2 = (1 \pm \epsilon) x^T L_G x, \quad \forall x \in \mathbb{R}^n$$

Spectral sparsification

spectral sparsifier H is sparse subgraph of G such that "quadratic forms" in Laplacian L_H approximate those in L_G:

$$x^T L_H x = \sum_{i<j} w_H(i,j)(x_i - x_j)^2 = (1 \pm \epsilon) x^T L_G x, \quad \forall x \in \mathbb{R}^n$$

e.g., if $x = 1_S$ for $S \subseteq [n]$ then

$$x^T L_H x = \sum_{i \in S, j \notin S} w_H(i,j) = \mathrm{val}_H(S)$$

Spectral sparsification

spectral sparsifier H is sparse subgraph of G such that "quadratic forms" in Laplacian L_H approximate those in L_G:

$$x^T L_H x = \sum_{i<j} w_H(i,j)(x_i - x_j)^2 = (1 \pm \epsilon) x^T L_G x, \quad \forall x \in \mathbb{R}^n$$

e.g., if $x = 1_S$ for $S \subseteq [n]$ then

$$x^T L_H x = \sum_{i \in S, j \notin S} w_H(i,j) = \mathrm{val}_H(S)$$

hence, spectral sparsifier $\Rightarrow$ cut sparsifier

Spectral sparsification

[Spielman-Teng '04]: exists ϵ-spectral sparsifier with $\widetilde{O}(n/\epsilon^2)$ edges

Spectral sparsification

[Spielman-Teng '04]: exists ϵ-spectral sparsifier with $\widetilde{O}(n/\epsilon^2)$ edges

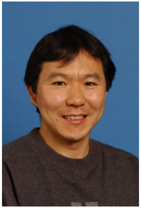
$\rightarrow$ building block of near-linear time algorithm
for solving Laplacian system $L_G x = b$

Spectral sparsification

[Spielman-Teng '04]: exists ϵ-spectral sparsifier with $\widetilde{O}(n/\epsilon^2)$ edges

$\rightarrow$ building block of near-linear time algorithm
for solving Laplacian system $L_G x = b$

= Gödel prize 2015

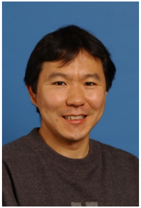



Spectral sparsification

[Spielman-Teng '04]: exists ϵ-spectral sparsifier with $\widetilde{O}(n/\epsilon^2)$ edges

$\rightarrow$ building block of near-linear time algorithm for solving Laplacian system $L_G x = b$

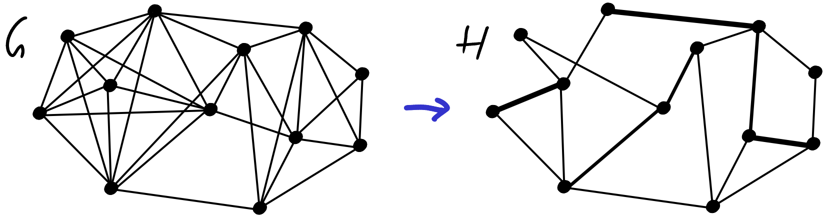
= Gödel prize 2015



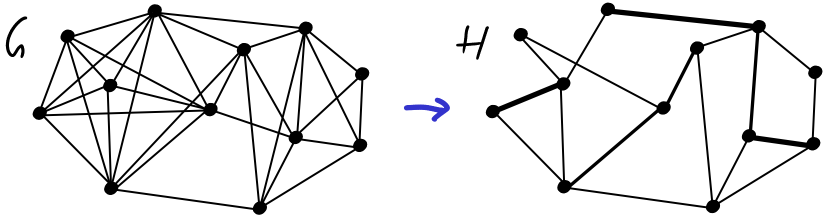
Theorem (A-de Wolf '20)

There is a quantum algorithm for constructing an ϵ-spectral sparsifier in time $\widetilde{O}(n^{3/2}/\epsilon)$, which is optimal.

Quantum speedup for Laplacian solving

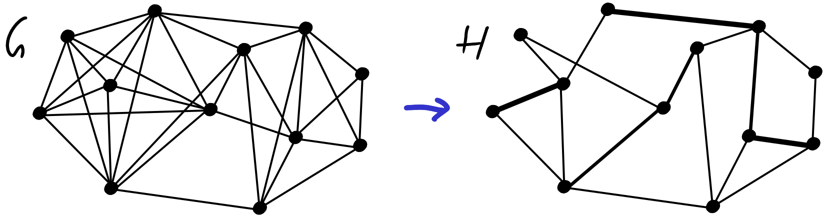


Quantum speedup for Laplacian solving



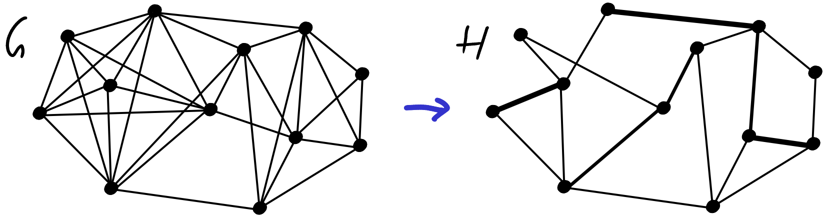
1. construct ϵ-spectral sparsifier H

Quantum speedup for Laplacian solving



1. construct ϵ-spectral sparsifier H
2. Classically solve Laplacian system $L_H x = b$

Quantum speedup for Laplacian solving

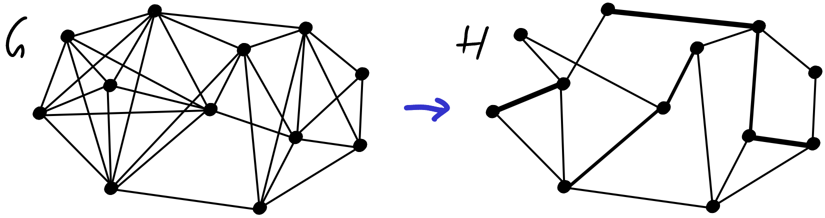


1. construct ϵ-spectral sparsifier H
2. Classically solve Laplacian system $L_H x = b$

= ϵ-approximate solution of $L_G x = b$ in time

$$\widetilde{O}(n^{3/2}/\epsilon)$$

Quantum speedup for Laplacian solving



1. construct ϵ-spectral sparsifier H
2. Classically solve Laplacian system $L_H x = b$

= ϵ-approximate solution of $L_G x = b$ in time

$$\widetilde{O}(n^{3/2}/\epsilon)$$

similarly, quantum speedups for spectral clustering, RW properties, ...

Quantum algorithms for

1) cut sparsification and cut problems (‘90-‘00)

2) spectral sparsification and Laplacian solving (‘00-‘10)

3) matrix scaling and second-order methods (‘10-‘20)

Matrix scaling

Input:

matrix $A \in \mathbb{R}^{n\times n}_{>0}$, target marginals $r, c \in \mathbb{R}^n_{>0}$

Matrix scaling

Input:

matrix $A \in \mathbb{R}^{n \times n}_{>0}$, target marginals $r, c \in \mathbb{R}^n_{>0}$

Goal:

find "rescaling"

$$XAY = \underbrace{\begin{bmatrix} e^{x_1} & & 0 \\ & \ddots & \\ 0 & & e^{x_n} \end{bmatrix}}_{\text{row rescaling}} A \underbrace{\begin{bmatrix} e^{y_1} & & 0 \\ & \ddots & \\ 0 & & e^{y_n} \end{bmatrix}}_{\text{column rescaling}}$$

such that XAY has row sums r and column sums c

Matrix scaling

Input:

matrix $A \in \mathbb{R}^{n\times n}_{>0}$, target marginals $r, c \in \mathbb{R}^n_{>0}$

Goal:

find "rescaling"

$$XAY = \underbrace{\begin{bmatrix} e^{x_1} & & 0 \\ & \ddots & \\ 0 & & e^{x_n} \end{bmatrix}}_{\text{row rescaling}} A \underbrace{\begin{bmatrix} e^{y_1} & & 0 \\ & \ddots & \\ 0 & & e^{y_n} \end{bmatrix}}_{\text{column rescaling}}$$

such that XAY has row sums r and column sums c

Applications:

approximating matrix permanent, optimal transport in machine learning, numerical linear algebra, ...

Algorithms for matrix scaling

"folklore" classical algorithm:

complexity $\widetilde{O}(n^2/\epsilon)$ for ϵ-approximate solution
by iterative rescaling

[Sinkhorn '64]

$$A \to \underbrace{X_1 A}_{\text{fix } r} \to \underbrace{X_1 A Y_1}_{\text{fix } c} \to \underbrace{X_2 X_1 A Y_1}_{\text{fix } r} \to \dots$$

Algorithms for matrix scaling

"folklore" classical algorithm:

complexity $\widetilde{O}(n^2/\epsilon)$ for ϵ-approximate solution
by iterative rescaling
[Sinkhorn '64]

$$A \to \underbrace{X_1 A}_{\text{fix } r} \to \underbrace{X_1 A Y_1}_{\text{fix } c} \to \underbrace{X_2 X_1 A Y_1}_{\text{fix } r} \to \ldots$$

$$\downarrow$$

quantum algorithm:

complexity $\widetilde{O}(n^{3/2}/\epsilon^3)$ for ϵ-approximate solution
by Sinkhorn + quantum approximate counting
[van Apeldoorn-Gribling-Li-Nieuwboer-Walter-de Wolf '21]

Algorithms for matrix scaling

observation:

matrix scaling = convex optimization problem

Algorithms for matrix scaling

observation:

matrix scaling = convex optimization problem

$$f(x,y) = \sum_{i,j} A_{i,j} e^{x_i + y_j} - r^T x - c^T y$$

$\nabla f(x,y) = 0 \;\Leftrightarrow\; (x,y)$ describes rescaling

Algorithms for matrix scaling

observation:

matrix scaling = convex optimization problem

$$f(x,y) = \sum_{i,j} A_{i,j} e^{x_i + y_j} - r^T x - c^T y$$

$\nabla f(x,y) = 0 \;\Leftrightarrow\; (x,y)$ describes rescaling

$$\downarrow$$

Sinkhorn, $\widetilde{O}(n^2/\epsilon)$: first-order method

Algorithms for matrix scaling

observation:

matrix scaling = convex optimization problem

$$f(x,y) = \sum_{i,j} A_{i,j} e^{x_i + y_j} - r^T x - c^T y$$

$\nabla f(x,y) = 0 \;\Leftrightarrow\; (x,y)$ describes rescaling

$$\downarrow$$

Sinkhorn, $\widetilde{O}(n^2/\epsilon)$: first-order method

improved algorithm:

complexity $\widetilde{O}(n^2 \log(1/\epsilon))$ using *second-order* method

[Cohen-Mądry-Tsipras-Vladu '17]

Algorithms for matrix scaling

observation:

matrix scaling = convex optimization problem

$$f(x,y) = \sum_{i,j} A_{i,j} e^{x_i + y_j} - r^T x - c^T y$$

$$\nabla f(x,y) = 0 \;\Leftrightarrow\; (x,y) \text{ describes rescaling}$$

$$\downarrow$$

Sinkhorn, $\widetilde{O}(n^2/\epsilon)$: first-order method

improved algorithm:

complexity $\widetilde{O}(n^2 \log(1/\epsilon))$ using *second-order* method

[Cohen-Mądry-Tsipras-Vladu '17]

key tool: Hessian of f is Laplacian matrix

$\rightarrow$ can use efficient Laplacian solving!

Algorithms for matrix scaling

complexity $\widetilde{O}(n^2 \log(1/\epsilon))$ using *second-order* method
[Cohen-Mądry-Tsipras-Vladu ’17]

? improved quantum algorithm for matrix scaling ?

Algorithms for matrix scaling

complexity $\widetilde{O}(n^2 \log(1/\epsilon))$ using *second-order* method
[Cohen-Mądry-Tsipras-Vladu ’17]

? improved quantum algorithm for matrix scaling ?

[Gribling-Nieuwboer ’21]:
complexity $\widetilde{O}(n^{3/2}/\epsilon^2)$ using second-order method
+ quantum sparsification and Laplacian solving

Algorithms for matrix scaling

complexity $\widetilde{O}(n^2 \log(1/\epsilon))$ using *second-order* method

[Cohen-Mądry-Tsipras-Vladu '17]

? improved quantum algorithm for matrix scaling ?

[Gribling-Nieuwboer '21]:

complexity $\widetilde{O}(n^{3/2}/\epsilon^2)$ using second-order method

+ quantum sparsification and Laplacian solving

also, no quantum speedup for $\epsilon \ll 1$

Summary:

quantum algorithms for

1) cut sparsification and cut problems (‘90-‘00)
2) spectral sparsification and Laplacian solving (‘00-‘10)
3) matrix scaling and second-order methods (‘10-‘20)

Summary:

quantum algorithms for

1) cut sparsification and cut problems (‘90-‘00)

2) spectral sparsification and Laplacian solving (‘00-‘10)

3) matrix scaling and second-order methods (‘10-‘20)

Future directions:

1) quantum speedup in interior point methods?
flagship problem = maximum flow (‘20-. . .)

Summary:

quantum algorithms for

1) cut sparsification and cut problems (‘90-‘00)

2) spectral sparsification and Laplacian solving (‘00-‘10)

3) matrix scaling and second-order methods (‘10-‘20)

Future directions:

1) quantum speedup in interior point methods?
flagship problem = maximum flow (‘20-...)

2) continuous $\leftrightarrow$ discrete trends also in sampling algorithms
e.g., logconcave sampling, estimating volume of convex bodies, ...