A deterministic PTAS for commutative rank of matrix spaces

Markus Bläser[1], Gorav Jindal[2] and Anurag Pandey[2]

[1]Saarland University
[2]Max-Planck-Institute for Informatics

CCC 2017
09/07/2017

1. Introduction
   - Basic Problem
   - Motivation
   - Previous work

2. Main algorithm
   - A simple $\frac{1}{2}$-approximation algorithm
   - Ideas for better approximation

Setup

- $\mathbb{F}$ be any field, $n \in \mathbb{Z}_{>0}$.
  - $\mathbb{F}^{n\times n}$ is the (vector) space of all $n \times n$ matrices with entries in $\mathbb{F}$.
- For vector spaces V, W
  - Use notation $V \leq W$ to denote that V is a subspace of W.

Definition (Matrix space)

A vector space $\mathcal{B} \leq \mathbb{F}^{n\times n}$ is called a matrix space.

COMPUTATIONAL COMPLEXITY CONFERENCE

Setup

- $\mathbb{F}$ be any field, $n \in \mathbb{Z}_{>0}$.
  - $\mathbb{F}^{n \times n}$ is the (vector) space of all $n \times n$ matrices with entries in $\mathbb{F}$.
- For vector spaces V, W
  - Use notation $V \leq W$ to denote that V is a subspace of W.

Definition (Matrix space)

A vector space $\mathcal{B} \leq \mathbb{F}^{n \times n}$ is called a matrix space.

Setup

- $\mathbb{F}$ be any field, $n \in \mathbb{Z}_{>0}$.
  - $\mathbb{F}^{n \times n}$ is the (vector) space of all $n \times n$ matrices with entries in $\mathbb{F}$.
- For vector spaces V, W
  - Use notation $V \leq W$ to denote that V is a subspace of W.

Definition (Matrix space)

A vector space $\mathcal{B} \leq \mathbb{F}^{n \times n}$ is called a matrix space.

Problem

Problem

Given a matrix space $\mathcal{B} \leq \mathbb{F}^{n\times n}$ as input, compute its "rank". $\mathcal{B}$ is given as input by its set of generators, i.e, $\mathcal{B} = \langle B_1, B_2, \ldots, B_m \rangle$.

- Two notions of rank.
  - Commutative rank.
  - Non-commutative rank.

Problem

Problem

Given a matrix space $\mathcal{B} \leq \mathbb{F}^{n \times n}$ as input, compute its "rank". $\mathcal{B}$ is given as input by its set of generators, i.e, $\mathcal{B} = \langle B_1, B_2, \ldots, B_m \rangle$.

- Two notions of rank.
  - Commutative rank.
  - Non-commutative rank.

Commutative rank

Definition (Commutative rank)

$\mathcal{B} \leq \mathbb{F}^{n \times n}$ any matrix space, then
Commutaive rank of $\mathcal{B} = \text{rank}(\mathcal{B}) = \max\{\text{rank}(B) \mid B \in \mathcal{B}\}$.

- $\mathcal{B} \leq \mathbb{F}^{n \times n}$ is called **full-rank** if $\text{rank}(\mathcal{B}) = n$.

Commutative rank

Definition (Commutative rank)

$\mathcal{B} \leq \mathbb{F}^{n \times n}$ any matrix space, then
Commutaive rank of $\mathcal{B} = \mathrm{rank}(\mathcal{B}) = \max\{\mathrm{rank}(B) \mid B \in \mathcal{B}\}$.

- $\mathcal{B} \leq \mathbb{F}^{n \times n}$ is called **full-rank** if $\mathrm{rank}(\mathcal{B}) = n$.

A different Formulation

- Matrix space $\mathcal{B} = \langle B_1, B_2, \ldots, B_m \rangle \leq \mathbb{F}^{n \times n}$, consider the matrix
  - $B = x_1 B_1 + x_2 B_2 + \ldots + x_m B_m$ over the field $\mathbb{F}(x_1, x_2, \ldots, x_m)$ of rational functions.

Fact

If $|\mathbb{F}| > n$ *then* $\text{rank}(\mathcal{B}) = \text{rank}(B)$.

- Gives a randomized polynomial time algorithm using Schwartz–Zippel lemma.
  - Even an RNC algorithm.

A different Formulation

- Matrix space $\mathcal{B} = \langle B_1, B_2, \ldots, B_m \rangle \leq \mathbb{F}^{n \times n}$, consider the matrix
  - $B = x_1 B_1 + x_2 B_2 + \ldots + x_m B_m$ over the field $\mathbb{F}(x_1, x_2, \ldots, x_m)$ of rational functions.

Fact

If $|\mathbb{F}| > n$ *then* $\mathrm{rank}(\mathcal{B}) = \mathrm{rank}(B)$.

- Gives a randomized polynomial time algorithm using Schwartz–Zippel lemma.
  - Even an RNC algorithm.

A different Formulation

- Matrix space $\mathcal{B} = \langle B_1, B_2, \ldots, B_m \rangle \leq \mathbb{F}^{n\times n}$, consider the matrix
  - $B = x_1 B_1 + x_2 B_2 + \ldots + x_m B_m$ over the field $\mathbb{F}(x_1, x_2, \ldots, x_m)$ of rational functions.

Fact

If $|\mathbb{F}| > n$ *then* $\mathrm{rank}(\mathcal{B}) = \mathrm{rank}(B)$.

- Gives a randomized polynomial time algorithm using Schwartz–Zippel lemma.
  - Even an RNC algorithm.

Our contribution

- A deterministic PTAS for computing the Commutative rank.

Theorem

For any Matrix space $\mathcal{B} \leq \mathbb{F}^{n \times n}$ as input, a deterministic poly-time algorithm which outputs a matrix $A \in \mathcal{B}$ such that

$$\operatorname{rank}(A) \geq (1 - \epsilon) \operatorname{rank}(\mathcal{B}).$$

Algorithm runs in time $n^{O(\frac{1}{\epsilon})}$.

COMPUTATIONAL COMPLEXITY CONFERENCE

Non-commutative rank

Definition (c-shrunk subspace)

$V \leq \mathbb{F}^n$ is a c-shrunk subspace of $\mathcal{B} \leq \mathbb{F}^{n\times n}$, if $\mathrm{rank}(\mathcal{B}V) \leq \dim(V) - c$.

Definition (Non-commutative rank)

$\mathcal{B} \leq \mathbb{F}^{n\times n}$ any matrix space, if
$r = \max\{c \mid \exists\, c\text{-shrunk subspaceof } \mathcal{B}\}$ then
Non-commutaive rank of $\mathcal{B} = \mathrm{ncr}(\mathcal{B}) = n - r$.

Non-commutative rank

Definition (c-shrunk subspace)

$V \leq \mathbb{F}^n$ is a c-shrunk subspace of $\mathcal{B} \leq \mathbb{F}^{n \times n}$, if $\text{rank}(\mathcal{B}V) \leq \dim(V) - c$.

Definition (Non-commutative rank)

$\mathcal{B} \leq \mathbb{F}^{n \times n}$ any matrix space, if
$r = \max\{c \mid \exists\, c\text{-shrunk subspaceof } \mathcal{B}\}$ then
Non-commutaive rank of $\mathcal{B} = \text{ncr}(\mathcal{B}) = n - r$.

Problem

Lemma (Fortin and Reutenauer, 2004)

$$\mathrm{rank}(\mathcal{B}) \leq \mathrm{ncr}(\mathcal{B}) \leq 2 \cdot \mathrm{rank}(\mathcal{B})$$

Lemma (Derksen and Makam, 2016)

There exist $\mathcal{B} \leq \mathbb{F}^{n \times n}$ *such that* $\frac{\mathrm{ncr}(\mathcal{B})}{\mathrm{rank}(\mathcal{B})}$ *gets arbitrarily close to 2 as* $n \to \infty$.

Problem

Lemma (Fortin and Reutenauer, 2004)

$\text{rank}(\mathcal{B}) \leq \text{ncr}(\mathcal{B}) \leq 2 \cdot \text{rank}(\mathcal{B})$

Lemma (Derksen and Makam, 2016)

There exist $\mathcal{B} \leq \mathbb{F}^{n \times n}$ *such that* $\frac{\text{ncr}(\mathcal{B})}{\text{rank}(\mathcal{B})}$ *gets arbitrarily close to* 2 *as* $n \to \infty$.

Why study this problem?

- Generalizes several computational problems from algebra and combinatorics.
  - Bipartite matching
  - Linear Matroid intersection.
  - Maximum matching
  - Linear matroid parity problem
- Polynomial identity testing(PIT) of Algebraic branching programs(ABP)

Special cases

- NP-complete when the field $\mathbb{F}$ is of constant size.
- Deterministic polynomial time algorithms when B_i's all are of rank 1.
  - Subsumes bipartite maximum matching, linear matroid intersection.
  - Even a quasi-NC algorithm by [Gurjar and Thierauf, 2016].

Special cases

- NP-complete when the field $\mathbb{F}$ is of constant size.
- Deterministic polynomial time algorithms when B_i's all are of rank 1.
    - Subsumes bipartite maximum matching, linear matroid intersection.
    - Even a quasi-NC algorithm by [Gurjar and Thierauf, 2016].

Special cases

- NP-complete when the field $\mathbb{F}$ is of constant size.
- Deterministic polynomial time algorithms when B_i's all are of rank 1.
  - Subsumes bipartite maximum matching, linear matroid intersection.
  - Even a quasi-NC algorithm by [Gurjar and Thierauf, 2016].

Algorithms for Non-commutative rank

- Gurvits, 2004 : Deterministic poly-time algorithms for "compression spaces"
  - Matrix space $\mathcal{B}$ is a compression space if $\text{rank}(\mathcal{B}) = \text{ncr}(\mathcal{B})$.

Theorem (GGOW 2015, Ivanyos et al.,2015)

There is a deterministic poly-time algorithm which computes the $\text{ncr}(\mathcal{B})$ *for any matrix space* $\mathcal{B} \leq \mathbb{F}^{n \times n}$.

Algorithms for Non-commutative rank

- Gurvits, 2004 : Deterministic poly-time algorithms for "compression spaces"
  - Matrix space $\mathcal{B}$ is a compression space if $\mathrm{rank}(\mathcal{B}) = \mathrm{ncr}(\mathcal{B})$.

Theorem (GGOW 2015, Ivanyos et al.,2015)

There is a deterministic poly-time algorithm which computes the $\mathrm{ncr}(\mathcal{B})$ *for any matrix space* $\mathcal{B} \leq \mathbb{F}^{n \times n}$.

Approximation algorithms for Commutative rank

- Using $\text{rank}(\mathcal{B}) \leq \text{ncr}(\mathcal{B}) \leq 2 \cdot \text{rank}(\mathcal{B})$, one gets a deterministic poly-time algorithms for $\frac{1}{2}$-approximation of Commutative rank.
- These Non-commutative rank computation algorithms were the only algorithms which compute any constant factor approximation of the commutative rank.

Approximation algorithms for Commutative rank

- Leads to a natural question whether this approximation ratio of $\frac{1}{2}$ can be improved?
- We devise a deterministic poly-time algorithm which improves this approximation ratio to $1-\epsilon$ for arbitrary constant $0<\epsilon<1$.

Main Idea

- $\mathcal{B} = \langle B_1, B_2, \ldots, B_m \rangle \leq \mathbb{F}^{n \times n}$.
  - $B = x_1 B_1 + x_2 B_2 + \ldots + x_m B_m$ over the field $\mathbb{F}(x_1, x_2, \ldots, x_m)$.
- We have some $A \in \mathcal{B}$ with some rank r.
  - Want to find $A' \in \mathcal{B}$ with $\text{rank}(A') > r$.
- WLOG assume $A = \begin{bmatrix} I_r & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \ldots & 0 & 0 \end{bmatrix}$.
- Consider the matrix $A + B \in \mathbb{F}(x_1, x_2, \ldots, x_m)^{n \times n}$.

Main Idea

- $\mathcal{B} = \langle B_1, B_2, \ldots, B_m \rangle \leq \mathbb{F}^{n \times n}$.
  - $B = x_1 B_1 + x_2 B_2 + \ldots + x_m B_m$ over the field $\mathbb{F}(x_1, x_2, \ldots, x_m)$.
- We have some $A \in \mathcal{B}$ with some rank r.
  - Want to find $A' \in \mathcal{B}$ with $\text{rank}(A') > r$.
- WLOG assume $A = \begin{bmatrix} I_r & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \ldots & 0 & 0 \end{bmatrix}$.
- Consider the matrix $A + B \in \mathbb{F}(x_1, x_2, \ldots, x_m)^{n \times n}$.

Main idea(Cont.)

- $A + B = \begin{bmatrix} I_r + B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$.
- Suppose $B_{22} = 0$ then $\text{rank}(A+B) = \text{rank}(B) \leq 2r$.
  - $\text{rank}(A)$ is already $\frac{1}{2}$-approximation of $\text{rank}(B)$.
- Otherwise $B_{22} \neq 0$, $c(x_1, x_2, \ldots, x_m)$ be a non-zero entry of B_{22}.

COMPUTATIONAL COMPLEXITY CONFERENCE

Main idea(Cont.)

- $A + B = \begin{bmatrix} I_r + B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$.
- Suppose $B_{22} = 0$ then $\text{rank}(A + B) = \text{rank}(B) \leq 2r$.
  - $\text{rank}(A)$ is already $\frac{1}{2}$-approximation of $\text{rank}(B)$.
- Otherwise $B_{22} \neq 0$, $c(x_1, x_2, \ldots, x_m)$ be a non-zero entry of B_{22}.

Main idea(Cont.)

- $A + B = \begin{bmatrix} I_r + B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$.
- Suppose $B_{22} = 0$ then $\text{rank}(A + B) = \text{rank}(B) \leq 2r$.
  - $\text{rank}(A)$ is already $\frac{1}{2}$-approximation of $\text{rank}(B)$.
- Otherwise $B_{22} \neq 0$, $c(x_1, x_2, \ldots, x_m)$ be a non-zero entry of B_{22}.

COMPUTATIONAL COMPLEXITY CONFERENCE

Main idea(Cont.)

- Consider the Minor M of $A+B$ which has $c(x_1, x_2, \ldots, x_m)$ as the last entry.
  - $M =$

$$\begin{bmatrix} 1+\ell_{11} & \ell_{12} & \ldots & a_1 \\ \ell_{21} & 1+\ell_{22} & \ldots & a_2 \\ \vdots & \vdots & \ddots & \vdots \\ b_1 & b_2 & \ldots & c(x_1, x_2, \ldots, x_m) \end{bmatrix}_{(r+1)\times(r+1)}$$

- $\det(M(x_1, x_2, \ldots, x_m)) = c(x_1, x_2, \ldots, x_m) +$ terms of degree at least 2.

COMPUTATIONAL COMPLEXITY CONFERENCE

Main idea(Cont.)

- Consider the Minor M of $A+B$ which has $c(x_1, x_2, \ldots, x_m)$ as the last entry.
  - $M =$
$$\begin{bmatrix} 1+\ell_{11} & \ell_{12} & \ldots & a_1 \\ \ell_{21} & 1+\ell_{22} & \ldots & a_2 \\ \vdots & \vdots & \ddots & \vdots \\ b_1 & b_2 & \ldots & c(x_1, x_2, \ldots, x_m) \end{bmatrix}_{(r+1)\times(r+1)}$$
- $\det(M(x_1, x_2, \ldots, x_m)) =$ $c(x_1, x_2, \ldots, x_m) +$ terms of degree at least 2.

Final Step

- If we can find a setting of $x = \lambda_1, x_2 = \lambda_2, \ldots, x_m = \lambda_m$ such that $\det(M(\lambda_1, \lambda_2, \ldots, \lambda_m)) \neq 0$.
  - Then we get a rank $r+1$ matrix in $\mathcal{B}$.
  - $\det(M(x_1, x_2, \ldots, x_m))$ has degree 1 monomials.

Fact

If a non-zero polynomial $f(x_1, x_2, \ldots, x_m)$ has a degree k monomial and $\deg(f) \leq n$, then one can find a non-zero assignment $x_1 = \lambda_1, x_2 = \lambda_2, \ldots, x_m = \lambda_m$ for f, by trying $O((mn)^k)$ choices.

- Gives a "rank increasing assignment of x_i's" by trying $O(mn)$ choices.
- Gives a matrix of bigger rank in $\mathcal{B}$.

COMPUTATIONAL COMPLEXITY CONFERENCE

Final Step

- If we can find a setting of $x = \lambda_1, x_2 = \lambda_2, \ldots, x_m = \lambda_m$ such that $\det(M(\lambda_1, \lambda_2, \ldots, \lambda_m)) \neq 0$.
  - Then we get a rank $r + 1$ matrix in $\mathcal{B}$.
  - $\det(M(x_1, x_2, \ldots, x_m))$ has degree 1 monomials.

Fact

If a non-zero polynomial $f(x_1, x_2, \ldots, x_m)$ has a degree k monomial and $\deg(f) \leq n$, then one can find a non-zero assignment $x_1 = \lambda_1, x_2 = \lambda_2, \ldots, x_m = \lambda_m$ for f, by trying $O((mn)^k)$ choices.

- Gives a "rank increasing assignment of x_i's" by trying $O(mn)$ choices.
- Gives a matrix of bigger rank in $\mathcal{B}$.

Final Step

- If we can find a setting of $x = \lambda_1, x_2 = \lambda_2, \ldots, x_m = \lambda_m$ such that $\det(M(\lambda_1, \lambda_2, \ldots, \lambda_m)) \neq 0$.
  - Then we get a rank $r + 1$ matrix in $\mathcal{B}$.
  - $\det(M(x_1, x_2, \ldots, x_m))$ has degree 1 monomials.

Fact

If a non-zero polynomial $f(x_1, x_2, \ldots, x_m)$ has a degree k monomial and $\deg(f) \leq n$, then one can find a non-zero assignment $x_1 = \lambda_1, x_2 = \lambda_2, \ldots, x_m = \lambda_m$ for f, by trying $O((mn)^k)$ choices.

- Gives a "rank increasing assignment of x_i's" by trying $O(mn)$ choices.
- Gives a matrix of bigger rank in $\mathcal{B}$.

What if $B_{22} = 0$

- $B_{22} \neq 0$ was needed for rank increase.
- What if $B_{22} = 0$?$\Longrightarrow$ Only $\frac{1}{2}$-approximation.

- $B_{22} \neq 0$ made sure that $\det(M)$ has degree 1 monomials.
- What if we look for degree 2 monomials?
  - When does $\det(M)$ has degree two monomials?

COMPUTATIONAL COMPLEXITY CONFERENCE

What if $B_{22} = 0$

- $B_{22} \neq 0$ was needed for rank increase.
- What if $B_{22} = 0$?$\Longrightarrow$ Only $\frac{1}{2}$-approximation.

- $B_{22} \neq 0$ made sure that $\det(M)$ has degree 1 monomials.
- What if we look for degree 2 monomials?
  - When does $\det(M)$ has degree two monomials?

What if $B_{22} = 0$

- $B_{22} = 0$, consider any $(r+1) \times (r+1)$ minor M of $A + B$ with $I_r + B_{11}$ always being there.
- $M = \begin{bmatrix} 1+\ell_{11} & \ell_{12} & \dots & a_1 \\ \ell_{21} & 1+\ell_{22} & \dots & a_2 \\ \vdots & \vdots & \ddots & \vdots \\ b_1 & b_2 & \dots & 0 \end{bmatrix}_{(r+1)\times(r+1)}$

Lemma

If $B_{22} = 0$ then
$\det(M(x_1, x_2, \dots, x_m)) = -\sum_{i=1}^{r} a_i b_i +$ *terms of degree at least 3.*

COMPUTATIONAL COMPLEXITY CONFERENCE

What if $B_{22} = 0$

- $B_{22} = 0$, consider any $(r+1) \times (r+1)$ minor M of $A + B$ with $I_r + B_{11}$ always being there.
- $M = \begin{bmatrix} 1+\ell_{11} & \ell_{12} & \dots & a_1 \\ \ell_{21} & 1+\ell_{22} & \dots & a_2 \\ \vdots & \vdots & \ddots & \vdots \\ b_1 & b_2 & \dots & 0 \end{bmatrix}_{(r+1)\times(r+1)}$

Lemma

If $B_{22} = 0$ *then*
$\det(M(x_1, x_2, \dots, x_m)) = -\sum_{i=1}^{r} a_i b_i +$ *terms of degree at least 3.*

COMPUTATIONAL COMPLEXITY CONFERENCE

$\frac{2}{3}$-approximation

- If degree two terms for all choices of M are zero then
  - $B_{21}B_{12} = 0$
  - $B_{22} = 0$

Lemma

Above conditions imply that $\operatorname{rank}(B) \leq \frac{3}{2}r$.

Proof.

If $\operatorname{rank}(B_{12}) \leq \frac{r}{2}$ then trivial. Otherwise rank $\operatorname{rank}(B_{21}) \leq \frac{r}{2}$ by rank-nullity theorem. Either way, $\operatorname{rank}(B) \leq \frac{3}{2}r$. □

- Thus if no degree 2 terms then we are done already
  - Otherwise increase the rank by trying $O((mn)^2)$ choices.

$\frac{2}{3}$-approximation

- If degree two terms for all choices of M are zero then
  - $B_{21}B_{12} = 0$
  - $B_{22} = 0$

Lemma

Above conditions imply that $\mathrm{rank}(B) \leq \frac{3}{2}r$.

Proof.

If $\mathrm{rank}(B_{12}) \leq \frac{r}{2}$ then trivial. Otherwise rank $\mathrm{rank}(B_{21}) \leq \frac{r}{2}$ by rank-nullity theorem. Either way, $\mathrm{rank}(B) \leq \frac{3}{2}r$. □

- Thus if no degree 2 terms then we are done already
  - Otherwise increase the rank by trying $O((mn)^2)$ choices.

$\frac{2}{3}$-approximation

- If degree two terms for all choices of M are zero then
  - $B_{21}B_{12} = 0$
  - $B_{22} = 0$

Lemma

Above conditions imply that $\mathrm{rank}(B) \leq \frac{3}{2}r$.

Proof.

If $\mathrm{rank}(B_{12}) \leq \frac{r}{2}$ then trivial. Otherwise rank $\mathrm{rank}(B_{21}) \leq \frac{r}{2}$ by rank-nullity theorem. Either way, $\mathrm{rank}(B) \leq \frac{3}{2}r$. □

- Thus if no degree 2 terms then we are done already
  - Otherwise increase the rank by trying $O((mn)^2)$ choices.

Degree 3 terms

- We saw that if degree one and degree two terms for all choices of M are zero then
  - $B_{21}B_{12} = 0$
  - $B_{22} = 0$
- What if degree three terms are also zero?

Lemma

If degree 1,2 and 3 terms are all zero in $\det(M)$ for all M then $B_{22} = 0$, $B_{21}B_{12} = 0$ and $B_{21}B_{11}B_{12} = 0$.

$\frac{3}{4}$-approximation

Lemma

Above conditions imply that $\text{rank}(B) \leq \frac{4}{3}r$.

- Thus if no degree 1,2,3 terms then we are done already.
  - Otherwise increase the rank by trying $O((mn)^3)$ choices.

COMPUTATIONAL COMPLEXITY CONFERENCE

$\frac{3}{4}$-approximation

Lemma

Above conditions imply that $\mathrm{rank}(B) \leq \frac{4}{3}r$.

- Thus if no degree 1,2,3 terms then we are done already.
  - Otherwise increase the rank by trying $O((mn)^3)$ choices.

Generalizing above ideas

- We have some $A \in \mathcal{B}$, with $\mathrm{rank}(A) = r$.
- Above discussion hints to the following conjecture.

Conjecture

For any $k \leq n$, either $\mathrm{rank}(\mathcal{B}) \leq r\left(1 + \frac{1}{k}\right)$ or we can increase the rank by trying $O((mn)^k)$ choices.

- We prove this conjecture by so called "Wong Sequences".

COMPUTATIONAL COMPLEXITY CONFERENCE

Generalizing above ideas

- We have some $A \in \mathcal{B}$, with $\mathrm{rank}(A) = r$.
- Above discussion hints to the following conjecture.

Conjecture

For any $k \leq n$, either $\mathrm{rank}(\mathcal{B}) \leq r\left(1 + \frac{1}{k}\right)$ or we can increase the rank by trying $O((mn)^k)$ choices.

- We prove this conjecture by so called "Wong Sequences".

Final algorithm

- Set $k = O\left(\frac{1}{\epsilon}\right)$ and we get the desired approximation ratio.
- Running time is $n^{O\left(\frac{1}{\epsilon}\right)}$.
- We also show tight examples where this approach does not give better than $(1-\epsilon)$ approximation ratio.
  - So analysis above is tight.

Final algorithm

- Set $k = O\left(\frac{1}{\epsilon}\right)$ and we get the desired approximation ratio.
- Running time is $n^{O\left(\frac{1}{\epsilon}\right)}$.
- We also show tight examples where this approach does not give better than $(1-\epsilon)$ approximation ratio.
  - So analysis above is tight.

Thanks

Thanks for listening