A polynomial time interior point method for problems with nonconvex constraints

Oliver Hinder, Yinyu Ye

Department of Management Science and Engineering
Stanford University

July 19, 2018
The problem

Consider problem

\[
\min_{x \in \mathbb{R}^d} f(x) \text{ s.t. } a(x) \leq 0
\]

where \( f : \mathbb{R}^d \to \mathbb{R} \) and \( a : \mathbb{R}^d \to \mathbb{R}^m \) have Lipschitz first and second derivatives.
Consider problem

$$\min_{x \in \mathbb{R}^d} f(x) \text{ s.t. } a(x) \leq 0$$

where $f : \mathbb{R}^d \to \mathbb{R}$ and $a : \mathbb{R}^d \to \mathbb{R}^m$ have Lipschitz first and second derivatives.

In worst-case finding global optima takes an exponential amount of time.
The problem

Consider problem

\[
\min_{x \in \mathbb{R}^d} f(x) \text{ s.t. } a(x) \leq 0
\]

where \( f : \mathbb{R}^d \to \mathbb{R} \) and \( a : \mathbb{R}^d \to \mathbb{R}^m \) have Lipschitz first and second derivatives.

- In worst-case finding global optima takes an exponential amount of time.
- Instead we want to find an ‘approximate local optima’, more precisely a Fritz John point.
µ-approximate Fritz John point

Want to find primal variables $x$, dual variables $y$ for which

$$a(x) < 0$$

$$\|\nabla_x f(x) - y^T \nabla a(x)\|_2 \leq \mu \sqrt{\|y\|_1 + 1} \quad y > 0$$

$$\frac{y_i a_i(x)}{\mu} \in [1/2, 3/2] \quad \forall i \in \{1, \ldots, m\}$$

where $\mu > 0$ is a small constant.
**μ-approximate Fritz John point**

Want to find primal variables $x$, dual variables $y$ for which

\[
a(x) < 0
\]

\[
\left\| \nabla_x f(x) - y^T \nabla a(x) \right\|_2 \leq \mu \sqrt{\|y\|_1 + 1} \quad y > 0
\]

\[
\frac{y_i a_i(x)}{\mu} \in \left[\frac{1}{2}, \frac{3}{2}\right] \quad \forall i \in \{1, \ldots, m\}
\]

where $\mu > 0$ is a small constant.

- Fritz John is a necessary condition for local optimality
- MFCQ constraint qualification, i.e., dual multipliers are bounded then Fritz John = KKT point
How do we solve this problem?

One popular approach, move constraints into a log barrier:

$$\psi(\mu(x)) := f(x) - \mu \sum_{i=1}^{m} \log(-a_i(x))$$

and apply (modified) Newton's method to find an approximate local optima.

Examples of practical codes using this method include IPOPT, KNITRO, LOQO, one-phase (my code), etc.

Local superlinear convergence is known.

It'd be nice to understand how reliable existing algorithms are and develop more reliable algorithms (e.g., one-phase).

Reliability is important for practical problems, e.g., AC optimal powerflow.

Understand worst-case global convergence?
How do we solve this problem?

One popular approach, move constraints into a log barrier:

\[ \psi_\mu(x) := f(x) - \mu \sum_{i=1}^{m} \log(-a_i(x)) \]

and apply (modified) Newton’s method to find an approximate local optima.
How do we solve this problem?

▶ One popular approach, move constraints into a log barrier:

\[
\psi_\mu(x) := f(x) - \mu \sum_{i=1}^{m} \log(-a_i(x))
\]

and apply (modified) Newton’s method to find an approximate local optima.

▶ Examples of practical codes using this method include IPOPT, KNITRO, LOQO, one-phase (my code), etc.
一个问题的解决方法是什么？

▶ 一种流行的方法是将约束项移到对数障碍上：

\[ \psi_\mu(x) := f(x) - \mu \sum_{i=1}^{m} \log(-a_i(x)) \]

然后应用（修改后的）牛顿方法找到一个近似局部最优解。

▶ 实际上使用这种方法的示例代码包括 IPOPT、KNITRO、LOQO、一阶段（我的代码），等等。

▶ 确认局部超线性收敛是已知的。

▶ 了解现有算法的可靠性，并开发更可靠的算法（例如一阶段）是有利的。

▶ 可靠性对于实际问题（例如 AC 最优功率流）非常重要。

▶ 了解最坏情况下的全局收敛性是很有价值的。
One popular approach, move constraints into a log barrier:

\[ \psi_\mu(x) := f(x) - \mu \sum_{i=1}^{m} \log(-a_i(x)) \]

and apply (modified) Newton’s method to find an approximate local optima.

Examples of practical codes using this method include IPOPT, KNITRO, LOQO, one-phase (my code), etc.

Local superlinear convergence is known.

It’d be nice to understand how reliable existing algorithms are and develop more reliable algorithms (e.g., one-phase).
How do we solve this problem?

- One popular approach, move constraints into a log barrier:
  \[
  \psi_\mu(x) := f(x) - \mu \sum_{i=1}^{m} \log(-a_i(x))
  \]
  and apply (modified) Newton’s method to find an approximate local optima.
- Examples of practical codes using this method include IPOPT, KNITRO, LOQO, one-phase (my code), etc.
- Local superlinear convergence is known.
- It’d be nice to understand how reliable existing algorithms are and develop more reliable algorithms (e.g., one-phase).
- Reliability is important for practical problems, e.g., AC optimal powerflow.
How do we solve this problem?

- One popular approach, move constraints into a log barrier:

\[
\psi_\mu(x) := f(x) - \mu \sum_{i=1}^{m} \log(-a_i(x))
\]

and apply (modified) Newton’s method to find an approximate local optima.

- Examples of practical codes using this method include IPOPT, KNITRO, LOQO, one-phase (my code), etc.

- Local superlinear convergence is known.

- It’d be nice to understand how reliable existing algorithms are and develop more reliable algorithms (e.g., one-phase).

- Reliability is important for practical problems, e.g., AC optimal powerflow.

- Understand worst-case global convergence?
For log barrier methods

\[ \psi_\mu(x) := f(x) - \mu \sum_{i=1}^{m} \log(-a_i(x)) \]

we show:

1. Existing convergence proofs give (implicit) exponential runtime bounds to find \( \mu \)-approximate Fritz John point.
2. Gradient descent with careful analysis requires \( O(\mu^{-3}) \) iterations to find \( \mu \)-approximate Fritz John point.
3. Trust-region method (similar flavor to existing IPMs) requires
   - \( O(\mu^{-7/4}) \) iterations to find \( \mu \)-approximate Fritz John point.
   - In convex case, \( O(\epsilon^{-2/3}) \) iterations to find approximate \( \epsilon \)-optimal solution.
Literature review

- IPM theory
- complexity of nonconvex optimization
Brief history of IPM theory
Brief history of IPM theory

Linear programming

Birth of interior point methods
[Karmarkar, 1984]

\[ O(m \log(1/\epsilon)) \]
Brief history of IPM theory

Linear programming

Birth of interior point methods
[Karmarkar, 1984]

\[ O(m \log(1/\epsilon)) \]

Log barrier + Newton
[Regnar, 1988]

\[ O(\sqrt{m \log(1/\epsilon)}) \]
Brief history of IPM theory

Conic optimization

Linear programming

General objective/constraints

Birth of interior point methods
[Karmarkar, 1984]

\[ O(m \log(1/\epsilon)) \]

Log barrier + Newton
[Regnar, 1988]

\[ O(\sqrt{m \log(1/\epsilon)}) \]
Brief history of IPM theory

**Conic optimization**

- Birth of interior point methods [Karmarkar, 1984]
  \[ O\left(m \log\left(\frac{1}{\epsilon}\right)\right) \]
- Log barrier + Newton [Regnar, 1988]
  \[ O\left(\sqrt{m} \log\left(\frac{1}{\epsilon}\right)\right) \]
- Self-concordant barriers [Nesterov, Nemirovskii, 1994]
  \[ O\left(\sqrt{v} \log\left(\frac{1}{\epsilon}\right)\right) \]

**Linear programming**

- Successful implementations: MOSEK, GUROBI, ...

**General objective/constraints**

- Successful implementations: LOQO, KNTIRO, IPOPT, ...

- Successful implementations: SEDUMI, MOSEK, GUROBI, ...
Brief history of IPM theory

**Conic optimization**

Self-concordant barriers
[Nesterov, Nemirovskii, 1994]

\[O(\sqrt{v} \log(1/\epsilon))\]

**Linear programming**

Birth of interior point methods
[Karmarkar, 1984]

\[O(m \log(1/\epsilon))\]

Log barrier + Newton
[Regnar, 1988]

\[O(\sqrt{m} \log(1/\epsilon))\]

**General objective/constraints**

\[O(m \log(1/\epsilon))\]
Brief history of IPM theory

- **Conic optimization**
  - Birth of interior point methods [Karmarkar, 1984]
    - $O(m \log(1/\epsilon))$
  - Log barrier + Newton [Regnar, 1988]
    - $O(\sqrt{m} \log(1/\epsilon))$
  - Self-concordant barriers [Nesterov, Nemirovskii, 1994]
    - $O(\sqrt{v} \log(1/\epsilon))$

- **Linear programming**
  - Successful implementations: MOSEK, GUROBI, ...

- **General objective/constraints**
  - Successful implementations: LOQO, KNTIRO, IPOPT, ...

- **Good theoretical understanding**
  - Linear programming
  - Conic optimization

- **Less theoretical understanding**
  - General objective/constraints
Unconstrained. Goal: find a point with \( \| \nabla f(x) \|_2 \leq \epsilon \).
Unconstrained. Goal: find a point with $\|\nabla f(x)\|_2 \leq \epsilon$.

- Gradient descent $O(\epsilon^{-2})$. 

Cubic regularization $O(\epsilon^{-3/2})$ [Nesterov, Polyak, 2006].

Best achievable dimension-free runtimes for $\nabla f$ and $\nabla^2 f$ Lipschitz respectively [Carmon, Duchi, Hinder, Sidford, 2017].

Nonconvex objective, linear inequality constraints. Goal: find KKT point.


Nonconvex objective and constraints. Goal: find (scaled) KKT point.

- Apply cubic regularization to quadratic penalty function [Birgin et al, 2016].
- Sequential linear programming [Cartis et al, 2015].

Both these methods ‘plug and play’ directly with unconstrained optimization methods. For example, [Birgin et al, 2016] use that their penalty functions have Lipschitz derivatives.
Unconstrained. Goal: find a point with $\|\nabla f(x)\|_2 \leq \epsilon$.

- Gradient descent $O(\epsilon^{-2})$.
- Cubic regularization $O(\epsilon^{-3/2})$ [Nesterov, Polyak, 2006].
Unconstrained. Goal: find a point with $\|\nabla f(x)\|_2 \leq \epsilon$.

- Gradient descent $O(\epsilon^{-2})$.
- Cubic regularization $O(\epsilon^{-3/2})$ [Nesterov, Polyak, 2006].
- Best achievable dimension-free runtimes for $\nabla f$ and $\nabla^2 f$ Lipschitz respectively [Carmon, Duchi, Hinder, Sidford, 2017].
Unconstrained. Goal: find a point with $\|\nabla f(x)\|_2 \leq \epsilon$.

- Gradient descent $O(\epsilon^{-2})$.
- Cubic regularization $O(\epsilon^{-3/2})$ [Nesterov, Polyak, 2006].
- Best achievable dimension-free runtimes for $\nabla f$ and $\nabla^2 f$ Lipschitz respectively [Carmon, Duchi, Hinder, Sidford, 2017].

Nonconvex objective, linear inequality constraints. Goal: find KKT point.

Literature review: complexity of nonconvex optimization

- Unconstrained. Goal: find a point with $\|\nabla f(x)\|_2 \leq \epsilon$.
  - Gradient descent $O(\epsilon^{-2})$.
  - Cubic regularization $O(\epsilon^{-3/2})$ [Nesterov, Polyak, 2006].
  - Best achievable dimension-free runtimes for $\nabla f$ and $\nabla^2 f$ Lipschitz respectively [Carmon, Duchi, Hinder, Sidford, 2017].
- Nonconvex objective, linear inequality constraints. Goal: find KKT point.
- Nonconvex objective and constraints. Goal: find (scaled) KKT point.
Unconstrained. Goal: find a point with $\|\nabla f(x)\|_2 \leq \epsilon$.
- Gradient descent $O(\epsilon^{-2})$.
- Cubic regularization $O(\epsilon^{-3/2})$ [Nesterov, Polyak, 2006].
- Best achievable dimension-free runtimes for $\nabla f$ and $\nabla^2 f$ Lipschitz respectively [Carmon, Duchi, Hinder, Sidford, 2017].

Nonconvex objective, linear inequality constraints. Goal: find KKT point.

Nonconvex objective and constraints. Goal: find (scaled) KKT point.
- Apply cubic regularization to quadratic penalty function [Birgin et al, 2016].
Unconstrained. Goal: find a point with $\|\nabla f(x)\|_2 \leq \epsilon$.

- Gradient descent $O(\epsilon^{-2})$.
- Cubic regularization $O(\epsilon^{-3/2})$ [Nesterov, Polyak, 2006].
- Best achievable dimension-free runtimes for $\nabla f$ and $\nabla^2 f$ Lipschitz respectively [Carmon, Duchi, Hinder, Sidford, 2017].

Nonconvex objective, linear inequality constraints. Goal: find KKT point.


Nonconvex objective and constraints. Goal: find (scaled) KKT point.

- Apply cubic regularization to quadratic penalty function [Birgin et al, 2016].
- Sequential linear programming [Cartis et al, 2015].
Unconstrained. Goal: find a point with $\|\nabla f(x)\|_2 \leq \epsilon$.
- Gradient descent $O(\epsilon^{-2})$.
- Cubic regularization $O(\epsilon^{-3/2})$ [Nesterov, Polyak, 2006].
- Best achievable dimension-free runtimes for $\nabla f$ and $\nabla^2 f$ Lipschitz respectively [Carmon, Duchi, Hinder, Sidford, 2017].

Nonconvex objective, linear inequality constraints. Goal: find KKT point.

Nonconvex objective and constraints. Goal: find (scaled) KKT point.
- Apply cubic regularization to quadratic penalty function [Birgin et al, 2016].
- Sequential linear programming [Cartis et al, 2015].
- Both these methods ‘plug and play’ directly with unconstrained optimization methods. For example, [Birgin et al, 2016] use that their penalty functions have Lipschitz derivatives.
Why traditional analysis fails and how to fix it for gradient descent!
Why a traditional nonlinear programming analysis fails

Derivatives are moving very quickly and have exponentially large Lipshitz constant in $\mu$. Region iterates must lie in:

$$S = \{x : \psi_{\mu}(x) \leq \psi_{\mu}(1)\}$$

Traditional approach [Byrd, et al. (2000), Chen and Goldfarb (2006), ...] to proving global convergence:
Why a traditional nonlinear programming analysis fails

Traditional approach [Byrd, et al. (2000), Chen and Goldfarb (2006), ...] to proving global convergence:

- Define a set $S$ where the iterates must lie.

Derivatives are moving very quickly and have exponentially large Lipshitz constant in $\mu$.

Initial point $x^{(0)} = 1$

Region iterates must lie in:

$$S = \{ x : \psi_\mu(x) \leq \psi_\mu(1) \}$$
Why a traditional nonlinear programming analysis fails

Traditional approach [Byrd, et al. (2000), Chen and Goldfarb (2006), ...] to proving global convergence:

- Define a set $S$ where the iterates must lie.
- Prove derivatives are Lipschitz on $S$ and apply a standard descent method.

Derivatives are moving very quickly and have exponentially large Lipschitz constant in $\mu$. The initial point $x^{(0)} = 1$.

Region iterates must lie in:

$$S = \{ x : \psi_{\mu}(x) \leq \psi_{\mu}(1) \}$$
Why a traditional nonlinear programming analysis fails

Derivatives are moving very quickly and have exponentially large Lipschitz constant in $\mu$.

Region iterates must lie in:
$$S = \{ x : \psi_\mu(x) \leq \psi_\mu(1) \}$$

Traditional approach [Byrd, et al. (2000), Chen and Goldfarb (2006), ...] to proving global convergence:

- Define a set $S$ where the iterates must lie.
- Prove derivatives are Lipschitz on $S$ and apply a standard descent method.
- Gives runtime bound is worse than $\exp(1/\mu)$!
Gradient descent on the log barrier

Gradient descent:

\[ x^{(k+1)} \leftarrow x^{(k)} - \alpha^{(k)} \nabla \psi_\mu(x^{(k)}). \]

Using a fixed Lipschitz constant for the whole set \( S \) is analogous to using a fixed step size, i.e., \( \alpha = \alpha^{(k)} \),
Gradient descent on the log barrier

Gradient descent:

\[ x^{(k+1)} \leftarrow x^{(k)} - \alpha^{(k)} \nabla \psi_{\mu}(x^{(k)}). \]

Using a fixed Lipschitz constant for the whole set \( S \) is analogous to using a fixed step size, i.e., \( \alpha = \alpha^{(k)} \), leads to exponential runtime bounds!
Gradient descent on the log barrier

Gradient descent:

\[ x^{(k+1)} \leftarrow x^{(k)} - \alpha^{(k)} \nabla \psi_{\mu}(x^{(k)}). \]

Using a fixed Lipschitz constant for the whole set \( S \) is analogous to using a fixed step size, i.e., \( \alpha = \alpha^{(k)} \), leads to exponential runtime bounds!

![Derivatives diagram](image)
Gradient descent on the log barrier

Gradient descent:

\[ x^{(k+1)} \leftarrow x^{(k)} - \alpha^{(k)} \nabla \psi_\mu(x^{(k)}). \]

Using a fixed Lipschitz constant for the whole set \( S \) is analogous to using a fixed step size, i.e., \( \alpha = \alpha^{(k)} \), leads to exponential runtime bounds!

Choose \( \alpha^{(k)} \) with backtracking line search s.t. Armijo rule holds, i.e,

\[ \psi_\mu(x^{(k)} + \alpha^{(k)} d^{(k)}) \leq \psi_\mu(x^{(k)}) + c\alpha^{(k)} \nabla \psi_\mu(x^{(k)})^T d^{(k)} \]

for fixed \( c \in (0, 1) \).
We prove gradient descent using Armijo rule obtains $O(\mu^{-3})$ runtime to find point with $\|\nabla \psi_\mu(x)\|_2 \leq \mu(\|y\|_1 + 1)$. 

Since $\psi_\mu$ is bounded below result follows.
We prove gradient descent using Armijo rule obtains $O(\mu^{-3})$ runtime to find a point with $\|\nabla \psi_\mu(x)\|_2 \leq \mu(\|y\|_1 + 1)$. 

Intuition for what Armjio rule does:

- $\|\nabla \psi_\mu(x)\|_2$ and local Lipschitz constant are small.
- $\|\nabla \psi_\mu(x)\|_2$ and local Lipschitz constant are big.

Since $\psi_\mu$ is bounded below, the result follows.
We prove gradient descent using Armijo rule obtains $O(\mu^{-3})$ runtime to find point with $\|\nabla \psi_\mu(x)\|_2 \leq \mu(\|y\|_1 + 1)$.

Intuition for what Armjio rule does:

- If termination criterion is not satisfied we reduce $\psi_\mu$ by
  
  $\frac{\|\nabla \psi(x^{(k)})\|_2^2}{L(x^{(k)})} = \alpha^{(k)}\|\nabla \psi(x^{(k)})\|_2^2 = \Omega(\mu^3)$
We prove gradient descent using Armijo rule obtains $O(\mu^{-3})$ runtime to find point with $\|\nabla \psi_\mu(x)\|_2 \leq \mu(\|y\|_1 + 1)$.

Intuition for what Armijo rule does

If termination criterion is not satisfied we reduce $\psi_\mu$ by

$$\frac{\|\nabla \psi(x^{(k)})\|^2}{L(x^{(k)})} = \alpha^{(k)}\|\nabla \psi(x^{(k)})\|^2 = \Omega(\mu^3)$$

Since $\psi_\mu$ is bounded below result follows.

We don’t use gradient descent in practice ...
Trust-region log barrier method

1. The method.
2. Nonconvex runtime bounds.
3. Convex runtime bounds.
Each iteration of our IPM we:

1. Pick trust region size $r \approx \mu^{3/4} (1 + \|y\|_1)^{-1/2}$.
2. Compute direction $d_x$ by solving trust region problem.
3. Pick step size $\alpha \in (0, 1]$ to ensure $x_{\text{new}} = x + \alpha d_x$ satisfies $a(x) < 0$.
4. Is new point Fritz John point? If no then return to step one, otherwise terminate.
Our trust-region log barrier method

Each iteration of our IPM we:

1. Pick trust region size $r \approx \mu^{3/4} (1 + \|y\|_1)^{-1/2}$.

2. Compute direction $d_x$ by solving trust region problem.

3. Pick step size $\alpha \in (0, 1]$ to ensure $x_{\text{new}} = x + \alpha d_x$ satisfies $a(x) < 0$.

4. Is new point Fritz John point? If no then return to step one, otherwise terminate.
Our trust-region log barrier method

1. Pick trust region size $r \approx \mu^{3/4}(1 + \|y\|_1)^{-1/2}$.
2. Compute direction $d_x$ by solving trust region problem.
3. Pick step size $\alpha \in (0, 1]$ to ensure $x_{\text{new}} = x + \alpha d_x$ satisfies $a(x) < 0$.
4. Is new point Fritz John point? If no then return to step one, otherwise terminate.
Our trust-region log barrier method

Reminder:

\[ \psi_\mu(x) := f(x) - \mu \sum_{i=1}^{m} \log(-a_i(x)). \]
Reminder:

\[ \psi_\mu(x) := f(x) - \mu \sum_{i=1}^{m} \log(-a_i(x)). \]
Each iteration of our IPM we:

1. Pick trust region size $r \approx \mu^{3/4}(1 + \|y\|_1)^{-1/2}$.
2. Compute direction $d_x$ by solving trust region problem.
3. Pick step size $\alpha \in (0, 1]$ to ensure $x_{new} = x + \alpha d_x$ satisfies $a(x) < 0$.
4. Is new point Fritz John point? If no then return to step one, otherwise terminate.
Our trust-region log barrier method

Each iteration of our IPM we:

1. Pick trust region size $r \approx \mu^{3/4}(1 + \|y\|_1)^{-1/2}$.
2. Compute direction $d_x$ by solving trust region problem.
3. Pick step size $\alpha \in (0, 1]$ to ensure $x_{\text{new}} = x + \alpha d_x$ satisfies $a(x) < 0$.
4. Is new point Fritz John point? If no then return to step one, otherwise terminate.
Our trust-region log barrier method

Each iteration of our IPM we:

1. Pick trust region size $r \approx \mu^{3/4}(1 + \|y\|_1)^{-1/2}$.
2. **Compute direction $d_x$ by solving trust region problem.**
3. Pick step size $\alpha \in (0, 1]$ to ensure $x_{\text{new}} = x + \alpha d_x$ satisfies $a(x) < 0$.
4. Is new point Fritz John point? If no then return to step one, otherwise terminate.
Our trust-region log barrier method

Each iteration of our IPM we:

1. Pick trust region size \( r \approx \mu^{3/4}(1 + \|y\|_1)^{-1/2} \).
2. Compute direction \( d_x \) by solving trust region problem.
3. **Pick step size** \( \alpha \in (0, 1] \) **to ensure** \( x_{\text{new}} = x + \alpha d_x \) **satisfies** \( a(x) < 0 \).
4. Is new point Fritz John point? If no then return to step one, otherwise terminate.
Each iteration of our IPM we:

1. Pick trust region size $r \approx \mu^{3/4}(1 + \|y\|_1)^{-1/2}$.
2. Compute direction $d_x$ by solving trust region problem.
3. **Pick step size** $\alpha \in (0, 1]$ **to ensure** $x_{\text{new}} = x + \alpha d_x$ **satisfies** $a(x) < 0$.
4. Is new point Fritz John point? If no then return to step one, otherwise terminate.
Each iteration of our IPM we:

1. Pick trust region size $r \approx \mu^{3/4}(1 + \|y\|_1)^{-1/2}$.
2. Compute direction $d_x$ by solving trust region problem.
3. Pick step size $\alpha \in (0, 1]$ to ensure $x_{\text{new}} = x + \alpha d_x$ satisfies $a(x) < 0$.
4. Is new point Fritz John point? If no then return to step one, otherwise terminate.
Each iteration of our IPM we:

1. Pick trust region size $r \approx \mu^{3/4}(1 + \|y\|_1)^{-1/2}$.
2. Compute direction $d_x$ by solving trust region problem.
3. Pick step size $\alpha \in (0, 1]$ to ensure $x_{\text{new}} = x + \alpha d_x$ satisfies $a(x) < 0$.
4. Is new point Fritz John point? If no then return to step one, otherwise terminate.
Theorem 1

Assume:

- Strictly feasible initial point $x^{(0)}$.
- $\nabla f, \nabla a, \nabla^2 f, \nabla^2 a$ are Lipschitz.

Then our IPM takes as most

$$\mathcal{O} \left( \left( \psi_\mu(x^{(0)}) - \inf_z \psi_\mu(z) \right) \mu^{-7/4} \right)$$

trust region steps to terminate with a $\mu$-approximate second-order Fritz John point.
Theorem 1

Assume:
- Strictly feasible initial point $x^{(0)}$.
- $\nabla f$, $\nabla a$, $\nabla^2 f$, $\nabla^2 a$ are Lipschitz.

Then our IPM takes as most

$$
O\left(\left(\psi_\mu(x^{(0)}) - \inf_z \psi_\mu(z)\right) \mu^{-7/4}\right)
$$

trust region steps to terminate with a $\mu$-approximate second-order Fritz John point.

<table>
<thead>
<tr>
<th>algorithm</th>
<th># iteration</th>
<th>primitive</th>
<th>evaluates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Birgin et al., 2016</td>
<td>$O(\mu^{-3})$</td>
<td>vector operation</td>
<td>$\nabla$</td>
</tr>
<tr>
<td>Cartis et al., 2011</td>
<td>$O(\mu^{-2})^*$</td>
<td>linear program</td>
<td>$\nabla$</td>
</tr>
<tr>
<td>Birgin et al., 2016</td>
<td>$O(\mu^{-2})$</td>
<td>KKT of NCQP</td>
<td>$\nabla, \nabla^2$</td>
</tr>
<tr>
<td><strong>IPM (this paper)</strong></td>
<td>$O(\mu^{-7/4})^{**}$</td>
<td>linear system</td>
<td>$\nabla, \nabla^2$</td>
</tr>
</tbody>
</table>

*Weaker scaled KKT criterion. **Using a Birgin et al., 2016 style termination criterion.
Convex result

Modify our IPM have sequence of decreasing $\mu$, i.e., $\mu^{(j)}$ with $\mu^{(j)} \to 0$.

**Theorem 2**

Assume:

- Objective and constraints are convex.
- Strictly feasible initial point $x^{(0)}$ and feasible region is bounded.
- $f$, $\nabla f$, $\nabla a$, $\nabla^2 f$, $\nabla^2 a$ are Lipschitz.
- Slaters condition holds.
- $m$ is number of constraints.

Then our IPM starting takes at most

$$\tilde{O} \left( m^{1/3} \epsilon^{-2/3} + m^2 \right)$$

trust region steps to terminate with a $\epsilon$-optimal solution.
For log barrier methods

\[ \psi_\mu(x) := f(x) - \mu \sum_{i=1}^{m} \log(-a_i(x)) \]

we show:

1. Existing convergence proofs give exponential runtime bounds to find \( \mu \)-approximate Fritz John point.
2. Gradient descent with careful analysis requires \( O(\mu^{-3}) \) iterations to find \( \mu \)-approximate Fritz John point.
3. Trust-region method (similar flavor to existing IPMs) requires
   - \( O(\mu^{-7/4}) \) iterations to find \( \mu \)-approximate Fritz John point.
   - \( O(\epsilon^{-2/3}) \) iterations to find approximate \( \epsilon \)-optimal solution.
Summary

For log barrier methods

\[ \psi_\mu(x) := f(x) - \mu \sum_{i=1}^{m} \log(-a_i(x)) \]

we show:

1. Existing convergence proofs give exponential runtime bounds to find \( \mu \)-approximate Fritz John point.
2. Gradient descent with careful analysis requires \( O(\mu^{-3}) \) iterations to find \( \mu \)-approximate Fritz John point.
3. Trust-region method (similar flavor to existing IPMs) requires
   - \( O(\mu^{-7/4}) \) iterations to find \( \mu \)-approximate Fritz John point.
   - \( O(\epsilon^{-2/3}) \) iterations to find approximate \( \epsilon \)-optimal solution.

Although what we present here is a naive algorithm it influenced the development of our practical one-phase IPM code.
Questions?
References


