TOR Group Presentation

Power Optimisation in Wireless Sensor Networks

Continuous Chris, Differentiable Dominic, GretaTeX & MATLAB Mark

May 27, 2021

Problem Adaptation

The NLP is non-differentiable where the max term flips, *i.e.*, when it moves into the region consisting of all points of the plane further from one sensor than to any other. The solution holds for a system with n sensors, with the lines drawn at the boundary of each Voronoi cell.

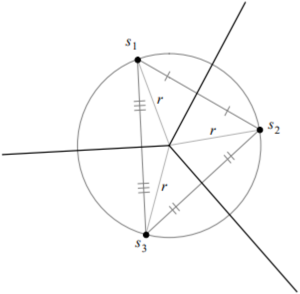

Figure: $|X| = 3$; $\mathrm{s}, X \in \mathbb{R}^{2\times n}$: Lines consisting of all points at which the problem is not differentiable.

Problem Adaptation

Our adapted NLP:

$$P(s) = \sum_{i=1}^{n} \|s - x_i\|^2 + \beta$$

$$\text{s.t.} \quad g_1(\mathrm{s}) = \|\mathrm{s} - \mathrm{x}_1\|^2 \leq \beta$$

$$\vdots$$

$$g_n(\mathrm{s}) = \|\mathrm{s} - \mathrm{x}_n\|^2 \leq \beta$$

$$\text{where } \beta = \max\{\|\mathrm{s} - \mathrm{x}_i\|^2\} \ \ \forall i \in \{1, n\}$$

Problem Adaptation

Restructuring this into an l_2-penalty function, we get:

$$\beta = \max\{\|\mathrm{s} - \mathrm{x}_i\|^2\} \;\; \forall i \in \{1, n\}$$

$$\mathcal{P}(s) = \sum_{i=1}^{n} \|s - x_i\|^2 + \beta + \frac{\alpha}{2} \sum_{i=1}^{n} g_i(\mathrm{s})_+^2$$

$$\text{where } g_i(\mathrm{s}) = \|\mathrm{s} - \mathrm{x}_i\|^2 - \beta, \quad \forall i \in \{1, n\}$$

$$\text{and } g_i(\mathrm{s})_+ := \begin{cases} g_i(\mathrm{s}) & \text{if } g_i(\mathrm{s}) > 0 \\ 0 & \text{if } g_i(\mathrm{s}) \leq 0 \end{cases}$$

Algorithms

Algorithm 1 A hybrid of BFGS and Barzilai-Borwein step size

Input: $k \leftarrow 0, \mathsf{s}^0 \in \mathbb{R}^d, X \in \mathbb{R}^{d \times n}, \lambda_0 > 0, H_0 \leftarrow I^d$, **Choose** ϵ

$d^0 \leftarrow -H_0 \nabla \mathcal{P}(\mathsf{s}^0)$

$\mathsf{s}^1 \leftarrow \mathsf{s}^0 + \lambda_0 d^0$

$k \leftarrow k + 1$

while $\|\nabla \mathcal{P}(\mathsf{s}^k)\| > \epsilon$ **do**

$d^k \leftarrow -H_k \nabla \mathcal{P}(\mathsf{s}^k)$

$\lambda_k \leftarrow \frac{\langle \mathsf{s}^k - \mathsf{s}^{k-1}, \nabla \mathcal{P}(\mathsf{s}^k) - \nabla \mathcal{P}(\mathsf{s}^{k-1}) \rangle}{\|\nabla \mathcal{P}(\mathsf{s}^k) - \nabla \mathcal{P}(\mathsf{s}^{k-1})\|^2}$

$\mathsf{s}^{k+1} \leftarrow \mathsf{s}^k + \lambda_k d^k$

$k \leftarrow k + 1$

end while

Algorithms

Algorithm 2 Steepest Descent with Barzilai-Borwein step size

Input: $k \leftarrow 0, \mathsf{s}^0 \in \mathbb{R}^d, X \in \mathbb{R}^{d \times n}, \lambda_0 > 0$, **Choose** ϵ

$s^1 \leftarrow s^0 - \lambda_0 \nabla \mathcal{P}(\mathsf{s}^0)$

$k \leftarrow 1$

while $\|\nabla \mathcal{P}(\mathsf{s}^k)\| > \epsilon$ **do**

$\lambda_k \leftarrow \frac{\langle \mathsf{s}^k - \mathsf{s}^{k-1}, \nabla \mathcal{P}(\mathsf{s}^k) - \nabla \mathcal{P}(\mathsf{s}^{k-1}) \rangle}{\|\nabla \mathcal{P}(\mathsf{s}^k) - \nabla \mathcal{P}(\mathsf{s}^{k-1})\|^2}$

$s^{k+1} \leftarrow s^k - \lambda_k \nabla \mathcal{P}(\mathsf{s}^0)$

$k \leftarrow k + 1$

end while

Algorithms

Algorithm 3 Steepest Descent with Adaptive step size

Input: $k \leftarrow 0, \mathsf{s}^0 \in \mathbb{R}^d, X \in \mathbb{R}^{d \times n}, \lambda_0 > 0, \theta_0 \leftarrow \infty,$ **Choose** ϵ

$s^1 \leftarrow s^0 - \lambda_0 \nabla \mathcal{P}(\mathsf{s}^0)$

$k \leftarrow 1$

while $\|\nabla \mathcal{P}(\mathsf{s}^k)\| > \epsilon$ **do**

$\lambda_k \leftarrow \min\left\{\sqrt{1+\theta_{k-1}}\lambda_{k-1}, \frac{\left\|\mathsf{s}^k - \mathsf{s}^{k-1}\right\|}{2\left\|\nabla \mathcal{P}(\mathsf{s}^k) - \nabla \mathcal{P}(\mathsf{s}^{k-1})\right\|}\right\}$

$s^{k+1} \leftarrow s^k - \lambda_k \nabla \mathcal{P}(\mathsf{s}^0)$

$\theta_k \leftarrow \frac{\lambda_k}{\lambda_{k-1}}$

$k \leftarrow k + 1$

end while

Line Search

Algorithm 4 Gradient-based Line Search

Input: $\lambda > 0, \rho \in (0,1), \mathsf{s} \leftarrow \mathsf{s}_k$

while

$$\lambda > \frac{\|\lambda \nabla \mathcal{P}(\mathsf{s})\|}{\|\nabla \mathcal{P}(\mathsf{s} - \lambda \nabla \mathcal{P}(\mathsf{s})) - \nabla \mathcal{P}(\mathsf{s}))\|}$$

do

$\lambda \leftarrow \rho\lambda$

end while

return λ

Algorithms

Algorithm 5 Steepest Descent with Line Search step size

Input: $k \leftarrow 0, \mathrm{s}^0 \in \mathbb{R}^d, X \in \mathbb{R}^{d\times n}, \lambda_0 > 0, \theta_0 \leftarrow \infty$, **Choose** ϵ

$s^1 \leftarrow s^0 - \lambda_0 \nabla \mathcal{P}(\mathrm{s}^0)$

$k \leftarrow 1$

while $\|\nabla \mathcal{P}(\mathrm{s}^k)\| > \epsilon$ **do**

  $\lambda_k \leftarrow$ Algorithm 3

  $s^{k+1} \leftarrow s^k - \lambda_k \nabla \mathcal{P}(\mathrm{s}^0)$

  $k \leftarrow k + 1$

end while

Implementation: Successful Algorithms

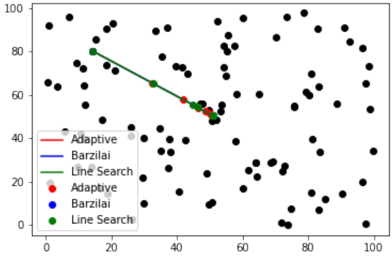

Figure: 100 sensors

$X = \{X \in \mathbb{R}^2 | x \in (0, 100)^2\}$.

Algorithm 3 and 5 show similar performance in terms of convergence.

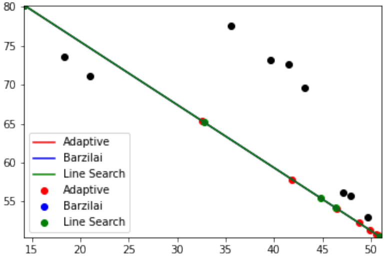

Figure: Plot from s^0 to $s^\star$.

Algorithm 5 reaches the neighbourhood of $s^\star$ faster than 3, but Algorithm 2 does so in essentially 1 step.

Implementation: Successful Algorithms

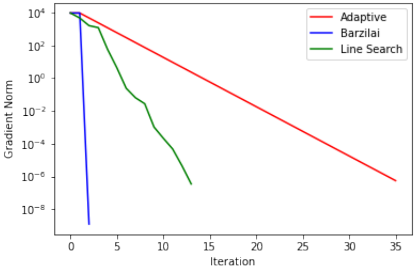

Figure: Algorithm 2 fastest convergence (1 step), Algorithm 5 next (13 steps), then Algorithm 3 (35 steps). (Not including initial step)

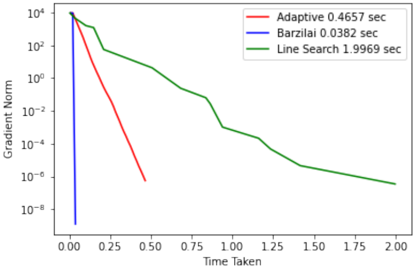

Figure: Algorithm 2 inexpensive, Algorithm 3 also quite cheap, and Algorithm 5 quite expensive.

Implementation: 'Failed BFGS'

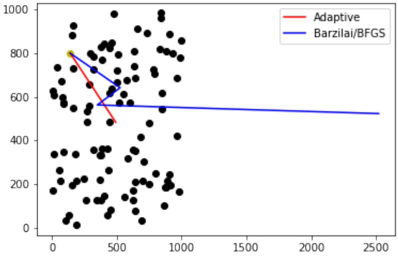

Figure: 100 sensors

$X = \{X \in \mathbb{R}^2 | x \in (0, 1000)^2\}$.

Only the first 4 steps of the Algorithm 1 method are shown.

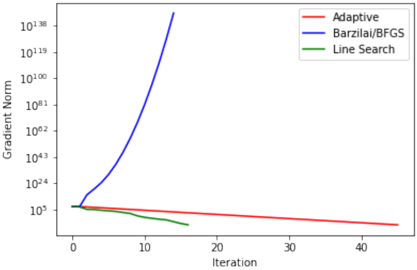

Figure: 1000 sensors

$X = \{X \in \mathbb{R}^2 | x \in (0, 10^4)^2\}$.

Gradient norm *grows* very quickly.

Implementation: 'Failed BFGS'

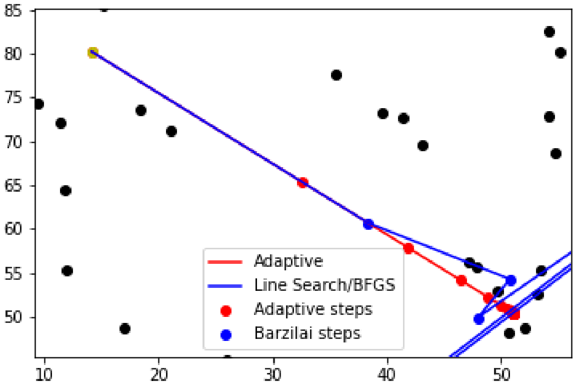

Figure: 100 sensors $X = \{X \in \mathbb{R}^2 | x \in (0, 100)^2\}$. Plot from s^0 to $s^\star$.

References

[1] Larry Armijo,Minimization of functions having Lipschitz continuous first partial derivatives, Pacific Journal ofMathematics16(1966), no. 1, 1–3.

[2] Jonathan Barzilai and Jonathan M. Borwein,Two-Point Step Size Gradient Methods, IMA Journal of NumericalAnalysis8(1988), no. 1, 141–148.

[3] Yura Malitsky and Konstantin Mishchenko,Adaptive gradient descent without descent, Proceedings of the 37thInternational Conference on Machine Learning (ICML)119(2020), 1–4.

[4] Jorge Nocedal and Stephen J. Wright,Numerical optimization, Springer, New York, 1999.