UNIVERSITÄT BERN

Applied Optimization
Exercise 11
Inequality Constrained Optimization

December 8, 2021

Hand-in instructions:

Please hand-in **only one** compressed file named after the following convention: `Exercise`n`-GroupMemberNames.zip`, where n is the number of the current exercise sheet. This file should contain:

- The complete code folder except for the `build` subfolder. Make sure that your code submission compiles on your machine. Zip the code folder.
- A `readme.txt` file containing a description on how you solved each exercise (use the same numbers and titles) and the encountered problems.
- Other files that are required by your `readme.txt` file. For example, if you mention some screenshot images in `readme.txt`, these images need to be submitted too.
- Submit your solutions to ILIAS before the submission deadline.

Algorithms Overview (5 pts)

Consider the task of optimizing a **large-scale** problem with $n = 10^6$ variables, given that state-of-the-art implementations of the following methods are available:

(a) Gradient Descent method
(b) Newton's method as on Page 12, Slides-07
(c) Augmented Lagrangian method using Newton's solver
(d) Infeasible Start Newton's method
(e) Active Set Method using Newton's solver
(f) Primal-Dual Interior Point method using Newton's solver with Hessian modification
(g) Gauss-Newton method
(h) L-BFGS method

For every of the following scenarios, list algorithms that can find a local minimum and specify the algorithm for which you expect the shortest runtime (assuming infinite memory). Note that every problem starts from a feasible point. ***Briefly* justify your choice**.

1. A quadratic problem which is strictly convex with linear equality constraints.
2. A linear program with inequality constraints.
3. A convex quadratic problem with linear inequality constraints which can be solved via a sequence of convex quadratic subproblems with small number of linear equality constraints.
4. An unconstrained non-convex problem
5. A non-convex least-squares problem of the form $f(x) = \sum_{i=1}^{m} f_i(x)^2$ with $x \in \mathbb{R}^n$. The gradient of f_i is dense and the hessian is not available.

Programming (5 pts)

Interior Point Method

Implement the Interior Point Method as discussed in the lecture by providing source code for:

- All `ToDos` in file `Functions/InteriorPointProblem.hh`.
  Note: You have multiple functions to help you compute the gradient and hessian of the log-barrier form of the constraints.
- `InteriorPoint::solve(...)` in `Algorithms/InteriorPoint.hh`

The new class `InteriorPointProblem` converts a given objective function and a set of constraints into an unconstrained log-barrier objective function.

In the test, we add penalty terms to the objective to fix four corner nodes as before and another penalty term to drag the node with index $(n\_grid\_x/2,\ \ n\_grid\_y/2)$ to the coordinate $(2.5 * n\_grid\_x,\ \ n\_grid\_y)$. The nodes are marked as red dots in the figure.

Additionally, we add inequality constraints to make sure the area of each triangle in the spring graph is positive. Specifically, for each triangle face (e.g. marked red in the figure) in the graph, we require the determinant of the matrix $M = \begin{pmatrix} V_{01} | V_{02} \end{pmatrix}$ to be larger than some epsilon (e.g. 10^{-10}). V_{01} and V_{02} are shown in the figure.
Important note: There are <u>four</u> triangles per grid cell.

Those constraint functions need to be implemented in the file `AreaConstraint2D.hh`. By calling the function `add_area_constraints()` in the `MassSpringSystem`, one can add the area constraints to the system. The implementation of this constraints setup function is also part of your exercise. Make sure that the orientation of all triangles is consistent!

Switch between test cases while executing the executable between with area constraints and without area constraints. Compare the results and submit the screenshots of spring graph of 10×10.

As usual, you have the visualization website and the unit tests to help you.