EE 5239 Nonlinear Optimization Homework 2 Cover Sheet

Instructor name: Mingyi Hong

Student name: Graham Droge

- Date assigned: Thursday 9/17/2019
- Date due: Sunday 9/29/2019, at 11:59PM
- This cover sheet must be signed and submitted along with the homework answers on additional sheets.
- By submitting this homework with my name affixed above,
  - I understand that late homework will not be accepted,
  - I acknowledge that I am aware of the University's policy concerning academic misconduct (appended below),
  - I attest that the work I am submitting for this homework assignment is solely my own, and
  - I understand that suspiciously similar homework submitted by multiple individuals will be reported to the Dean of Students Office for investigation.
- Academic Misconduct in any form is in violation of the University's Disciplinary Regulations and will not be tolerated. This includes, but is not limited to: copying or sharing answers on tests or assignments, plagiarism, having someone else do your academic work or working with someone on homework when not permitted to do so by the instructor. Depending on the act, a student could receive an F grade on the test/assignment, F grade for the course, and could be suspended or expelled from the University.

Exercise 1.2.1

We are given the function f of two variables

$$f(x,y) = 3x^2 + y^4$$

a) Apply steepest descent starting at (1,-2) using Armijo's rule

Armijo's rule is a successive step reduction rule where we want to reduce our step-size until our next value is smaller than our previous by a good enough amount. The following equation gives us a test for our choice for the next step.

$$f(x^k) - f(x^k + \beta^m s d^k) \geq -\sigma s \boldsymbol{\nabla} f(x^k)' d^k$$

We pick m to be 0 initially and check if this satisfies the equality. If it does we set our next step size to be $\alpha^k = \beta^m s$ if not we move to $m = 1$ and so on. First we calculate the values that we will need to evaluate the inequality with $d^k = -\boldsymbol{\nabla} f(x^k)$

$$f(x^0) = 19 \quad \boldsymbol{\nabla} f(x^0) = [6, -32] \quad -\sigma s \boldsymbol{\nabla} f(x^0)' d^0 = 106$$

Now we need to find an m value that solves

$$19 - f([1,-2]' + (.5)^m[-6, 32]') \geq 106\beta^m$$

The m that satisfies the following equation is $m = 4$ so our next step size would be $\alpha^k = \beta^4 s = .0625$ and the next iterate would be $x^{k+1} = x^k - \alpha^k \boldsymbol{\nabla} f(x^k) = [.625, 0]$.

b) Change Armijo parameters to $s = 1, \sigma = 0.1, \beta = 0.1$

The new m value that satisfies this new equation is $m = 1$ so the next step size $\alpha^k = \beta^1 s = 0.1$ so the next iterate becomes $[.4, 1.2]$. Obviously we can see that the choice of β has an effect on the iterative algorithm. The β gives us a measure of the "jump" size in the direction of the negative gradient. Initially we start with a large jump to hopefully transverse a larger line and thus converging faster. If we find that the point at this jump size and negative gradient direction is small enough compared to the previous point we use that at our next iteration and if not we reduce and test the inequality again. This method allows the iterative algorithm to move quickly when larger jump sizes equate to smaller function values. The cost of such an iterative step size is that shown in our first step calculation. We had to check the inequality 4 times before we found a satisfiable β for our other parameters and the gradient. This can greatly reduce the speed of the algorithm especially when you are near the minimum and have a large β since you will have to have a small jump size to find a smaller function value.

c) Same step size but using Newton's Method

Newton's method calls for the computing of the inverse of the Hessian matrix at the iteration point x^k. I calculated this for the first point

$$D^k = \begin{pmatrix} 6 & 0 \\ 0 & 12y^2 \end{pmatrix}^{-1} = \begin{pmatrix} 6 & 0 \\ 0 & 48 \end{pmatrix}^{-1} = \begin{pmatrix} .166 & 0 \\ 0 & .0238 \end{pmatrix}$$

Now our new d^k for the Armijo inequality and next iterate calculation becomes

$$d^k = -D^k \nabla f(x^k)$$

Using the same values for the other Armijo parameters in a) I obtain an m value of 0 and the step size becomes $\alpha^k = (.5)^0 = 1$ and the next iterate becomes $x^{k+1} = x^k - D^k \nabla f(x^k) = [0, -1.33]$. With Newton's method a β of .5 is enough to satisfy the inequality and our $m = 0$. This reduction in the amount of searching through m values comes at a cost and that is computing the Hessian on each iteration to get a better approximation for $f(x^k)$

Exercise 1.2.2

Describe the behavior of the steepest descent method with the constant step size s for the f

$$f(x) = \|x\|^{2+\beta}$$

The first thing I did is rewrite f to make computing the gradient easier from my point of view

$$f(x) = (\|x\|^2)^{\frac{2+\beta}{2}} \quad (1)$$

We are suppose to relate our answer to the assumptions of Prop 1.2.3 i.e using the Lipschitz condition.

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \quad (2)$$

So lets compute the gradient of f first

$$\nabla f(x) = \nabla\big(x_1^2 + x_2^2 + x_3^2 ...\big)^{\frac{2+\beta}{2}} = (2)(\frac{2+\beta}{2})\|x\|^\beta x = (2+\beta)\|x\|^\beta x$$

Now we will plug this into the Lipschitz equation (2). We want to analyze this inequality to see if it holds for all x,y. If we choose a y vector input that is the opposite direction of x i.e y = -x we get the equation (3)

$$(2+\beta)\Big\|\|x\|^\beta x - \|y\|^\beta y\Big\| \leq L\|x - y\|$$

$$(2+\beta)\|x\|^\beta \leq L \quad (3)$$

(3) tells us that an L (constant) $>$ 0 needs to be greater than or equal to $(2+\beta)\|x\|^\beta$ for the condition to hold, and we can see that this not satisfied for all x,y since we can choose an x such that $\|x\| > (\frac{L}{(2+\beta)})^{1/\beta}$

Now let's analyze the steepest descent behavior for this function.

$$x^{k+1} = x^k + s\nabla f(x^k) = x^k(1 - s(2+\beta)\Big\|x^k\Big\|^\beta) \quad (4)$$

Our original function (1) is just the norm of the input vector taken to a power so for our convergence analysis we can by induction say that if $\|x^1\| < \|x^0\|$ than $\|x^{k+1}\| < \|x^k\|$. This makes sense since with steepest descent we are moving "down" the objective value function and should obtain smaller values for our objective function. Thus (4) converges if

$$\Big|1 - s(2+\beta)\|x^0\|^\beta\Big| < 1 \quad (5)$$

$$s(2+\beta)\left\|x^0\right\|^\beta < 2$$

So for the values that satisfy (5) we know that the sequence of x^{k+1} is monotonically decreasing and we want to show that it converges to 0. So our intuition tells us that if the limit does not equal 1 it must be zero. To start take the limit of the iterates and there next values

$$\lim_{k\to\infty} \frac{\left\|x^{k+1}\right\|}{\left\|x^k\right\|} = d \leq \left\|x^0\right\| \quad (6)$$

if we use (6) in (5) which tells us the next iterate is less than the previous we can say that the limit cannot equal 1 and must equal 0.

$$\lim_{k\to\infty} \frac{\left\|x^{k+1}\right\|}{\left\|x^k\right\|} = \left|1 - s(2+\beta)d^\beta\right| < 1$$

Exercise 1.2.3

Considering the function

$$f(x) = \|x\|^{\frac{3}{2}}$$

we need to show that the Lipschitz condition is not satisfied for any L. To do this first compute the gradient.

$$\nabla f(x) = \frac{3}{2}\|x\|^{-1/2}x$$

Now we try a value for y that is equal to $-x$ and plug it into the Lipschitz condition

$$\left\|\frac{3}{2}\|x\|^{-1/2} + \frac{3}{2}\|x\|^{-1/2}\right\| \leq L\|2x\|$$

$$3\|x\|^{1/2} < 2L\|x\|$$

Now we could pick a x vector that satisfies this inequality $\|x\|^{1/2} < \frac{3}{2L}$ which would make the Lipschitz condition unsatisfied.

Now we analyze the convergence of f given a constant step-size. We start by writing the equation for the next iterate in the sequence

$$x^{k+1} = x^k - \alpha\nabla f(x^k) \quad = \quad x^k(1 - \frac{3}{2}\alpha\left\|x^k\right\|^{-1/2})$$

To converge in a finite number of iterations we need $x^{k+1} = 0$. We can just use the vector values since one of the properties of a norm is that it is 0 only for the 0-vector. This would make the term $1 - \frac{3}{2}\alpha\left\|x^k\right\|^{-1/2} = 0$ as well. Solving for $\left\|x^k\right\|$ we get

$$\left\|x^k\right\| = \frac{9\alpha^2}{4}$$

Lastly we analyze what happens if $\left\|x^k\right\| \neq \frac{9\alpha^2}{4}$. The $\left\|x^{k+1}\right\|$ will be greater than $\left\|x^k\right\|$ when $(1 - \frac{3}{2}\alpha\left\|x^k\right\|^{-1/2})$ is greater than ± 1 and $\left\|x^{k+1}\right\|$ will be smaller than $\left\|x^k\right\|$ when $(1 - \frac{3}{2}\alpha\left\|x^k\right\|^{-1/2})$ is less than 1. Observing this we note the value at which this oscillation between greater than and smaller than happens $\frac{9\alpha^2}{16}$. When $\left\|x^k\right\| < \frac{9\alpha^2}{16}$ the next iterate will be bigger and when $\left\|x^k\right\| > \frac{9\alpha^2}{16}$ the next iterate will be smaller. Thus the function oscillates and it does not converge. The same for when $\left\|x^k\right\| = \frac{9\alpha^2}{16}$ since this would make $x^{k+1} = -x^k$ and it would oscillate again about those two points.

Exercise 1.2.7

The engineers process for maximizing the complex circuit is one that is closely related to how the steepest descent algorithm works. What the engineer is essentially doing is doing a descent/ascent method where at each iteration point she does an exact line search across one dimension. She is working her way down/up the objective value by finding min/maxs along alternating variable paths. Steepest descent's similar algorithm computes a direction that is n-dimensional so it can move across the n-dimensional space in the direction of the gradient at that point. Her iterative approach is a cheap and provides a simple and intuitive way to maximize a function, but it comes at the cost of being slow and depending on the mathematical model of the network it could take a large amount of iterations to converge .

Exercise 2

We suppose that $f(x)$ is convex.

a) $f(x) - f(y) \geq (\boldsymbol{\nabla} f(y))^\intercal (x-y)$

Start by writing down the equation for the definition of convexity where it holds for any vector in the convex set of x, y. Then rearrange terms to be in a different form

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y)$$

$$f(y + \lambda(x-y)) \leq f(y) + \lambda(f(x) - f(y))$$

$$\frac{f(y+\lambda(x-y)) - f(y)}{\lambda} \leq f(x) - f(y) \quad (1)$$

The final equations (1) left term looks a lot like the definition of the derivative. So we will take the limit ast $\lambda \to 0$. Defining $g(\lambda) = f(y + \lambda(x-y))$ we see that $f(y) = g(0)$

$$\lim_{\lambda \to 0} \frac{g(\lambda) - g(0)}{\lambda} = g^{'}(0)$$

$$g^{'}(\lambda) = (\boldsymbol{\nabla}_y f(y + \lambda(x-y))^\intercal (x-y)$$

$$g^{'}(0) = (\boldsymbol{\nabla}_y f(y))^\intercal (x-y) \quad (2)$$

Now plugging (2) into (1) after taking the limit of both sides with respect to λ we have the equation below which proves our initial goal.

$$(\boldsymbol{\nabla} f(y))^\intercal (x-y) \leq f(x) - f(y)$$

b) $g(y)$ **is monotonically non-decreasing convex function then g(f(x)) is also a convex function**

Start by writing the definition of convexity for f(x)

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$$

Now using the fact that $g(y)$ is non decreasing we can write it as

$$g(f(\lambda x + (1-\lambda)y)) \leq g(\lambda f(x) + (1-\lambda)f(y))$$

And since g is convex we can use the right term to write the equation below which shows the function is still convex

$$g(\lambda f(x) + (1-\lambda)f(y)) \leq \lambda g(f(x)) + (1-\lambda)g(f(y))$$

c) Show an example

If we say that $f(\mathbf{x}) = e^{\|x\|^2}$ which is convex and $g(y) = e^{-y}$ which is also convex and decreasing then the composition of them $g(f(\mathbf{x}))$ will not be a convex function.

Exercise 3

Given the objective function

$$min_{x,y}\frac{1}{2}\|A - xy^\intercal\|_F^2$$

To better analyze this function I rewrite it using the Frobeius norm properties where λ_i are the non-zero eigenvalues of $(A - xy^\intercal)$.

$$\|A - xy^\intercal\|_F^2 = \sum_{i=1}^{R} \lambda_i$$

Since we are given that A and $xy^\intercal$ are rank one and the rank is at most 1 after subtraction this means that the non-zero eigenvalue of $(A - xy^\intercal)$ is $Tr(A - xy^\intercal)$

$$Tr(A - xy^\intercal) = \sum_{i=1}^{n} a_{ii} - x_i y_i$$

$$\frac{\partial f}{\partial \mathbf{x}} = -\mathbf{y} \quad \frac{\partial f}{\partial \mathbf{y}} = -\mathbf{x}$$

Since we are essentially minimizing a function of two vectors the hessian will be composed of cross vector entries and as we can see from the first order partial derivatives we will get negative entries and 0 entries. This means our Hessian is not positive semi definite so it is not a convex optimization problem.

b) Find the stationary points

Using the equations in a) we can observe that the vectors for which the gradient with respect to input vectors x, y is the 0 vector and if the vectors are orthogonal due to the fact that the dot product sum will be zero.

Programming Assignment

3)

Below shows various plots for the difference values between the iterates and the known min as well as the CPU time as a function of the iterates. The objective value difference plots are zoomed in over a range of iterates to show where most of the descending is happening. The CPU time plots are zoomed in as well to give an idea of the slope of these lines which tells us how long it takes to perform an iteration given a specific algorithm. The results are consistent with what we would expect for the different algorithms. First off the size of the matrix and vector under consideration greatly effected the speed of the algorithms. Armijo steepest descent algorithm was greatly effected by this large matrix size due to it having to perform matrix multiplication for each m value check. The diagonally scaled gradient method would also see time increases due to the computation of the hessian for each iterate. When looking at the objective value differences as a function of iterates we notice some interesting things about the algorithms. The Armijo steepest descent method and the diagonally scaled gradient method generally followed closely for the different n and c values. Convergence happened relatively quickly for each of the methods. I'll mention that the convergence depended on the Q and b matrix that was obtained when running the MATLAB script and there were some instances where the discrepancy between the two was greater. The conjugate gradient method on the other hand starts off with a sharp difference in objective values. This is due to the fact that the algorithm builds up directions to travel based on previous directions so we would expect a large different at the start. I quickly settles to the objective value minimum across all of the tests. You can see that in the case when c = 10 the conjugate gradient method sharply bounces higher to lower but converges much quicker than the Armijo and diagonal gradient descent. This is expected since the conjugate gradient method converges by the n iterate where n is equal to the nxn dimension of Q. Lastly I'll mention the effect of condition number on convergence. Since the condition number gives a measure of the ”sensitivity” of the function to inputs we expect that the algorithm will have a harder time converging for larger condition numbers. The results show a slightly more jagged descent to the minimum value but this could be more pronounced if run with a different matrix and value vector. On the flip side you can see how when the condition number was low the Armijo and diagonal scaled gradient method took a considerable amount of time to transverse there way to the minimum.

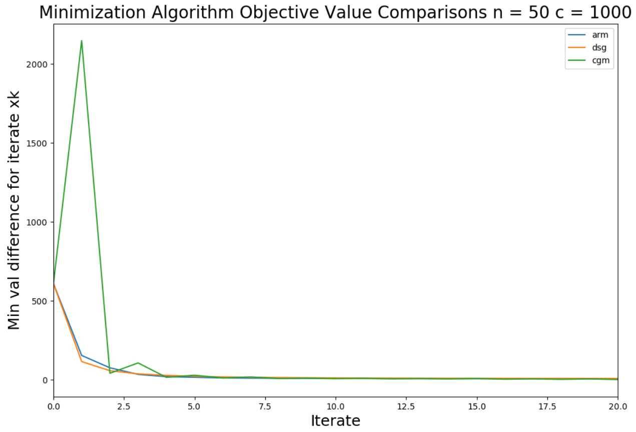

Figure 1: Objective value difference for n = 50 c = 1000

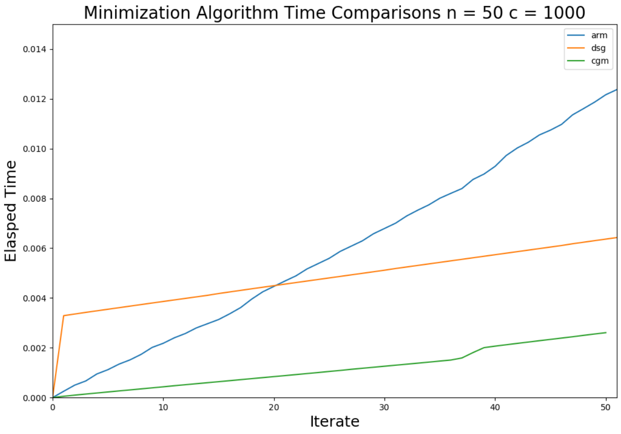

Figure 2: Elapsed time snippet plot for n = 50 c = 1000

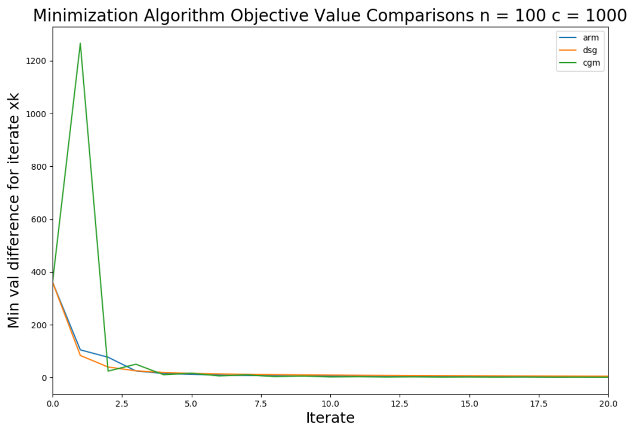

Figure 3: Objective value difference for n = 100 c = 1000

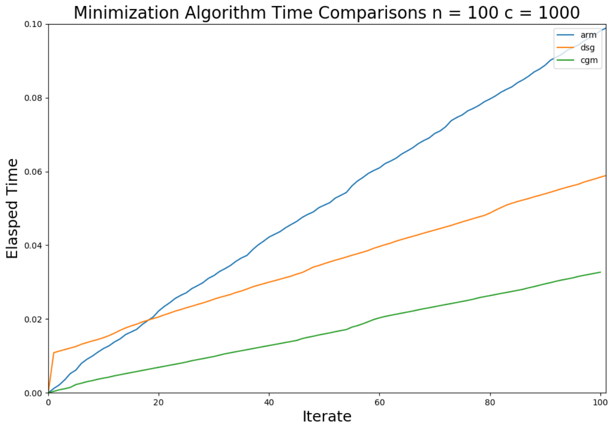

Figure 4: Elapsed time snippet plot for n = 100 c = 1000

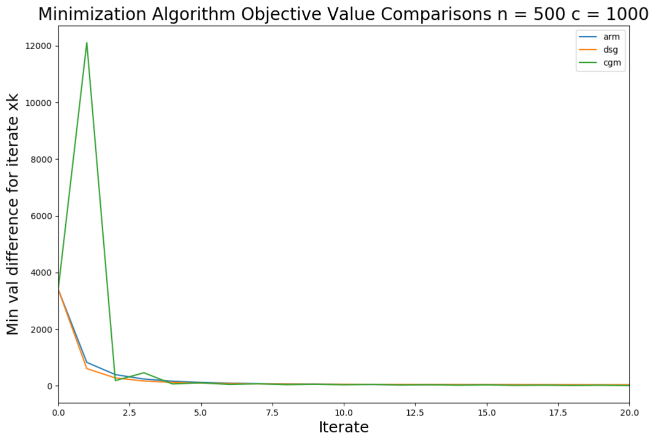

Figure 5: Objective value difference for n = 500 c = 1000

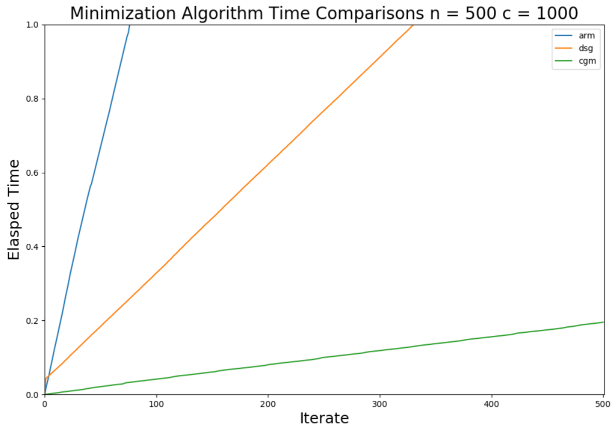

Figure 6: Elapsed time snippet plot for n = 500 c = 1000

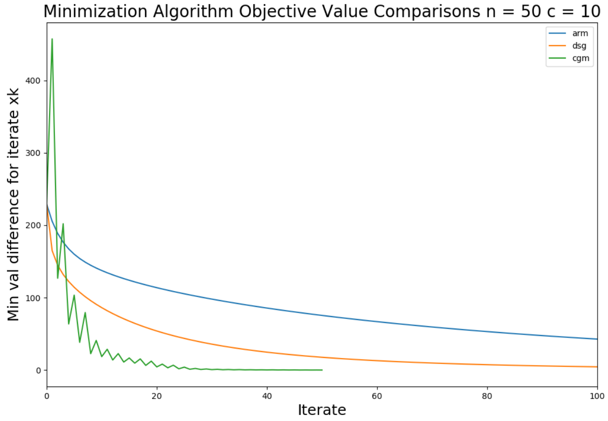

Figure 7: Objective value difference for n = 50 c = 10

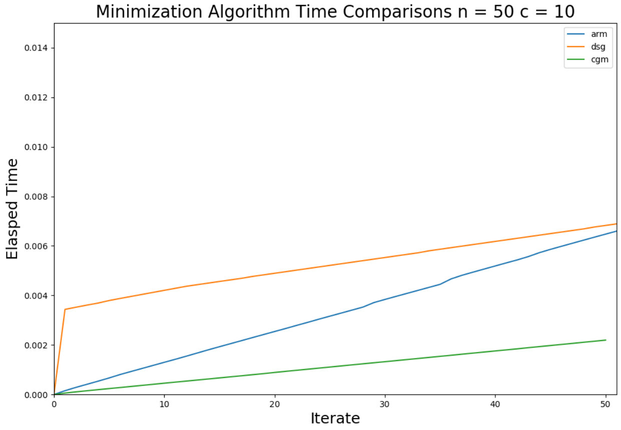

Figure 8: Elapsed time snippet plot for n = 50 c = 10

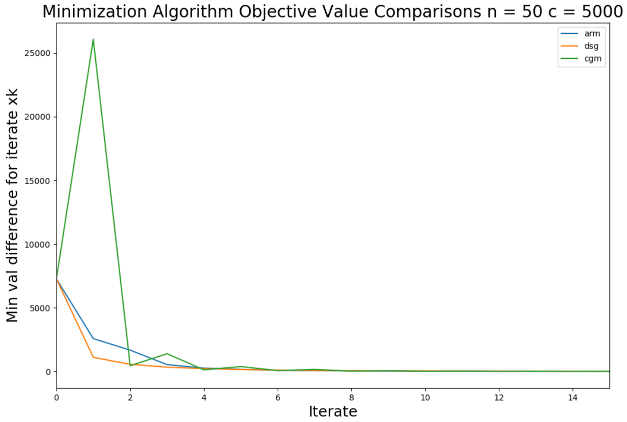

Figure 9: Objective value difference for n = 500 c = 5000

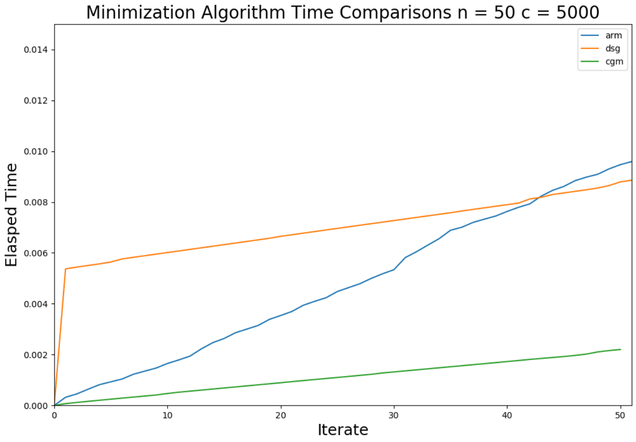

Figure 10: Elapsed time snippet plot for n = 500 c = 5000