Comparison of Optimization Methods with Application to a Network Containing Malicious Agents.

Iyanuoluwa Emiola, Robson Adem

Abstract—There are different methods of solving unconstrained optimization problems, but there have been disparities in convergence speed for most of these methods. First-order methods such as the steepest descent method are very common in solving unconstrained problems but second-order methods such as the Newton-type methods enable faster convergence especially for quadratic functions. In this paper, we compare and analyse first and second order methods in solving two unconstrained problems that differ by a multiplicative perturbation parameter to show how malicious agents in a network can cause disruption. We also explore the advantages and disadvantages of the steepest descent, Newton, and Conjugate gradient methods in terms of their convergence attributes by comparing a strictly convex function with a banana-type function.

I. INTRODUCTION

Solutions to unconstrained optimization problems can be applied to multi-agent systems and machine learning problems, especially if the problem is in a decentralized or distributed fashion [1], [2], [3] and [4]. Sometimes, malicious agents can be present in a network that will slow down convergence rates to optimal points. Therefore, the need for a fast convergence and cost associated with it are usually necessities by recent researchers. The steepest descent method is a good first order method for obtaining optimal solutions if an appropriate step size is chosen. Some methods of choosing step sizes include the fixed step size, the variable step size, the polynomial fit, and the golden section method that will be discussed in details in subsequent sections. Nonetheless, these methods have their own merits and demerits. A second order method such as the Newton method is very suitable for quadratic problems and attains optimality in a small number of iterations [5]. However, the Newton method requires computing the inverse of the hessian which is often a bottleneck. Moreover, the Newton method will not be suitable for solving large scale optimization problems.

To address the lapses that the Newton method poses, some methods have been recently proposed such as matrix splitting seen in [6]. Another way of obtaining fast convergence properties of second-order methods using the structure of first-order methods are the so-called quasi-Newton methods. These methods incorporate second-order (curvature information) in the first-order approaches. Examples of these methods include the BFGS [7] and the Barzilai-Borwein (BB) [8] and [9]. However, these methods usually require additional assumptions on the objective function to be strongly convex and that the gradient of the objective function to be Lipschitz continuous to improve convergence rate as seen in [10].

The conjugate gradient method on the other hand does not have as much restrictions as the Newton method in terms of computing the inverse of the hessian. The method computes the direction of search at each iteration and the direction is expressed as a linear combination of previous search direction calculation and the present gradient at the present iteration. Therefore, the conjugate direction will be more suitable for large scale optimization problems than the Newton method. The other interesting attribute of the conjugate gradient method is that it gives different methods of calculating the search directions and it is not only limited to quadratic functions as we will use that while performing simulations using the Banana function [5].

Fast convergence is a necessity as we will explore in an application where there are malicious agents in a network (that has a central coordinator) that will try to deviate the optimal solution of the regular agents. Some researchers have explored the roles adversarial agents play in such scenarios. One of them is seen in [11] where the authors explore the convergence properties using a consensus framework. Another one is seen in [12] where the author devised a detection metric to identify what nodes are malicious within a network. We use a perturbation method by changing and increasing the coefficient of a strictly convex function and the new coefficient will result in a perturbed optimal solution parameter. In this paper, we explore the convergence properties of the regular agents and malicious agents central coordinator’s objective functions by using first order methods such as the steepest descent with fixed step size, variable step size, polynomial quadratic fit and golden section method. We also explore convergence properties using the Newton Method and the Conjugate Gradient Method. For both first and second methods as described, we deduce how the malicious agents perturbed parameter can either prevent convergence or increase the number of iterations to achieve convergence through some simulations.

A. Contributions

In this paper, we compare and analyze the convergence attributes of a non-quadratic function and in a scenario where a multiplicative perturbation parameter is used to denote the central coordinator of malicious agents solving a different function as the regular with the hope of obstructing the malicious agent goal of optimality. We analyze the convergence numerically and show that convergence is attained despite the presence of the malicious agents. We explore different types of methods as described to analyze convergence without restricting the analysis to quadratic functions. Moreover, this is the first work that will compare convergence of perturbed non-quadratic functions in the manner described.
B. Paper Pattern

Section II presents the problem formulation, section introduces the different types of first and second type methods, and their convergence analysis, numerical experiments are performed in section IV and we conclude in section V.

C. Notation

We denote the set of positive and negative reals as $\mathbb{R}_+$ and $\mathbb{R}_-$, the transpose of a vector or matrix as $(\cdot)^T$, and the L2-norm of a vector by $||\cdot||$. We let the gradient of a function $f(\cdot)$ as $\nabla f(\cdot)$, and $\langle \cdot, \cdot \rangle$ denotes the inner product of two vectors. $x^*$ and $x^a$ represent the optimal points for the regular agent central central coordinator and malicious agent central coordinator respectively. The variable $\epsilon$ represents the perturbation parameter.

II. PROBLEM FORMULATION

Suppose there is a central coordinator of regular agents in a network trying to solve the following objective minimization problem:

$$\min(f_r = f(x_1, x_2))$$

(1)

where $f_r$ is a strictly convex function. To solve problem (1) using gradient methods, we use the iterative equation below:

$$x(k + 1) = x(k) - \alpha \nabla f(x(k)),$$

(2)

where $k$ is the iterative time step, $\alpha$ is the step size that the central coordinator is using to achieve convergence, and $f(x(k))$ is the gradient of $f$ at each iterate $x(k)$. Different first order and second order methods for solving problem (1) will be explored in the later part of the paper. The goal of the regular agents is to get to the optimal point $x^*$.

Suppose there are malicious agents in a network and the central coordinator decides to obstruct the regular agent from obtaining their desired location. To facilitate this obstruction, the malicious agent central coordinator used the banana type function of the form:

$$\min(f_m = m f_r)$$

(3)

where $m$ is a positive constant and the malicious objective function $f$ is also a strictly convex function. To solve problem (3) using gradient methods, and noting that the malicious agents are trying to get to the optimal point, $x^* + \epsilon$, we use the iterative equation below:

$$x(k + 1) = x(k) - \alpha \nabla f(x(k)) + \epsilon,$$

(4)

where $\epsilon$ is the perturbed parameter that the central coordinator of the regular agents is using to distort the usual optimal solution of the regular agents, $x^*$, $k$ is the iterative time step, $\alpha$ is the step size that the central coordinator of malicious agents is using to achieve convergence, and $f(x(k))$ is also the gradient of $f$ at each iterate $x(k)$.

In later part of the paper, we will use the Banana function with $\epsilon = 100$ to test how the optimal solutions are being shifted and that convergence is still obtained after some iterations. We now state the assumptions needed before analysis.

Assumption 1. The decision set of the coordinators of malicious and regular agents $X$ is bounded. This means there exists some positive constant $0 \leq B < \infty$ such that $|X| \leq B$.

Assumption 2. $f(x)$ in problems (1) and (3) are strictly convex and twice differentiable.

III. ANALYSIS OF FIRST AND SECOND ORDER METHODS

We now discuss a first order method known as the steepest descent method for solving problem (1).

A. The Steepest Descent Method

To solve a problem such as the one described in (1), a sequence of guesses $x(0), x(1), ..., x(k), x(k + 1)$... will be generated in a descent manner such that

$$f(x(0)) > f(x(1)) > ... > f(x(k + 1))$$

It can be often tedious to obtain optimality after some $K$ iterations, when $\nabla f(x(K)) = 0$. Therefore it suffices to actually modify the gradient stopping condition to satisfy $||f(x(K))|| \leq \epsilon$ where $\epsilon > 0$ and very small which is often referred to as the stopping criterion for convergence of Hold. While using descent methods under each $k$th iterative guess, we search along a direction $d(k)$ which is an $n \times 1$ vector and also using the optimum step size, $\alpha(k)$ according to:

$$x(k + 1) = x(k) + \alpha(k)d(k)$$

The goal is to ensure that $f(x(k + 1)) \leq f(x(k))$. Moreover, if the direction of negative of the gradient is taken such that $d(k) = -\nabla f(x(k))$, we obtain the steepest descent algorithm. Different ways of choosing the step size will be explored below:

B. Steepest Descent With a Constant Step Size

The constant step size is constructed in a manner where you simply use one value of $\alpha$ in all iterations. Although it looks relatively easy to implement, the demerits of using a fixed step size is not knowing apriori whether the choice of step size is numerically adequate. In problem (1), if the central coordinator chooses a very small $\alpha$ value, the algorithm will be slow and if he chooses an extremely high $\alpha$ value, the algorithm might diverge. To illustrate the fixed step size principle in problem (1), we will pick an $\alpha$ value between 0 and 1 and show numerically how convergence is attained.

To demonstrate that a suitable step size results in a neighborhood convergence to optimal solution, we will show a proof of a gradient descent algorithm where malicious agents are present provided the function in problem (2) is strongly convex. This is shown below:

C. Convergence proof for a strongly convex function with malicious perturbation parameter

Theorem 1. Let Assumptions (1), and (2) hold where the function in (1) is strongly convex with strong convexity parameters $\mu$ and $L$ with $\mu \leq L$. Let the perturbation parameter satisfy $\epsilon > 0$, and let each agent's estimates when malicious agents
are present satisfy \( x(0) - x^* < 0 < c_{\text{max}} \). If \( c_1 = \frac{2}{\mu + L} \), \( c_2 = \frac{2\mu L}{\mu + L} \), and \( \alpha < c_1 \), the iterates generated from (2) converge to the neighborhood of the optimal solution \( x^* \).

\[ \text{Proof.} \ \text{Refer to [13] for the proof.} \]

Other methods of choosing the step size in a steepest descent algorithm are shown below:

\section*{D. Steepest Descent with Variable Step Size}

In the variable step size method, 3 or 4 values of \( \alpha \) are chosen at each iteration and the values that produce the smallest \( g(\alpha) \) value will be chosen where \( g(\alpha(k)) = f(x(k + 1)) \). The variable step size algorithm is also easy to implement and has a better convergence probability than the fixed step size method. The results of simulating problems (1) and (3) via the variable step size is shown in (IV).

Another variation of the variable step size method is the polynomial fit methods but we will focus mostly on a subclass of the polynomial methods which is the quadratic fit method below:

\section*{E. Steepest Descent with Quadratic Fit Method}

For the quadratic fit method, three values of \( \alpha \) are guessed at each iteration and the values of the corresponding \( g(\alpha) \) values are computed, where \( g(\alpha(k)) = f(x(k + 1)) \) For example, suppose the three values of \( \alpha \) values chosen are \( \alpha_1, \alpha_2, \alpha_3 \). To fit a quadratic model of the form:

\[ g(\alpha) = a\alpha^2 + b\alpha + c, \] (5)

we write the quadratic model with respect to the \( \alpha \) values as:

\[ g(\alpha_1) = a\alpha_1^2 + b\alpha_1 + c, \] (6)

\[ g(\alpha_2) = a\alpha_2^2 + b\alpha_2 + c, \] (7)

and

\[ g(\alpha_3) = a\alpha_3^2 + b\alpha_3 + c. \] (8)

where \( a, b, c \) are constants. After solving the constants \( a, b, c \) in equations (6), (7), and (8), it will be used in equation (5) and the quadratic fit according to equation (5) is then minimized. Another method for solving problem in (1) and unconstrained optimization problems is the golden section search method. We briefly explain this as follows:

\section*{F. Steepest Descent with the Golden Section Search}

In this algorithm, we use a range between two values and divide the range into sections. We then eliminate some sections within the sections in the range to shrink the region where the convergence might occur. For this algorithm to be implemented as we will see in section (IV), the initial region of uncertainty and the stopping criterion have to be defined. An example where a golden section search was applied to minimize a function in a closed interval is seen in [5]. Second-order methods have been an improvement in terms of convergence speeds when it comes to solving unconstrained optimization problems such as problems (1) and (3). We will show by simulations in section IV that convergence is faster with the second-order methods than in the first order type methods despite the presence of malicious agents. We now analyze the two second order methods below:

\section*{G. Newton Methods}

The Newton Method is very useful in obtaining fast convergence of an unconstrained problem like in equation (1) especially when the initial starting point is very close to the minimum. The main disadvantage of this method is the cost and difficulty associated with finding the inverse of the hessian and also ensuring that the hessian inverse matrix is positive definite. The update equation for the Newton Method is given by:

\[ x(k + 1) = x(k) - \nabla f(x)(\nabla^2 f(x))^{-1}. \] (9)

Some of the methods of approximating the term that contains the inverse of the hessian, \( (\nabla^2 f(x))^{-1} \) in equation (9) are the Quasi-Newton methods such as the BFGS and the Barzilai-Borwein methods [10].

\section*{H. Conjugate Gradient Methods}

For the class of quadratic functions:

\[ f(x) = 0.5x^T Q x - x^T b, \]

and \( x \in \mathbb{R}^n \), the conjugate gradient algorithm uses a direction expressed in terms of the current gradient and the previous direction at each iteration by ensuring that the directions are mutually Q-conjugate, where \( Q \) is a real symmetric \( n \times n \) matrix. We note that the directions \( d(0), d(1), \ldots, d(m) \) are Q-conjugate if \( d(i)^T Q d(j) = 0 \), and \( i \neq j \). The conjugate gradient method also exhibits fast convergence property for non-quadratic problems like problem in (1) and (3). In the simulation in section IV, we use the Fletcher-Reeves Formula given by:

\[ \beta(k) = \frac{g(k + 1)^T g(k + 1)}{g(k)^T g(k)} \]

where \( g(k) = \nabla f(x(k)) \) and \( \beta(k) \) are constants picked such that the directional iteration \( d(k + 1) \) is Q-conjugate to \( d(0), d(1), \ldots, d(k) \) according to the following iterations:

\[ x(k + 1) = x(k) + \alpha(k)d(k), \]

and \( \alpha(k) \) is the step size. We will show in section IV that the conjugate gradient method performs better than the steepest descent method in terms of convergence rate when we use the same fixed step size.

\section*{IV. Numerical Experiments Insights}

In this section, we will compare the methods discussed in section III in terms of convergence and the number of iterations taken to reach optimality. In this section, we let the central coordinator of the regular agents minimization problem be given by:

\[ f_r = f(x_1, x_2) = (x_1^2 - x_2)^2 + (x_1 - 1)^2 \] (10)
and due to the optimal solution disruption goal of the malicious agents, we let the central coordinator of the malicious agents problem be:

\[ f_m = f(x_1, x_2) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 \]  

(11)

where \( f_r \) and \( f_m \) are strictly convex in the neighborhood of the optimal solution. For these two functions, initial conditions of (2, 2) and (5, 5) and a stopping criterion of \( \nabla f(x(K)) \leq 0.001 \) are used across all these methods discussed in section III. Here, \( K \) is the maximum iteration number to achieve convergence. By inspection, the optimal solutions of both equations (10) and (11) are \( x_1 = x_2 = 1 \). To compare the methods discussed in section III, a geometric sequence of fixed step sizes is used. For the steepest descent, Newton and Conjugate Methods, we use the fixed step sizes, 0.123, 0.0123, 0.00123 and 0.000123.

A. Case when \( \alpha = 0.123 \)

When the step size \( \alpha = 0.123 \) is used, the iteration in equation (2) generated by (10) converges for the Newton method with both of the initial conditions criteria, converges for the steepest descent with the (2, 2) initial condition and diverges with the initial condition of (5, 5). The iteration also diverges with the conjugate gradient with both of the initial starting points.

For the iteration generated by (11) when the step size \( \alpha = 0.123 \), it converges by the Newton method using both of the initial starting points, diverges by the steepest descent for both initial points, and also diverges by the conjugate gradient using both initial points, (2, 2) and (5, 5).

B. Case when \( \alpha = 0.0123 \)

It is expected that the case when \( \alpha = 0.0123 \) will provide an improvement regarding convergence compared to the case when \( \alpha = 0.123 \) because 0.0123 < 0.123. When \( \alpha = 0.0123 \) is used, the iteration in equation (2) generated by (10) converges for the Newton method by using the two starting points, (2, 2) and (5, 5). By using the steepest descent on the first function, the iteration generated by (10) converges with both of the starting initial points which is an improvement over the case when \( \alpha = 0.123 \), where the steepest method diverges using the initial point (5, 5). By the conjugate method on the first function, convergence is obtained with both starting points which is also an improvement over the case when \( \alpha = 0.123 \), where it diverges for both starting values.

For the iteration generated by (11) when the step size \( \alpha = 0.0123 \) is used, there was no improvement in convergence attributes because divergence is obtained by steepest descent and conjugate gradient methods for both of the initial points.

C. Case when \( \alpha = 0.00123 \)

At the stage when \( \alpha = 0.00123 \) is used, the three methods, conjugate, Newton, and the steepest descent all converge with both starting points for the iteration in generated by (10).

By using the iteration generated by (11) when \( \alpha = 0.00123 \), there is an improvement over the case when \( \alpha = 0.0123 \). This is evident because convergence is obtained by starting from the initial point (2, 2) compared to divergence result obtained by using the steepest descent and conjugate gradient methods when \( \alpha = 0.0123 \).

D. Case when \( \alpha = 0.000123 \)

If the step size \( \alpha = 0.000123 \) is used, the three methods: steepest descent, Newton, and conjugate gradient all converge as shown in figures 1 and 2 for both the iterations generated by (10) and (11). In addition, We start with both initial points (2, 2) and (5, 5) to obtain convergence with the step size \( \alpha = 0.000123 \). Therefore we will use this step size as a case study to compare these methods.

E. Significance of the Newton Method

The significance of the Newton method should not be overlooked even for non-quadratic functions such as in (10) and (11) because the method guarantees convergence to the optimal solution for the geometric step sizes of 0.123, 0.0123, 0.00123 and 0.000123. Moreover, it achieves convergence in just few iterations for both of the starting points. This affirms the unique convergence attribute of the Newton method when the starting point is not far away from the optimal solution.

To compare the performance of the fixed step sizes with other variable metrics, an illustration of the variable step size, quadratic fit and golden section methods are presented below:

F. Comparison of the variable step size, quadratic fit and golden section with other methods

By starting with the two initial points (2, 2) and (5, 5) using the variable step size method, convergence was obtained for the two functions (10) and (11) when three varying step sizes of 0.000123, 0.0123 and 0.123 are used. When the quadratic fit is used, three values of the step sizes are used in each iteration and selected from the range (0.00001, 0.000123). The result from the quadratic fit shows that a better convergence is achieved for function (10) but shows a weaker convergence for the second function. This explains that fluctuations in the random selection of step sizes between a range can alter convergence rate. Moreover, the perturbation from function (11) can also slow down convergence rates. For the golden section method, a range of (0.00000123, 1.5) is used to locate the value of the step size that result in the solution to problems (10) and (11). By using the initial points of (2, 2) and (5, 5) on the first function in (10), the golden section method resulted in the fastest convergence rate by comparing with the steepest descent with fixed step size, variable step size and the quadratic fit methods. However for the second function in (11), convergence with the golden section than the variable, fixed step size and the quadratic fit methods when the same initial starting conditions are used.

The simulations for all of these methods are shown below:
| Function 1 | Function 2 |
|------------|------------|
| $x_0 = (2.2)$ | $x_0 = (2.2)$ |
| 154,049 Iterations | 156,541 Iterations |
| 169 Iterations | 170 Iterations |
| 34 Iterations | 96 Iterations |
| 16 Iterations | 38 Iterations |

Fig 1: Contour plots of the Steepest Gradient Descent Method with various choices of step-size. The plots also show the trajectory of guesses converging to the minimum.
V. CONCLUSIONS

We analyze convergence attributes of some selected first and second order methods such as the steepest descent, Newton, and conjugate gradient and apply it to a strictly convex and a Banana-type function. We show through different optimization methods for functions (11) and (11) that it is still possible for the regular agents to converge to their optimal solution despite the presence of malicious agents by showing that function (11) converges for some methods. We obtain a geometric test for a fixed step size that guarantee convergence while comparing the optimization methods discussed. Numerical experiments affirm that the Newton method has the fastest convergence rate for the two strictly convex functions used in this paper and that it is useful when the initial starting point is close to the optimal point as seen with the starting points used. Out of the three optimization methods considered, the conjugate gradient is the most beneficial method than the others since it does not have the convergence restrictions as the steepest descent and the Newton methods.

REFERENCES

[1] A. Nedić, A. Olshevsky, and M. G. Rabbat, “Network topology and communication-computation tradeoffs in decentralized optimization,” Proceedings of the IEEE, vol. 106, no. 5, pp. 953–976, 2018.

[2] S. Yang, Q. Liu, and J. Wang, “Distributed optimization based on a multiagent system in the presence of communication delays,” IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 47, no. 5, pp. 717–728, 2016.

[3] E. Montijano and A. R. Mosteo, “Efficient multi-robot formations using distributed optimization,” in 53rd IEEE Conference on Decision and Control. IEEE, 2014, pp. 6167–6172.

[4] K. I. Tsianos, S. Lawlor, and M. G. Rabbat, “Consensus-based distributed optimization: Practical issues and applications in large-scale machine learning,” in 2012 50th annual allerton conference on communication, control, and computing (allerton). IEEE, 2012, pp. 1543–1550.

[5] E. K. Chong and S. H. Zak, An introduction to optimization. John Wiley & Sons, 2004.

[6] M. Zargham, A. Ribeiro, A. Ozdaglar, and A. Jadabaie, “Accelerated dual descent for network flow optimization,” IEEE Transactions on Automatic Control, vol. 59, no. 4, pp. 905–920, 2013.

[7] M. Eisen, A. Mokhtari, and A. Ribeiro, “Decentralized quasi-newton methods,” IEEE Transactions on Signal Processing, vol. 65, no. 10, pp. 2613–2628, 2017.

[8] Y.-H. Dai and R. Fletcher, “Projected barzilai-borwein methods for large-scale box-constrained quadratic programming,” Numerische Mathematik, vol. 100, no. 1, pp. 21–47, 2005.

[9] P. E. Gill and W. Murray, “Quasi-newton methods for unconstrained optimization,” IMA Journal of Applied Mathematics, vol. 9, no. 1, pp. 91–108, 1972.

[10] J. Gao, X. Liu, Y.-H. Dai, Y. Huang, and P. Yang, “Geometric convergence for distributed optimization with barzilai-borwein step sizes,” arXiv preprint arXiv:1907.07852, 2019.

[11] S. Sundaram and B. Gharesifard, “Distributed optimization under adversarial nodes,” IEEE Transactions on Automatic Control, vol. 64, no. 3, pp. 1063–1076, 2018.

[12] N. Ravi, A. Scaglione, and A. Nedić, “A case of distributed optimization in adversarial environment,” in ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019, pp. 5252–5256.

[13] I. Emiola, L. Njilla, and C. Enyioha, “On distributed optimization in the presence of malicious agents,” 2021.