The Consequences of Communication Deficiencies on Distributed Optimization Networks

Ethan Liu

BASIS Independent Silicon Valley, 1290 Parkmoor Ave, San Jose, CA 95126
ethantsliu@gmail.com

Abstract

There are many different types of online networks, one of which is multi-agent systems. A multi-agent system or network consists of multiple decision-making agents which interact in some environment to achieve a certain goal. Agents or nodes may be collaborative or antagonistic in achieving their goals. In distributed optimization of multi-agent systems, nodes cooperate to minimize a global function which is a sum of agents' local objective functions. Each agent has a connection or edge with some or all other agents. These systems occur in settings such as client-server architecture, machine learning, power systems, and smart manufacturing. Metaheuristic algorithms such as distributed stochastic gradient descent (D-SGD) and consensus-based optimization (CBO) are commonly used in distributed optimization in order to optimize the global function. However, some networks may have irregularities that affect performance: 1) stragglers, in which nodes inconsistently optimize the local function due to subpar computation power, and 2) faltering edges, where agents or nodes may have issues communicating due to device or network compatibility. We term these communication deficiencies and study them in various convex and non-convex multivariable benchmark function settings. We conclude that the two algorithms perform equally for benchmark functions in the non-convex setting. However, in all other settings, including the convex setting and all settings incorporating both types of communication deficiencies, CBO outperforms D-SGD.

1 Introduction

There are many different types of online networks, one of which is multi-agent systems. A multi-agent system consists of multiple decision-making agents which interact in some environment to achieve a certain goal. Agents may be collaborative or antagonistic in achieving their goals. In distributed optimization of multi-agent systems, agents collaboratively minimize a global function which is the sum of their local objective functions, subject to some constraints such as the communication network itself, which governs which agents may communicate with one another. Distributed optimization increases the speed of the network as it allows for parallel computing, with numerous agents optimizing the function at the same time [1]. With the advancement of machine learning and cloud computing, as well as other applications such as smart manufacturing, big data, and power systems, distributed optimization has become more crucial than ever [1, 2]. In this work, we focus on a specific issue within distributed optimization of multi-agent systems.

One of the most significant factors harming the performance of distributed optimization networks is communication deficiencies, which we will be focusing on in this paper. Due to the heterogeneous nature of distributed optimization, agents may have different hardware and characteristics, leading to varying local computation powers [3]. As a result, some machines may have weaker computation power than others. They can become stragglers and slow the network, which can lead to numerous issues including delayed communication with its neighbors and resource inefficiencies [3]. The second type of communication deficiency we will be focusing on is related to the edges themselves—the communication links from each node or agent to its neighbor. These links can be prone to failing or faltering in real-world situations due to reasons such as network outages and agent compatibility issues [3]. As a result of these two types of communication deficiencies, network performance may decrease substantially.

Communication deficiencies in distributed optimization have been examined in a variety of contexts. One example is machine learning. Distributed optimization lends concepts to federated learning, a distributed machine learning network in which the local objective functions are the cost functions in machine learning [4]. In federated learning, the use of a distributed optimization network overcomes the challenges of data privacy while also training a central machine learning model. However, federated learning also encounters communication deficiencies such as stragglers, due to the same concepts: lack of local computation power, or the incompatibility of data due to heterogeneity, as different agents may have differing subsets of data. One solution to this problem exists: simply identifying the stragglers, then dropping them entirely, excluding them from contributing to the network [5]. However, this solution still produces issues since the straggler's model update is lost and the straggler's data effectively no longer contributes to the global model. There are currently numerous asynchronous algorithms being developed in federated learning in order to combat stragglers [6].

Another example is electric power systems. Previously, Alkhraijah et al. examined the impact of intermittent communication failure on distributed optimization networks by creating a chance of edge failure model. They also compared the performance of several algorithms under those conditions, including the alternating direction method of multipliers (ADMM), Analytical Target Cascading (ATC), and Auxiliary Problem Principle (APP) algorithms in the context of electric power systems and optimal power flow algorithms [7]. However, this intermittent communication failure only examined the faltering of edges due to compatibility, rather than the straggler case.

Our gap in the current literature is the lack of research surrounding the consequences of communication deficiencies stragglers and faltering edges in distributed optimization. Therefore, the primary goal of this paper is to study them in detail. We will be examining three commonly-used metaheuristic algorithms—a general class of optimization techniques that seek an optimal, or near-optimal, solution to a problem, but that cannot guarantee that the solution they find will be optimal. These include decentralized stochastic gradient descent (D-SGD) and consensus-based optimization (CBO). Both D-SGD and CBO are burgeoning optimization algorithms that enable numerous agents or nodes to work on tasks without the need for a central server. In D-SGD, each node keeps a copy of the local objective function and communicates with other nodes to exchange gradient steps and ultimately average each node's gradient step [8]. In CBO, agents interleave local gradient descent steps with consensus iterations to attempt to reach the global minimum [9]. Gradient steps drive the solution to a global minimum, while the consensus iterations synchronize the values so that all

nodes come to a consensus on the final solution [9]. By combining these metaheuristic algorithms with rigorous benchmark functions, we will be able to determine the extent to which communication deficiencies such as stragglers and faltering edges have on the network.

Therefore, our research question is: To what extent do communication deficiencies such as stragglers and faltering edges affect the network performance of optimization algorithms on 2D, 4D, and 8D benchmark functions?

2 Methods

We choose a simulation-based method a theoretical method since we are examining the effect of communication deficiencies on a network's accuracy. Moreover, a simulation-based method allows for parameter exploration and reproducibility in future work. We will be simulating using Python code.

We test two metaheuristic optimization algorithms (D-SGD and CBO) on convex and non-convex 2D, 4D, and 8D benchmark objective functions, which will constitute six benchmark functions in total. In terms of communication, for the sake of simplicity, we test only fully interconnected networks of 100 nodes, all with learning rate $\alpha = 0.01$ and the number of epochs or rounds at 10,000. Thus, the weights between nodes will all be set to 1, and each node has equal sway over its neighbors. To simulate the communication deficiencies, we set the chances of stragglers and faltering edges from 0% to 100%. For the sake of simplicity, each node has the same chance of a communication deficiency occurring.

2.1 An Overview of Gradient Descent

The basic objective of a distributed optimization network is to minimize the global function which is the sum of the local objective functions. Suppose we have some network configuration G, in which every node may be connected to every other node, or some edges may be missing. This network $G \in \{N, E\}$ contains N nodes and E edges. We denote (θ_i, θ_j) as the edge between node θ_i and node θ_j, where all edges are bi-directional. We define a system state as $\{\theta_i\}_i \in \mathcal{N}_i$, and we seek to find a system state that reaches consensus $\theta_i = \theta_j \forall \mathcal{N}_i$, where $j \in \mathcal{N}_i$ which minimizes the global function which is the weighted (hence the w_i in Equation (1), depicted below) sum of the local objective functions.

$$\begin{aligned} \theta^* \in \arg\min \sum_{i=1}^{N} w_i f(\theta_i) \\ \text{s.t. } \theta_1 = \theta_2 = \ldots = \theta_N \end{aligned} \tag{1}$$

In Equation (1), θ^* is the optimal solution, and we attempt to converge all other final θ values to that θ^* value. This optimal state can easily be found in a centralized manner by solving Equation

(1); however, we seek to understand the performance of decentralized, local optimization algorithms.

To adequately understand the algorithms we will be using throughout this paper, we first need to examine the basic algorithm gradient descent. Gradient descent is a standard practice in the field of optimization, so it is crucial to understand its inner workings. It is used to minimize an objective function $f(\theta)$ parameterized by a model's parameters $\theta \in R_d$ by updating the parameters in the opposite direction of the gradient of the objective function $\nabla f(\theta)$ with reference to the parameters. The learning rate α, as the name implies, determines the rate that which the node "learns" the function, and also determines the size of the steps it takes to reach a local minimum. In other words, the node follows the direction of the slope of the surface created by the local objective function downhill until it reaches a valley and hits a local minimum point [10]. Each node in a network performs gradient descent in order to minimize some objective function in an attempt to reach the global minimum. The formula for the standard gradient descent algorithm is below. In order to assist the reader's understanding of the algorithm, Figure 1 is depicted below Equation 2 [11]. The gradient descent algorithm attempts to converge to the global minimum.

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \nabla f(\theta^{(t)}) \tag{2}$$

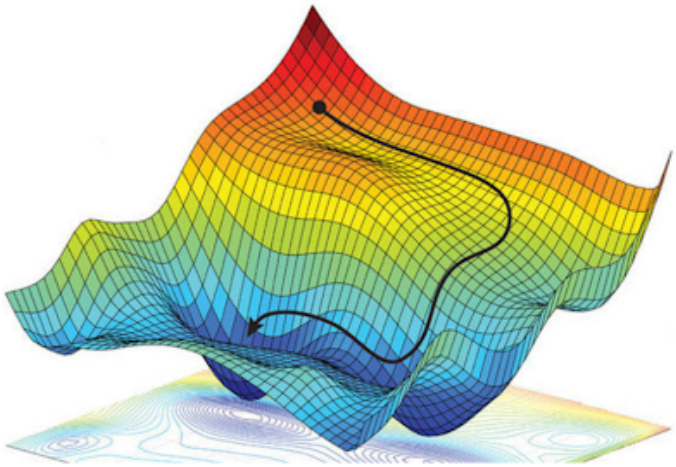

Figure 1: Gradient Descent
Source: [11]

2.2 Metaheuristic Optimization Algorithms

In this section, we familiarize ourselves with the two popular optimization algorithms that we will be using throughout the course of this paper. We will also utilize gradient clipping, in which if the gradient exceeds a certain threshold, we clip the gradient to minimize the step size.

2.2.1 Decentralized Stochastic Gradient Descent (D-SGD)

Decentralized Stochastic Gradient Descent is a form of the original gradient descent algorithm in which the algorithm is more stochastic, or random, and frequently checks the optimization of the function, making it faster and more preferable to traditional gradient descent [9, 12]. D-SGD, as the name implies, is simply the decentralized version of SGD, where all agents perform stochastic gradient descent and exchange their resulting gradients with each other, removing the need for a central model [12]. As each node exchanges its gradient $\nabla f(\theta)$ with its neighbors, it compiles the resulting gradient by weighing all of the gradients (w_{ij}) that it receives over the neighbors that it has (including its own gradient), depending on the edges of the communication network. The set of N_i neighbors for node θ_i is denoted as $\mathcal{N}_i$, where the cardinality is denoted as $|\mathcal{N}_i|$. The notation w_{ij} refers to the weight exerted on node θ_i by node θ_j, which can depend on the communication network. For example, the weight might be higher if node θ_i is receiving a gradient from its neighbor θ_j which has more neighbors itself than node θ_i. This is known as the mixing step. After that gradient is calculated, it takes the gradient step by multiplying the gradient against the learning rate α. Equation (3), shown below, may provide some clarity.

$$\theta_i^{(t+1)} = \theta_i^{(t)} - \alpha \times \frac{1}{|\mathcal{N}_i| + 1}(\nabla f(\theta_i^{(t)}) + \sum_{j \in \mathcal{N}_i} w_{ij} \nabla f(\theta_j^{(t)})) \tag{3}$$

2.2.2 Consensus-based Optimization (CBO)

Consensus-based optimization averages are based on the node's state or position instead of the gradients. In other words, in comparison to D-SGD, the mixing step in CBO does not calculate the weighted average of its neighbors' gradients, but the weighted average of its neighbors' positions along with its own [9]. After computing, the node takes a standard stochastic gradient descent step, multiplied by its learning rate. Equation (4), depicted below, may provide some clarity.

$$\begin{aligned} \theta_i^{(t+\frac{1}{2})} &= \frac{1}{|\mathcal{N}_i| + 1}(\theta_i^{(t)} + \sum_{j \in \mathcal{N}_i} w_{ij} \theta_j^{(t)}) \\ \theta_i^{(t+1)} &= \theta_i^{(t+\frac{1}{2})} - \alpha \nabla f(\theta_i^{(t+\frac{1}{2})}) \end{aligned} \tag{4}$$

2.3 Additional Methods

In this section, we outline some methods, or add-ons to the algorithm that we will be using in this paper.

2.3.1 Neighborhood Radius

Neighborhood radius limits the number of nodes that a node communicates with. This radius is the Euclidean Distance between nodes, essentially the straight-line path between two nodes. This is useful because nodes' information is more relevant if they are located closer to each other. The neighborhood radius is typically 10%-30% of the input space, and we'll set it to 30% in this experiment because we are studying communication settings.

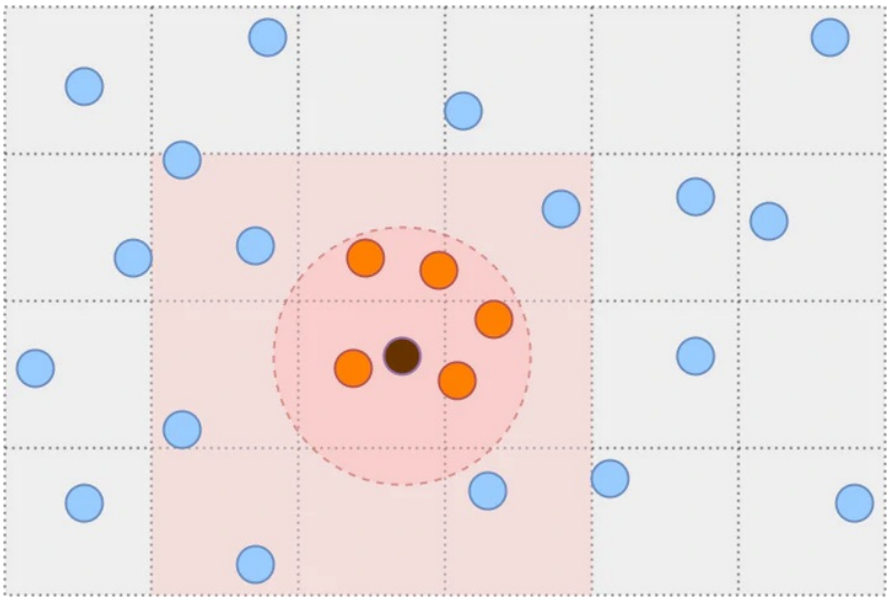

Figure 2: Neighborhood Radius
Source: [13]

2.3.2 Gradient Clipping

We will also be using gradient clipping, which limits the size of the gradient for each round. This is important because, without it, gradients can overshoot, as shown here. Gradient clipping is necessary for large-value functions that can cause algorithms to overshoot the minima, so we will only be using it for one benchmark function which will be discussed in the next section.

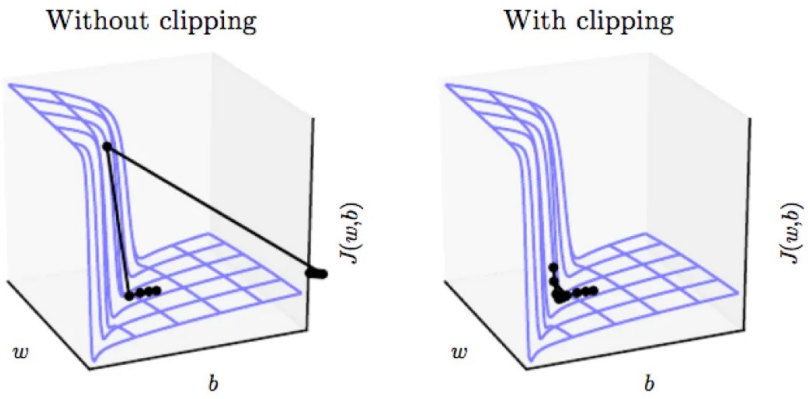

Figure 3: Gradient Clipping
Source: [14]

2.4 Benchmark Functions

We examine communication deficiencies in two types of optimization functions: convex and non-convex functions. Convex, unimodal functions have a unique global minimum, making optimization much easier, as one does not have to worry about becoming "trapped" at local minima.

On the other hand, non-convex functions can have multiple minima (multimodal), one of which is the optimal global minimum; hence, optimization is more challenging in this setting. Therefore, it is important to examine both types of optimization functions in multiple dimensions to simulate the challenges of optimization in any given setting. The more complex the setting, the more number of dimensions are required. The input space will depend on the space typically used for each benchmark function.

2.4.1 2D Functions

Two-dimensional input (2D) benchmark functions have the benefit of being graphed since they are in a 3D space, with the z plane representing the output of the function. As such, we will present the graphs of the convex and the non-convex benchmark functions, but this will not be true in later sections since it is impossible to accurately depict a high-dimensionality space (4 dimensions or higher). Instead, we will present the 3D space of those graphs purely for visualization purposes.

There are some criteria that must be met for our benchmark functions, both convex and non-convex. For the convex function, it must be unimodal and have only one minimum, the global minimum. There should be no local minima.

Therefore, for the convex function, we choose the Rosenbrock Valley Function, which is a commonly used function in optimization [15, 16]. The global minimum lies at $x^* = (1, ..., 1)$, and the function value at global minimum is $f(x^*) = 0$. The input space tested is $x^* \in [-5, 10]$ for however many input dimensions are tested. It is considered trivial to find the valley; however, it is difficult to find the global minimum within the valley. We will use gradient clipping on both algorithms for this function to prevent overshooting due to the rapidly ascending large values, and also the neighborhood radius.

Similarly, there are also criteria that must be met for the non-convex function. We desire the function to have numerous local minima and one global minimum in order to challenge the optimization algorithm.

For the non-convex function, we select the Ackley function, another widely used function in the field of optimization. This function has numerous local minima and one global minimum, making it non-convex and difficult to correctly optimize [17]. The global minimum occurs at $x^* = (0, ..., 0)$, where $f(x^*) = 0$. The input space tested is $x^* \in [-5, 5]$ for however many input dimensions are tested. Both functions are written below as Equations (5) and (6), respectively, and depicted in Figure 3 and Figure 4 for the reader's reference.

Convex:

$$f(x) = \sum_{i=1}^{d-1} [100((x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \quad \text{where d} = 2 \tag{5}$$

Non-convex:

$$f(x, y) = -20 \times \exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right) + 20 + e \quad \text{where d} = 2 \tag{6}$$

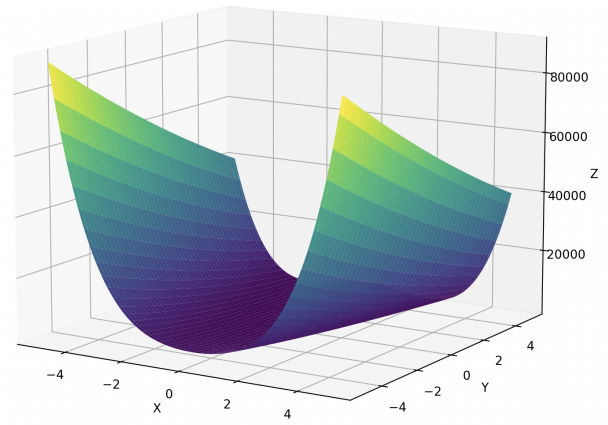

Figure 4: Rosenbrock Valley Function

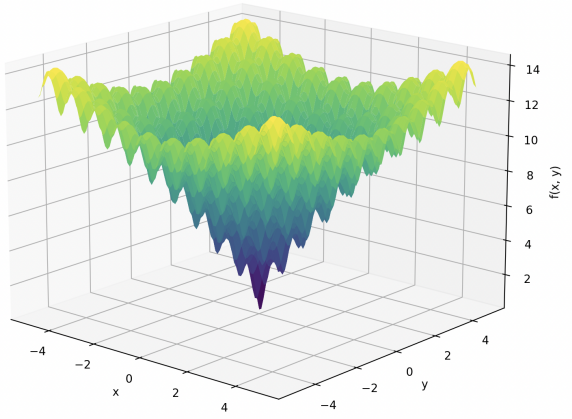

Figure 5: Ackley Function

2.4.2 4D Functions

The convex 4D function we choose will be the Rosenbrock Valley function, which is a commonly used unimodal, convex test function [15]. We set the dimensionality to 4 for this test case. As for the non-convex 4D function, we retain the same criteria of having numerous local minima and one global minimum in order to challenge the optimization algorithm. Since the Ackley function can be tuned to higher dimensions, it will suffice for the 4D non-convex function.

2.4.3 8D Functions

Since the criteria for the non-convex and convex functions have been fulfilled previously in the 4D function section and the previous 4D functions are scalable to higher dimensions, we choose to continue with the sum of squares for the convex function and the Ackley function for the non-convex function, only while changing the dimensions to 8, setting d = 8 in the equations.

3 Results

3.1 Ackley 3D Results

We will first study some cases of 2D input spaces on the non-convex Ackley function in order to understand how the two algorithms (D-SGD and CBO) are performing with and without communication deficiencies.

3.1.1 Base Cases

We present the base cases of both optimization algorithms with no communication deficiencies introduced. D-SGD (Figure 6) is on the left, below, and CBO (Figure 7) is on the right.

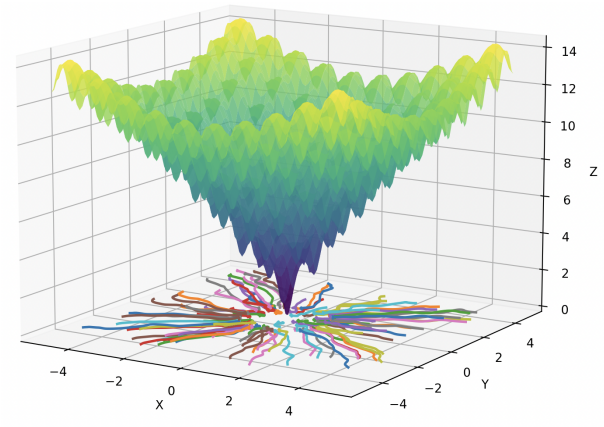

Figure 6: D-SGD, 100 nodes

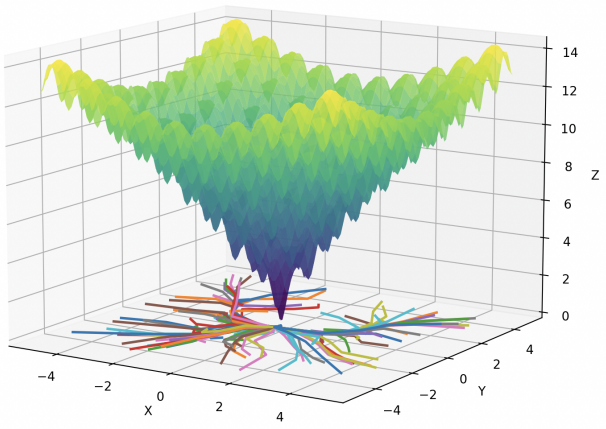

Figure 7: CBO, 100 nodes

These are the base cases with no communication deficiencies, with the trajectories of the nodes plotted on the xy plane. CBO is on the left, where we see rapid, smooth convergence to the global minimum, indicating that the nodes had no issue getting over the local minima because they were interacting with others and averaging the positions.

On the other hand, the right side is D-SGD, which has a "snaking" convergence to the minimum but also performs very well, although it doesn't completely reach the global minimum. Neither algorithm has a substantial issue with becoming trapped at local minima due to the collaborative nature of their network steps.

3.1.2 Stragglers, 50%

Here, in figures 8 and 9, we present one type of communication deficiency: stragglers, at 50%. In other words, for each round of the 10,000 rounds, about half of the 100 nodes will be stragglers.

Now we introduce some deficiencies. Interestingly enough, neither algorithm is impacted much by the 50% straggler deficiency. This is possibly due to the stragglers not affecting the system as much because there are 100 nodes populated in a small, [-5, 5] space. If there is a 50% straggler

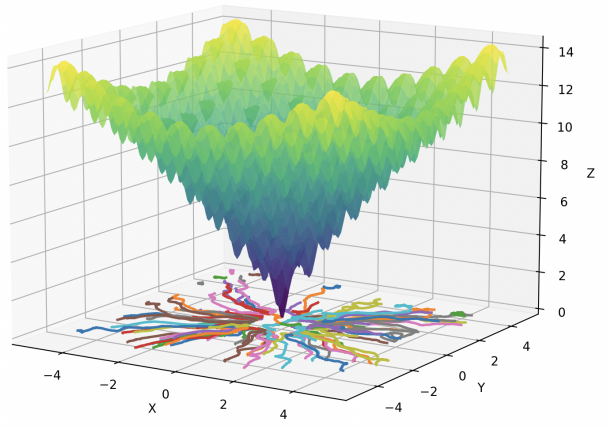

Figure 8: D-SGD, 100 nodes

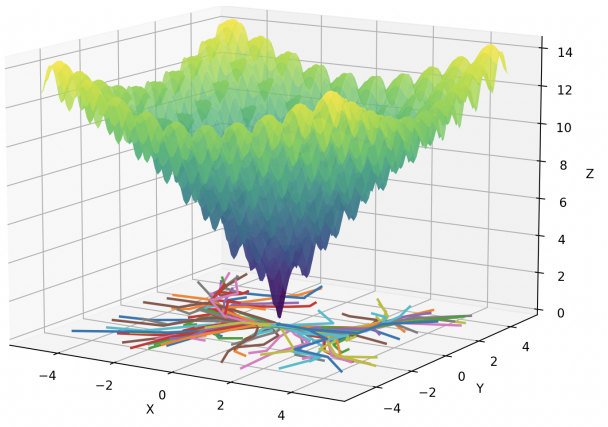

Figure 9: CBO, 100 nodes

issue, then there still remain 50 nodes, which allows the algorithm a great amount of flexibility.

Therefore, we choose to examine another case in the same setting, but this time with 10 nodes. As depicted below in Figure 10 and Figure 11, we can clearly see that both algorithms are clearly negatively affected by the stragglers. Due to neighborhood radius and stragglers, they lack nodes to interact with, and thus get trapped at local minima.

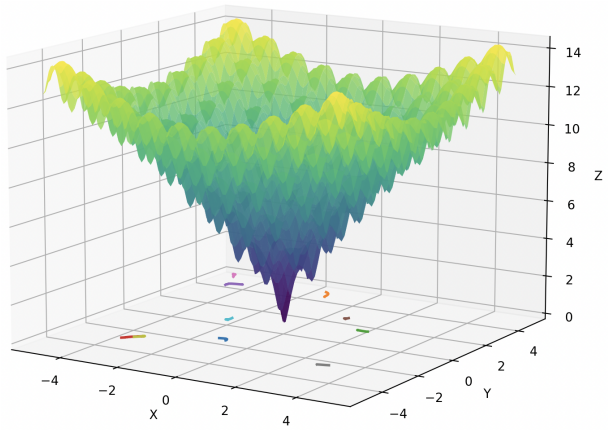

Figure 10: D-SGD, 10 nodes

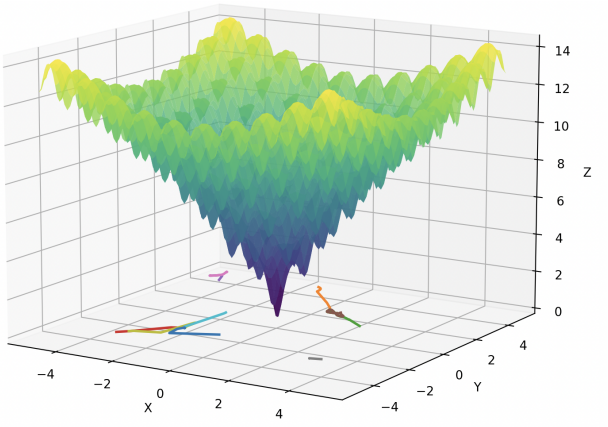

Figure 11: CBO, 10 nodes

As depicted in Figures 10 and 11, due to the low node count, we can see that both algorithms are clearly negatively affected by the stragglers. Due to neighborhood radius and stragglers, they lack nodes to interact with, and thus get trapped at local minima.

3.1.3 Faltering Edges, 50%

Now, we will examine the other type of communication deficiency, faltering edges. As depicted in Figures 12 and 13 below, CBO has slightly more difficulty with faltering edges, as not all nodes converge to the minimum, and some become stuck in the local minima. However, overall, it still performs superior to that of D-SGD. It is clear that D-SGD suffers from the communication deficiency heavily, as nearly all nodes become trapped in local minima.

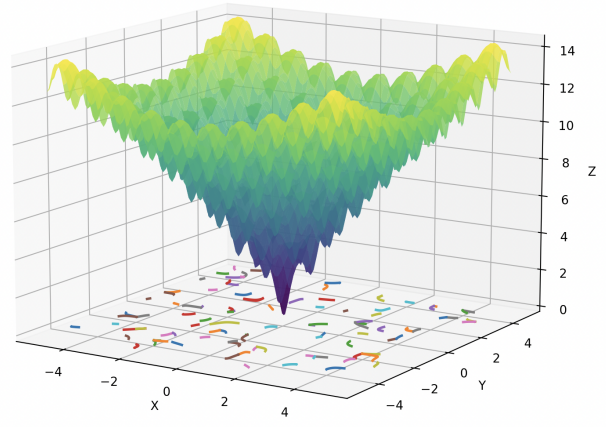

Figure 12: D-SGD, 100 nodes

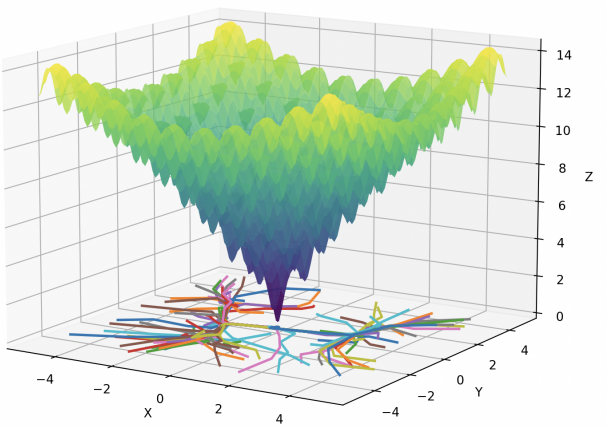

Figure 13: CBO, 100 nodes

Similar to the previous section, we will examine how the algorithms perform with 10 nodes instead of 100 nodes. As shown below in Figures 14 and 15, both D-SGD and CBO struggle with fewer nodes in the input space.

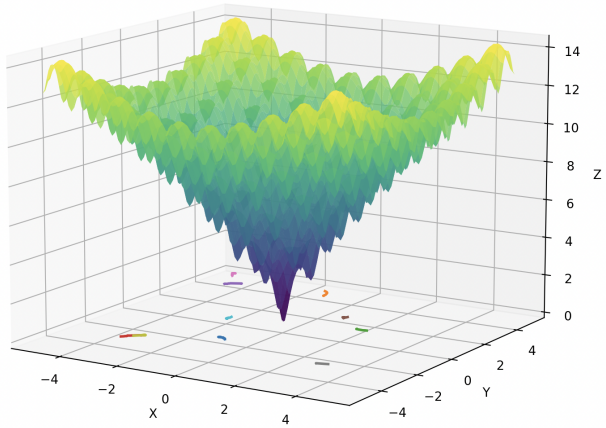

Figure 14: D-SGD, 10 nodes

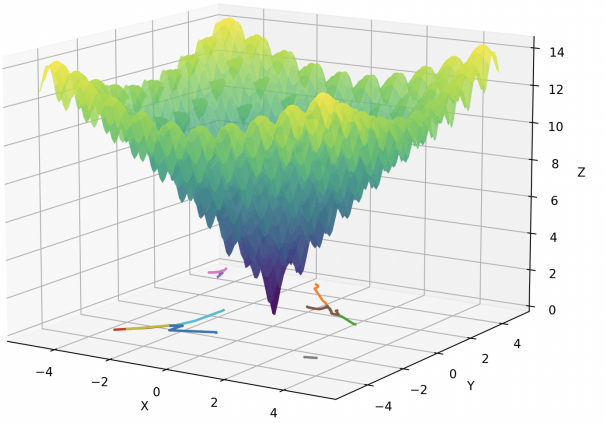

Figure 15: CBO, 10 nodes

3.1.4 Stragglers, 100%

At 100% deficiency, stragglers cause the network to not run at all, so there is no use depicting the results there. Instead, we will examine the results of faltering edges at 100%.

3.1.5 Faltering Edges, 100%

By setting the faltering edges to 100%, this is essentially what the networks look like without communication at all, and exclusively using stochastic gradient descent.

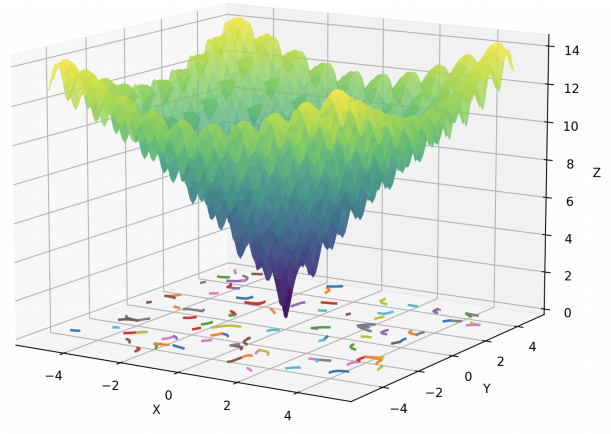

Figure 16: D-SGD, 100 nodes

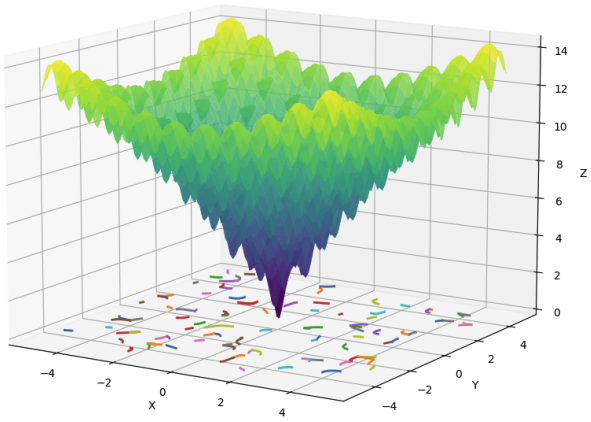

Figure 17: CBO, 100 nodes

4 Main Results

Now we'll move onto the comprehensive results, where we test the chance of faltering from 0% to 100% using L2 norms, which are the Euclidean distance between the global minimum and the final average value of the network. This is the straight-line distance between the two points, and so the higher the L2 norm, the worse the network performance. The further right the point, the higher the communication deficiency. The source code is provided on Github in the left side of the footer for the reader's convenience.

4.1 Ackley Results: Stragglers

4.1.1 2D

We see that, in a 2D input space, affected by stragglers, In the 2D input space,the performance of both CBO and D-SGD declines.

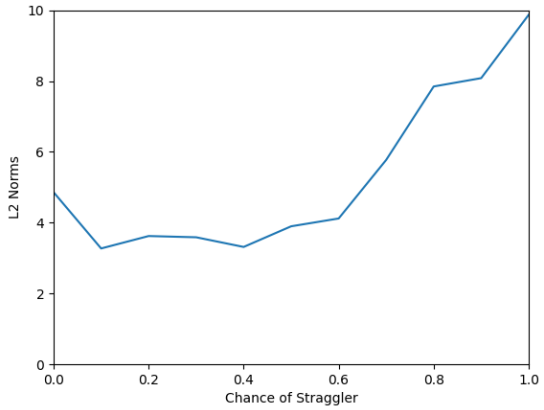

Figure 18: D-SGD, 100 nodes

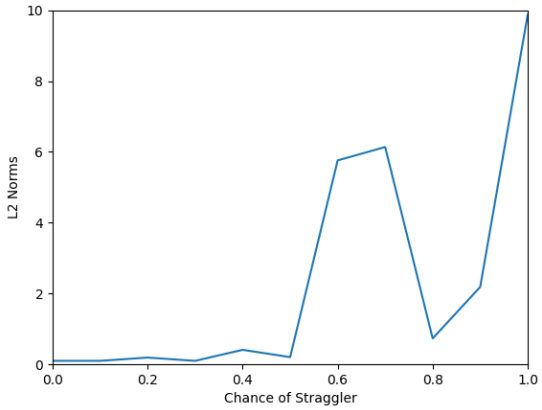

Figure 19: CBO, 100 nodes

4.1.2 4D

The efficacies for both algorithms are about the same throughout for 4D input spaces. However, once again CBO is not heavily affected by communication deficiencies, whereas D-SGD clearly suffers from a lack of communication.

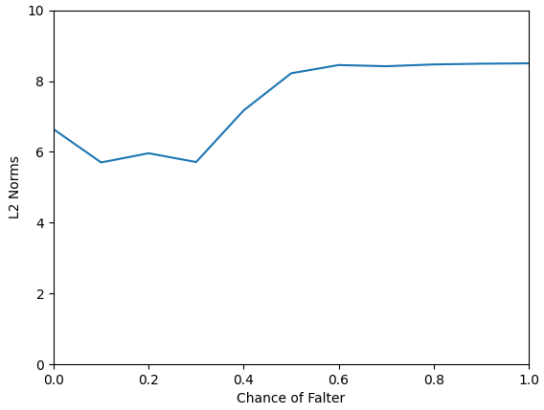

Figure 20: D-SGD, 100 nodes

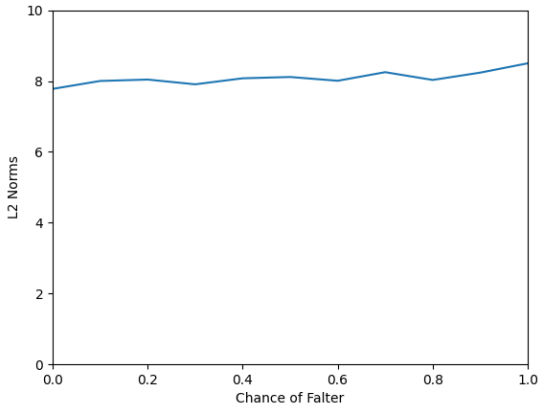

Figure 21: CBO, 100 nodes

4.1.3 8D

In the 8D input space, the L2 norms are already too high and constant for stragglers to affect these algorithms, since it is incredibly difficult to optimize in higher dimensions with the Ackley function, which is multimodal. Since these nodes are not moving anywhere in the first place due to being trapped at local minima, the stragglers do not affect them.

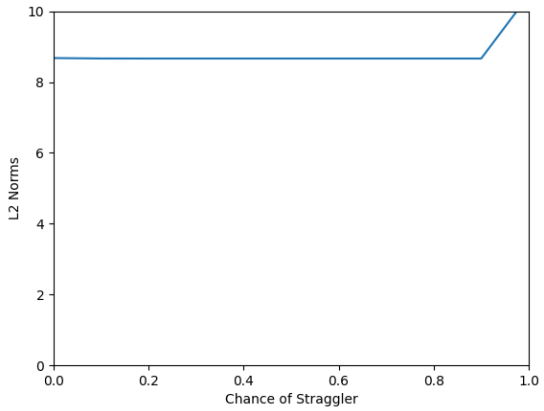

Figure 22: D-SGD, 100 nodes

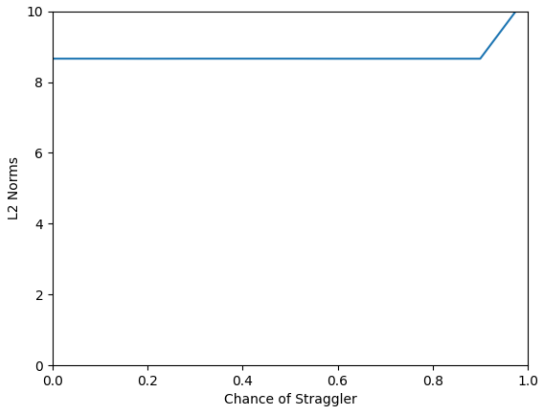

Figure 23: CBO, 100 nodes

4.2 Ackley Results: Faltering Edges

In this section, we interpret the effects of faltering edges on the two optimization algorithms in the non-convex Ackley function.

4.2.1 2D

We see that, in a 2D space, CBO performs much better than D-SGD. CBO isn't heavily affected by the faltering edges communication deficiency, and the network is only negatively affected with very high amounts of faltering edges, that being from 80% and onwards.

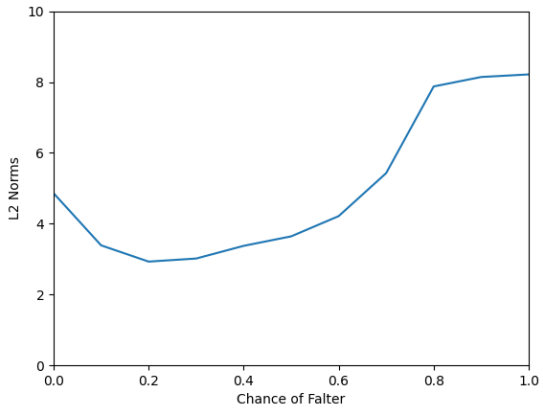

Figure 24: D-SGD, 100 nodes

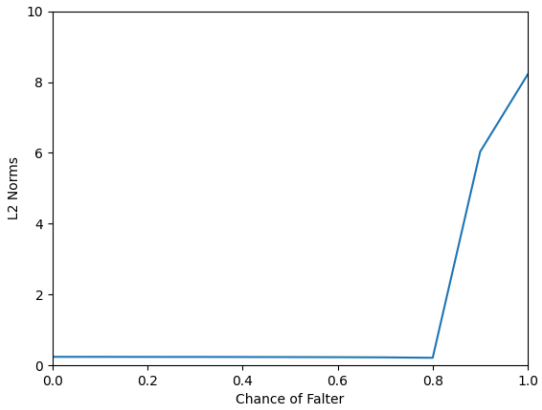

Figure 25: CBO, 100 nodes

4.2.2 4D

Looking at 4D results in Figures 26 and 27, the efficacies are about the same throughout. D-SGD is affected, as the L2 norm increases with more faltering edges. However, CBO is not significantly affected by the higher amounts of faltering edges; it is consistent regardless of the amount.

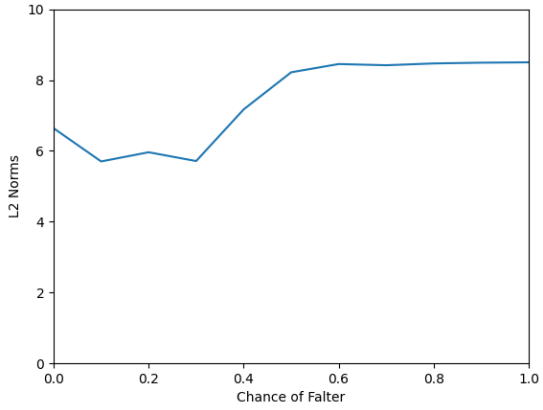

Figure 26: D-SGD, 100 nodes

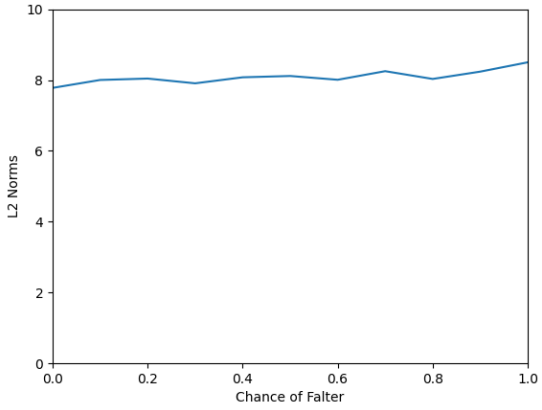

Figure 27: CBO, 100 nodes

4.2.3 8D

Looking at 8D results, the efficacies are about the same throughout. However, once again CBO is not affected by faltering edges.

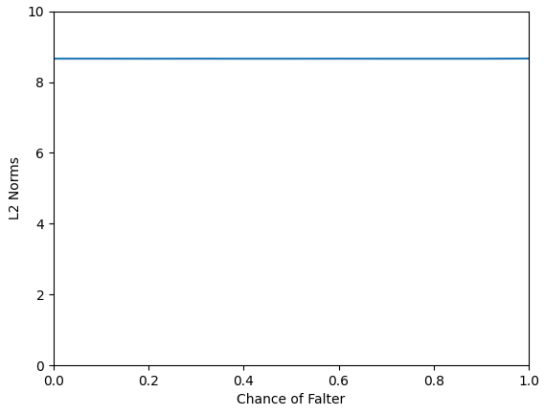

Figure 28: D-SGD, 100 nodes

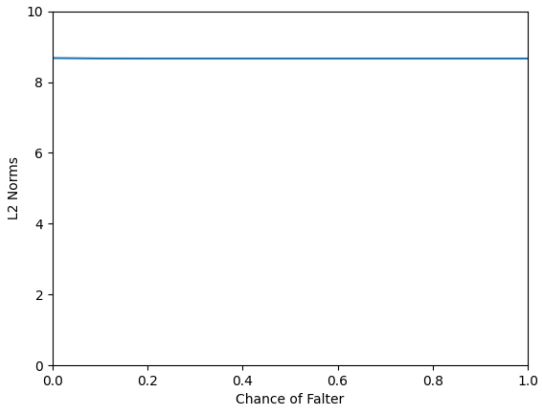

Figure 29: CBO, 100 nodes

4.3 Rosenbrock Results: Stragglers

Note that D-SGD performs significantly worse for the Rosenbrock function, so much so that y-axis of the D-SGD graph on the right is scaled to 200,000 for the y axis. This could be due to the high values of the Rosenbrock valley and the sharing of gradients confusing the algorithm on the direction toward the minima. In comparison, the CBO is at nearly zero.

4.3.1 2D

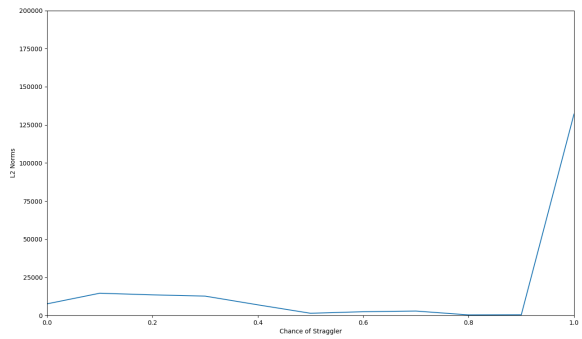

Figure 30: D-SGD, 100 nodes

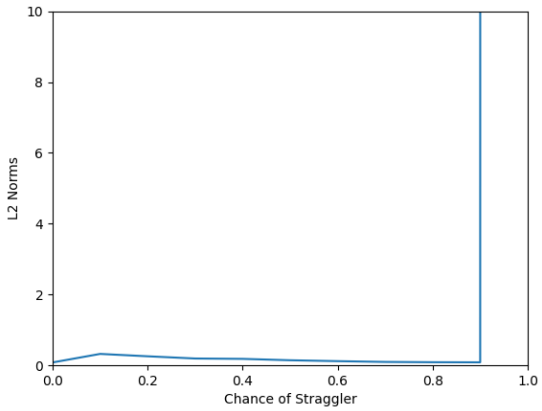

Figure 31: CBO, 100 nodes

4.3.2 4D

The same trend in the 2D space is seen in the 4D space. Once again, note that the y-axis of the D-SGD graph on the right is scaled to 200,000 in order to accommodate the extremely large values of the L2 norms.

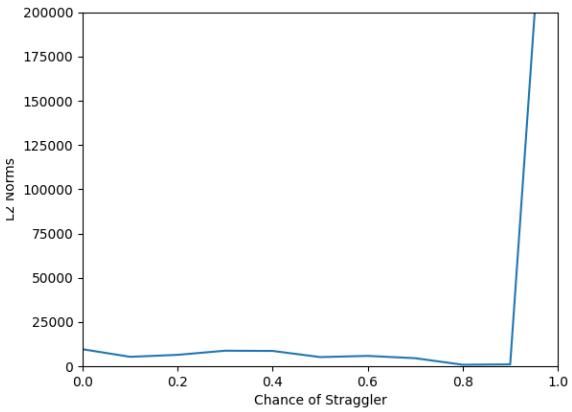

Figure 32: D-SGD, 100 nodes

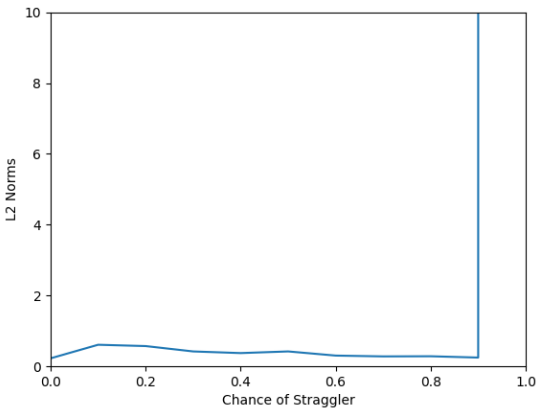

Figure 33: CBO, 100 nodes

4.3.3 8D

The trend continues in the 8D space.

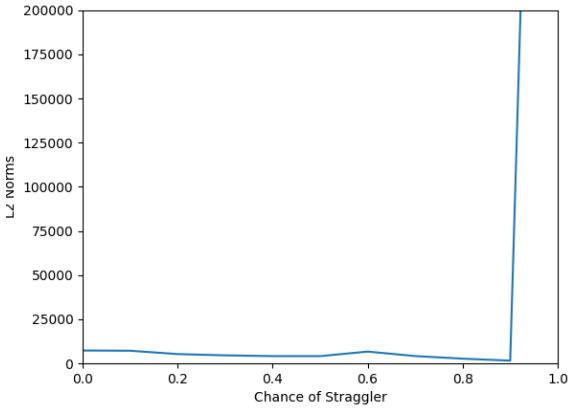

Figure 34: D-SGD, 100 nodes

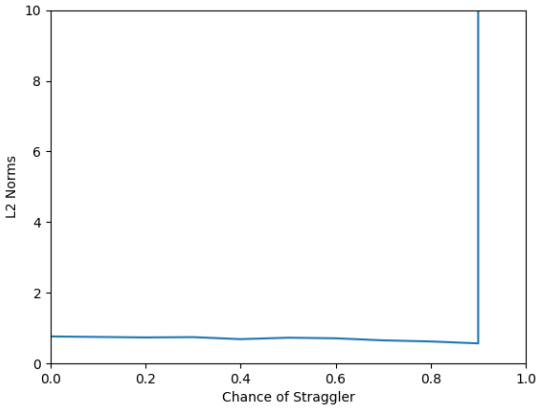

Figure 35: CBO, 100 nodes

4.4 Rosenbrock Results: Faltering Edges

4.4.1 2D

As shown below in Figures 36 and 37, even with the change of the type of communication deficiency, D-SGD still performs badly for the Rosenbrock function, whereas CBO performs well.

One interesting detail to examine is that D-SGD on the Rosenbrock function performs the opposite of what one would expect: with more faltering edges and a higher chance of communication deficiencies, it actually begins to perform better on the Rosenbrock function. This signifies that D-SGD is more effective as SGD, and that communication could be interfering with the algorithm.

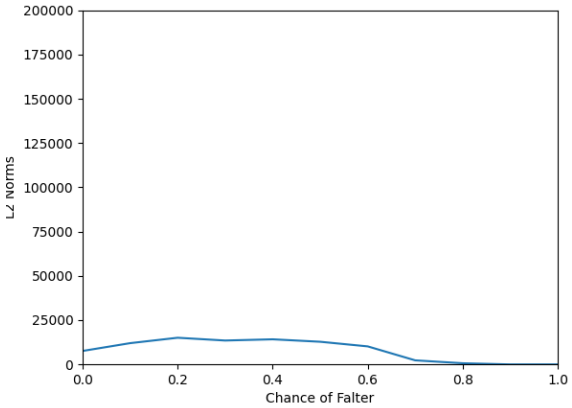

Figure 36: D-SGD, 100 nodes

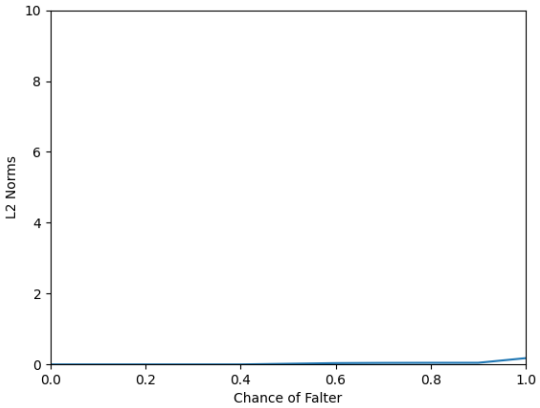

Figure 37: CBO, 100 nodes

4.4.2 4D

The same trend continues in the 4D input space.

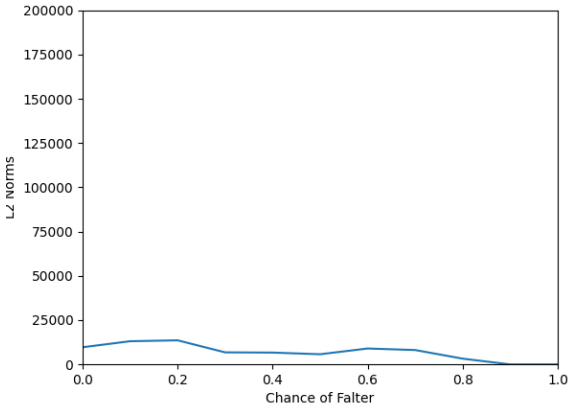

Figure 38: D-SGD, 100 nodes

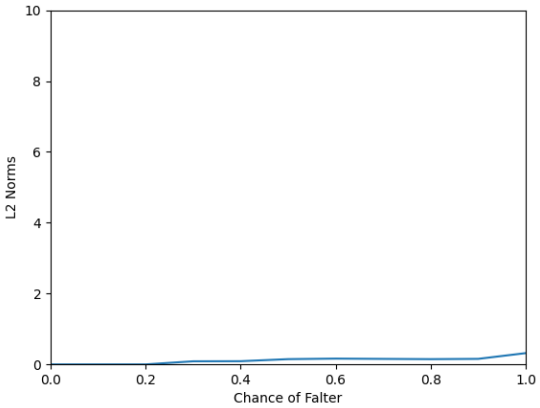

Figure 39: CBO, 100 nodes

4.4.3 8D

The same trend continues in the 8D input space.

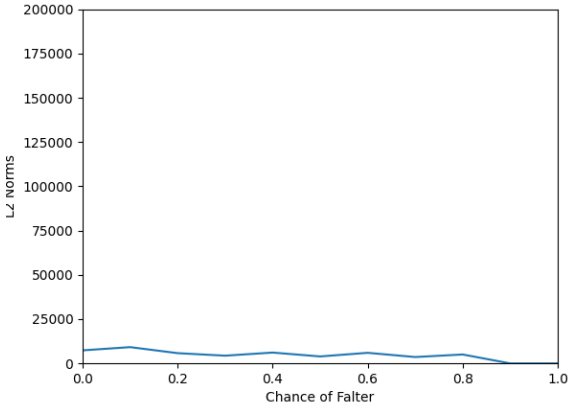

Figure 40: D-SGD, 100 nodes

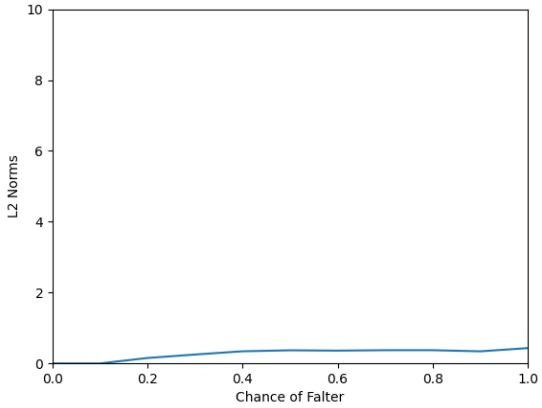

Figure 41: CBO, 100 nodes

5 Discussion

Ultimately, CBO generally outperforms D-SGD for the Rosenbrock function, even though the Rosenbrock function is considered easy to optimize. This could be due to the large number of nodes in a small input space, which allows the CBO algorithm an advantage since position-based averaging aids the network in converging to the center, which is coincidentally the global minimum in this case. Another reason is that CBO helps nodes achieve more decisive action, whereas D-SGD can cause "rubber-banding" gradients where the nodes take steps that negate each other, adding up to no progress made. With large gradients due to the rapidly ascending values of the Rosenbrock function, the D-SGD algorithm performs poorly, even with the incorporation of gradient clipping. Instead, incorporating normal SGD without the network seems to rectify the issue. This is shown in all of the figures in Section 4.4, where more faltering edge communication deficiencies lead to less communication between nodes, which lead to lower L2 norms as the nodes just optimize on the SGD algorithm.

However, CBO and D-SGD perform about equally on the Ackley function at base value, but CBO performs better with communication deficiencies, as it manages to avoid being “trapped” at local minima using its position-averaging algorithm.

Overall, CBO outperforms D-GSD by a significant amount in multivariable, convex and non-convex benchmark function settings. Communication deficiencies such as stragglers and faltering edges negatively imapct the optimization of these algorithms to a significant degree in non-convex benchmark functions, but in convex benchmark functions, they have a varied impact. D-SGD does not optimize well in the base case, and the straggler communication deficiency causes the L2 norm to skyrocket, whereas the faltering edges communication deficiency paradoxically causes the L2 norm to dip. On the other hand, CBO remains almost completely unaffected by the communication deficiencies, except for the setting where the stragglers are very high, since that prevents the algorithm from optimizing in the end.

6 Conclusion and Future Work

This work contributes to the field of distributed optimization and distributed systems, allowing researchers to better understand the consequences of communication deficiencies in these systems. By improving the comprehension of researchers, this work can help inform researchers of which main issues to tackle when developing new algorithms to combat communication deficiencies that occur in distributed optimization.

Future work could consist of analyzing when an algorithm is interrupted mid-step while calculating the gradient steps. It could also analyze other algorithms such as the Alternating Direction of Multipliers Method (ADMM), and the ADAM optimizer, a popular machine learning technique that utilizes adaptive learning rates. More examples of multivariable convex and non-convex optimization benchmark functions could be used in order to ensure the rigor of this study.

7 References

[1] T. Yang et al., "A survey of distributed optimization," Annual Reviews in Control, vol. 47, pp. 278–305, Jan. 2019, doi:10.1016/j.arcontrol.2019.05.006.
[2] Patari et al., "Distributed Optimization in Distribution Systems: Use Cases, Limitations, and Research Needs | IEEE Journals & Magazine | IEEE Xplore." Accessed: Mar. 19, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/9635694
[3] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data." arXiv, Jan. 26, 2023. Accessed: Jul. 15, 2023. [Online]. Available: http://arxiv.org/abs/1602.05629
[4] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, "A review of applications in federated learning," Computers & Industrial Engineering, vol. 149, p. 106854, Nov. 2020, doi:10.1016/j.cie.2020.106854.
[5] C. Li, H. Shen, and T. Huang, "Learning to Diagnose Stragglers in Distributed Computing," in 2016 9th Workshop on Many-Task Computing on Clouds, Grids, and Supercomputers (MTAGS), Nov. 2016, pp. 1–6. doi:10.1109/MTAGS.2016.04.
[6] "Straggler-Resilient Federated Learning: Leveraging the Interplay Between Statistical Accuracy and System Heterogeneity | IEEE Journals & Magazine | IEEE Xplore." Accessed: Mar. 21, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/9887795
[7] M. Alkhraijah, C. Menendez, and D. K. Molzahn, "Assessing the impacts of nonideal communications on distributed optimal power flow algorithms," Electric Power Systems Research, vol. 212, p. 108297, Nov. 2022, doi:10.1016/j.epsr.2022.108297.
[8] A. Koloskova, N. Loizou, S. Boreiri, M. Jaggi, and S. Stich, "A Unified Theory of Decentralized SGD with Changing Topology and Local Updates," in Proceedings of the 37th International Conference on Machine Learning, PMLR, Nov. 2020, pp. 5381–5393. Accessed: Feb. 06, 2024. [Online]. Available: https://proceedings.mlr.press/v119/koloskova20a.html
[9] 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), Oct. 2012, pp. 1543–1550. doi:10.1109/Allerton.2012.6483403.

[10] S. Ruder, "An overview of gradient descent optimization algorithms." arXiv, Jun. 15, 2017. Accessed: Mar. 22, 2024. [Online]. Available: http://arxiv.org/abs/1609.04747
[11] A. Amini, A. Soleimany, S. Karaman, and D. Rus, "Spatial Uncertainty Sampling for End-to-End Control." arXiv, May 23, 2019. Accessed: Mar. 25, 2024. [Online]. Available: http://arxiv.org/abs/1805.04829
[12] L. Ding, K. Jin, B. Ying, K. Yuan, and W. Yin, "DSGD-CECA: Decentralized SGD with Communication-Optimal Exact Consensus Algorithm," in Proceedings of the 40th International Conference on Machine Learning, PMLR, Jul. 2023, pp. 8067–8089. Accessed: Mar. 22, 2024. [Online]. Available: https://proceedings.mlr.press/v202/ding23b.html
[13] J. T. de Balsch, "Searching for Fixed-Radius Near Neighbors with Cell Lists Algorithm in Julia Language," Jaan Tollander de Balsch. Accessed: Apr. 24, 2024. [Online].
Available: https://jaantollander.com/post/searching-for-fixed-radius-near-neighbors-with-cell-lists-algorithm-in-julia-language/
[14] W. Wong, "What is Gradient Clipping?," Medium. Accessed: Apr. 24, 2024. [Online]. Available: https://towardsdatascience.com/what-is-gradient-clipping-b8e815cdfb48
[15] Ma, J. and Li, H. (2019) Research on Rosenbrock Function Optimization Problem Based on Improved Differential Evolution Algorithm. Journal of Computer and Communications, 7, 107-120. doi:10.4236/jcc.2019.711008.
[16] M. Molga and C. Smutnicki, "Test functions for optimization needs". Apr. 2005, [Online]. Available: https://robertmarks.org/Classes/ENGR5358/Papers/functions.pdf
[17] J. Xing, Y. Luo, and Z. Gao, "A global optimization strategy based on the Kriging surrogate model and parallel computing," Structural and Multidisciplinary Optimization, vol. 62, Jul. 2020, doi:10.1007/s00158-020-02495-6.