P2P-LLM: Collaborative LLM Inference

Andrew Boessen BOESSENA@BC.EDU

Abstract

This project explores a framework that enables distributed running of large language models across personal computing devices connected in a peer-to-peer network. While current advanced AI models require substantial computing resources typically available only to large organizations, P2P-LLM offers an alternative approach by dividing these models into smaller parts that can be processed across multiple personal computers. The system includes efficient routing mechanisms to find optimal paths through the network, balancing computational speed and cost. To encourage participation, the project develops a fair pricing system where participants are rewarded for contributing their computing power. This marketplace is secured by blockchain technology, which handles payments and ensures that agreements between participants are honored. The P2P-LLM approach demonstrates how collaborative computing could make advanced AI more accessible while enhancing privacy and user autonomy, potentially transforming how these technologies are accessed in the future.

1 Introduction

In recent years, large language models (LLMs) have demonstrated remarkable capabilities across diverse tasks, exhibiting unprecedented rates of improvement within a relatively brief developmental timeline. This extraordinary progress can be primarily attributed to the immense computational resources allocated to training these models on datasets comprising trillions of tokens, resulting in systems that subsequently demand substantial computing power for deployment. The computational intensity of these models has resulted in significant investment, with billions of dollars directed toward the construction of specialized super-clusters housing hundreds of thousands of GPUs dedicated to the training and serving of LLMs. Given the substantial memory and computational requirements of state-of-the-art LLMs, most personal computing devices lack the necessary specifications to run or train these models independently. Consequently, access to LLM capabilities is predominantly channeled through services offered by large technology corporations, which may present concerns regarding privacy, autonomy, and accessibility.

Crowdsourcing methodologies offer a promising alternative to address these limitations. The crowdsourcing approach has demonstrated considerable success across various domains, including knowledge aggregation (Wikipedia), real-time traffic information collection (Waze), and venture capital formation (Kickstarter). These same foundational principles can be effectively applied to computational resource allocation and utilization.

My work explores fully decentralized crowd computing architectures, where users establish a peer-to-peer network in which each participant can leverage the collective computational resources for individual tasks. This structure shares conceptual similarities with BitTorrent file-sharing networks. While current state-of-the-art language models exceed the

capacity of individual personal computers, the aggregation of distributed computational resources through a decentralized network could potentially enable users to host their own LLMs, thereby democratizing access to advanced artificial intelligence capabilities.

2 Related Work

2.1 Crowd Computing

The idea of pooling compute power from many computers to form a distributed supercomputer is not new. This concept of Crowd Computing has been used ever since large networks of computer were formed. The predominant use of this method is to facilitate academic research. Researchers publish problem that that they are working on, and anyone can help contribute by donating their computing power. Projects that have successfully used crowd computing include SETI@Home (Anderson et al., 2002) and Einstein@Home (Steltner et al., 2021). This approach has proven to be successful in the past, but it has become less popular as the computational power of computers and access to supercomputers has increased. However, the use of idle compute power still has untapped applications, and the principles from crowd computing can be applied to modern tasks like LLM inference.

2.2 P2P Networks

LLM inference is a perfect candidate for P2P network because it has technical requirements that prohibit most personal computers from being able to run large models with billions of parameters, and it can easily be split into small chunks that can be run on several computers at the same time. BitWorker (Durand et al., 2015) is a protocol designed to compute any parallel computing task across a peer-to-peer network. Although it is possible to apply LLM inference to this protocol, BitWorker is designed for general tasks and would not able to efficiently scale to the demands of many users. The most notable research being done in this area is Petals (Borzunov et al., 2023) which is a BitTorrent like peer-to-peer network for LLM inference and finetuning. Petals creates a network with client and servers. Clients send requests that will be processed by the servers on the network. The model is split into multiple sequential blocks of the LLM's layers, and each server in the network is capable of serving one or multiple of these blocks. In order to run an inference step, a client will find a optimal path of servers in the network that forms the complete model and minimizes the time required to compute the LLM. In my work, I explore a similar approach with a network of distributed computers that serve a specific part of the LLM, but I expand upon this and add incentive mechanisms that reward nodes for donating their computational power. This incentive mechanism also facilitates optimal allocation of nodes to maximize throughput, by each node maximizing expected revenue individually.

3 Peer-to-Peer Network

Fundamentally, a peer-to-peer network is a decentralized system where each devices acts as a client and a server. Each devices can send and receive data which eliminates the need for a centralized server. The central benefit of a P2P network is that resources are pooled and shared between peers, so the combined resources available to an individual peer is

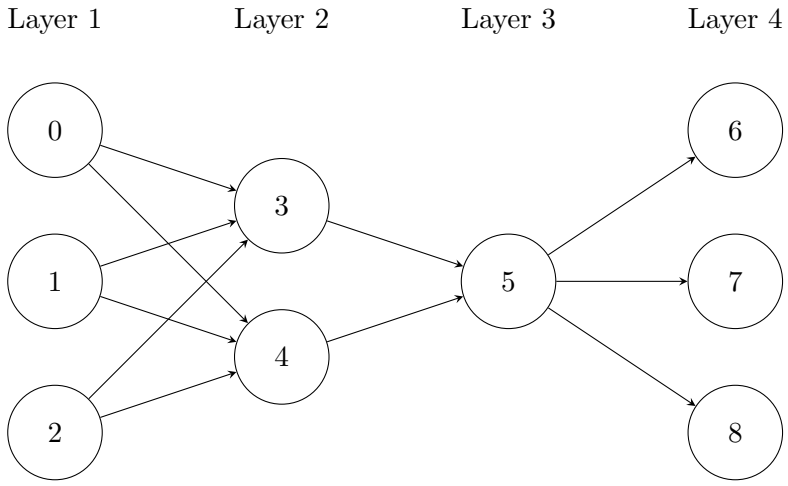

Figure 1: P2P-LLM Network Directed Graph

greater than that peer's own resources. A peer-to-peer network can be built to share any type of resource. In the case of BitTorrent, bandwidth and files are shared between peers. For the case of LLM inference, peers will need to be able to share bandwidth, data, and computational power.

A peer-to-peer network is defined as having n nodes in the network. There is an LLM model with m total layers which will be divided between nodes. Each node N in the network is a tuple $N(\ell, l, c, p)$ where

- $\ell_i = (x, y) : x, y \in [0, m); y > x$ is the range of layers the node is able to serve
- $l_{ij} \in \mathbb{R}$ is the network latency between nodes i and j,
- $c_i \in \mathbb{R}$ is the computational cost of the node, measured in milliseconds
- $p_i \in \mathbb{R}$ is the price the nodes charges for its compute.

3.1 Path Finding

In order to use the network to run the LLM, a peer must first find a sequence of servers in the network that form a complete model. Because the LLM layers must be computed sequentially, the peer must find a path through the network that will ideally compute the LLM in the least amount of time possible and at the lowest price. This problem can be solved by viewing the network as a directed graph where nodes are peers serving the LLM layers and weights correspond to the computational cost and latency of nodes. The edges of the graph correspond to the compatibility between the layers that each node in serving. Specifically, each node is connected to every node that is serving the consecutive layer of the model. A visualization of this directed graph is shown in Figure 1.

Due to this property, the P2P network graph is a directed acyclic graph (DAG). This means that there are efficient algorithms to find the shortest path between two nodes. The

first step is to perform a topological sort on the nodes in the network. This is an ordering of the nodes such that all neighbors of a node come after the node in the ordering. For any graph $G = (V, E)$ a topological ordering is an order $\prec$ on V such that if $(u \to v) \in E$ then $u \prec v$. For example, a topological ordering of the DAG in Figure 1 is $[0, 1, 2, 3, 4, 5, 6, 7, 8]$. The complete algorithm for topological sort is shown in Appendix A. With the topological order, dynamic programming can be used to find the shortest path in the network. This algorithm will iterate over the ordering, and search all the node's neighbors for the one with the lowest weight. The full algorithm is shown in Appendix B. Most importantly, both the topological sort and path finding algorithm run in linear time with a computational complexity of $O(|V| + |E|)$.

3.2 Pareto Efficiency

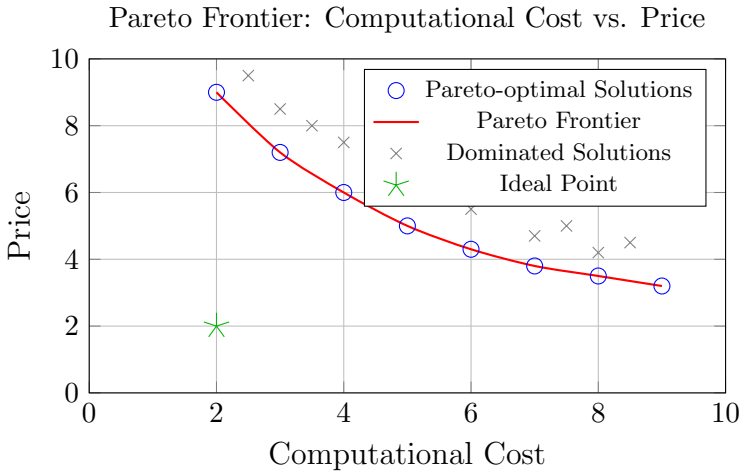

Figure 2: Pareto Frontier

The optimal path is well defined when weights for nodes are clearly defined, but there is not an apparent way to define weights for nodes in this case. In the P2P-LLM network, there are two competing criterion that define the weight of node: computational cost and price. The algorithm described above uses a single objective (SO), while the optimal path in the network is a multi-objective (MO) optimization problem. There are several way to find optimal solutions to MO optimization problems (Ngatchou et al., 2005). The simplest way is to convert it to a SO optimization problem, and since the shortest path algorithm is designed for a single objective, it is the most practical solution for this task. In order to reframe the problem as a single objective, a weighed aggregate of the multiple objectives is used. In the case of the peer-to-peer network, the SO is a weighed sum of computational cost c_i and price p_i. The objective, or weight, of a node w_i is defined as.

$$w_i = \alpha \cdot p_i + (1 - \alpha) \cdot c_i \tag{1}$$

However, this assumes that the parameter α is known a priori. In general, there are many equivalent optimal solutions in a MO optimization problem. These are known as

Pareto-optimal solutions. They are solutions such that there is no way to improve one objective without degrading any other objectives. A solution is said to be Pareto-optimal if there are no other solutions that dominate it. A solution x_1 dominates x_2 if

1. $\forall i, f_i(x_1) \leq f_i(x_2)$ and
2. $\exists i, f_i(x_1) < f_i(x_2)$

where $f_i()$ is the utility for the ith objective.

The set of all Pareto-optimal solutions forms a frontier (Figure 2) of equivalent solutions, but there is not a clear choice on which one should be used. In the case of the SO weighted aggregate used in the P2P-LLM network, the parameter α decides where on the Pareto frontier the optimal solution lies. Due to this, each node can independently decide the importance of price and computational cost in the optimal path through the network by choosing the parameter α.

3.3 Revenue Optimization

Each peer additionally has to choose a price to charge for other peers to use their computation. Logically, each node will act in its self-interest and choose the price that maximizes revenue. If the price is too high, peers will choose equivalent nodes serving the same layers, while if the price is too low, the total revenue will be less than ideal. The price they choose should reflect the current state of the network, and adapt to changing conditions. In this context, price optimization become difficult as it is a dynamic environment and a uniform pricing rule would not result in optimal revenue. In the P2P-LLM network, a peer only has to compete with other peers serving the same layers. This can be exploited to narrow down the complexities of this problem. Working with this leads to the conclusion that a node's price should be determined by its own price and computational cost and the price and computational cost of all other peers serving the same layers.

If we can define a revenue function based on these parameters, then the optimal price can be found by maximizing this function with respect to p_i. The revenue a node receives for a single peer using the network would be its price multiplied by the probability of being chosen in the optimal path.

$$R(p_i) = p_i \cdot \mathbb{P}(w_i) \tag{2}$$

The probability that a node is chosen is related to its weight (Equation 1). In order to get a distribution across all weights in the node's layer, the Boltzmann distribution (Boltzmann, 1868) can be used. This distribution give the probability that a system will be in certain state.

$$\mathbb{P}(\epsilon_i) = \frac{e^{-\epsilon_i/kT}}{\sum_{j=1}^{M} e^{-\epsilon_j/kT}} \tag{3}$$

where ϵ_i is the energy for state i, k is the Boltzmann constant, and T is the temperature of the system. For the purposes of the P2P-LLM network, k and T are dropped and ϵ_i is w_i (Equation 1). Using this method gives a smooth expected revenue function, and prices will reflect how much each nodes contributes to the network. Specifically, if a nodes contributes

more compute power to the network, demonstrated as a lower computational cost c_i, the optimal price they will charge will be higher, and they will be rewarded with a higher expected revenue. A visualization of how c_i effects the expected revenue curve is shown in Figure 3.

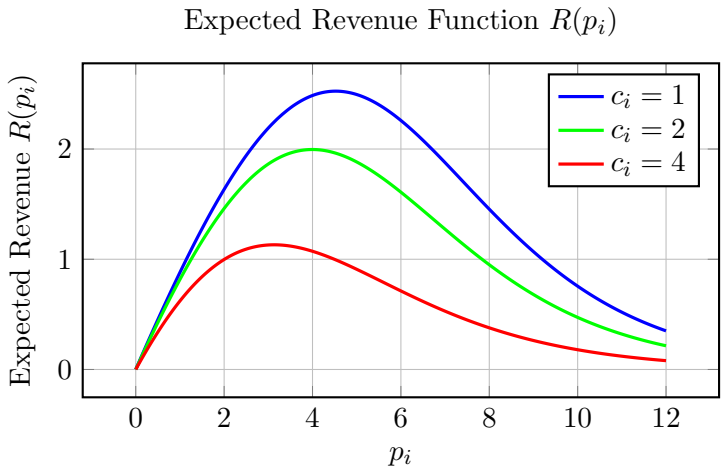

Figure 3: Effect of Computational Cost on Expected Revenue

Now with the expected revenue function, the optimal price can be found by finding the price that maximizes $R(p_i)$. Unfortunately, there is not a closed form solution to this problem, but there are methods that can be used. Iterative pricing mechanism (Saari, 1985) are used to find optimal prices by making iterative changes to the price until it converges at an optimal point. These methods work well for dynamic environment where the state of the market is continuously changing. For example, gradient ascent has been used for forex investing with algorithmic trading (Murtza et al., 2022). The gradient ascent algorithm is very similar to the gradient descent algorithm (Cauchy, 1847) commonly used in machine learning. The key difference is that it seeks to maximize a function's value rather than minimize it. Gradient ascent can be applied in the P2P-LLM network to maximize the expected revenue function, $R(p_i)$, with respect to price. It is defined as

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \eta \nabla f(\mathbf{x}_t) \tag{4}$$

where $\mathbf{x}_t$ is the parameter vector at iteration t, $\mathbf{x}_{t+1}$ is the updated parameter vector, η is the learning rate (step size), and $\nabla f(\mathbf{x}_t)$ is the gradient of the objective function f at point $\mathbf{x}_t$.

Applying this to the revenue function, $R(p_i)$, we can get an update rule for the price. The gradient of R with respect to p is

$$\frac{\partial R}{\partial p} = \mathbb{P}(w)(1 + \alpha p \cdot (\mathbb{P}(w) - 1))$$

Using this with the gradient ascent algorithm (Equation 4), the price update rule becomes

$$p_{t+1} = p_t + \eta \frac{\partial R}{\partial p_t} \tag{5}$$

As all nodes continually update their prices, they will converge on the price that gives maximum revenue for each node individually. Once this happens the network will be in a steady state where no node will want to change its price.

3.4 Network Equilibrium

When all nodes are at the state of equilibrium, each layer in the network is at a Nash-Equilibrium (NE), meaning that any deviation in price will result in a lower expected revenue. Formally this is defined as the utility of the optimal strategy being greater than or equal to any single deviation. In the P2P-LLM network, the utility is just the revenue function R. A price profile p^* for a layer is a NE when

$$u_i(p_i^*, p_{-i}^*) \geq u_i(p_i, p_{-i}^*) \forall p_i \in \mathbb{R} \tag{6}$$

This is a equilibrium for an individual layer in the network, but the whole network may not be at equilibrium. If the peers in the network are not evenly distributed, some layers may have less peers serving them, which will lead to inefficiencies. Ideally the computational power in the network will be evenly split between all layers in the LLM. In an unbalanced state, this will lead to some nodes benefiting by changing which layers of the LLM they serve.

The optimal layer for a given node is the one in which they will have the largest share of computation. The percentage of computation a node has can be found by normalizing the computational cost across a layer. A reciprocal norm is used to find the share of computational power in the layer. It is defined as

$$s_l(c_i) = \frac{c_i^{-1}}{\sum_{j=i}^{n_l} c_j^{-1}} \tag{7}$$

where l is the layer, n_l is the number of nodes serving the layer l, and c_i is the computational cost. The optimal layer for a node, is the one that maximizes this value.

$$l^* = \max_l \quad s_l(c_i) \tag{8}$$

In the P2P-LLM network, peers will periodically change layers if they are not already in the optimal state. Changing layers will increase the peers expected revenue while also increasing the throughput of the network overall. In this case the incentives of the individual nodes are aligned with the whole network overall. When no peers have a useful change then the network is in an efficient state.

4 Cryptoeconomics

The P2P-LLM network requires a secure, transparent, and decentralized mechanism to facilitate transactions between peers. The blockchain provides a secure and decentralized way to allow peers to form contracts with eachother and exchange payment in cryto-currencies

for computation. Cryptoeconomics has proven to be successful in the past with some application being in resources trading through the blockchain. Specifically, research has been done to enable peer-to-peer surplus energy trading on the blockchain (Wongthongtham et al., 2021). For the P2P-LLM network, the blockchain serves three primary functions: (1) recording the state of the network including available nodes and their capabilities, (2) facilitating secure payments between peers, and (3) executing smart contracts that govern computational resource sharing.

4.1 Blockchain

Blockchain technology provides an ideal foundation for this system, as it offers immutability, transparency, and security without relying on a central authority.

The blockchain implementation for P2P-LLM employs a Proof-of-Stake (PoS) consensus mechanism rather than the energy-intensive Proof-of-Work (PoW) used in systems like Bitcoin (Nakamoto, 2009). This design choice aligns with the goal of resource efficiency, as PoS validators are selected based on the quantity of tokens they have staked, consuming significantly less energy than PoW mining operations (Saleh, 2020). This approach is particularly appropriate for a system focused on computational resource optimization.

Each block in the P2P-LLM blockchain contains transactions representing computational resource exchanges between peers. These transactions include metadata such as the identity of the participating nodes, the layers being computed, the computational cost, and the agreed-upon price. Additionally, the blockchain maintains a distributed registry of all active nodes and their capabilities (ℓ_i, c_i, p_i), which is critical for the path-finding algorithm described in Section 3.1.

To ensure scalability, the P2P-LLM blockchain implements a sharding mechanism that partitions the network state by model layers. This design allows transactions related to specific model layers to be processed in parallel, significantly increasing throughput as the network grows. Each shard maintains its own state and processes transactions independently, with periodic cross-shard communication to maintain overall network consistency (Yu et al., 2020).

4.2 Contracts

Smart contracts form the backbone of the interaction mechanism in the P2P-LLM network. These self-executing contracts with the terms of agreement directly written into code automate the negotiation, execution, and settlement processes between peers without requiring trust or intermediaries.

In the P2P-LLM network, three types of smart contracts are employed:

1. **Registration Contracts**: These contracts handle the onboarding of new nodes to the network. When a node joins, it executes a registration contract that records its capabilities (ℓ_i, c_i, p_i) on the blockchain. This contract also requires the node to stake a minimum amount of tokens as collateral, which can be forfeited if the node behaves maliciously or fails to deliver promised computational resources.

2. **Service Level Agreements (SLAs)**: These contracts formalize the terms of computational resource sharing between peers. An SLA contract specifies the model layers

to be computed, the expected computational performance (in terms of latency and throughput), the price per computation, and penalties for non-compliance. The contract is automatically executed when a peer requests computation, with payment held in escrow until the computation is successfully completed and verified.

3. **Payment Channels**: To minimize transaction costs and latency, the P2P-LLM network implements payment channels similar to Bitcoin's Lightning Network (Poon and Dryja, 2016). These channels allow peers to conduct multiple transactions off-chain, only settling the final balance on the blockchain when the channel is closed. This approach is particularly valuable for high-frequency, low-value transactions that are common in LLM inference operations.

The execution of these contracts is governed by a verification mechanism that ensures computational integrity. When a node performs computation for another peer, the input, output, and intermediate states are cryptographically hashed and recorded on the blockchain. This creates a verifiable proof of computation that can be used to resolve disputes and ensure that nodes are correctly performing their assigned tasks.

To optimize contract execution in the dynamic environment of the P2P-LLM network, an adaptive pricing mechanism based on the concepts described in Section 3.3 is used. This mechanism allows for continuous adjustment of prices based on network conditions, ensuring that nodes are appropriately incentivized to provide computational resources where they are most needed.

The combination of these contract mechanisms creates a self-regulating marketplace for computational resources that aligns individual incentives with the overall efficiency of the network. Nodes are rewarded for providing reliable, cost-effective computation, while clients can access LLM capabilities at competitive prices without relying on centralized service providers.

5 Experiment Setup

To evaluate the performance, stability, and efficiency of the propose P2P-LLM protocol, I implemented a simulated environment to model the dynamics of the network. The simulation explores the relationship between computational resource allocation, price optimization, and network equilibrium under various conditions without requiring the deployment of an actual LLM across distributed hardware.

5.1 Simulation Framework

The simulation framework implements a discrete event-based model of the P2P-LLM network, where each event corresponds to a peer action such as joining the network, processing requests, updating prices, or switching layers. The simulation progresses in timesteps, with each timestep representing a fixed duration (e.g., 1 second in simulated time). At each timestep, the following operations are performed:

1. Process pending inference requests through the network

2. Update node prices according to the gradient ascent algorithm (Equation 5)

3. Periodically, reallocate nodes to new layers (Equation 8)

4. Create new contracts for peers without a pending contract

The simulation parameters can be configured to model various network sizes, LLM architectures, and peer behaviors, allowing for comprehensive sensitivity analysis.

5.2 Network Configuration

For the simulation, I chose to create a small network, with $n = 28$ total nodes, and a simulated LLM with $m = 4$ layers. The parameters of the nodes in the network were randomly initialized where $c_i \in (1, 50)$, $l_i \in (1, 20)$. All prices are initially set at 1. The initial state of the network is not optimal with the peers being distributed between the 4 layers in the configuration $[10, 4, 10, 4]$. For simplicity, all nodes have an α value in Equation 1 of 0.5 and the η used in the iterative pricing function (Equation 5) is 1.

6 Results

Table 1: P2P-LLM Network Metrics at Different Iteration Steps (Model with 4 Layers, 28 Peers)

Iteration	Metric	Layer 1	Layer 2	Layer 3	Layer 4
0	Nodes	10	4	10	4
	Avg. Price	1	1	1	1
100	Nodes	7	6	10	5
	Avg. Price	1.92	2.20	1.80	2.31
500	Nodes	7	6	8	7
	Avg. Price	2.33	2.40	2.29	2.33
1000	Nodes	7	7	7	7
	Avg. Price	2.33	2.34	2.33	2.33

When running the simulated environment, the behavior of the peers aligns with what is expected based on the theory described above. The starting state of the network has uniform pricing across all nodes, and there is a high concentration of peers in the first and third layers of the LLM. As the network progresses toward a state of equilibrium, the prices are expected to rise to meet the demand of the network, and reach a maximum where the expected revenue for the respective node is highest. Nodes are also expected to change the layer of the LLM they serve based on the location of other nodes in the network. Ideally, the nodes will be evenly distributed between layers in the network. Looking at Table 1, the expected actions occurs as the iteration of the simulation increases. It eventually reached a point of equilibrium where the prices and layers are constant.

Appendix A. Topological Sort

Algorithm 1 Topological Sort of a Directed Acyclic Graph (DAG)

Require: A directed acyclic graph $G = (V, E)$

Ensure: A linear ordering of vertices such that for every directed edge (u, v), vertex u comes before vertex v

result[] ▷ An empty list to store the sorted vertices
visited[v] →false for all $v \in V$ ▷ A map to track visited vertices
temp[v] → false for all$v \in V$ ▷ A map to track vertices in the current recursion stack
for each vertex $v \in V$ **do**
    if visited[v] = false **then**
        DFS-Visit(G, v, visited, temp, result)
    end if
end for
return Reverse(result)
function DFS-Visit(G, u, visited, temp, result)
    temp[u] ← true ▷ Mark current vertex as being processed
    for each vertex v such that $(u, v) \in E$ **do**
        DFS-Visit(G, v, visited, temp, result)
    end for
    temp[u] ← false ▷ Mark u as processed
    visited[u] ← true ▷ Mark u as visited
    Append u to result
end function

Appendix B. Shortest Path

Algorithm 2 Shortest Path in a DAG

Require: Graph $G = (V, E)$ with edge weights w, source vertex s, target vertex t, topological ordering $topo[]$ of G

Ensure: Length of shortest path from s to t and the path itself

Initialize $dist[v] \leftarrow \infty$ for all $v \in V$
Initialize $prev[v] \leftarrow$ nil for all $v \in V$
$dist[s] \leftarrow 0$
for each vertex u in topological order $topo[]$ **do**
    for each outgoing edge $(u, v) \in E$ **do**
        if $dist[v] > dist[u] + w(u, v)$ **then**
            $dist[v] \leftarrow dist[u] + w(u, v)$
            $prev[v] \leftarrow u$
        end if
    end for
end for
$path \leftarrow \emptyset$ ▷ Reconstruct the shortest path
$curr \leftarrow t$
while $curr \neq$ nil **do**
    Prepend $curr$ to $path$
    $curr \leftarrow prev[curr]$
end while
return $dist[t]$, $path$

References

David P. Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, and Dan Werthimer. Seti@home: an experiment in public-resource computing. *Commun. ACM*, 45(11): 56–61, November 2002. ISSN 0001-0782. doi: 10.1145/581571.581573. URL `https://doi.org/10.1145/581571.581573`.

Ludwig Boltzmann. Studien über das gleichgewicht der lebendigen kraft zwischen bewegten materiellen punkten. *Wiener Berichte*, 58:517–560, 1868. [Studies on the balance of living force between moving material points].

Alexander Borzunov, Dmitry Baranchuk, Tim Dettmers, Max Ryabinin, Younes Belkada, Artem Chumachenko, Pavel Samygin, and Colin Raffel. Petals: Collaborative inference and fine-tuning of large models, 2023. URL `https://arxiv.org/abs/2209.01188`.

Augustin-Louis Cauchy. Méthode générale pour la résolution des systèmes d'équations simultanées. *Comptes Rendus de l'Académie des Sciences*, 25:536–538, 1847. Early description of the gradient descent method.

Arnaud Durand, Mikael Gasparyan, Thomas Rouvinez, Imad Rafic Aad, Torsten Braun, and Tuan Anh Trinh. Bitworker, a decentralized distributed computing system based on

bittorrent, May 2015. URL https://boris-portal.unibe.ch/handle/20.500.12422/134610.

Iqbal Murtza, Ayesha Saadia, Rabia Basri, Azhar Imran, Abdullah Almuhaimeed, and Abdulkareem Alzahrani. Forex investment optimization using instantaneous stochastic gradient ascent—formulation of an adaptive machine learning approach. *Sustainability*, 14(22), 2022. ISSN 2071-1050. doi: 10.3390/su142215328. URL https://www.mdpi.com/2071-1050/14/22/15328.

Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. May 2009. URL http://www.bitcoin.org/bitcoin.pdf.

P. Ngatchou, A. Zarei, and A. El-Sharkawi. Pareto multi objective optimization. In *Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems*, pages 84–91, 2005. doi: 10.1109/ISAP.2005.1599245.

Joseph Poon and Thaddeus Dryja. The bitcoin lightning network, 2016. URL https://lightning.network/lightning-network-paper.pdf. Accessed: 2016-07-07.

Donald G. Saari. Iterative price mechanisms. *Econometrica*, 53(5):1117–1131, 1985. ISSN 00129682, 14680262. URL http://www.jstor.org/stable/1911014.

Fahad Saleh. Blockchain without waste: Proof-of-stake. *The Review of Financial Studies*, 34(3):1156–1190, 07 2020. ISSN 0893-9454. doi: 10.1093/rfs/hhaa075. URL https://doi.org/10.1093/rfs/hhaa075.

B. Steltner, M. A. Papa, H.-B. Eggenstein, B. Allen, V. Dergachev, R. Prix, B. Machenschalk, S. Walsh, S. J. Zhu, O. Behnke, and S. Kwang. Einstein@home all-sky search for continuous gravitational waves in ligo o2 public data. *The Astrophysical Journal*, 909(1):79, March 2021. ISSN 1538-4357. doi: 10.3847/1538-4357/abc7c9. URL http://dx.doi.org/10.3847/1538-4357/abc7c9.

Pornpit Wongthongtham, Daniel Marrable, Bilal Abu-Salih, Xin Liu, and Greg Morrison. Blockchain-enabled peer-to-peer energy trading. *Computers & Electrical Engineering*, 94: 107299, 2021. ISSN 0045-7906. doi: https://doi.org/10.1016/j.compeleceng.2021.107299. URL https://www.sciencedirect.com/science/article/pii/S0045790621002780.

Guangsheng Yu, Xu Wang, Kan Yu, Wei Ni, J. Andrew Zhang, and Ren Ping Liu. Survey: Sharding in blockchains. *IEEE Access*, 8:14155–14181, 2020. doi: 10.1109/ACCESS.2020.2965147.