A Study on 'A Compact Routing Scheme and Approximate Distance Oracle for Power-Law Graphs': Summary and Experimental Insights

Ohad Heines
Ori Katzir

1 Introduction

In the realm of network communications, efficient routing and distance querying are fundamental challenges. In the article "A Compact Routing Scheme and Approximate Distance Oracle for Power-Law Graphs" by Wei Chen, Christian Sommer, Shang-Hua Teng, and Yajun Wang (abbreviated CSTW from this point onward), an innovative approach is presented to address these challenges. CSTW introduce an adapted version of Thorup and Zwick's universal compact routing scheme[11], specifically optimized for unweighted, undirected power-law graphs, optimizing for both theoretical efficiency and practical applicability. This adaptation includes selecting high-degree nodes as landmarks and encoding shortest paths directly in node labels and message headers, rather than relying on a tree routing scheme. By leveraging the unique structural properties of these graphs, the proposed scheme achieves significant improvements in routing table sizes and preprocessing times, while maintaining low stretch factors for path lengths.

CSTW claim to provide the first theoretical analysis that directly links the power-law exponent τ of a random power-law graph to the bounds on the routing table sizes. Additionally, they extend their techniques to optimize an approximate distance oracle for power-law graphs[11, 12], achieving similar improvements in space and preprocessing requirements.

The theoretical analysis demonstrates that the new scheme significantly reduces routing table sizes and preprocessing times compared to the general case. For instance, it achieves routing tables of size $O\left(n^{\gamma} \log n\right)$, with $\gamma = \frac{\tau-2}{2\tau-3} + \epsilon$, for $\epsilon > 0$ and τ, and an expected construction time of $O\left(n^{1+\gamma} \log n\right)$. The routing scheme requires a stretch-5 handshaking step, similar to a DNS lookup in TCP/IP, and achieves a stretch-3 for the actual routing paths. Additionally, the addresses and message headers in this scheme are optimized to use $O\left(\log n \log \log n\right)$ bits, ensuring efficient communication overhead.

To verify CSTW's results and claims, we replicated their experiment and analyzed the data compared to theirs. Mode detailed discussions on these replications and our additional findings will be provided later in this report.

2 Terms and Preliminaries

2.1 Power Law

Power law is a statistical term describing a functional relationship between two quantities, where a relative change in one quantity results in a change in the other quantity proportional to a power of the change, independent of the initial size of those quantities. This relationship is often expressed as $P\left(x\right) = Cx^{-\tau}$, where τ is a positive constant known as the exponent or scaling parameter, x is a variable of interest, and C is a normalizing constant.

Power law distributions, also known as heavy-tail distributions, Pareto distributions, or Zipfian distributions, are characterized by their scale-free nature, meaning that changing the scale does not alter the shape of the distribution plot. Another key property of power law distributions is their long tail, indicating a non-negligible probability of observing very large values, even though most of the samples are small.

2.2 Power Law Graphs

In power-law graphs, also termed scale-free networks or graphs, the number of nodes with degree x is proportional to $x^{-\tau}$ for some constant τ typically between 2 and 3. This means that the degrees of the nodes in these graphs follow a power law distribution.

Power-law graphs are sparse due to their degree distribution, where most nodes have few connections. When τ is between 2 and 3, the power-law distribution has a finite mean but an infinite variance[2]. This balance makes it suitable for modelling systems where the average value is well-defined, but there are occasional extreme events that lead to high variability. The distribution's tail is heavy but not excessively so, ensuring that while there are extreme values, they do not dominate the overall behaviour of the system. Such distributions strike a balance between having a significant number of high-degree nodes (hubs) and a majority of low-degree nodes, a common feature in many networks.

Despite their sparsity and unique features, power-law graphs pose challenges for algorithm design. Problems such as colouring or finding cliques remain NP-Hard for power-law graphs[4], highlighting the complexity and difficulty in tailoring algorithms for these types of networks.

2.3 Compact Routing Schemes

A routing scheme is a systematic method used to determine the path that data packets should take from a source node to a destination node within a network. It involves algorithms and protocols that govern what information is kept at each node, as well as how routing decisions are made dynamically as packets traverse the network.

A compact routing scheme is an advanced type of routing strategy designed to efficiently route data packets in a network while minimizing the storage requirements for routing information at each node. The main goal is to find a balance between the space required to store routing information and the efficiency of the routes taken by the messages. The increase in the path length taken by packets, known as the stretch, is a key consideration. Stretch is a metric that quantifies the efficiency of the routing paths used compared to the shortest possible paths. Specifically, the stretch factor is defined as the ratio of the length of the path taken by a routing scheme to the length of the shortest possible path between the same source and destination nodes.

In the context of the article, we are introduced to a specific kind of compact routing scheme called a "labelled routing scheme". These schemes allow nodes to have additional labels that encode useful information for routing, achieving lower stretch factors by using this extra information.

2.4 Bound for General Compact Routing Schemes

Labelled compact routing schemes have been studied extensively. Universal schemes work for all network topologies[6, 7, 8]. On graphs with n nodes and for parameter k, it has been shown that with $\tilde{O}\left(n^{\frac{1}{k}}\right)$-bit routing tables, one can achieve a stretch of $O\left(k\right)$, and that this trade-off is essentially tight due to a girth conjecture by Erdős. The conjecture itself is about the girth (the length of the shortest cycle) of a graph with a given number of nodes and edges. Specifically, it posits that for any given average degree, there exists a graph with a girth that is logarithmic in the number of nodes.

In dense graphs with a large number of vertices and edges, the Erdős girth conjecture implies the existence of graphs with a large girth, meaning that the shortest cycle in such graphs is relatively long. In the worst-case scenario, an approximated shortest path between two nodes could become excessively long if one of the nodes is on a cycle and the other is not. This is because the path might have to traverse almost the entire cycle to reach the destination, bypassing all but one vertex of the cycle. Consequently, this results in at least one node having to store an extremely large routing table to accommodate these lengthy paths.

In sparse graphs, like power-law graphs, this conjecture sets a theoretical bound that is not tight because of their unique properties. Specialized schemes can take advantage of these properties to optimize routing tables and minimize stretch more effectively than general schemes.

2.5 Random Power-Law Graph Model

CSTW adapted the random graph model for fixed expected degree sequences as defined by Aiello, Chung, and Lu [Aiello et al. 2000; Chung and Lu 2002, 2006; Lu 2002b] using the following definition from Chung and Lu [2002, Sect. 2]:

For a constant $\tau \in (2, 3)$, the random power-law graph distribution RPLG (n, τ) is defined as follows, Let the sequence of generating parameters $\overrightarrow{w} = \{w_1, w_2, \ldots, w_n\}$ obey a power law:

$$w_i = \left(\frac{n}{i}\right)^{\frac{1}{\tau-1}}$$

The edge between two vertices v_i and $v_{i'}$ is present in the random graph with probability:

$$\Pr\left[\{v_i, v_{i'}\} \in E\right] = \min\left\{w_i \cdot w_{i'} \cdot \rho, 1\right\}, \text{ where } \rho = \frac{1}{\sum_j w_j}$$

This definition ensures that the edges in the graph are independent. This model allows CSTW to implicitly rely in their profs on a property called assortativity, which is the tendency of high-degree nodes to be connected to each other. Formally, the assortativity of a graph is defined as[9]:

$$s\left(G\right) := \sum_{\{v_i, v_{i'}\} \in E} \deg\left(v_i\right) \cdot \deg\left(v_{i'}\right)$$

Graphs sampled from the FDRG model tend to have a high $s(G)$ value, since high-degree nodes are likely to be attached to other highly connected nodes. This property measures the extent to which a graph has a "hub-like core,"[9] where the core consists of nodes with large degrees.

CSTW defined the core with the assumption that the number of vertices in the graph is sufficiently large, specifically, for some $\epsilon > 0$:

$$n^{\frac{\epsilon(2\tau-3)}{\tau-1}} \geq \frac{2(\tau-1)}{\tau-2} \ln n$$

Let $\gamma = \frac{\tau-2}{2\tau-3} + \epsilon$ and $\gamma' = \frac{1-\gamma}{\tau-1}$, the core is defined as follows:

For a power-law degree sequence $\vec{w}$ and a graph G with n nodes, the core with degree threshold $n^{\gamma'}$, $\gamma' \in (0,1)$ is[10]:

$$\mathrm{Core}_{\gamma'}\left(\vec{w}\right) := \left\{v_i | w_i > n^{\gamma'}\right\}, \mathrm{Core}_{\gamma'}(G) := \left\{v_i | \deg_G(v_i) > \frac{n^{\gamma'}}{4}\right\}$$

In the article, CSTW used $\mathrm{Core}_{\gamma'}(G)$ as the landmark set for the routing scheme. An important aspect of the core in the routing scheme is the specific threshold that determines the minimum degree of a node needed for it to qualify as a member of the core.

Another important definition used by CSTW is the ball of a vertex $u \in V$, which is the set of all vertices whose distance from u is shorter than the distance from u to the closest landmark:

$$\forall_{u \in V_G} : B(u) := \{v \in V_G | d(u,v) < d(u, \ell(u))\}$$

Where $\ell(v)$ denotes the landmark that is the closest to node u (ties are resolved arbitrarily):

$$\forall_{u \in V} : \ell(u) := \underset{v \in L}{\mathrm{argmin}} d(u,v), \ L = \text{set of landmarks}$$

In the preposed CSTW compact routing scheme of the referred the vertices included in the core as landmarks $L := \mathrm{Core}_{\gamma'}(G)$.

3 Description of the Compact Routing Scheme

3.1 Preprocessing

To describe the proposed compact routing scheme, we first need to understand the network to which it applies. The network is represented as an unweighted and undirected graph. It is static, meaning that we do not handle the addition or removal of nodes. Each vertex in the power-law graph is considered a "computer" with a unique $\lceil \log_2 n \rceil$-bit static name that identifies it. Every vertex $v \in V$ has $\deg(v)$ ports connecting it to its neighbours, numbered from 1 to $\deg(v)$. For every packet, the routing scheme needs to decide which port the packet is to be forwarded to, and each node maintains a routing table with the necessary information.

The proposed routing scheme is a labelled, fixed-port scheme, meaning that it works for any permutation of port number assignments on any node. The labels serve as the addresses of the nodes within the network.

The general idea of this scheme is to use a subset of vertices as landmarks and the ball of every node in the network to create local data that will help each node route messages efficiently. When a message is routed to a distant target node, it is first sent to the closest landmark of the target and then from the landmark to the target. This scheme adapts the Thorup and Zwick and the Cowen compact routing framework[11, 3] by considering the high variance in node degrees in power-law graphs and selecting high-degree nodes as landmarks instead of random sampling.

The use of nodes utilises the “hub-like” behaviour of the graph’s core and its assortativity property, setting high-degree, well-connected vertices as intermediaries. Using landmarks significantly reduces the size of the routing tables. The use of landmarks helps define the ball of each node, where every vertex in a node’s ball is considered a local target of that node.

The labelled nature of the scheme allows encoding shortest paths directly in vertex labels and message headers, rather than relying on a tree routing scheme like Thorup and Zwick’s. This enables each node $v \in V$ to encode the shortest path from $\ell(v)$ (the closest landmark to v) to v itself. This encoding consists of the sequence of port numbers on the path, helping nodes pass messages to non-local targets by routing them through the closest landmark to the target node. One could think of a port as a “step” in a shortest path from the current node to the closest landmark to the target node the message is intended for. The encoding of shortest paths is a recurring element in the proposed scheme and is denoted as follows: $\mathrm{SP}(s,t)$.

The labels, in fact, serve as addresses of the nodes inside the network. The address of every node $v \in V$ in the network includes the static identifying name of the node itself, the static identifying name of $\ell(v)$, and the encoding of a shortest path from $\ell(v)$ to v itself - $\mathrm{SP}(s,t)$.

The local routing table for each node u consists of information about shortest routes to all of the landmarks and to local targets. Formally:

$$\forall_{u \in V} : \mathrm{tbl}(u) := \{(v, \mathrm{port}_u(v)) \mid v \in \mathrm{Core}_{\gamma'}(G)\} \cup \{(v, \mathrm{port}_u(v)) \mid v \in B_G(u)\}$$

Where $\mathrm{port}_u(v)$ is the local port of u to route messages towards node v along some shortest path from u to v.

The header of a message from node s (source) to node t (target) can be in one of the following formats (the first parameter marks its type):

1. $\mathrm{Header} = (\text{"local"}, s, t)$
2. $\mathrm{Header} = (\text{"toLandmark"}, s, \mathrm{addr}(t))$, where $\mathrm{addr}(t)$ is the address of node t

3. Header $=$ ("fromLandmark"$, s, t, \mathrm{pos}, \mathrm{SP}$), where pos is a non-negative integer that may be modifier along the route, and SP is the encoding of a shortest path from $\ell(t)$ to t itself.
4. Header $=$ ("direct"$, s, t, \mathrm{pos}, \mathrm{SP}$), where pos is a non-negative integer that may be modifier along the route, and SP is the encoding of a shortest path from s to t itself.
5. Header $=$ ("handshake"$, s, t, \mathrm{SP}$), where route SP is a reversed shortest path from t to s to be encoded along the path from s to t during a handshake process.

Nodes s and t are represented in the headers by their unique identifiers. These headers help route messages efficiently along optimal paths.

3.2 The Routing Procedure

The procedure itself could be looked at as three sub procedures:

1. Sending a message
2. Forwarding a message
3. Handshake process

The process of sending a message involves determining the appropriate port for the initial routing decision based on the target node's distance from the source node. This decision depends on whether the message's target is a local target, a landmark, or a distant node.

When a node receives a message, it must decide how to forward it based on the message header and the routing table. The forwarding decision varies depending on whether the target is a local target, a landmark, or a distant node. If the message has been previously routed from the source to the target, the handshake process comes into play.

The handshake process is a coordination step akin to "meeting someone for the first time." It involves caching paths used to route messages through a node, storing this information locally, outside the routing table for efficient routing in future interactions. The handshake process ensures that paths are established and maintained for subsequent messages between nodes, optimizing routing efficiency.

3.2.1 Sending a Message (Algorithm 1 in the artice):

Local Target:

If the target node t is in the ball of the source node s, denoted as $t \in B_G(s)$, then t is considered a local target of s. This means node s has stored in its routing table the port that it uses to initiate a shortest path to node t.

Node s will pass the message through the relevant port, $\mathrm{port}_s(t)$, with the header ("local"$, s, t$). From that point onward, each vertex u, which is different

from t, that receives the message will forward it with the same header using $\text{port}_u(t)$. This algorithm is based on the observation that if the target node is inside the ball of s, then all nodes on a shortest path from s to t will also have t inside their balls. Otherwise, there would be a contradiction for $t \in B_G(s)$ as a sub-path of a shortest path is itself a shortest path.
Non-Local Target:

If node t is not inside$B_G(s)$, and if s has stored a shortest path from itself to t, $\text{SP}(s,t)$, locally after a handshake was made, then s will send the message with the header $(\text{"direct"}, s, t, \text{pos}, \text{SP})$ via the port $\text{SP}[0]$.

Otherwise, s will send the message with the header $(\text{"fromLandmark"}, s, t, \text{pos}, \text{SP})$ using the port connecting s to the closest landmark to t, that is $\text{port}_s(\ell(t))$. This covers the case where t is too far away to count as a local target, as well as the case where t is a landmark.

3.2.2 Forwarding a Message (Algorithm 2 in the article):

When a node receives a message, it first checks if it is the target of that message. If so, the routing procedure is complete. Before stopping the entire routing procedure, if the header is of type "fromLandmark" and the source node of the message is a local target of the receiving node or a landmark, a handshake process will begin (more on this in a moment). Otherwise, the forwarding process is determined by the value of the route field in the message header.
Forwarding with "toLandmark" Header:

If the message header is of type "toLandmark" the receiving node checks to see if it's the closest landmark to t by comparing its unique identifier with the identifier of $\ell(t)$ stored in the message header.

- If the receiving node is not $\ell(t)$, it simply passes the message onward via the port that connects it to $\ell(t)$, which is $\text{port}_v(\ell(t))$ where v is the receiving node. This is possible because every node stores this information in its routing table for every landmark in the scheme, and $\ell(t)$ doesn't change.

- If the receiving node is $\ell(t)$, the message will be forwarded with a new header using the path encoded in the address of node t, which is a shortest path from $\ell(t)$ to t. The new header will be $(\text{"fromLandmark"}, s, t, \text{pos} = 0, \text{SP}(\ell(t), t))$ and it will be passed using the port $\text{SP}(\ell(t), t)[0]$.

Every node that receives a message with a "fromLandmark" header will increment the pos value stored in the header by one and use the port indicated in SP(l(t),t)[pos]. This can be thought of as making one step along the path. Every node that receives a message with the header ("local", s, t) will forward the message using the port connected to t that is stored in its routing table, just as if it sent the message itself.

Lastly, there are two special cases involving handshake headers:

1. When a node receives a message with header $(\text{"handshake"}, s, t, \text{SP})$

2. When a node receives a message with header $(\text{"direct"}, s, t, \text{pos}, \text{SP})$

These cases are part of the handshaking sub-procedure, which is defined by how these two header types are handled.

3.2.3 Handshake (Algorithm 3 in the article):

The handshake sub-procedure begins when a node u receives a message and the following conditions are met:

1. The receiving node is the target of the message, that is, u is t.
2. The message has a header of type "fromLandmark".
3. The target node t is not a local target of s, that is $t \notin B_G(s)$.
4. The target node t is not a landmark, that is $t \notin \text{Core}_{\gamma'}(G)$.
5. The source node of the message, s, is either a landmark or is inside the ball of t, that is $s \in \text{Core}_{\gamma'}(G) \cup B_G(t)$.

If t is a landmark or is inside the ball of s, there wouldn't be a need to start a handshake since, as described earlier, all the information needed for routing is kept inside the routing tables of s and the other relevant nodes.

If the source node s is inside the ball of t, meaning the distance from s to $\ell(t)$ is greater than the distance from s to t itself, it would be more efficient for future messaging to save a shortest path from s to t locally (and temporarily) in s itself.

However, if s is a landmark, it might not be the closest landmark to t. According to the routing scheme, the message should first be routed to $\ell(t)$, but this could be inefficient compared to sending it directly from s to t. Since landmarks are central, well-connected nodes that play a crucial role in the routing scheme, ensuring that landmarks cache paths efficiently enhance overall network performance, as landmarks are frequently used as intermediaries in routing. Note that if s is $\ell(t)$, then there is a shortest path from s to t in the address of t, and the handshake sub-procedure is pointless.

In these cases, node t will start a handshake process by sending a message with a header of the fifth type, through the port stated in the routing table that is connected to s, with the following information: $(\text{"handshake"}, t, s, \text{NIL})$. This indicates that the message is of type "handshake", it is from node t, and it is addressed to the source node of the previous message. The last field in the header is an empty path that will be encoded in reverse during the handshaking process.

Every node that receives a message with "handshake" in its header will first prepend the port identifier to the last field in the header, that is, the port it received the message from. Then the receiving node will check if it is the target node. If it is, it will store the path stored in the header locally. Otherwise, it will send the message with the updated header through the port that its routing table indicates as connected to the target node. This information will be in the table because the target node of this handshake process is either a landmark (for which this information is stored in all nodes) or is in the ball of the handshaking process source node, as explained in the first sub-procedure.

4 Experiment Replication

4.1 The Original Experiment

CSTW conducted an experimental evaluation of the proposed compact routing scheme using several real-world graphs[5] and randomly generated graphs[1]. All graphs used in the experiments were unweighted and undirected. The scheme was tested on the largest connected components of these graphs. In this section we detail our attempt at replicating their experiment and compare the results. In addition we propose a new threshold function as opposed to $n^{\frac{\gamma'}{4}}$, and we analyze it's performance.

4.2 Replicating the Algorithm

4.2.1 The Schemes

First, we programmed the Routing Scheme detailed by CSTW in both of the versions specified - the theoretical, and the practical, and the scheme defined by Thorup and Zwick[11]. All schemes function in a similar manner, except for the way they select the nodes that are considered landmarks. All of our code can be found at [http://bit.ly/3LxtN03]

Theoretical Landmark Selection (TLS) - Is the main method in used by CSTW. In this method, landmarks are chosen to be all nodes with a degree larger than some threshold $\frac{n^{\gamma'}}{4}$ (for further details see section [2.5]). Not only is this selection method straightforward and only takes $O(n)$ time to calculate, they also proved that with high probability, for the model of power-law graphs they used, $|\text{Core}_{\gamma'}(G)| = \Theta(n^{\gamma})$ which might be considered optimal in some sense that is explained in section [4.4.2].

Practical Landmark Selection (PLS) - In this method, we fixate $|\text{Core}_{\gamma'}(G)| = \Theta(n^{\gamma})$ by picking the $\lceil n^{\gamma} \rceil$ nodes with the largest degrees in the graph. As CSTW mention, empirically, "The largest connected components of the graphs generated by Brady and Cowen[1] [...] do not contain nodes with such a high degree (relative to the threshold)", which is why they used in their experiment this method only. The reason for this discrepancy is further explored in section [4.3].

Thorup and Zwick Landmark Selection (T&Z) - This method is general, and not tailor made for power law graphs. In this method, the landmarks are selected by a random sampling process which we won't go into in detail.

For completeness sake, we compared the performance of all three methods with this scheme relative to one another.

For each graph, we calculated the tables for each node in the network, and then randomly generated pairs of nodes, and simulated the message forwarding procedure to calculate the distance streches. Finally, we generated a file consisting of the expected value and variance of the table size (in rows) and the distance stretch values (relative to the shortest path between the nodes).

4.2.2 Message Forwarding

We simulated the message forwarding procedure from s to t with an overhead look on the graph. Instead of simulating each node in a distributed manner, we kept track of the message's current position and then simulated a single step from the position it is currently in. This saves unnecesary computation, since a single message is only affected by the node currently processing it. Formally, our forwarding simulation

Algorithm 1 ForwardingSimulation(Node s, Node t)

```
$pos \leftarrow s$
$msg \leftarrow$ InitMessage($s$, $t$)
While (CurrentPosition $\neq t$):
    $msg$, $pos \leftarrow$ ProcessMessage($msg$, $pos$)
```

Where the ProcessMessage function runs a single iteration of the distributed algorithm from node *pos* on *msg*.

Notice that in all of our code we neglect handshakes, since we only simulate the first message in each conversation. Our code contains another variant of this process which assumes conversations are long and the first message is negligible relative to the amount of direct messages - this assumption is realized with a detection process for pairs that would undergo a handshaking process, and then setting the distance stretch of the message to be 1 (since most of the messages would be sent on the direct path between them).

4.3 Dataset

In our experiments, we used the power law graphs generated by Brady and Cowen[1], and generated new graphs using their power law graph generator. From each graph generated, we extracted the largest connected component, and simulated the three schemes aforementioned on it. As mentioned by CSTW, this power law graph generator, lead to different results than the theoretical model used in the their analysis. This is due to two main reasons: the graphs not satisfying the condition for large enough n ($n^{\frac{\epsilon(2\tau-3)}{\tau-1}} \geq \frac{2(\tau-1)}{\tau-2} \ln n$), and the fact that we use the parameter n as the original amount of nodes in the graph, even though we remove all of the nodes not contained in the largest connected component. The second reason, can't be easily fixed by setting n to the amount of nodes in the largest connected component, since the expected degree sequence is defined by the original n. The assumption that the graph is connected, rarely happens in graphs generated by the Brady and Cowen generator.

4.4 Results

4.4.1 Comparison with the General Thorup and Zwick Scheme

First, we'll discuss the performance of both methods tailor made for power-law graphs TLS and PLS, in comparison to the T&Z method. The results can be seen in Figures 1 and 2, where we compare performance of the selections for the pregenerated graphs with $10,000$ nodes. The blue points are for PLS, the orange points are for TLS, and the green points are for T&Z. It is important to notice how in these graphs, and all graphs that follow, for τ values smaller than 2.5 there are no orange points, as no nodes were selected to be landmarks using TLS in these graphs, so we removed these redundant cases.

It is clear from the graphs, that both TLS and PLS perform much better than T&Z, which is as expected since it is a general method while the other two are tailor-made for power law graphs. Because of this, we will be omitting the green points from here onward, since it just makes the comparison between the two other schemes in the graph less clear.

4.4.2 Comparison Between the Theoretical and Practical Landmark Selection Methods

Now, we'll further analyze the performance of the theoretical and practical methods in relation to each other. It is first important to notice that both algorithms pick the nodes with the highest degrees in the graph to be landmarks, the difference being the amount of landmarks they pick - in PLS this amount being $\lceil n^\gamma \rceil$, while in TLS this being the amount of nodes with a degree higher than some threshold. This means, that for each graph, one of the landmarks sets selected is a subset of the other. This observation lets us conclude that, at least for the distance stretches, the graph with the higher amount of landmarks will have smaller stretches, i.e. will have better performance in this regard. This can be seen in Figure 3, where we compare the distance stretches of both methods, and TLS has better performance than PLS for $\tau > 2.5$, since for $\tau > 2.5$, the amount of nodes with a degree higher than the threshold ended up being bigger than $\lceil n^\gamma \rceil$. In other words, assuming the goal of TLS was to approximate PLS, the threshold was *underestimating* the degrees of nodes in the graph. This is further explored in section [4.4.5].

Again, it is important to note that for $\tau < 2.5$, we omitted the graphs, since there were no landmarks selected using TLS. In other words, in contrast to $\tau > 2.5$, in the case that $\tau < 2.5$ the threshold was *overestimating* the degrees of nodes in the graph.

Although it is quite clear which landmark selection method results in better distance stretches, this comes at the cost of the table sizes. We'll analyze those next.

As earlier, for $\tau = 2.5$ both methods 'converge', because the landmark selection is almost exactly the same. For $\tau > 2.5$, the theoretical algorith consistently picks more landmarks than the practical algorithm, which leads to bigger expected table sizes, but smaller variance. This is because every node saves a single line for each landmark, so the baseline for the size of each table rises, but the distance to the closest landmark decreases significantly, which leads to much smaller balls, inducing smaller additions to the baseline size of each table. Intuitively, everyone has to keep more landmarks, but since everyone is close to a landmark they have much smaller local nodes to save in their table. This can be seen in Figure 4

Interestingly enough, on the graph comparing expected value with variance, the table size of the practical algorithm seems to stay on the diagonal, regardless of the value of τ. This alludes that the practical algorithm is 'optimal' in some sense, since it minimizes the amount of landmarks as much as possible without leading to nodes that are extremely 'loaded' with huge tables.

4.4.3 Comparing Our Data with the Original Paper's Data

In Figure 5, we plotted a graph of the relation between the expected table size and the expected stretch value for each value of τ. In it, we compared our results for the practical algorithm (orange points), with their results (blue points). Our results seem to have consistently better performances, but not by a lot.

In addition, one might notice that both sizes $n = 10,000$ and $n = 100,000$ the points seem to follow a paraboloid, with it's extremum being around $\tau = 2.5$. While this is not exact for small values of n (i.e. for $n = 10,000$ the shapes are more chaotic with $\tau = 2.9$ being an outlier and 2.3 being a point of extremum in their data while 2.4 is the extremum in our data), for large n this trend is much smoother.

4.4.4 Larger Graphs

Up to this point, we only showed data concerning the pregenerated graphs with 10,000 nodes since we tried to replicate the original experiment done by CSTW. Still, we ran the same experiments on larger graphs with 100,000 nodes that we generated using Brady and Cowen power-law graph generator[1] to see if they led to similar results. We'll omit most of these graphs since they have similar distributions and implications, though all of the graphs can be found at [http://bit.ly/3LxtN03].

4.4.5 Better Threshold Approximation of the Practical Algorithm

After noticing that the threshold proposed by CSTW in TLS seemed to not represent the ideal core sizes (and after seeing CSTW also pivot to a different algorithm - i.e. PLS - because of the poor results), we tried to find a better threshold function statistically. For this, we ran the practical algorithm on graphs of various sizes and τ values, and recorded the smallest degree from the landmark set as the ideal threshold (every node with a higher degree is in the core, and every node with a lower degree is outside of the core, if we ignore nodes with the same degrees since this is highly unlikely for the high degrees). As can be seen in Figure 6, if we look at a specific n and plot the optimal approximation threshold as a function of τ, on a **log scale**, we can visually see a straight slope with indicates an exponential relationship!

From this observation, we can conclude that a function for the threshold better approximating the sizes of the practical algorithm, should be of the form

$$\text{Threshold}\left(n, \tau\right) = f\left(n\right) \cdot e^{g(n) \cdot \tau}$$

where $f\left(n\right)$ and $g\left(n\right)$ are some functions that are constant for different values of τ.

To find $f\left(n\right)$ and $g\left(n\right)$, we let excel generate the best interpolation of the data, and then plotting graphs of the coefficients and the exponents based on n. This can be seen in Figure 7. As seen in the graphs, we again interpolated the points, and found the approximations

$$f\left(n\right) = 0.294n + 2586.3$$

$$g\left(n\right) = -0.118 \ln\left(n\right) + -1.054$$

which in turn, after substitution and some simplification, leads to the following threshold function

$$\text{Threshold}\left(n, \tau\right) = \left(0.294n + 2586.3\right) \cdot \left(\frac{1}{e^{1.05} \cdot n^{0.118}}\right)^{\tau}$$

Finally, in Figures 8 and 9 we compare the performance of landmark selection based on our proposed threshold function, relative to the performance of both PLS abd TLS. It can be seen that for graphs with $10,000$ nodes our proposal almost converges with PLS meaning our threshold estimates it pretty well while still being relatively easy to calculate (since to calculate the core we don't need to sort all of the nodes based on their degrees in $O\left(n \log n\right)$ time). On the other hand, for large graphs with $100,000$ nodes, while our proposal still generates better results than TLS, it is far from optimal. We even have the same problem that for $\tau < 2.3$ no nodes were selected as landmarks because our function *overestimated* the degrees of nodes in the graph. Overall, this means our function is a good estimation for PLS in small to medium sized graphs, but it doesn't scale well for large graphs (with $100,000$ or more nodes). Further research is needed to find a function that scales better.

References

[1] A. Brady and L. Cowen. Compact routing on power law graphs with additive stretch. *Proceedings of the 9th Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 119–128, 2006.

[2] A. Clauset, C. Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Rev. 51*, pages 4, 661–703, 2009.

[3] L. Cowen. Compact routing with minimum stretch. *J. Algor. 38*, pages 1, 170–183, 2001.

[4] A. Ferrante, G. Pandurangan, and K. Park. On the hardness of optimization in power-law graphs. *Theoret. Comput. Sci. 393*, pages 1–3,220–230, 2008.

[5] CAIDA Cooperative Association for Internet Data Analysis. Caida's router-level topology measurements. *https://www.caida.org/tools/measurement/skitter/router$_t$opology/*, 2003.

[6] Abraham I., C. Gavoille, and D. Malkhi. On space-stretch trade-offs: Lower bounds. *Proceedings of the 18th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 207–216, 2006.

[7] Abraham I., C. Gavoille, and D. Malkhi. On space-stretch trade-offs: Upper bounds. *Proceedings of the 18th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 217–224, 2006.

[8] Abraham I., C. Gavoille, D. Malkhi, N. Nisan, and M. Thorup. Compact name-independent routing with minimum stretch. *ACM Trans. Algor. 4, 3*, 2008.

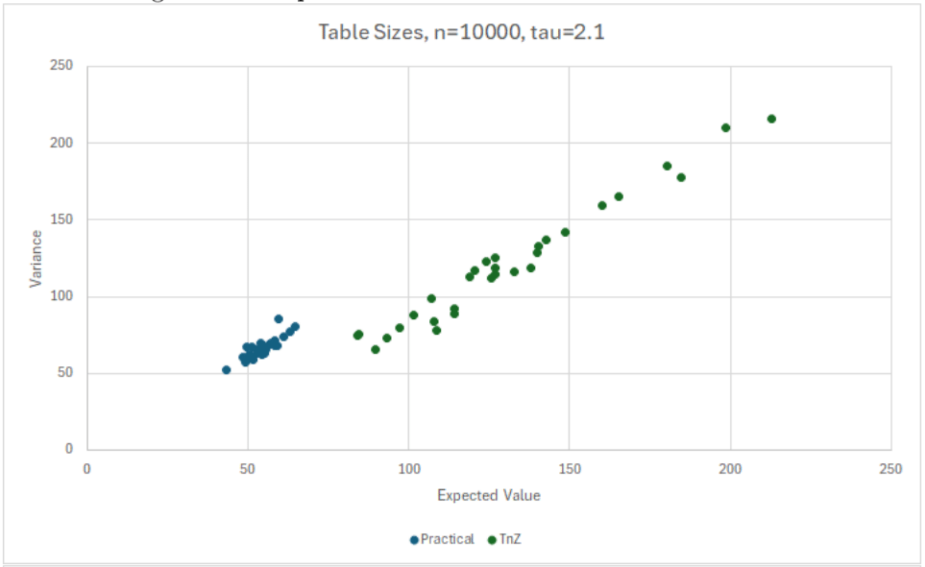
[9] L. Li, D. Alderson, J. C. Doyle, and W. Willinger. Towards a theory of scale-free graphs: Definitions, properties, and implications. *Internet Math*, pages 2, 4, 431–523, 2005.

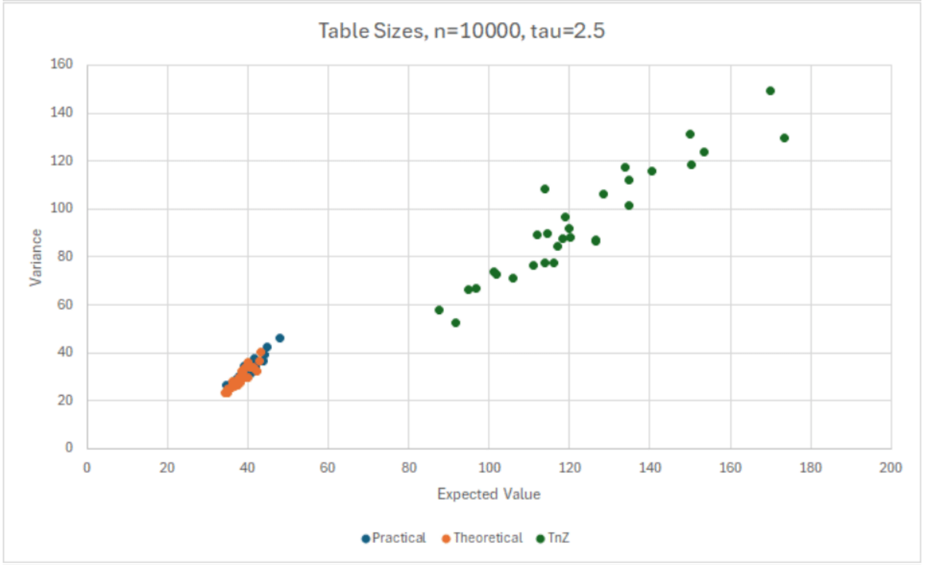
[10] L. L. Lu. Probabalistic methods in massive graphs and internet computing. *Ph.D. Dissertation. University of California San Diego*, 2002.

[11] M. Thorup and Zwick U. Compact routing schemes. *Proceedings of the 13th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 1–10, 2001.

[12] M. Thorup and Zwick U. Approximate distance oracles. *J. ACM 52*, pages 1–24, 2005.

Figure 1: Comparison of Table Sizes for all three schemes





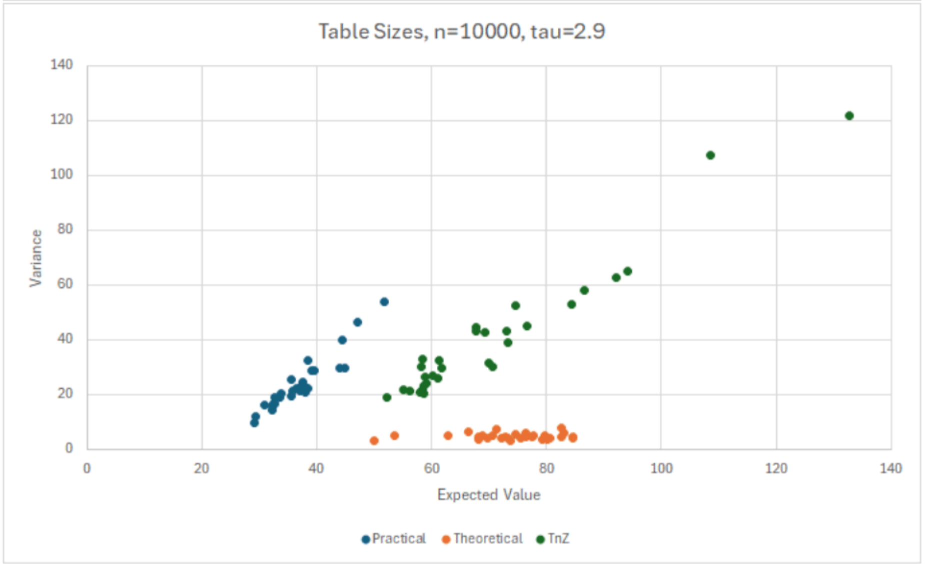


Figure 2: Comparison of Distance Stretches for all three schemes

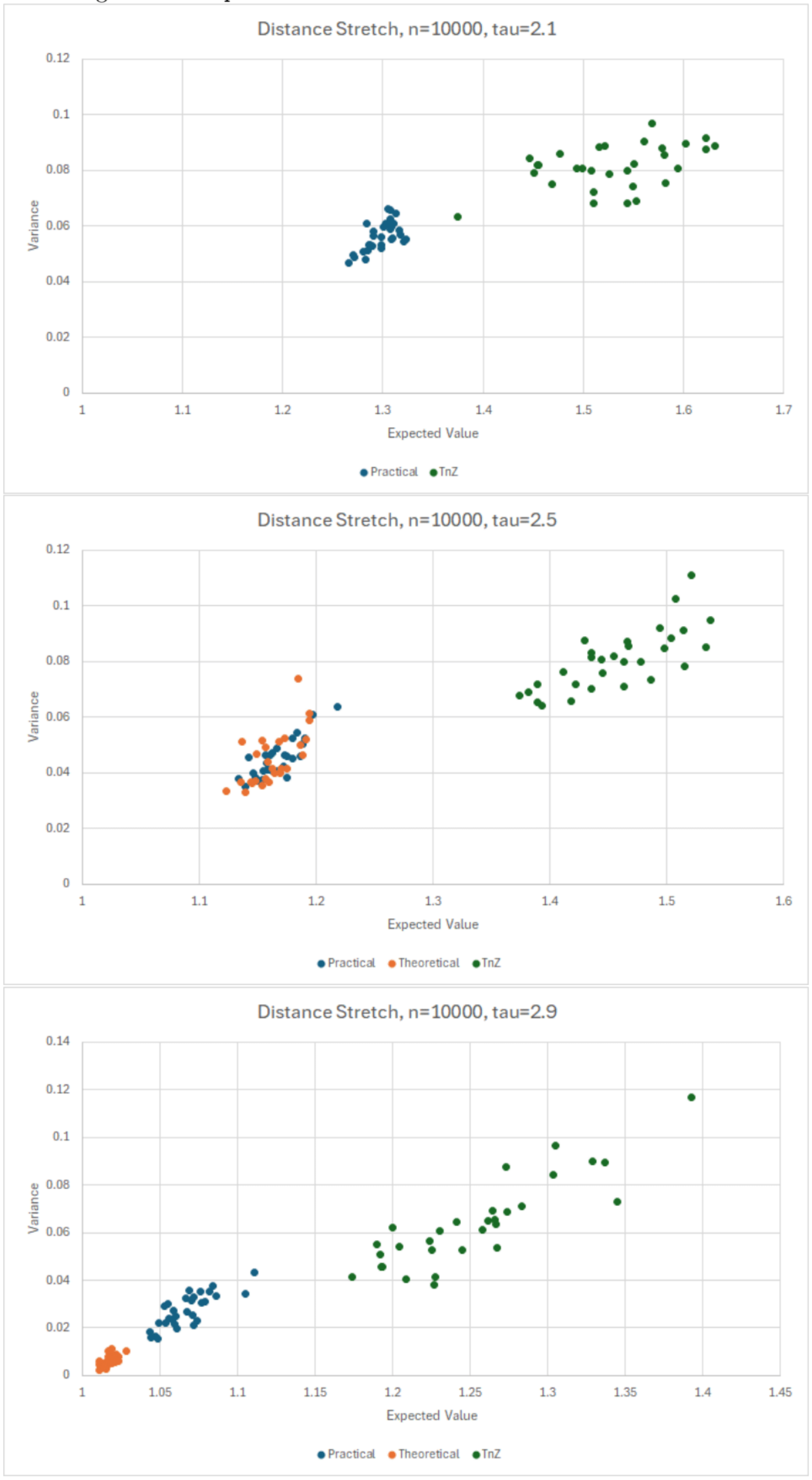
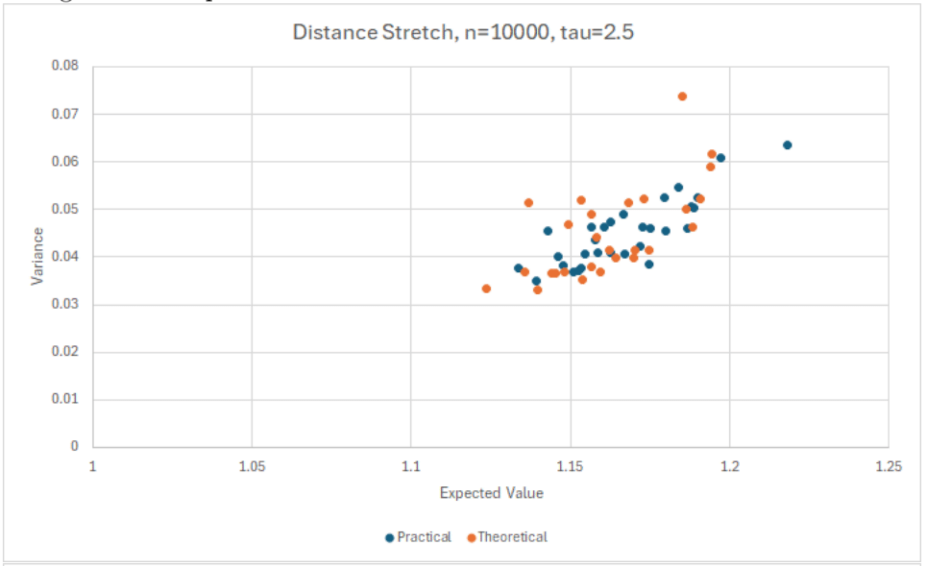
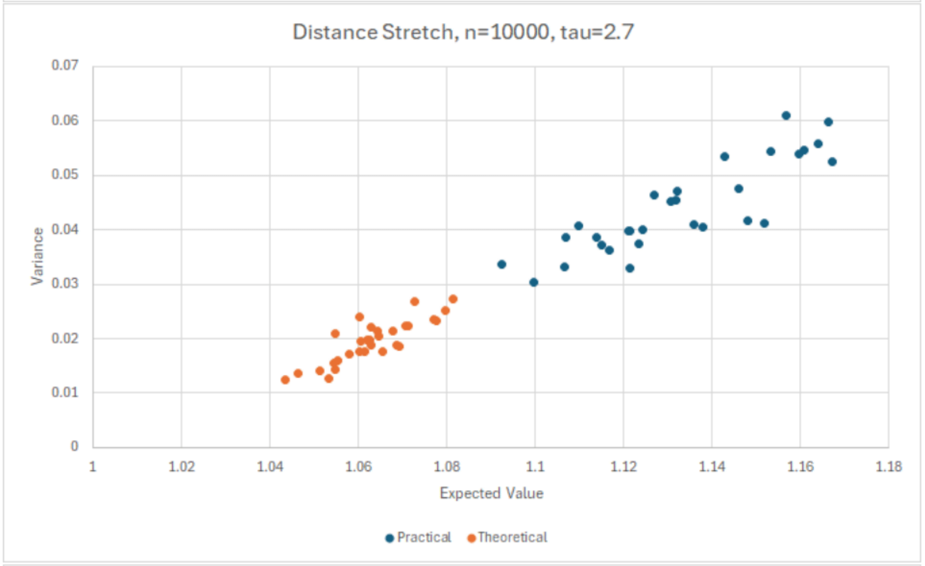


Figure 3: Comparison of Distance Stretches for Theoretical and Practical





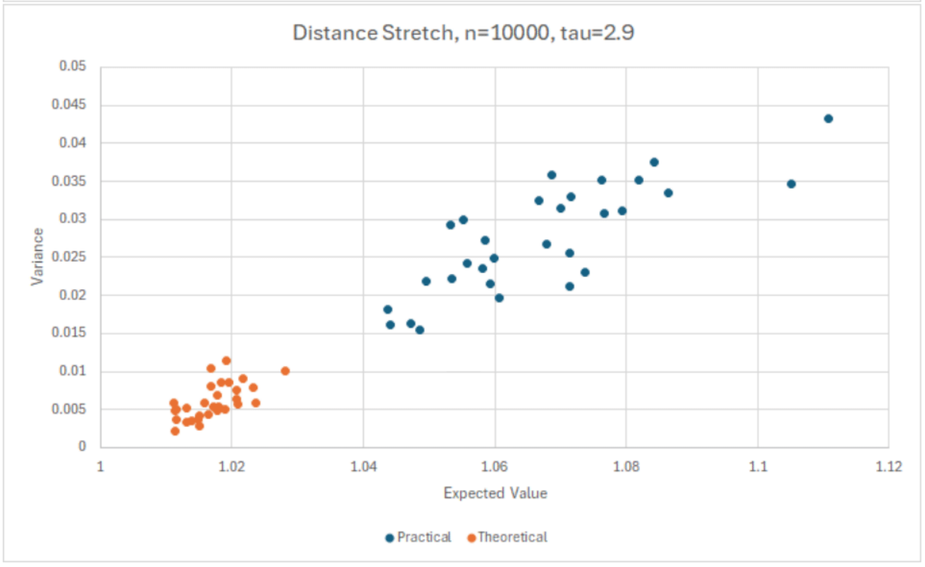


Figure 4: Comparison of Table Sizes for Theoretical and Practical

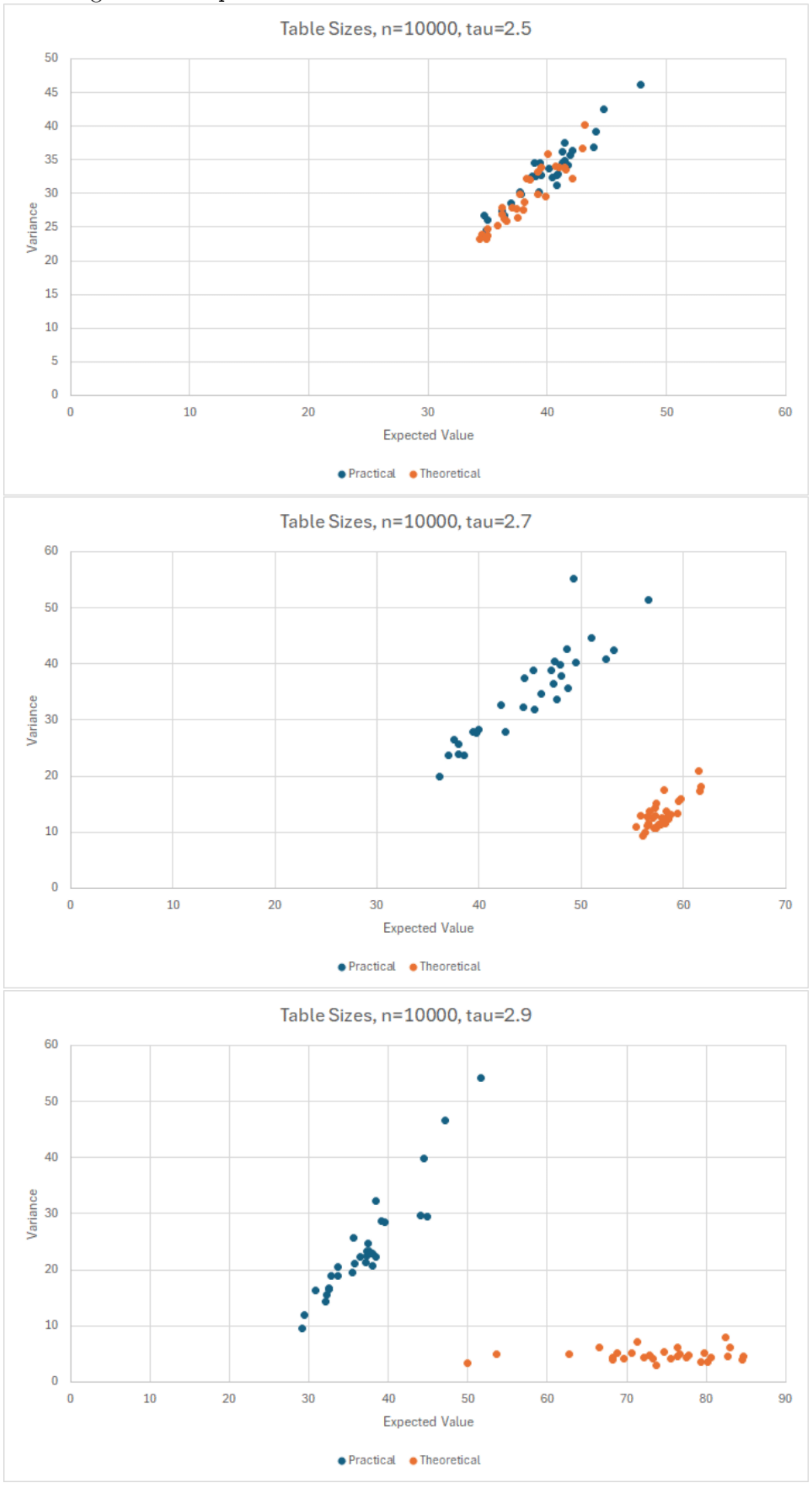
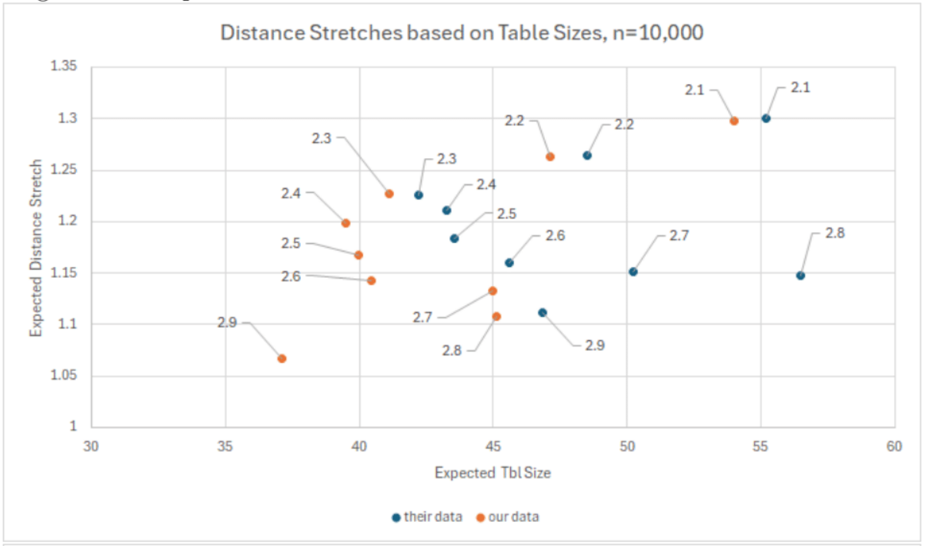


Figure 5: Comparison of Overall Tradeoff Between Our Data and Their Data



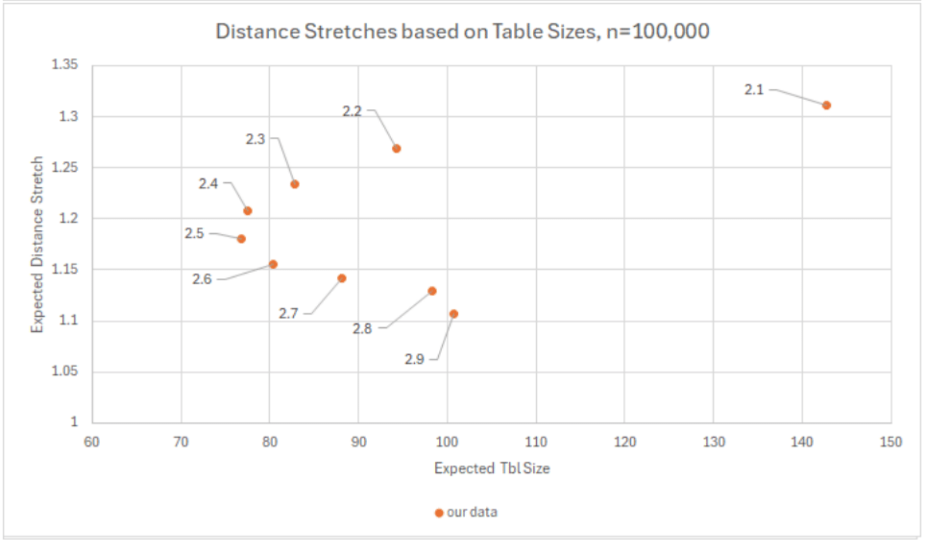


Figure 6: Better Threshold as a function of Tau

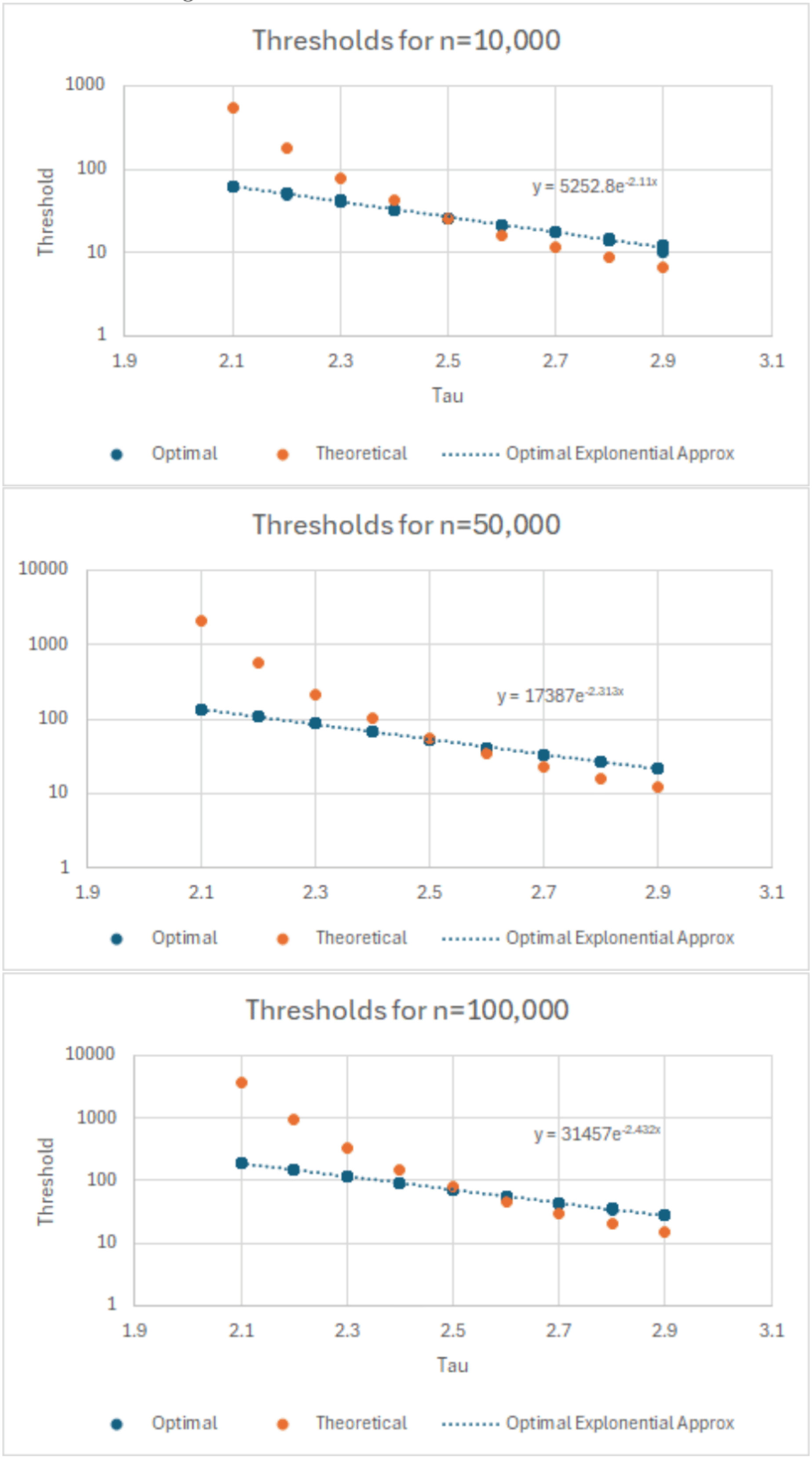
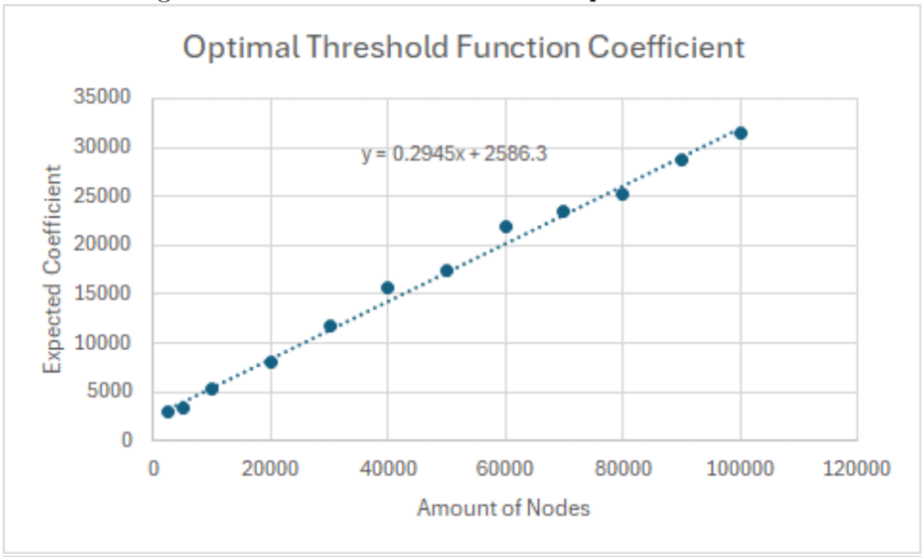


Figure 7: Threshold Coefficient and Exponent Functions



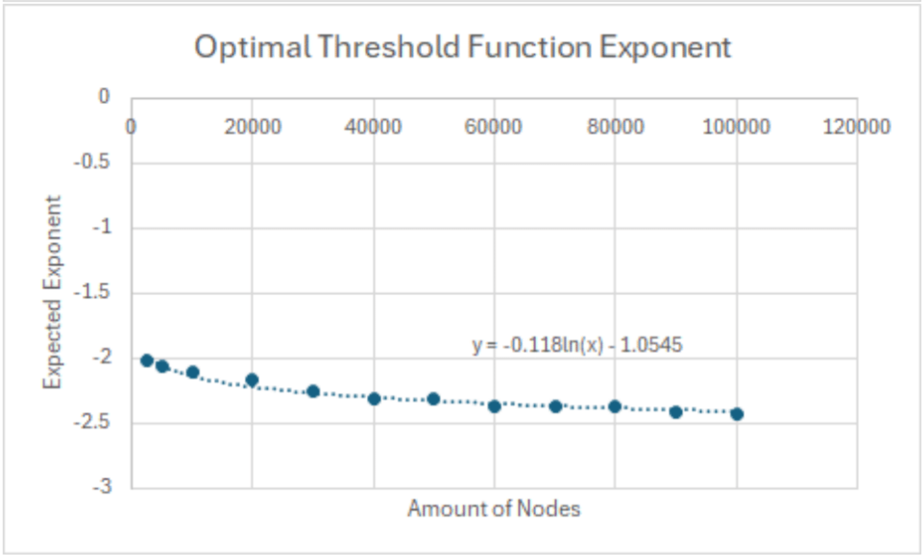


Figure 8: Performance Comparisong of Our Proposal for $n = 10,000$

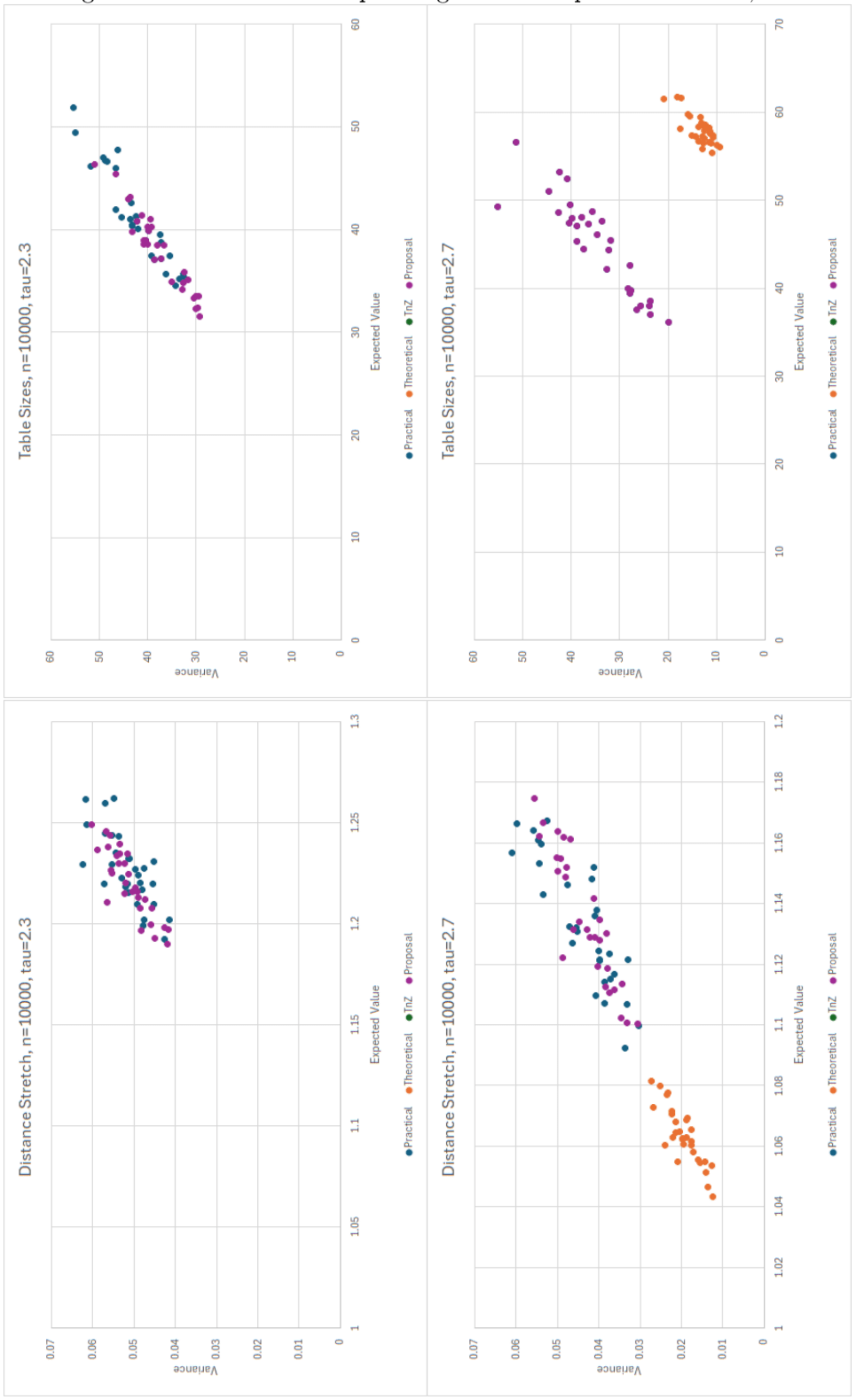


Figure 9: Performance Comparisong of Our Proposal for $n = 100,000$

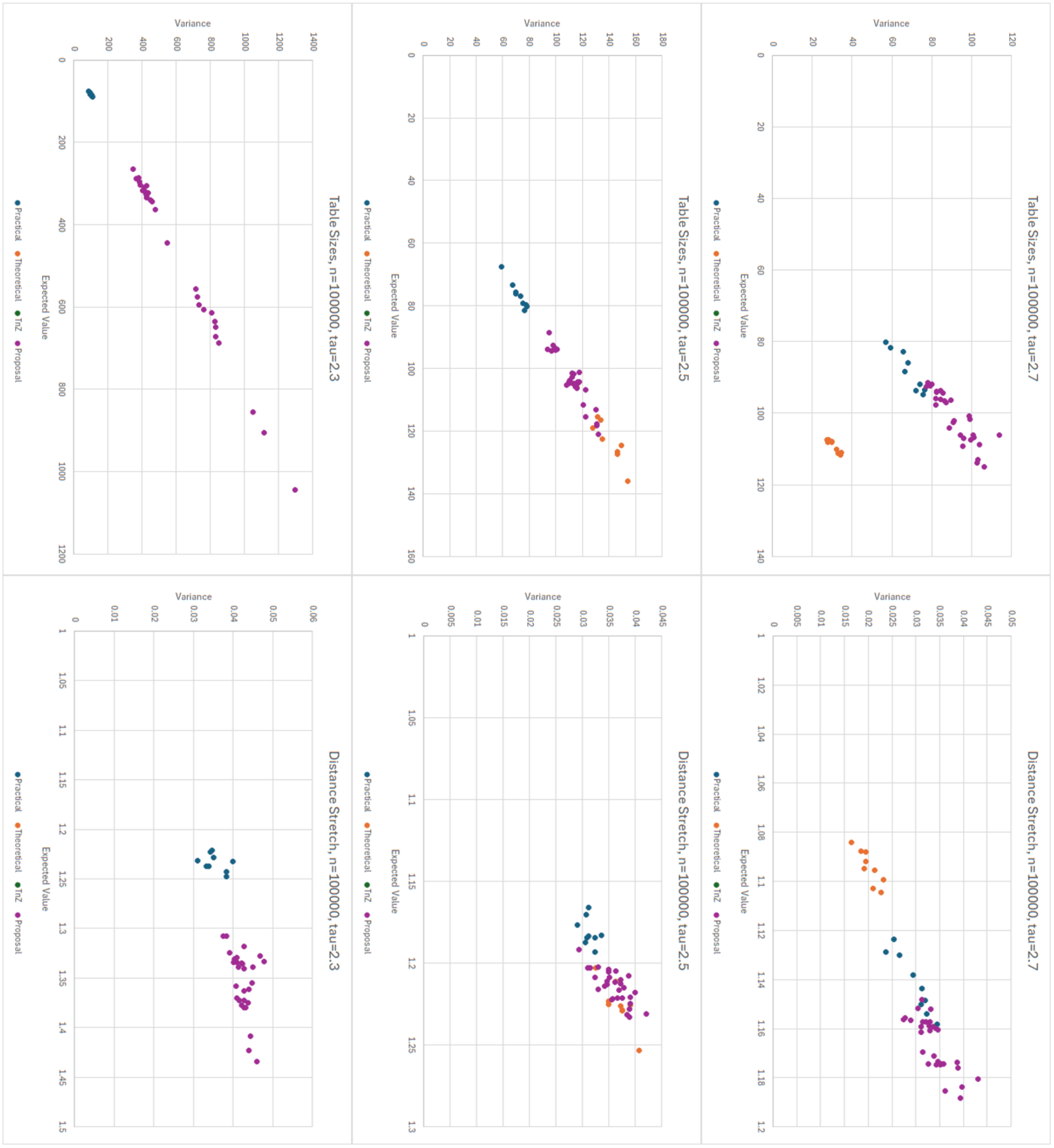


Table Sizes, n=100000, tau=2.7
Table Sizes, n=100000, tau=2.5
Table Sizes, n=100000, tau=2.3
Distance Stretch, n=100000, tau=2.7
Distance Stretch, n=100000, tau=2.5
Distance Stretch, n=100000, tau=2.3
Variance
Expected Value
Practical
Theoretical
TnZ
Proposal