The K Shortest Paths Problem with Application to Routing

David Burstein\textsuperscript{1,2} Leigh Metcalf \textsuperscript{1}

May 5, 2017

Abstract

While much work has proposed novel solutions to finding the $k$ shortest simple paths between two nodes, very little research has addressed the challenges in designing efficient solutions for real world networks. More specifically, the computational complexity for solutions to the $k$ shortest simple paths problem on positively weighted graphs typically include an $O(km)$ term, where $m$ is the number of edges in the graph. And even though an almost shortest simple path in an arbitrary graph could consist of $O(m)$ edges, in real world networks, such a solution appears asymptotically inefficient as such paths often scale logarithmically with the size of the network.

To address this gap, we provide both theoretical results and simulations illustrating that the ratio between simple paths and weakly simple paths of the same length is well behaved for a range of parameters in the Chung-Lu random graph model. Since the Chung-Lu random graph model captures many properties of real world networks, these results strongly suggest that in application, solutions to the $k$ shortest simple path problem should first identify a slightly larger collection of almost shortest paths and then remove the paths that contain loops. Furthermore, we present a simple algorithm for computing all (weakly simple) paths between two nodes bounded by length $D$ in $O(m \log m + kL)$ time for graphs with positive edge weights, where $L$ is an upperbound for the number of nodes in any returned path and $k$ is the number paths computed. We then consider an application to the almost shortest paths algorithm to measure path diversity for internet routing in a snapshot of the Autonomous System graph subject to an edge deletion process.

Keywords: $k$ shortest paths, internet routing, path sampling, edge deletion, simple paths, random graphs

MSC: 05C38, 05C85, 68R10, 90C35

\textsuperscript{1} Software Engineering Institute. Carnegie Mellon University. 4500 Fifth Avenue. Pittsburgh, PA 15213
\textsuperscript{2} Department of Mathematics and Statistics. Swarthmore College. 500 College Avenue, Swarthmore, PA 19081
\textsuperscript{3} Questions or comments may be sent to dburste1@swarthmore.edu
\textsuperscript{4} Distribution Statement A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US government use and distribution.
1 Introduction

Calculating the $k$ shortest simple paths between two nodes on positively weighted graphs arises in many applications; such applications include, inferring the spreading path of a pathogen in a social network [36], proposing novel complex relationships between biological entities [20, 22, 45], identifying membership of hidden communities in a graph [37, 40] and routing in the autonomous system (AS) graph, as discussed in this work and others [28, 44]. Even though research has demonstrated through simulation that exponentially slow solutions to the $k$ shortest simple paths problem often out-perform their polynomial time counterparts on many synthetic and real world networks [24], very little work has explored how properties in these empirically observed networks suggest efficient solutions to finding the $k$ shortest simple paths. Consequently, we address this gap by proposing an efficient method in finding almost shortest simple paths for a specific family of graphs that emulate many features found in real world networks.

When considering solutions to the almost shortest paths problem on a real world network, as opposed to an arbitrary graph, such a solution should exploit the small diameter and locally tree-like properties of the graph. Furthermore, the number of paths between two fixed nodes grows exponentially in terms of path length. The former property emphasizes that the optimal runtime for finding explicit representations for the $k$ shortest simple paths between two nodes, where the graph has $m$ edges and paths consist of at most $L$ nodes, should roughly be $O(m + kL)$. In addition, the latter property highlights the fact that constructing a set of just the $k$ shortest paths could ultimately exclude many paths of equal length. To analyze the almost shortest simple path problem for real world networks, we introduce the notion of weakly simple paths, which informally allows us to revisit nodes under certain constraints. We then provide both theoretical and numerical results identifying conditions in the Chung-Lu random graph model such that the number of simple paths is comparable to the number of weakly simple paths of the same length connecting two fixed nodes. As a result, we can construct an efficient algorithm for finding $k$ simple shortest paths by identifying a slightly larger set of (weakly simple) paths and deleting the paths that are not simple.

We emphasize that while our choice for considering the Chung-Lu model may appear arbitrary, Chung-Lu random graphs are closely related to the Stochastic Kronecker Graph model [10, 29, 30, 38], a commonly used random graph model for evaluating the efficiency of graph algorithms [3, 16, 47]. Additionally, Chung-Lu random graphs emulate many of the properties observed in real world networks; more specifically, realizations possess a small diameter along with de-
gree heterogeneity. We also anticipate that a similar relationship pertaining to
the ratio between the number of simple and weakly simple paths should carry
over for other random graph models that are also locally tree-like [8]. While
the last statement may appear obvious, proving precise theoretical upperbounds
for the ratio between the number of simple and non-simple paths, is intimately
related to constructing asymptotics for the dominating eigenvalue of the adja-
cency matrix, a highly nontrivial problem [12, 42, 46].

In application, many existing algorithms for identifying the $k$ shortest simple
paths are not designed to exploit properties often found in real world networks.
Such approaches often require deleting edges from the graph and running a
shortest path algorithm on the newly formed graph. For example, Yen [49] pro-
vides an $\mathcal{O}(kn(m + n \log n))$ solution that works for weighted, directed graphs,
while Katoh [27] and Roditty [43] provide $\mathcal{O}(k|m + n \log n|)$ and $\mathcal{O}(km\sqrt{n})$ so-
lutions respectively for undirected graphs. More recently, Bernstein [5] provides
an $\mathcal{O}(km/\epsilon)$ algorithm for computing approximate replacement paths. But since
we want to calculate many paths, such solutions can be asymptotically expen-
sive.

In contrast, the asymptotic performance for computing the $k$ shortest paths
is more appropriate for implementation on real world networks. Eppstein pro-
vides both $\mathcal{O}(m + n \log n + k \log k + Lk)$ and $\mathcal{O}(m + n \log n + Lk)$ solutions [17, 18]
for finding explicit representations of the $k$ shortest paths between two nodes,
where $L$ is an upperbound on the number of nodes that appear in a path. Never-
theless, recent variations to Eppstein’s solution often emphasize the asymp-
totically inferior version, due to the large constant factor behind the asymptot-
ically worst-case performance and the sophistication of the $\mathcal{O}(m + n \log n + Lk)$
solution [2, 19, 25, 26]. Consequently, our theoretical and numerical results
demonstrating that in Chung-Lu random graphs the number of almost shortest
non-simple paths makes a small contribution to the number of almost shortest
(weakly simple) paths strongly suggests that for real world networks we should
use an almost shortest path algorithm to solve the almost shortest simple path
problem.

Furthermore, by building upon the work of Byers and Waterman [6, 33],
we provide a simple $\mathcal{O}(m \log m + Lk)$ solution, for finding all (weakly simple)
paths bounded by length $D$ between two nodes in a directed positively weighted
graph, where $L$ is an upperbound for the number of nodes in a path and $k$ is
the number of paths returned. We stress that in practical application finding
all paths bounded by a prescribed length is often more useful than finding the
$k$ shortest paths as real world networks exhibit an exponential growth of the
number of paths in terms of length; hence, identifying (only) the $k$ shortest
paths could exclude many paths of equal length.
An outline of the our paper is as follows:

- In Section 2 we present a simple algorithm for finding all paths no greater than a prescribed length in $O(m \log m + kL)$ time respectively, where $L$ is an upperbound on the number of nodes appearing in any of the $k$ shortest paths. Furthermore, we illustrate that given a shortest path tree, for graphs with degree sequences resembling a scale-free distribution, as commonly observed in real world networks, that the complexity is $O(m + kL)$. We also stress that the algorithm works for positively weighted directed and undirected graphs.

- Then in Section 3, we introduce the notion of weakly simple paths. In particular, we explore properties of Chung-Lu random graphs in context to the $k$ shortest simple path problem and prove Corollary 1, an asymptotic result that bounds the ratio between the number of simple paths and the total number of weakly simple paths of a given length, where we sum over a collection of sources and targets of sufficiently large expected degree. Subsequently, we illustrate how to extend the algorithm in Section 2 to compute almost shortest weakly simple paths with time complexity $O(m \log m + kL)$ and space complexity $O(m + kL)$. These results strongly suggests that it is often more efficient to use an almost shortest weakly simple path algorithm, such as the solution provided in Section 2, to solve for almost shortest simple paths, than an almost shortest simple path algorithm.

- And finally in Section 4, we explore applications to the almost shortest paths problem in context to internet routing, where we use our solution for finding almost shortest paths to measure the diversity of surviving paths under an edge deletion process. We compare the AS graph to realizations of an Erdos-Renyi and Chung-Lu random graph and find that the AS graph behaves remarkably similar to realizations of the Chung-Lu random graph model under the edge deletion process.

2 K Shortest Path Algorithm

2.1 Strategy for Finding Almost Shortest Paths

Even though many algorithms have been proposed for finding almost shortest simple paths between two nodes, very little work has considered the implications for implementing such a solution on real world networks. We will argue in Section 3 that we should first compute almost shortest weakly simple paths
between two nodes to solve the almost shortest simple paths problem. In this section, we present a simple asymptotically efficient solution for finding all paths between two nodes less than a certain length. We will then argue in Section 3 how to extend this algorithm to compute almost shortest weakly simple paths. Before presenting the algorithm, we first sketch the solution strategy.

To find all paths between two nodes with length less than a given value, \( D \), our solution constructs a path tree illustrating all possible choices in identifying paths from the source to the target. As an example, consider finding all paths from node \( S \) to node \( T \) with length less than 3 in the graph on the left side of the first panel in Figure 1. We stress that while this example focuses on an undirected unweighted graph, the algorithm will work for directed positively weighted graphs as well.

As mentioned before, informally, the path tree identifies all possible options for constructing almost shortest paths from node \( S \) to node \( T \). Foremost, the algorithm assigns each nodes on the path tree to nodes in the original graph. As all such paths must end with the node \( T \), the algorithm maps the root of the tree to the node \( T \) in the graph. The right side of the first panel illustrates this initialization of the path tree.

Subsequently in the second panel, the algorithm identifies the node(s), \( T \), which corresponds to the node(s) recently added to the path tree in the prior panel; we marked such nodes in blue. Since all paths from \( S \) to \( T \) have length at most 3, it follows that the node that precedes \( T \) at the end of the path must have distance at most 2 from \( S \). Consequently, we mark in yellow the neighbors of the blue node, \( T \), with distance at most 2 from the source. Then the algorithm adds children to the blue node in the tree that correspond to the yellow neighbors of \( T \) in the original graph. In this case, \( T \) has three neighbors that have distance 2 from the node \( S \): \( A \), \( B \) and \( C \).

In Step 3, we repeat the same argument. For each of the newly added nodes in the tree from the prior step, now marked in blue, we identify that node’s neighbors in the original graph with distance at most 1 from \( S \). Nodes \( A \) and \( B \) only have one such neighbor, \( C \), that satisfies this constraint, so the algorithm adds a child that corresponds to \( C \) to those respective blue nodes in the path tree. Node \( C \) in the original graph, which is both a blue and yellow node, has three neighbors that have distance at most 1: \( D, E \) and \( S \).

Finally in Step 4, for each of the newly added nodes in the tree from the prior step, the algorithm identifies the neighbors in the original graph with distance at most 0 from \( S \). Step 4 completes the construction of the path tree. We now demonstrate how to efficiently extract all paths from node \( S \) to node \( T \) with length at most 3 using the path tree. See Figure 2. First, while constructing the path tree, we record all nodes that correspond to \( S \) in the original graph.
Figure 1: An illustration on how to construct a path tree (on the right) for finding paths with length at most 3 between source $S$ and target $T$ in an unweighted graph (on the left). Each numbered panel corresponds to a step in constructing the path tree. In the path tree, yellow nodes indicate newly added nodes, while blue nodes indicate that the nodes were added to the path tree in the prior step. Similarly, the nodes that correspond to the yellow (blue) nodes in the path tree are also shaded yellow (blue). Nodes that correspond to both a blue and yellow node in the path tree are shaded yellow with a blue border.

In the first panel, we highlighted all nodes that correspond to $S$ either in green or yellow. We focus on the yellow node. In the second panel, by looking at the parent of the yellow node in the tree, we can identify the next node on the path, $E$. Continuing this process for the third and fourth panels yields the path $S, E, C, T$.

Now that we have explained the intuition behind the proposed solution, at this juncture we specify the inputs (and outputs) for the algorithm, Pathfind.
Figure 2: An illustration on how to extract paths from the path tree. We record all nodes that correspond to $S$ while constructing the path tree. Then for each such node that corresponds to $S$, we traverse the path tree to the root to identify the corresponding path in the original graph.

**Pathfind** requires the following inputs:

- $V$, a list of nodes in the graph.
- $InNbrs_x$, a list of the incoming neighbors for each node $x \in V$.
- The source and target, for the almost shortest paths.
- $d(source, \cdot)$, the distance function from the source to any node in the graph.
- $d(n, x)$, the positive edge weights in the graph where $n \in InNbrs_x$.
- $D$, the upperbound on the lengths for the almost shortest paths

**Pathfind** then outputs a list of all paths from the source to the target with length at most $D$.

See Algorithm 1 for an outline of the algorithm, **Pathfind**. Foremost, to achieve the desired asymptotic complexity, Step 1 in **Pathfind** sorts the incoming neighbors $n$ of each node $x$ according to $d(source, n) + d(n, x)$, the distance...
from the source to the neighbor \( n \) plus the weight of the edge connecting the incoming neighbor \( n \) to \( x \). Since nodes can have many neighbors, adding this step will prevent \texttt{Pathfind} from considering a potentially large number of neighbors that are not sufficiently close to the source to form an almost shortest path.

Subsequently, Step 2 initializes the path tree described in the first panel in Figure 1. Whenever \texttt{Pathfind} adds nodes to the path tree, \texttt{Pathfind} defines the following attributes for such newly added nodes. The \texttt{Parent} attribute returns the parent of a node on the path tree and the \texttt{ID} attribute of a node in the path tree identifies the corresponding node on the original graph. Furthermore by construction, edges in the path tree also correspond to edges in the original graph, as neighbors in the path tree correspond to neighbors in the original graph. Hence, traversing up a given node in the path tree to the root corresponds to a path in the original graph. Consequently, the \texttt{trackdistance} attribute of a node in the path tree keeps track of the distance of that path in the original graph. At the conclusion of Step 2 \texttt{Pathfind} also initializes a set \texttt{Pathstart} to keep track of any nodes in the path tree that correspond to the source, as mentioned in the discussion of Figure 2.

Then in Step 3, \texttt{Pathfind} builds the path tree as illustrated in the second to fourth panels in Figure 1. More specifically, for each recently added node, \( l \), to the path tree, \texttt{Pathfind} identifies all neighbors \( n \) of \( l.ID \) in the original graph.
such that \( d(source, n) + d(n, l.ID) \) is sufficiently small as illustrated in Figure 3. More precisely, suppose we know that there exists a path with length at most \( D \) of the form \((s, ?, \ldots, ?, l_1.ID, \ldots, l_k.ID, t)\), where the nodes \( s, l_1.ID, \ldots, l_k.ID, t \) are fixed and there are no constraints on the nodes connecting \( s \) to \( l_1.ID \). (In practice, they are unknown.) We then wish to determine if there exists a path with length at most \( D \) of the form \((s, ?, \ldots, ?, n, l_1.ID, \ldots, l_k.ID, t)\), where \( n \) is an incoming neighbor of \( l_1.ID \). Furthermore, suppose that we record the length of the path \((l_1.ID, \ldots, l_k.ID, t)\) as \( l_1.trackdistance \). Consequently, if there exists a path of the form \((s, ?, \ldots, ?, n, l_1.ID, \ldots, l_k.ID, t)\) with length at most \( D \), it follows that \( d(s, n) + d(n, l_1.ID) + l_1.trackdistance \leq D \), or equivalently

\[
d(s, n) + d(n, l_1.ID) \leq D - l_1.trackdistance. \tag{1}
\]

For each such neighbor \( n \) that satisfies (1), \textbf{Pathfind} adds a new node to the path tree, defining the attributes, \textbf{Parent}, \textbf{ID} and \textbf{trackdistance} accordingly. In the event the neighbor \( n \) under consideration is the \textbf{source}, \textbf{Pathfind} adds the corresponding node in the path tree to the set \textbf{Pathstart}.

Finally, Step 4 extracts paths from the path tree, as described in Figure 2. That is for each \textit{treenode}, where \textit{treenode.ID} is the \textbf{source}, \textbf{Pathfind} traverses up the tree to the root to construct an explicit representation of a path from the \textbf{source} to the \textbf{target}. \textbf{Pathfind} then returns all such paths with length bounded by the prescribed parameter \( D \).
Algorithm 1: Pathfind

1. (Sort adjacency list). For each node $x \in V$, sort the nodes $n \in InNbrs_x$ by $d(source, n) + d(n, x)$ in non-decreasing order. We can then easily identify incoming neighbors of $x$ that are close to the source.

2. (Initialize path tree.) Initialize a path tree with the single node $root$. We will construct a correspondence between paths on the path tree to almost shortest paths between the source and target in the original graph.
   
   (a) Define the following attributes for nodes in the path tree.
      
      i. **Parent** returns the parent of a node on the tree. Set $root.Parent \leftarrow \emptyset$.
      
      ii. **ID** maps nodes in the tree to nodes in the graph. Set $root.ID \leftarrow target$.
      
      iii. **trackdistance** tracks the distance traveled so far in the original graph. Set $root.trackdistance \leftarrow 0$.
      
   (b) Initialize a set $PathStart \leftarrow \emptyset$, where $PathStart$ will contain all nodes $t$ in the path tree, such that $t.ID$ is the source.
      
   (c) Initialize a queue, $Q$, with node root. Similar to a breadth-first search, $Q$ will help us identify almost shortest paths between the source and target.

3. (Search for almost shortest paths by constructing path tree). While $Q \neq \emptyset$, remove the last element $l$ from $Q$.
   
   (a) For each $n \in InNbrs_{l.ID}$, check if there exists a path from source to $l.ID$, where the last edge connects $n$ to $l.ID$ and the path has length at most $D - l.trackdistance$. If there does not exist such a path, we should exit this for loop as all remaining nodes in $InNbrs_{l.ID}$ are too far away from the source by Step 1. Else, we should add a new node $z$ to the tree with the following attributes.
      
      i. $z.parent \leftarrow l$
      
      ii. $z.ID \leftarrow n$
      
      iii. $z.trackdistance \leftarrow d(n, l.ID) + l.trackdistance$ and
      
      iv. $Q \leftarrow Q \cup z$.
      
      v. If $n == source$ then add $z$ to $PathStart$.

4. (Construct explicit representation for almost shortest paths). Initialize the output, the list of almost shortest paths, $PathList \leftarrow \emptyset$. Then for each $v \in PathStart$, find the shortest path from $v$ to the root in the tree path by using the parent attribute as mentioned in Figure 2.
   
   (a) Denote the path in the path tree as $(v, v_1, ..., v_k)$, where $v_k$ is the root. Then the corresponding path in the original graph is $(v.ID, v_1.ID, ..., v_k.ID)$. Note that by construction, $v.ID$ is the source and $v_k.ID = root.ID$ is the target.
      
   (b) $PathList \leftarrow PathList \cup (v.ID, v_1.ID, ..., v_k.ID)$.

5. Return $PathList$. 
2.2 Complexity Analysis

We now verify the claimed computational complexity of the algorithm for identifying all paths between two fixed nodes of length bounded by $L$.

**Theorem 1.** Denote $d_{in}(x)$ as the in-degree of node $x$, let $V$ be the set of all nodes in the graph and $m$ be the total number of edges in the graph. Then if the graph is nicely weighted or if $\sum_{x \in V} d_{in}(x) \log d_{in}(x) = O(m)$, then the computational runtime for the Pathfind algorithm (in section 2.1) to identify all shortest paths of length bounded by $D$ from node $s$ to node $t$ is $O(m + kL)$, where $k$ is the number of shortest paths in the output, $n$ is the number of nodes in $V$ and $L$ is an upperbound for the number of nodes in any outputted path. Otherwise, the computational runtime is $O(m \log m + n + kL)$.

**Proof.** Step 1 in Pathfind sorts the incoming neighbors for each node $x$, $InNbrs_x$. Denote $d_{in}(x)$ as the number of incoming neighbors for node $x$ and let $V$ be the set of all vertices. Then it follows that for a nicely weighted graph if Pathfind uses counting sort, the computational complexity of sorting the sets $InNbrs_x$ for all $x$ is precisely $\sum_{x \in V} O(d_{in}(x)) = O(m)$. Alternatively by using a heapsort, Pathfind can sort the sets $InNbrs_x$ for all $x$ in $\sum_{x \in V} O(d_{in}(x) \log d_{in}(x))$ time. This quantity is trivially bounded by $\sum_{x \in V} O(d_{in}(x) \log m) = O(m \log m)$, as the maximum in-degree is bounded by the number of edges. Alternatively, for many real world networks, $\sum_{x \in V} d_{in}(x) \log d_{in}(x) = O(m)$. Hence it would follow that if either the graph is nicely weighted or if $\sum_{x \in V} d_{in}(x) \log d_{in}(x) = O(m)$, then Step 1 takes $O(m)$ time. Otherwise, Step 1 takes $O(m \log m)$ time.

Step 2 takes constant time $O(1)$ as Pathfind initializes the path tree.

For Step 3, note that the complexity for evaluating the criteria to determine whether we add a node to our path tree is $O(1)$. Furthermore, for every neighbor of $l.ID$ that satisfies the criteria must yield at least one path. And since $InNbrs_{l.ID}$ is sorted, there is only an $O(1)$ penalty when Pathfind comes across a neighbor that is not sufficiently close to node $s$ in the graph as all other neighbors that have not been checked are too far to construct an almost shortest path. Since the time complexity for adding a node or edge to the path tree is $O(1)$, the complexity of Step 3 is proportional to the number of nodes and edges in the tree.

By construction of the algorithm, as illustrated in panel 4 in Figure 1 and in Figure 2, all leaves $z$ in the path tree have the property that $z.ID$ maps to node $s$ in the original graph. Since paths from $s$ to $t$ contain at most $L$ nodes, then the (unweighted) distance from any leaf to the root in the path tree must be at most $L$, as paths in the path tree from a leaf to the root correspond to a path from node $s$ to $t$ in the original graph. Note that every node in the path tree is in at least one path from a leaf to the root of the tree. As there
The K Shortest Paths Problem

are at most $k$ leaves, because there cannot be more than $k$ outputted paths, it then follows that the number of nodes is the path tree $O(kL)$. Furthermore, since the number of edges in a tree equals the number of nodes minus one, we conclude that the number of edges is also $O(kL)$. Hence the complexity of Step 3 is $O(kL)$.

Finally in Step 4, the time to reconstruct explicit representations for each path is $O(L)$ as the shortest path from any node to the root contains at most $L$ nodes by assumption. And since Pathfind outputs at most $k$ paths this step also costs $O(kL)$. The combined computational complexity of the algorithm is therefore $O(m + kL)$ or $O(m \log m + kL)$ depending on the assumptions from Step 1.

Remark: Many real world networks exhibit degree sequences that appear to follow a scale free distribution [4, 9, 48], that is $\Pr(d_{in}(x) = k) \propto k^{-\gamma}$, for some positive value of $\gamma$. (Typically $\gamma$ is greater than 2). Let $n$ be the number of nodes in the network. If for instance $\gamma > 2$, it follows that,

$$\sum_{x \in V} d_{in}(x) \log d_{in}(x) \approx n \int_{1}^{n} \Pr(d_{in}(x) = k)k \log(k)dk \propto n \int_{1}^{n} k^{-\gamma+1} \log(k)dk,$$

(2)

where $\int_{1}^{n} \Pr(d_{in}(x) = k)k \log(k)dk$ is roughly the expected value for the in-degree times the logarithm of the in-degree. Notice that we integrate up to $n$ as the in-degree of a node cannot exceed the number of nodes in the network.

Using integration by parts it follows that

$$n \int_{1}^{n} k^{-\gamma+1} \log(k)dk = O\left(\frac{n}{(\gamma - 2)^2}\right).$$

As for many real world networks $n < m$, we conclude that

$$\sum_{x \in V} d_{in}(x) \log d_{in}(x) = O(m),$$

holds for many real world networks.

Now that we have verified that the time complexity for Pathfind, we now consider the space complexity.

Lemma 1. The space complexity for Pathfind is $O(kL)$.

Proof. From Step 1, sorting takes $O(1)$ additional space. (If Pathfind uses count sort, we assume that the range of possible values is bounded by a constant). From Steps 2-5, constructing the path tree takes $O(kL)$ space as there are at most $kL$ nodes in the tree. Consequently, Pathfind has $O(kL)$ space complexity.
With the space and time complexity results for Pathfind at hand, we verify that Pathfind does indeed find all paths of length bounded by $L$.

**Theorem 2.** *Pathfind* finds all paths of length bounded by $D$ from node $s$ to node $t$.

*Proof.* By construction of the algorithm if two nodes $x$ and $y$ in the path tree are neighbors, then $x.ID$ and $y.ID$ are neighbors in the original graph $G$. Consequently, it follows that for any path $P = (v_1, ..., v_k)$ in the path tree, where $v_1.ID$ is the source and $v_k.ID$ is the target, then $(v_1.ID, ..., v_k.ID)$ is a path from the source, $s$, to the target, $t$. Hence, all paths returned by Pathfind are paths from $s$ to $t$.

Alternatively for any path in the original graph $(n_1, ..., n_k)$, with length bounded by $D$, where $n_1 = s$, the source and $n_k = t$, the target, we need to show that there is a corresponding path in the path tree. Inductively starting with the node $n_k$, let $v_k$ be the root of the tree and it follows that $v_k.ID = n_k$. Now by construction, since $d(n_{k-1}, n_k) + d(n_1, n_{k-1}) \leq D$, as $(n_1, ..., n_k)$ has length bounded above by $D$, it follows that there is a child $v_{k-1}$ of the root $(v_k)$, where $v_{k-1}.ID = n_{k-1}$.

Furthermore by construction, it follows that there is a unique child of $v_{k-1}$, $v_{k-2}$, where $v_{k-2}.ID = n_{k-2}$, as $d(n_1, n_{k-2}) + d(n_{k-1}, n_{k-2}) \leq D - d(n_{k-1}, n_k)$, where $d(n_{k-1}, n_k) = v_{k-1}.trackdistance$. Proceeding inductively, we conclude that for all $j \in \{1, ..., k\}$, there is a node $v_j$ in the path graph with the properties that if $j < k$, $v_j.Parent = v_{j+1}$ and $v_j.ID = n_j$. Furthermore, since $v_1.ID = s$, the source, this implies that $v_1 \in PathStart$. Hence we conclude that for every path from $s$ to $t$ of length at most $D$, there is a corresponding path in the path graph. Consequently, there is a bijective mapping between the paths from $s$ to $t$ with length at most $D$ and the paths from nodes in $PathStart$ to the root of the path tree. 

\[\square\]

3 The Ratio of Simple to Nonsimple Paths in Chung-Lu Random Graphs

Intuitively, since real world networks are locally tree like, to efficiently identify almost shortest simple paths, we should use an almost shortest path algorithm. Consequently, we employ the Chung-Lu random graph model as a convenient method for constructing a collection of graphs that emulate properties of real world networks. To this end, we seek conditions for realizations of the Chung-Lu random graph model, where the number of simple paths of fixed length is roughly the same as the number of paths of that length. Unfortunately, for many undirected random graphs with nodes of large degree, the aforementioned claim
is false [7]; short non-simple paths in undirected graphs can outnumber simple paths by considering paths that revisit nodes of large degree. To circumvent this issue, we introduce the notion of a weakly simple path, that excludes paths that traverse the same edge twice in a row. After illustrating that under a broad range of parameters that the number of weakly simple paths asymptotically approximates the number of simple paths of the same length (Corollary 1), we then demonstrate how to adapt the almost shortest paths algorithm in the prior section to compute the almost shortest weakly simple paths with the same time and space complexity (Lemma 6).

Definition 1. Chung-Lu Random Graph Model [11]: Let \( n \) be the number of nodes in an undirected graph and let \( w = (d_1, ..., d_n) \) be the expected degree sequence, where \( d_i \) corresponds to the expected degree of node \( i \). Denote \( S = \sum_{i=1}^{n} d_i \) and suppose that \( \max_i d_i^2 \leq S \). We then model edges in the graph as independent Bernoulli random variables, where we denote the probability an edge connects nodes \( i \) and \( j \) as \( p_{ij} \) and \( p_{ij} = \frac{d_i d_j}{S} \).

As a technical point in the above definition, nodes may have edges that connect to themselves. But before introducing any subsequent results regarding the Chung-Lu random graph model, the following notation will be helpful.

Definition 2. Define the random variable, \( SP_r(s, t) \), to be the number of simple paths from node \( s \) to \( t \) with length \( r \). In the event that \( s = t \), define \( SP_r(s, s) \) to be the number of simple cycles, where the path starts with node \( s \).

Notice that from the prior definition, we break the standard convention by classifying a simple cycle as a simple path, even though the last node repeats. To calculate the number of simple paths in the graph, we employ Hoare-Ramshaw notation for a closed set of integers, namely

\[ [a..b] = \{ x \in \mathbb{Z} : a \leq x \leq b \} \]

We now illustrate the expected exponential growth for the number of simple paths between any two nodes for the Chung-Lu random graph.

Lemma 2. For the Chung-Lu random graph model, define \( S_2 = \sum_{i=1}^{n} d_i^2 \) and \( d_{\max} = \max_i d_i \). Consider the expected number of simple paths of length \( r \) from node \( s \) to \( t \), \( E[SP_r(s, t)] \). It then follows that

\[
p_{st} \frac{S_2^{r-1}}{S} \left(1 - \frac{r(r+1)p_{\max}}{2} \frac{S}{S_2}\right) \leq E[SP_r(s, t)] \leq p_{st} \frac{S_2^{r-1}}{S},
\]

where \( p_{st} \) is the probability that an edge exists connecting the nodes \( s \) and \( t \).

Proof. Define the set \( B_{st} \) such that \( b \in B_{st} \) if \( b = (s, b_1, ..., b_{r-1}, t) \in \mathbb{N}^{r+1} \), where each entry in \( b \) is distinct, (except for the possibility that \( s = t \) and the
\[ b_i \in [1..n] \text{ for all } i \in [1..r - 1]. \] Informally, \( B_{st} \) consists of all simple paths from \( s \) to \( t \) of length \( r \) that could exist in a graph of \( n \) nodes. It then follows that the expected number of simple paths of length \( r \) between nodes \( s \) and \( t \) is the sum of probabilities that a given simple path from \( s \) to \( t \) exists.

\[
E[SP_r(s, t)] = \sum_{b \in B_{st}} p_{sb_1}(\Pi_{k=1}^{r-2} p_{b_k,b_{k+1}}) p_{r-1,t},
\]

Using the probabilities that two nodes share an edge in the Chung-Lu random graph model, we can rewrite the above expression.

\[
E[SP_r(s, t)] = \sum_{b \in B_{st}} \frac{d_s d_b}{S} (\Pi_{k=1}^{r-2} \frac{d_{b_k}d_{b_{k+1}}}{S}) \frac{d_{b_{r-1}}}{S}.
\]

Noticing that for each index \( i \) from 1 to \( r - 1 \), the term \( d_{b_i} \) appears twice in the product, we rearrange the terms to get the following,

\[
E[SP_r(s, t)] = \frac{d_s d_t}{S} \sum_{b \in B_{st}} \Pi_{k=1}^{r-1} \frac{d_{b_k}^2}{S} \geq p_{st}(\sum_{b_1,\ldots,b_{r-1}=1}^{n} \Pi_{k=1}^{r-1} \frac{d_{b_k}^2}{S} - \frac{(r + 1)}{2} \sum_{b_2,\ldots,b_{r-1}=1}^{n} \Pi_{k=1}^{r-1} \frac{d_{b_k}^2}{S}),
\]

where we derive the last inequality by inclusion-exclusion; we consider the contribution of the summation by removing the constraint of the distinctness of the \( b_i \) terms and then we subtract off terms where the \( b_i \) either equal \( s \), \( t \), or another \( b_j \). To compute the quantity we should subtract off, we first consider the contribution where \( b_1 = b_2 \) and then multiply that quantity by \( \binom{r+1}{2} \), corresponding to the number of ways any two of the \( r + 1 \) variables could equal each other and hence violate the constraints in the original summation. It then follows that

\[
E[SP_r(s, t)] \geq p_{st}(S_2^{r-1} \frac{S}{S} - \frac{(r + 1)}{2} \sum_{b_2,\ldots,b_{r-1}=1}^{n} p_{max} \Pi_{k=2}^{r-2} \frac{d_{b_k}^2}{S}) \geq p_{st}(S_2^{r-1} \frac{S}{S} - \frac{(r + 1)}{2} p_{max} \frac{S_2^{r-2}}{S_2}) = p_{st} \left( S_2^{r-1} \frac{S}{S} - \frac{r(r+1)}{2} p_{max} \frac{S}{S_2} \right).
\]

The proof for the upperbound of \( E[SP_r(s, t)] \) follows from the fact that,

\[
E[SP_r(s, t)] = \frac{d_s d_t}{S} \sum_{b \in B_{st}} \Pi_{k=1}^{r-1} \frac{d_{b_k}^2}{S} \leq \frac{d_s d_t}{S} \sum_{b_1,\ldots,b_{r-1}=1}^{n} \Pi_{k=1}^{r-1} \frac{d_{b_k}^2}{S} \leq \frac{d_s d_t}{S} \frac{S_2^{r-1}}{S}. \]
As a result of Lemma 2, when calculating the $k$ shortest paths, the expected number of simple paths grows exponentially in terms of length. Consequently, we may be arbitrarily or perhaps even systematically ignoring many paths of the same length. For this reason in many applications, it is often more informative to calculate all paths bounded by a fixed length as opposed to calculating just $k$ of them. In addition, we also note that if $r^2 \frac{S}{S^2} \to 0$, the difference between upper and lower bounds for the expected number of simple paths between two nodes approaches 0. In particular, we have that $E(\text{SP}(r,s,t)) \to 1$. Since we wish to show that the ratio between the number of simple paths and weakly simple paths of the same length is well behaved, we seek bounds for the expected number of (weakly simple) paths of length $r$ between two arbitrary nodes; however, since the same edge may appear multiple times on a path, we first provide an efficient method for computing the probability that an arbitrary path exists. To do so, we will need the following definitions.

First we introduce the notion of a multiset, or a list of objects that allows for repeats. For a multiset $X$, an object $x \in X$ if $x$ appears in the multiset $X$. If for example the multiset $X = [2, 4, 4]$ and the objects in $X$ are in the domain of a function $f$, then we seek the notation $\prod_{x \in X} f(x) = f(2) \cdot f(4) \cdot f(4)$, to include objects that repeat in the multiset. For this purpose, we define the cardinality $|X|$ as the number of objects in the list. Formally, we can denote the $i$th object in $X$ as $X_i$. Rigorously, we can now define,

$$\prod_{x \in X} f(x) = \prod_{i=1}^{|X|} f(X_i).$$

With this understanding of a multiset, we now have the following definition.

**Definition 3.** Consider an edge in a given path. If the edge has not appeared before, that edge is a **new edge**. Alternatively, if the edge has appeared before, that edge is a **repeating edge**. Furthermore, a list (multiset) of consecutive repeating edges of maximal size in a path is called a **repeating edge block**. Analogously, a list (multiset) of consecutive new edges of maximal size is called a **new edge block**. The **interior** of a new edge block is a multiset of all of the nodes in the new edge block except for the first and last nodes.

In order to construct a convenient formula for computing the probability that a given path exists, it will be helpful to identify which nodes appear elsewhere in the path.

**Lemma 3.** Let $x$ be a node in an undirected graph that appears in a repeating edge block of a path and is not the first node in that repeating edge block. Then
The K Shortest Paths Problem

Proof. Suppose that there exists a path \( P \) that contradicts the lemma. In particular, consider the first such contradiction, \( x \), where \( x \) is the \( m \)th node in the path. As \( x \) is not the first node in the repeating edge block, we know that there exists an edge of the form \((y, x)\) in the repeating edge block for some node \( y \). By the definition of a repeating edge, \((y, x)\) or \((x, y)\) appears earlier as a new edge in the path.

**Case 1:** \((y, x)\) appears earlier as a new edge in the path. If \((y, x)\) is a new edge in the path, then either \( x \) is in the interior of a new edge block or \( x \) is at the end of a new edge block and hence would also be the first node of a repeating edge block.

**Case 2:** \((x, y)\) appears earlier as a new edge in the path. Then \( x \) can either be the first node in the path, \( x \) can be in the interior of a new edge block, or \( x \) is at the beginning of a new edge block. If \( x \) is at the beginning of a new edge block, then \( x \) is at the end of a repeating edge block. But this implies that the first such contradiction happens before the \( m \)th node in the path, a contradiction.

With Lemma 3 at hand, we have the following lemma to compute the probability that a given path exists.

**Lemma 4.** Define \( X_{(i,j)} \) as an indicator random variable that equals 1 if the edge \((i, j)\) exists and 0 otherwise. Consequently, \( \Pi_{k=1}^{r} X_{(i_k, i_{k+1})} \) is an indicator random variable that equals 1 if there is a path \((i_1, \ldots, i_{r+1})\). Let \( N \) be the multiset of all nodes in the interior of a new edge block. Let \( R_1 \) be a list of pairs of the first and last nodes for each repeating edge block, where the first node has appeared before and let \( R_2 \) be a list of pairs of the first and last nodes for each repeating edge block, where the first node has not appeared before. If the first and last edges are new edges, then

\[
\Pr(\Pi_{k=1}^{r} X_{(i_k, i_{k+1})} = 1) = \frac{d_1 d_{i_r+1}}{S} \prod_{i \in N} \frac{d_i^2}{S} \prod_{(j, k) \in R_1} \frac{d_j d_k}{S} \prod_{(l, m) \in R_2} \frac{d_l d_m}{S}. \tag{3}
\]

Furthermore, if \( q_i \) is the number of repeating edge blocks of length \( i \), then the number of nodes in \( N \),

\[
|N| = r - 1 - \sum_{i=1}^{r-2} (i + 1) q_i. \tag{4}
\]

**Remark:** To correctly compute the probabilities, we will want to consider the number of times a node appears in a new or repeating edge block along with...
the identity of that node, as such the same node can appear multiple times in $N$ (or $R_1$, $R_2$).

**Proof.** To derive (3), we first consider some examples from Figure 4. For example in the first path in Figure 4, there are only new edges and the interior of the new edge block $N = [2,3,4,5,6]$. It follows that the probability that the first path in Figure 4 exists is precisely
\[
\frac{d_s d_t}{S} \prod_{i=2}^{6} d_i = \frac{d_s d_t}{S} \prod_{i \in N} d_i.
\]

More generally for an arbitrary path from node $s$ to node $t$ with no repeating edges, it follows that the probability the path exists is,
\[
\frac{d_s d_t}{S} \prod_{i \in N} d_i.
\]

Of course as in the second path in Figure 4, we may have repeating edges. By noting that the last node in a repeating edge block must appear earlier,
there are two types of repeating edge blocks; either the first node has appeared earlier in the path or the first node has not appeared earlier in the path.

Define a multiset $R_1$ consisting of the first and last nodes for each repeating edge block, where those first node has appeared earlier in the path. Consequently, the probability such a path exists is,

$$\frac{d_sd_t}{S} \prod_{i \in N} \frac{d_i^2}{S} \Pi_{(j,k) \in R_1} \frac{d_jd_k}{S}.$$

Finally, since the first node in a repeating edge block may have not appeared before, define a multiset $R_2$ consisting of the the first and last nodes for each repeating edge block, where the first node has not been seen before. Then the probability such a path exists is

$$\frac{d_sd_t}{S} \prod_{i \in N} \frac{d_i^2}{S} \Pi_{(j,k) \in R_1} \frac{d_jd_k}{S} \Pi_{(l,m) \in R_2} \frac{d_ld_m}{S}.$$

This completes the proof of (3).

To verify $|N| = r - 1 - \sum_{i=1}^{r-2} (i + 1)q_i$,

we consider the number of times a node is not in an interior of a new edge block, $|N^c|$. Alternatively, $|N^c|$ counts the number of times a node appears in a repeating edge block in addition to the first and last nodes of the path. This quantity is precisely $2 + \sum (i + 1)q_i$, where $q_i$ is the number of repeating edge blocks of length $i$ and $i + 1$ are the number of nodes in a repeating edge block of length $i$. Consequently, since there are $r + 1$ nodes in a path of length $r$, $r + 1 - 2 - \sum_{i \geq 1} (i + 1)q_i$ is precisely the right hand side of equation (4). (We derive the upperlimit in the summation from the assumption that the first and last edges are new edges and that the path has length $r$, which implies that a maximal of length of a repeating edge block could be at most $r - 2$.)

At this juncture, we provide a formal definition for a weakly simple path, which we will show that such paths are both analytically tractable and easy to compute using an almost shortest path algorithm.

**Definition 4.** A weakly simple path $(x_1,...,x_{r+1})$ is a path where for all integers $i \in [1..r - 1]$, $x_i \neq x_{i+2}$. Furthermore, we denote the number of weakly simple paths of length $r$ from node $s$ to $t$ as $WSP_r(s,t)$.

In the following lemma, we demonstrate how the formula from Lemma 4 simplifies for computing the probability that a weakly simple path exists.
Lemma 5. Given a weakly simple path, then the first node in any repeating edge block in the path either appears in the interior of a new edge block or is the first node in the path. Alternatively in the language of Lemma 4 for a weakly simple path, \( R_2 = \emptyset \).

Proof. Suppose there exists a weakly simple path \((x_1, \ldots, x_{r+1})\) that violates the lemma and denote the first node in the path, \( x_i \), that is the first node in a repeating edge block that does not appear in the interior of a new edge block and is not the first node in the path. It follows that \((x_{i-1}, x_i)\) is a new edge and that \((x_i, x_{i+1})\) has appeared elsewhere. Once we prove that \( x_i \) cannot appear earlier in the path, it will follow that \( x_{i+1} = x_{i-1} \) and that the path will not be weakly simple, a contradiction.

Suppose that \( x_i \) has appeared earlier in the path. \( x_i \) cannot be the first node in the path or part of the interior of a new edge block. Consequently, \( x_j = x_i \) for some \( j < i \) and that \( x_j \) occurs in another repeating edge block. From Lemma 3, this would imply that there exists an \( x_k = x_j = x_i \), where \( x_k \) appears at the beginning of an earlier repeating edge block. But then this would imply that the first contradiction in the path occurs with \( x_k \) and hence \( x_i \) is not the first node in the path that violates the property stated in the lemma. Consequently, \( x_i \) cannot appear earlier in the path. \( \square \)

Now that we have demonstrated that for a weakly simple path, the first node in a repeating edge block appears either in the interior of a new edge block or is the first node in a path, we can invoke Lemma 4 to bound the expected number of weakly simple paths between two nodes.

Theorem 3. For the Chung-Lu random graph model, define \( S_2 = \sum_{i=1}^{n} d_i^2 \) and \( d_{\text{max}} = \max_i d_i \). Let \( S_2 > S \) and consider the expected number of weakly simple paths of length \( r \) from node \( s \) to \( t \), \( E[\text{WSP}_r(s,t)] \), where \( s \neq t \). If \( 2r < \frac{S_2}{S} \), then

\[
E[\text{WSP}_r(s,t)] \leq \frac{p_{st}(\frac{S_2}{S})^{r-1}}{1 - \frac{S_2}{S}} \exp\left(\frac{(2r \frac{S_2}{S})^2 p_{\text{max}}}{1 - \frac{2S_2}{S}}\right).
\]

Proof. The challenge in bounding the expected number of weakly simple paths of length \( r \) comes from the issue that a path may visit the same edge multiple times. As a result, we define the indicator random variable \( X_{(u,v)} \) to be 1 if the edge \((u,v)\) exists and 0 otherwise. For simplicity let \( i_0 = s \) and \( i_r = t \). Define a set \( B \subset \mathbb{N}^{r-1} \), where \( i \in B \) if for each \( j \in [1..r-1] \), \( i_j \in [1..r] \) and for all \( j \in [2..r] \), \( i_j \neq i_{j-1} \). (Alternatively, if \( i \in B \), then \( i \) corresponds to a weakly simple path from \( s \) to \( t \) that could exist in the graph.) We then have that
The K Shortest Paths Problem

\[ E[\text{WSP}_r(s,t)] = \sum_{i \in \mathcal{B}} \text{Pr}(\Pi_{j=0}^{r-1} X_{(i_j,i_{j+1})} = 1) = \sum_{i \in \mathcal{B}} \text{Pr}(X_{(i_0,i_1)} = 1) \Pi_{j=1}^{r-2} \text{Pr}(X_{(i_j,i_{j+1})} = 1) \Pi_{k=0}^{j-1} X_{(i_k,i_{k+1})} = 1), \quad (6) \]

where \( \Pi_{j=0}^{r-1} X_{(i_j,i_{j+1})} = 1 \) implies that there is a path (of length \( r \)) from \( i_0 = s \) to \( i_r = t \). Note that \( \text{Pr}(X_{(i_j,i_{j+1})} = 1) = 1 \) if \((i_j, i_{j+1}) = (i_k, i_{k+1}) \) or \((i_j, i_{j+1}) = (i_{k+1}, i_k) \) for some \( k \in [0, j-1] \) and \( \text{Pr}(X_{(i_j,i_{j+1})} = 1) \Pi_{k=0}^{j-1} X_{(i_k,i_{k+1})} = 1) = p_{i_j,i_{j+1}} \) otherwise by independence.

Now to prove the upperbound, we will modify the order in which we condition on the remaining edges in the path. More specifically, from (6) we have that

\[ E[\text{WSP}_r(s,t)] = \sum_{i \in \mathcal{B}} \text{Pr}(X_{(i_0,i_1)} = 1) \text{Pr}(X_{(i_{r-1},i_r)} = 1) \Pi_{j=1}^{r-2} \text{Pr}(X_{(i_j,i_{j+1})} = 1) \Pi_{k=0}^{j-1} X_{(i_k,i_{k+1})} = 1), \quad (8) \]

where we process the last edge immediately after the first edge and then resume the normal order for conditioning on the remaining edges in the path. In particular it will be helpful to assume that \( \text{Pr}(X_{(i_{r-1},i_r)} = 1) = 1 \) if \( (i_{r-1}, i_r) \) is not an edge we have visited before. Denote all such paths of length \( r \) from node \( s = i_0 \) to node \( t = i_r \) as \( \text{WSP}_r(s,t) \). We will argue that for \( r \geq 2 \),

\[ \text{WSP}_r(s,t) \leq \text{WSP}_{r-1}(s,t) + \text{WSP}_{r-2}(s,t), \quad (9) \]

where applying the (9) to itself iteratively yields the inequality,

\[ \text{WSP}_r(s,t) \leq \text{WSP}_1(s,t) + \sum_{m=2}^{r} \text{WSP}_m(s,t), \quad (10) \]

Consequently, to derive a formula for the expected number of paths of length \( r \) from node \( s \) to \( t \), it suffices to construct a formula for the expected number of paths of length \( r \) from node \( s \) to node \( t \), where the first edge is not the same as the last edge. (Note that computing the expected number of paths of length 1 is precisely the probability that nodes \( s \) and \( t \) are neighbors).

To show (9), consider all (weakly simple) paths of length \( r \) where the last edge is identical to the first edge. It then follows that the path must be of the form \( s, i_1, ..., i_{r-1}, t \) as the first and last nodes in the path must be \( s \) and \( t \) respectively. Furthermore, since the first edge and last edge are identical and by assumption \( s \neq t \), it follows that \( i_{r-1} = s \) and \( i_1 = t \). Hence all
paths where the last and first edges are identical are of the form, $s, t, ..., s, t$. Assuming that an edge from node $s$ to $t$ exists, the number of such paths is precisely $WSP_{r-2}(t, s)$. But since this is an undirected graph we have that $WSP_{r-2}(t, s) = WSP_{r-2}(s, t)$, which proves (9).

Since the first and last edges cannot be repeating edges, we can now invoke Lemma 4 to compute the probability that a given path exists. Define $k_0$ to be the number of new edges. Let $k_i$ be the number of repeating edge blocks of length $i$ (where the first node has already been seen before). So to compute $E[WSP_L^r(s, t)]$, we will fix (integer) values for $k_i$, consider all possible arrangements for each of the $k_i$, repeating edge blocks and then consider all possible choices of nodes for the corresponding multisets $N$ and $R_1$ from Lemmas 4 and 6. It then follows that we have the following upperbound,

$$E[WSP_L^r(s, t)] \leq \sum_{k_0 + \sum_{i=1}^{r-2} i(k_i)} \left( \sum_{i=0}^{r-2} k_i \right) \left( \sum_{i=0}^{r-2} k_i \right) \sum_{j_1, ..., j_{|N|}} \frac{d_j d_t}{S} \Pi_{i=1}^{N_j} d_j d_t \Pi_{m=1}^{r-2} (p_{max} (2r)^m)^{k_m},$$

(11)

where the inner sum represents all possible choices of nodes for constructing $N$ and $R_1$ that yield paths with the prescribed number of repeating edge blocks of various lengths. We can then construct an upperbound to (11) by identifying the nodes in $R_1$ that must equal other nodes in the summation and bound the contribution of that node’s expected degree by $d_{max}$. This yields the following inequality.

$$E[WSP_L^r(s, t)] \leq \sum_{k_0 + \sum_{i=1}^{r-2} i(k_i)} \left( \sum_{i=0}^{r-2} k_i \right) \sum_{j_1, ..., j_{|N|}} \frac{d_j d_t}{S} \Pi_{i=1}^{N_j} d_j d_t \Pi_{m=1}^{r-2} (p_{max} (2r)^m)^{k_m},$$

(12)

where there are $r - 2 - \sum_{i=1}^{r-2} (i + 1)(k_i)$ nodes in $N$ from Lemma 4. Note that the contribution from the term $\Pi_{(j,k)\in R_1} d_j d_t$ is replaced by $p_{max}$ as both nodes in $R_1$ appear elsewhere by definition. Furthermore, the contribution from the last node that appears in each pair in $R_1$ is replaced by a $d_{max}$ as that node appears elsewhere. We also note that for an arbitrary repeating edge block of length $l$, there are at most $(2r)^l$ choices for filling in the repeating edge block.

Summing over all possible choices of nodes and using the fact that \( \sum_{i=1}^{\sum_{i=1}^{r-2} k_i} \sum_{j_1, ..., j_{|N|}} \frac{d_j d_t}{S} \Pi_{i=1}^{N_j} d_j d_t \Pi_{m=1}^{r-2} (p_{max} (2r)^m)^{k_m}, \) yields the following upperbound for (11).
The K Shortest Paths Problem

We can then bound above (13) by removing the constraint under the summation by letting
\[ k_0 = r - \sum_{i=1}^{r-2} i k_i \]
and letting the other \( k_i \) take on any non-negative integer value.

Finally applying (14) to (10) yields the result.

From Lemma 2 and Theorem 3 we know that the number of simple or non-simple paths grows exponentially in terms of path length. In particular, we know that for a flexible range of parameters in Chung-Lu random graph model, the diameter is no greater than \( O(\log n) \), [11]. And since the number of paths grows exponentially (in terms of length), that for practical application, the length of the almost shortest paths will also be no greater than \( O(\log n) \). Consequently, we are interested in the ratio of the number of simple paths and non-simple paths, where the length \( r = O(\log n) \). To attain such results, we will need bounds on the variance for the number of simple and non-simple paths and hence we have the following theorem.

**Theorem 4.** Consider a collection of sources \( S \) and for each \( s \in S \), let \( T_s \) be a collection of targets for the source \( s \). Denote \( SP_r(S, T) = \sum_{s \in S} \sum_{t \in T_s} SP_r(s, t) \).

Then

\[
\text{var}(SP_r(S, T)) \leq E(SP_r(S, T))[1 + \frac{S_2}{S} \sum_{s \in S} \sum_{t \in T_s} \frac{d_s d_t}{S} (\exp\left(\frac{4r^2 p_{\max} S_2^2}{1 - \frac{S_2}{S}}\right) - 1) + \frac{S_2^{r-2}}{1 - \frac{S_2}{S}} \exp\left(\frac{4r^2 p_{\max} S_2^2}{1 - \frac{S_2}{S}}\right)]2(r+1)d_{max}(\sum_{s \in S} \frac{d_s}{S} + \max_{t \in T_s} \sum_{t \in T_s} \frac{d_t}{S} + 2(r+1)d_{max})].
\]

(15)
The K Shortest Paths Problem

Proof. We first define indicator variables, $X_\alpha$, that denote the existence of a path $\alpha$ that contributes to $SP_r(S, T)$. We will abuse notation slightly and denote $SP_r(S, T)$ as a set consisting of all such paths $\alpha$. Let $X_\alpha$ be an indicator variable that equals 1 if the path $\alpha$ exists and 0 otherwise. Furthermore, denote all paths $\beta$ in $SP_r(S, T)$ such that $X_\alpha$ and $X_\beta$ are dependent random variables as $D(\alpha)$. Then it follows that

$$\text{var}(SP_r(S, T)) = \text{var}(\sum_{\alpha \in SP_r(S, T)} X_\alpha) = \sum_{\alpha \in SP_r(S, T)} \text{var}(X_\alpha) + \sum_{\alpha \in SP_r(S, T)} E(X_\alpha X_\beta) \leq \sum_{\alpha \in SP_r(S, T)} E(X_\alpha^2) + \sum_{\alpha \in SP_r(S, T)} \sum_{\beta \in SP_r(S, T) \cap D(\alpha)} E(X_\alpha X_\beta) = E(SP_r(S, T)) + \sum_{\alpha \in SP_r(S, T)} \sum_{\beta \in SP_r(S, T) \cap D(\alpha)} E(X_\alpha X_\beta),$$

(16)

where the last equality follows from the fact that $X_\alpha^2 = X_\alpha$ as $X_\alpha$ is an indicator random variable. From (16), we now seek a bound for $\sum_{\alpha \in SP_r(S, T)} \sum_{\beta \in SP_r(S, T) \cap D(\alpha)} E(X_\alpha X_\beta)$. Note that

$$E(X_\alpha X_\beta) = Pr(X_\alpha = 1 \text{ and } X_\beta = 1) = Pr(X_\beta = 1|X_\alpha = 1) Pr(X_\alpha = 1).$$

Eventhough $X_\beta$ corresponds to the existence of a simple path, since we already know of the existence of edges, and since we are conditioning on $X_\alpha = 1$ we (slightly) ammend our definition of a repeating edge to be an edge that we already know exists. We consider four cases, where we stipulate whether the first and last edges are repeating or new edges. 

Case 1: The first and last edges are new edges. We denote all such $\beta \in D(\alpha)$ that have his property to also be in the set $C_1(\alpha)$. We will argue that

$$\sum_{\beta \in D(\alpha) \cap C_1(\alpha)} E(X_\alpha X_\beta) \leq Pr(X_\alpha = 1) \frac{S_r}{S} \sum_{s \in S} \sum_{t \in T_s} d_s d_t \left( \exp \left( \frac{4r^2 p_{\max} S^2}{1 - \frac{S}{S_r}} \right) - 1 \right) \Rightarrow$$

$$\sum_{\alpha \in SP_r(S, T)} \sum_{\beta \in D(\alpha) \cap C_1(\alpha)} E(X_\alpha X_\beta) \leq E(SP_r(S, T)) \frac{S_r}{S} \sum_{s \in S} \sum_{t \in T_s} d_s d_t \left( \exp \left( \frac{4r^2 p_{\max} S^2}{1 - \frac{S}{S_r}} \right) - 1 \right),$$

(17)

where the second line follows from the first as $\sum_{\alpha \in SP_r(S, T)} Pr(X_\alpha = 1) = E(SP_r(S, T))$. As the proof for (17) is similar to the proof strategy for Theorem 3, we provide a sketch for the argument. By fixing the lengths for the number of
repeating edge blocks and considering all possible configurations, from Lemma 4 and repeating an argument analogous to the derivation of (11) we have the upperbound

\[
Pr(X_\alpha = 1) \sum_{\beta \in D(\alpha) \cap C_1(\alpha) \atop \beta \neq \alpha} Pr(X_\beta = 1 | X_\alpha = 1) \leq Pr(X_\alpha = 1) \sum_{k_0 + \sum_{i=1}^{r-2} k_i = r \atop \forall i \in [0, r-2], k_i \in [0, r]} \left( \sum_{i=0}^{r-2} k_i \right) \sum_{N, R_1 \atop \beta_i \in S, \beta_{r+1} \in T, \beta_i \neq S} \frac{d_{\beta_i} d_{\beta_{r+1}}}{S} \Pi_{j \in N} \frac{d_j^2}{S} \Pi_{(l,m) \in R_1} \frac{d_l d_m}{S},
\]

where for \( i \geq 1 \), \( k_i \) corresponds to the number of repeating edge blocks of length \( i - 1 \) and \( k_0 \) corresponds to the number of new edges. In (18), since \( X_\beta \) and \( X_\alpha \) are dependent random variables, there must be at least one repeating edge and hence at least one repeating edge block. As a result, we require that \( \sum_{i \geq 1} k_i > 0 \). After selecting choices for the number of repeating edge blocks of various lengths, we sum over all possible choices of nodes for \( N \) and \( R_1 \), the multisets of nodes in the interior of a new edge block and the nodes at the beginning and end of a repeating edge block respectively. (Note that since we are dealing with simple paths, the only way to encounter a repeating edge is for those nodes to appear earlier in the simple path that corresponds to \( X_\alpha \).) We then use a similar argument in the proof of Theorem 3 to derive an upperbound for (18). In particular we bound the expected degree of each node that appears in \( R_1 \) by \( d_{\text{max}} \) and we note that there are at most \( 2(r+1) \) choices for deciding which edges appear in an entire repeating edge block for a simple path as illustrated in Figure 5.

More generally, in a simple path \( P_* \), we cannot revisit a node. So for a repeating edge block to exist in a simple path (given that we know another simple path \( P \) of length \( r \) exists), identify the first node in the repeating edge block. There are at most \( r + 1 \) choices for this first node as this node must appear in \( P \); denote this node as \( x \). The next node in the repeating edge block must be a neighbor of that node \( x \) in \( P \); call it \( y \). As \( P \) is simple, we have at most two such choices for neighbors of \( x \). Finally, the remaining choices are uniquely determined as \( y \) has at most two neighbors, \( x \) and \( z \). We cannot choose \( x \) as that would make \( P_* \) not simple. So we only have one choice; node \( z \). Proceeding in this way, it follows that the remaining nodes are uniquely determined as \( P_* \) must be simple. As a result, this yields an upperbound of \( 2(r+1) \) choices for deciding which edges appear in a repeating edge block.

We then have the following upperbound,
Figure 5: An illustration for the number of ways that we can fill in the nodes in a repeating edge block (marked in red) for a simple cycle (path), with the knowledge that a particular simple cycle (or path) of length 5 exists (top left). Once we decide on the nodes in the interior of the new edge block, we only have 6 choices for the first node in the repeating edge block and 2 choices for the second node in the repeating edge block. Once these choices are made, the rest of the nodes in the repeating edge block are uniquely determined as the cycle must be simple. More generally, if we know that a simple path of length 2r exists, then there will be 4r possible choices for filling in a given repeating edge block. We readily note that knowing the nodes in the interior of the new edge blocks, (8 and 9), in conjunction with the nodes in the repeating edge blocks, (3,2,1 and 6) uniquely determines the cycle.

\[ Pr(X_\alpha = 1) = \sum_{k_0 + \sum_{i>0} i k_i = r}^{\sum_{i=0}^{r-2} k_i} \Pi_{i=1}^{r-2} \frac{(2(r+1)p_{max})^{k_i}}{k_i!} \cdot \frac{r^{r-1-\sum_{i=1}^{r}(i+1)k_i}}{S} \cdot \sum_{\beta_1 \in S} \sum_{\beta_{r+1} \in T_{\alpha}} d_{\beta_1} d_{\beta_{r+1}} \cdot \frac{S}{S} \cdot (19) \]

Then by setting \( k_0 = r - \sum_{i>0} i k_i \), letting the \( k_i \) be any non-negative integer and applying the inequality \( 2r(r + 1) \leq 4r^2 \) for \( r \geq 1 \), we have the following
upperbound.

\[
Pr(X_\alpha = 1) \frac{S_2^{r-1}}{S} \sum_{\beta_1 \in S} \frac{d_{\beta_1} S_{\beta_{r+1}}}{S} \left( \prod_{i=1}^{\infty} \left( \sum_{k_i=0}^{s_i-2} k_i \right) \Pi_{i=1}^{r-2} \left( \frac{4r^2 p_{\max} S_2^{s_i+1}}{k_i!} \right) \right) - 1 \leq \Pr(X_\alpha = 1) \frac{S_2^{r-1}}{S} \sum_{\beta_1 \in S} \frac{d_{\beta_1} d_{\beta_{r+1}}}{S} \left( \exp \left( \frac{4r^2 p_{\max} S_2^2}{1 - \frac{S_2}{S}} \right) - 1 \right). \tag{20}
\]

This completes Case 1.

Case 2: The last edge is repeating and the first edge is not. We denote all such \( \beta \in D(\alpha) \) that have his property to also be in the set \( C_2(\alpha) \). Let \( LB \) denote the length of the last repeating edge block. Analogous to (17), we then claim that since there are at most \( 2(r+1) \) possible choices for filling in the last repeating edge block that,

\[
\sum_{\beta \in D(\alpha) \cap C_2(\alpha)} E(X_\alpha X_\beta) \leq Pr(X_\alpha = 1) \cdot \left( \sum_{LB=1}^{r-1} \frac{S_2^{r-1-LB}}{S} \sum_{s \in S} \frac{2(r+1) d_{\max} S_2}{S} \Pi_{j \in N \Pi_{(l,m) \in R_i} d_{l} d_{m}} \right) \sum_{k_0, \ldots, k_{r-2}} k_i \sum_{i=1}^{LB} k_i = r - LB \tag{21}
\]

where we apply Lemma 4 to compute the probability of a path once we fix the number of repeating edge blocks of length \( i \), identify the nodes in \( N \) and \( R_i \) and fix the length of last repeating edge block, \( LB \). Proceeding similarly to (17), we get the following upperbound that,

\[
Pr(X_\alpha = 1) \frac{S_2^{r-2}}{1 - \frac{S_2}{S}} \sum_{s \in S} \frac{2(r+1) d_{\max} S_2}{S} \exp \left( \frac{4r^2 p_{\max} S_2^2}{1 - \frac{S_2}{S}} \right). \tag{22}
\]

And it follows that by summing over all \( \alpha \) that,

\[
\sum_{\alpha \in SP(S, T) \beta \in D(\alpha) \cap C_1(\alpha)} E(X_\alpha X_\beta) \leq E(SP(S, T)) \frac{S_2^{r-2}}{1 - \frac{S_2}{S}} \sum_{s \in S} \frac{2rd_{\max} S_2}{S} \exp \left( \frac{4r^2 p_{\max} S_2^2}{1 - \frac{S_2}{S}} \right). \tag{23}
\]

Case 3: The first edge is repeating and the last edge is not. We denote all such \( \beta \in D(\alpha) \) that have his property to also be in the set \( C_3(\alpha) \). Let \( FB \)
denote the length of the first repeating edge block. Note that there are at most 
\(2(r + 1)\) choices for filling the first repeating edge block. As the first node, \(s\), in the first repeating edge block uniquely identifies a set \(T_s\) of possible targets, to bound \((23)\) we will need to sum over all possible targets \(T_s\). Instead we will just consider the set \(T_{s_\alpha}\) of targets that maximizes the probability that such a path exists to simplify the analysis. This then yields,

\[
\sum_{\beta \in D(\alpha) \cap C_4(\alpha)} E(X_{\alpha}X_{\beta}) \leq \Pr(X_{\alpha} = 1).
\]

\[
\sum_{\substack{FB, LB \in [1, r-1] \\
 k_0 + \sum_{i=1}^{r-2} k_i = r - FB - LB}} \left(\sum_{\max s \in S} \left(k_0, \ldots, k_{r-2}\right)\right) \sum_{t \in T_s} \max_{R, r} 2(r + 1)d_{\text{max}}d_t \frac{d_j}{S} \Pi_{l \in N_r} \frac{d_l^2}{S} \Pi_{(l, m) \in R} \frac{d_m}{S} \leq
\]

\[
\Pr(X_{\alpha} = 1) \sum_{FB = 1}^{r-1} S_2 \left(1 - \frac{S_2}{\sqrt{S}}\right)^{r-1-FB} \frac{4r^2p_{\text{max}} S_2^2}{1 - \frac{S_2}{\sqrt{S}}} \leq
\]

\[
\Pr(X_{\alpha} = 1) \sum_{FB = 1, LB = 1}^{r-1} S_2 \left(1 - \frac{S_2}{\sqrt{S}}\right)^{r-1-FB-LB} \frac{4r^2p_{\text{max}} S_2^2}{1 - \frac{S_2}{\sqrt{S}}} \leq
\]

Summing over all possible choices for \(X_\alpha\) completes Case 3.

**Case 4: The first edge and the last edge are both repeating.** We denote all such \(\beta \in D(\alpha)\) that have his property to also be in the set \(C_4(\alpha)\). Let \(FB\) denote the length of the first repeating edge block and \(LB\) is the length of the last repeating edge block. Note that there are at most \(2(r + 1)\) choices for filling the first and last repeating edge blocks. We then have that,

\[
\sum_{\beta \in D(\alpha) \cap C_4(\alpha)} E(X_{\alpha}X_{\beta}) \Pr(X_{\alpha} = 1) \leq
\]

\[
\sum_{\substack{FB, LB \in [1, r-1] \\
 k_0 + \sum_{i=1}^{r-2} k_i = r - FB - LB}} \left(\sum_{\max s \in S} \left(k_0, \ldots, k_{r-2}\right)\right) \sum_{t \in T_s} \max_{R, r} 2(r + 1)d_{\text{max}}d_t \frac{d_j}{S} \Pi_{l \in N_r} \frac{d_l^2}{S} \Pi_{(l, m) \in R} \frac{d_m}{S} \leq
\]

\[
\Pr(X_{\alpha} = 1) \sum_{FB = 1}^{r-1} S_2 \left(1 - \frac{S_2}{\sqrt{S}}\right)^{r-1-FB-LB} \frac{4r^2p_{\text{max}} S_2^2}{1 - \frac{S_2}{\sqrt{S}}} \leq
\]

Summing over all possible choices for \(\alpha\) proves the bound for Case 4. Applying the bounds from Cases 1-4 to \((16)\) yields the result.

Analogous to Theorem 4, we now consider the variance for the number of weakly simple paths, so that we can show that the ratio of the number of simple paths and paths of the same length should converge to 1.
Theorem 5. Consider a collection of sources $S$ and for each $s \in S$, let $T_s$ be a collection of targets for the source $s$. Denote $WSP_r(S, T) = \sum_{s \in S} \sum_{t \in T_s} WSP_r(s, t)$. Then

$$\text{var}(WSP_r(S, T)) \leq E(WSP_r(S, T)) \frac{S^2 r^{-2} \exp(\frac{4r^2 p_{max} S^2}{1 - 4r S^2})}{1 - 4r S^2} + \frac{S^2 r^{-1}}{S} \sum_{s \in S} \sum_{t \in T_s} \frac{d_s d_t}{S} \left(\exp(\frac{4r^2 p_{max} S^2}{1 - 4r S^2}) - 1\right) + E(WSP_r(S, T)).$$

(26)

Proof. The proof is nearly identical to Theorem 4. Notice that for both simple paths and weakly simple paths when we invoke Lemma 4, $R_2 = \emptyset$. The only difference is that when bounding the variance for the number of simple paths of length $r$, there were at most $2(r + 1)$ choices for filling in a repeating edge block of length $k$ if we know about the existence of another simple path of length $r$. In contrast for weakly simple paths, there are at most $(2r)^k$ choices for filling in a repeating edge block of length $k$ if we know about the existence of another weakly simple path of length $r$. (There are at most $2r$ edges in the two weakly simple paths of length $r$ and hence there are trivially at most $(2r)^k$ ways of filling in a repeating edge block of length $k$.) 

From Theorems 4 and 5, we can deduce asymptotic results regarding the distribution of the number of simple and weakly simple paths under the Chung-Lu random graph model by formulating conditions such that the variance approaches 0. Then by invoking Lemma 2 and Theorem 3, it follows that (under certain conditions) the expected number of weakly simple paths equals the expected number of simple paths. This provides a derivation of our main result.

Corollary 1. Suppose we possess a collection of sources $S$ and for each $s \in S$, a collection of targets $T_s$, where $s \notin T_s$. Let $r$ be a parameter corresponding to the lengths of the paths. Consider two fixed positive constants $c$ and $C$ and a sequence of expected degree sequences (indexed by $n$) that has the following properties:

- $cd_{max} \leq \sum_{s \in S} d_s$,
- For all $s \in S$, $c \leq \frac{\sum_{t \in T_s} d_t}{\sum_{s \in S} d_s} \leq C$,
- $\lim_{n \to \infty} r/\frac{S^2}{S} \to 0$,
- $\lim_{n \to \infty} E(WSP_r(S, T)) \to \infty$, 

• \( \lim_{n \to \infty} r^2 p_{\text{max}} / \frac{S^2}{S} \to 0. \)

Then asymptotically almost surely,

\[
\frac{SP_r(S, T)}{WSP_r(S, T)} \to 1.
\]

**Remark 1:** Note that the third and fifth conditions in the list ensure that the \( E(SP(S, T)) \) and \( E(WSP(S, T)) \) approach the same value. Furthermore, the first four conditions in Corollary 1 guarantee that \( \text{var}(SP(S, T)) \) and \( \text{var}(WSP(S, T)) \) are insignificant relative to \( E(SP(S, T))^2 \) and \( E(WSP(S, T))^2 \) respectively. These results in conjunction demonstrate that the number of weakly simple paths and simple paths are tightly concentrated around their expected values and that their expected values are the same, completing the proof.

For simplicity, we provided easy to interpret conditions so that the variance terms from Theorems 4 and 5 would approach 0. Consequently, it follows that a more delicate approach to finding conditions for the variance going to 0 would yield a generalization of Corollary 1 that holds under weaker assumptions.

**Remark 2:** From Corollary 1, in order for the ratio of the number of weakly simple paths and simple paths of the same length to equal 1 with high probability, we must consider a sufficiently 'large' collection of sources and targets. In particular, Corollary 1 holds even if there is precisely one source and one target, as long as the expected degree of the source and target is proportional to \( d_{\text{max}} \).

We can then consider extensions of Corollary 1 where nodes \( s \) and \( t \) have an arbitrary expected degrees, such that \( WSP_r(s, t) \) approximates \( SP_r(s, t) \), by projecting all nodes in an \( i \)th neighborhood around \( s \) or \( t \) into a single node with expected degree proportional to \( d_{\text{max}} \). Hence, Corollary 1 suggests that asymptotically almost surely the number of weakly simple paths and simple paths between the neighborhoods of two arbitrary nodes \( s \) and \( t \) approaches 1.

Now that we have shown that the number of weak simple paths (of prescribed length) approximates the number of simple paths for a wide range of parameters under the Chung-Lu random graph model, we illustrate how to compute almost shortest weakly simple paths using the algorithm from Section 2. In particular, we computed paths bounded by length \( D \) from node \( s \) to \( t \) by considering a partial path \( (x_m, x_{m-1}, \ldots, x_1, x_0) \), where \( x_0 = t \), measuring the distance traveled so far \( D_s = \sum_{i=0}^{m-1} d(x_{m-i}, x_{m-i}-1) \) and adding a new node to the partial path \( x_{m+1} \) if \( x_{m+1} \) is a neighbor of \( x_m \) and \( d(s, x_{m+1}) + d(x_{m+1}, x_m) \leq D - D_s \). Iteratively adding nodes in this way would yield a path from node \( s \) to node \( t \) with length at most \( D \). Consequently to find sufficiently short weakly simple paths, it will be helpful to compute the minimum length of a weakly simple path between two nodes under some constraints. This motivates the
**Definition 5.** For any edge \((a, b)\) in the graph \(G\), denote \(d_{WSP}(a \rightarrow b, t)\) as the length of the shortest weakly simple path of the form \((a, b, ..., t)\).

Now to efficiently compute \(d_{WSP}(a \rightarrow b, t)\), it will be helpful to consider a (directed) shortest path tree \(T\) for the graph \(G\). [As \(T\) is directed, note that if the edge \((a, b)\) is in \(T\), this does not imply that \((b, a)\) is in \(T\). Furthermore, given an edge \((a, b)\) in a directed graph, we read the edge as going from node \(a\) to node \(b\); that is, \(a\) is an incoming neighbor of \(b\).] In particular, if \((b, a)\) is an edge in \(T\), then the shortest path of the form \((a, b, ..., t)\) is \((a, b, a, ..., t)\), which would not be weakly simple. We claim that we have the following recursion for computing \(d_{WSP}(a \rightarrow b, t)\) when \((b, a)\) is an edge that appears in \(T\).

**Lemma 6.**
\[
d_{WSP}(a \rightarrow b, t) = d(a, b) + \min_{n \in Nbr(b)} \begin{cases} 
  d(b, n) + d(n, t) & \text{if } n \neq a \text{ and } (n, b) \notin T \\
  d_{WSP}(b \rightarrow n, t) & \text{if } n \neq a \text{ and } (n, b) \in T \\
  \infty & \text{if } n = a
\end{cases}
\]

**Proof.** To find the length of the shortest weakly simple path from \(a\) to \(t\) where the first edge is \((a, b)\), we can look at the lengths of the shortest weakly simple path of the form \((a, b, n, ..., t)\), where \(n\) is a neighbor of \(b\) and we minimize over all choices for \(n\).

If \(n = a\), then there is no such weakly simple path of the form \((a, b, n, ..., t)\) and we define the length as \(\infty\).

Alternatively, if \(n \neq a\) and \((n, b) \notin T\), then it follows that the shortest path of the form \((b, n, ..., t)\) is a simple path (and hence weakly simple). In particular the length of the path is \(d(b, n) + d(n, t)\).

Finally, if \((n, b) \in T\) and \(n \neq a\), then by definition we can denote the length of the shortest weakly simple path of the form \((a, b, n, ..., t)\) as \(d(a, b) + d_{WSP}(b \rightarrow n, t)\).

**Remark:** In the event that \((b, a)\) is not an edge in \(T\), then \(d(a \rightarrow b, t) = d(a, b) + d(b, t)\). Otherwise, (as mentioned previously) if \((b, a)\) is an edge in \(T\), then Lemma 6 is especially helpful for computing the weakly simple path distances, \(d_{WSP}(a \rightarrow b, t)\). To efficiently solve for the weakly simple path distances, we start by using Lemma 6 to solve for \(d_{WSP}(a \rightarrow b, t)\), where \(b\) has 0 incoming edges in \(T\). We can subsequently solve for the remaining weakly simple path distances by identifying nodes \(b\) such that for all incoming neighbors \(n\) in \(T\), \(d_{WSP}(b \rightarrow n, t)\) is known and invoke Lemma 6 to compute the weakly
simple path distance. Continuing this process yields an $O(m + n)$ algorithm for computing the distances of the shortest weakly simple paths between two nodes, where the first edge is fixed.

We can then generalize Algorithm 1 to find only weakly simple paths in the following manner. Initially, we compute the lengths of the shortest weakly simple paths $d_{WSP}(a \rightarrow b, t)$, for all pairs of edges $(a, b)$. Subsequently, for each node $a$, we sort $a$’s neighbors $n_i$ in the adjacency list according to $d_{WSP}(a \rightarrow n, t)$. Then once we have determined that there exists a weakly simple path of length bounded by $D$ of the form $(x_0, ..., x_m, ..., t)$, we can determine if there exists a weakly simple path of length bounded by $D$ of the form $(x_0, ..., x_m, x_{m+1}, ..., t)$, where $x_{m+1}$ is a neighbor of $x_m$ by checking that $x_{m+1} \neq x_{m-1}$ and that $d_{WSP}(x_m \rightarrow x_{m+1}, t) \leq D - \sum_{i=0}^{m-1} d(x_i, x_{i+1})$. It then follows from the analysis of Algorithm 1, that the time complexity for identifying almost shortest weakly simple paths between two nodes is the same for finding almost shortest paths between two nodes.

3.1 Simulations for the Ratio of Paths to Simple Paths

For an undirected graph, the presence of nodes of high degree can influence the ratio of the number of simple paths to non-simple paths, as illustrated in [7, 8]. For this reason, we introduced the notion of weakly simple paths, as they are easy to compute and for a more flexible parameter regime, the number of simple paths asymptotically approximates the number of weakly simple paths of the same length for a sufficiently large collection of sources and targets.

Even so, for many graphs we can approximate the number of simple paths by the number of paths between two nodes. To better understand the relationship, we ran numeric simulations. Intuitively speaking, given a collection of graphs with a fixed number of high degree nodes (to ensure a substantial number of nonsimple paths for sufficiently large $r$), the claim is that graphs associated with a larger $\frac{S^2}{\sigma}$ will result in a larger percentage of simple paths of length $r$; that is, $\frac{S^2}{\sigma}$ is related to the expected number of neighbors of a node and increasing the number of neighbors of a node will result in more ‘new’ (simple) paths.

To justify this claim, we constructed realizations of Chung-Lu graphs with expected degree sequences where we fixed the average degree, varied $\frac{S^2}{\sigma}$, and selected a fixed number of nodes to have expected degree equal to $\sqrt{S}$. In particular, we used a Markov Chain Monte Carlo method similar to [32] for randomly generating expected degree sequences such that any realization has a fixed expected average degree of 8. Expected degree sequences satisfied a specified value for $\frac{S^2}{\sigma}$ and consisted of at least four nodes with an expected degree of $\sqrt{S} = \sqrt{8n}$, where $n$ is the number of nodes. In these simulations, graphs
consisted of 800 nodes. We constructed 100 such expected degree sequences. Subsequently, from each expected degree sequence we constructed a realization from the Chung-Lu random graph model. We then randomly chose 50 pairs of nodes (each with a minimum degree of 5) and calculated the ratio of the number of simple and non-simple almost shortest paths for various lengths. Figure 6 presents clusters of three box plots of the ratio corresponding to realizations from each of these expected degree sequences.

Figure 6: A box plot of the ratio of the number of paths of length \( r \) compared to the number of simple paths of length \( r \). Each cluster of three box plots represents different values of \( r \), minimum distance +2, +3 and +4. Similarly a more rightwards cluster of box plots on the x-axis denotes an increased value of \( \frac{S_2}{S} \) in the expected degree sequence. Note that there appears to be an exponential growth in the number of non-simple paths (relative to the number of paths) as \( r \) increases but that for modest values of \( r \), this quantity is well controlled.

In Figure 6 the graphs corresponding to the last cluster of boxplots has an expected \( \frac{S_2}{S} \) of approximately 20, as opposed to the first cluster of boxplots affiliated with an expected \( \frac{S_2}{S} \) of roughly 12. We can non-rigorously estimate the number of paths of minimum distance +\( k \) by recalling Lemma 2 where the expected number of paths of length \( k \) grows like \( \frac{S_2}{S}^k \). When \( k = 4 \) and \( 10 \leq \frac{S_2}{S} \leq 20 \) this implies that the number of paths calculated is roughly bounded by \( 100,000 \leq \frac{S_2}{S}^4 \leq 160,000 \). Hence it is far more efficient to calculate the number of simple paths of length \( r \) by calculating the number of non-simple paths of length \( r \) (where the ratio of the number of paths to simple paths is roughly 2 when \( k = 4 \)) than using one of the existing algorithms for computing the number of simple paths directly as referenced in the introduction. Furthermore, while the ratio of non-simple paths to simple paths grows as we increase \( r \), in practice we must compute exponentially many paths to see an exponential growth in the penalty for computing both simple and nonsimple paths.
4 Connectivity Simulations in the AS Graph

We now consider an application of the almost shortest (simple) path problem to internet routing. More precisely, we wish to inquire the robustness of the AS Graph (and some random graph models) to an edge deletion process and assess the connectivity of the graph.

Measuring connectivity for this application is rather ambiguous. For example under an edge deletion process [31] construes connectivity between two nodes as a measure that solely depends on the existence of a path of length bounded by some constant multiple of the diameter. Alternatively, one can consider measuring connectivity by requiring the existence of a giant component or dynamical robustness [14, 15, 23, 35].

In this work as dynamical robustness (and path existence) may fail to capture the potential ramifications of the existence of only a modest number of short, viable paths, we would instead like to track the percentage (or number) of surviving almost shortest paths under an edge deletion process, where we delete each edge from the graph independently with probability \( p \). On the left column of Figure 7, we plotted box plots for the percentage of surviving almost shortest paths (y-axis) under an edge deletion process with probability \( p \) denoted on the x-axis. More specifically, we sampled 20 random pairs of nodes, one with 10 edges and another with 12 edges and repeated the edge deletion process 20 times for each of the 20 distinct node pairs with a given \( p \). When considering a collection of almost shortest paths, in practice we only included paths that were at most 3 or 4 edges longer than the path of minimal length. For a particular pair of nodes, a collection of almost shortest paths could consist of more than 50 million paths. In figure 7a, we construct an Erdos-Renyi random graph with average degree chosen to match the AS Graph. Subsequently, in figure 7c, we consider a Chung-Lu random graph with an expected degree sequence chosen to match the AS Graph as well [21, 34]. Finally in figure 7e, we consider a snapshot of the AS Graph from January 2015, where we compiled edges based on route announcements from the Ripe and Route Views data set.

On the right column of Figure 7, we consider five randomly chosen node pairs and plot the median percentage of surviving paths. Two perhaps surprising results emerge from Figure 7. Firstly, for a given \( p \), the median percentage of surviving paths appear to be roughly the same across all pairs of nodes (of sufficiently high degree) in spite of the fact that the existence of two paths under an edge deletion process is often dependent on one another. And secondly, while the Erdos-Renyi graph fails to capture the distribution of the percentages of surviving paths of almost shortest length in the AS Graph, the Chung-Lu
random graph model behaves remarkably similar to the AS Graph and heavily suggests that knowledge of the degree sequence plays a fundamental role in predicting the percentage of surviving almost shortest paths.
5 Conclusions

Identifying almost shortest paths between two nodes arises in numerous applications including internet routing and epidemiology. Since we want to find many almost shortest paths in these real world networks, we would like our algorithm to exploit properties commonly found in these networks. Consequently, we provided a simple algorithm for computing all paths bounded by length $D$ between two nodes in an graph with $m$ weighted edges. In particular, we demonstrated that the space and time complexity is $O(m + kL)$, where $L$ is an upper bound for the number of nodes that appear in any almost shortest path, for graphs that exhibit certain real world network features.

In contrast, for many applications, we want to find almost shortest simple paths, where we cannot visit a node more than once in a path. Since computing almost shortest simple paths can be computationally expensive, we presented a rigorous framework for explaining when we could use a variant of our solution to solve the almost shortest simple paths problem. More specifically, we analyzed the Chung-Lu random graph model, which emulates many of the properties frequently observed in real world networks, and demonstrated in Corollary 1 that for a flexible choice of parameters, the total number of simple paths approximately equals the total number of weakly simple paths of the same length. We also performed numeric simulations illustrating the ratio of the number of paths to simple paths for some realizations of Chung-Lu random graphs.

In an effort to provide rigorous arguments supporting the efficiency of our algorithm for solving the almost shortest simple paths problem on the Chung-Lu random graph model, other questions organically emerged in the process. While in this work we focused primarily on properties of the number of simple paths to weakly simple paths for Chung-Lu random graphs, we could ask about the ratio of simple paths to paths for other random graph models as well.

Finally, we considered an application to internet routing where we would like to assess the quality of the connectivity between two nodes under an edge deletion process. To measure the quality of connectivity we constructed large collections of almost shortest simple paths, consisting of potentially millions of paths, and then observed the number of paths that survive the edge deletion process through simulation. Of particular interest, we found that the edge deletion process on the snapshot of the AS Graph looked remarkably similar to the simulations on realizations in the Chung-Lu random graph model with the appropriate expected degree sequence, further supporting the notion that Chung-Lu random graphs can emulate many of the properties observed in real world networks.

Ultimately to find an efficient solution to the almost shortest path problem
on real world networks, we need to consider the performance of the algorithm on these types of networks. In this work, we not only provided an efficient solution to the almost shortest paths problems in terms of an important parameter of the problem for real world networks, the actual lengths of the paths, but also provided rigorous results relevant to the efficiency of using an almost shortest path algorithm to find the almost shortest simple paths for realizations of the Chung-Lu random graph model, a model that captures many of the qualities empirically observed in real world networks.

6 Acknowledgements

The authors would like to express their gratitude to Rachel Kartch and Rhiannon Weaver (CMU) for the many helpful conversations pertaining to the applications of the $k$ shortest path algorithms in context to internet routing in the Autonomous System graph. Furthermore, the authors would also like to thank Philip Garrison (CMU) for his input in terms of constructing efficient $k$ shortest path algorithms for real world networks.

Copyright 2016 Carnegie Mellon University

This material is based upon work funded and supported by Department of Homeland Security under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center sponsored by the United States Department of Defense.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[Distribution Statement A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.
References

[1] Réka Albert, Hawoong Jeong, and Albert-László Barabási. Error and attack tolerance of complex networks. *nature*, 406(6794):378–382, 2000.

[2] Husain Aljazzar and Stefan Leue. K*: A heuristic search algorithm for finding the k shortest paths. *Artificial Intelligence*, 175(18):2129–2154, 2011.

[3] David A Bader and Kamesh Madduri. Snap, small-world network analysis and partitioning: an open-source parallel graph framework for the exploration of large-scale networks. In *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pages 1–12. IEEE, 2008.

[4] Albert-László Barabási. Scale-free networks: a decade and beyond. *science*, 325(5939):412–413, 2009.

[5] Aaron Bernstein. A nearly optimal algorithm for approximating replacement paths and k shortest simple paths in general graphs. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 742–755. Society for Industrial and Applied Mathematics, 2010.

[6] TH Byers and MS Waterman. Determining all optimal and near-optimal solutions when solving shortest path problems by dynamic programming (in press) operat, 1984.

[7] Claudio Castellano and Romualdo Pastor-Satorras. Thresholds for epidemic spreading in networks. *Physical review letters*, 105(21):218701, 2010.

[8] Claudio Castellano and Romualdo Pastor-Satorras. Topological determinants of complex networks spectral properties: structural and dynamical effects. *arXiv preprint arXiv:1703.10438*, 2017.

[9] Deepayan Chakrabarti and Christos Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM computing surveys (CSUR)*, 38(1):2, 2006.

[10] Deepayan Chakrabarti, Yiping Zhan, and Christos Faloutsos. R-mat: A recursive model for graph mining. In *SDM*, volume 4, pages 442–446. SIAM, 2004.

[11] Fan Chung and Linyuan Lu. The average distances in random graphs with given expected degrees. *Proceedings of the National Academy of Sciences*, 99(25):15879–15882, 2002.
[12] Fan Chung, Linyuan Lu, and Van Vu. Spectra of random graphs with given expected degrees. *Proceedings of the National Academy of Sciences*, 100(11):6313–6318, 2003.

[13] Fan Chung and Mary Radcliffe. On the spectra of general random graphs. *Electronic Journal of Combinatorics*, 18(1):215–229, 2011.

[14] Paolo Crucitti, Vito Latora, Massimo Marchiori, and Andrea Rapisarda. Error and attack tolerance of complex networks. *Physica A: Statistical Mechanics and its Applications*, 340(1):388–394, 2004.

[15] Jordi Duch and Alex Arenas. Effect of random failures on traffic in complex networks. In *SPIE Fourth International Symposium on Fluctuations and Noise*, pages 66010O–66010O. International Society for Optics and Photonics, 2007.

[16] Nick Edmonds, Torsten Hoefler, and Andrew Lumsdaine. A space-efficient parallel algorithm for computing betweenness centrality in distributed memory. In *2010 International Conference on High Performance Computing*, pages 1–10. IEEE, 2010.

[17] D Eppstein. Finding the k shortest paths. In *Foundations of Computer Science, 1994 Proceedings.*, 35th Annual Symposium on, pages 154–165. IEEE, 1994.

[18] David Eppstein. Finding the k shortest paths. *SIAM Journal on computing*, 28(2):652–673, 1998.

[19] Greg N Frederickson. An optimal algorithm for selection in a min-heap. *Information and Computation*, 104(2):197–214, 1993.

[20] Raoul Frijters, Marianne Van Vugt, Ruben Smeets, René Van Schaik, Jacob De Vlieg, and Wynand Alkema. Literature mining for the discovery of hidden connections between drugs, genes and diseases. *PLoS Comput Biol*, 6(9):e1000943, 2010.

[21] Aric Hagberg and Nathan Lemons. Fast generation of sparse random kernel graphs. *PloS one*, 10(9):e0135177, 2015.

[22] Bing He, Jie Tang, Ying Ding, Huijun Wang, Yuyin Sun, Jae Hong Shin, Bin Chen, Ganesh Moorthy, Judy Qiu, Pankaj Desai, et al. Mining relational paths in integrated biomedical data. *PLoS One*, 6(12):e27506, 2011.

[23] Shan He, Sheng Li, and Hongru Ma. Effect of edge removal on topological and functional robustness of complex networks. *Physica A: Statistical Mechanics and its Applications*, 388(11):2243–2253, 2009.
[24] John Hershberger, Matthew Maxel, and Subhash Suri. Finding the k shortest simple paths: A new algorithm and its implementation. *ACM Transactions on Algorithms (TALG)*, 3(4):45, 2007.

[25] Víctor M Jiménez and Andrés Marzal. Computing the k shortest paths: A new algorithm and an experimental comparison. In *International Workshop on Algorithm Engineering*, pages 15–29. Springer, 1999.

[26] Víctor M Jiménez and Andrés Marzal. A lazy version of eppstein’s k shortest paths algorithm. In *International Workshop on Experimental and Efficient Algorithms*, pages 179–191. Springer, 2003.

[27] Naoki Katoh, Toshihide Ibaraki, and Hisashi Mine. An efficient algorithm for k shortest simple paths. *Networks*, 12(4):411–427, 1982.

[28] Jérémie Leguay, Matthieu Latapy, Timur Friedman, and Kavé Salamatian. Describing and simulating internet routes. In *International Conference on Research in Networking*, pages 659–670. Springer, 2005.

[29] Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. Kronecker graphs: An approach to modeling networks. *Journal of Machine Learning Research*, 11(Feb):985–1042, 2010.

[30] Jure Leskovec and Christos Faloutsos. Scalable modeling of real graphs using kronecker multiplication. In *Proceedings of the 24th international conference on Machine learning*, pages 497–504. ACM, 2007.

[31] Eduardo López, Roni Parshani, Reuven Cohen, Shai Carmi, and Shlomo Havlin. Limited path percolation in complex networks. *Physical review letters*, 99(18):188701, 2007.

[32] Xuesong Lu and Stéphane Bressan. Generating random graphic sequences. In *International Conference on Database Systems for Advanced Applications*, pages 570–579. Springer, 2011.

[33] W Matthew Carlyle and R Kevin Wood. Near-shortest and k-shortest simple paths. *Networks*, 46(2):98–109, 2005.

[34] Joel C Miller and Aric Hagberg. Efficient generation of networks with given expected degrees. In *International Workshop on Algorithms and Models for the Web-Graph*, pages 115–126. Springer, 2011.

[35] Adilson E Motter and Ying-Cheng Lai. Cascade-based attacks on complex networks. *Physical Review E*, 66(6):065102, 2002.
[36] Jukka-Pekka Onnela and Nicholas A Christakis. Spreading paths in partially observed social networks. *Physical Review E*, 85(3):036106, 2012.

[37] Gergely Palla, Imre Derényi, Illes Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

[38] A. Pinar, C. Seshadhri, and T. Kolda. The similarity between stochastic kronecker and chung-lu graph models. *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 127–135, 2014.

[39] Ali Pinar, C Seshadhri, and Tamara G Kolda. The similarity between stochastic kronecker and chung-lu graph models. *CoRR*, abs/1110.4925, 2011.

[40] Mason A Porter and James P Gleeson. *Dynamical systems on networks: A tutorial*, volume 4. Springer, 2016.

[41] José J Ramasco, S Marta, Eduardo López, and Stefan Boettcher. Optimization of transport protocols with path-length constraints in complex networks. *Physical Review E*, 82(3):036119, 2010.

[42] Juan G Restrepo, Edward Ott, and Brian R Hunt. Approximating the largest eigenvalue of network adjacency matrices. *Physical Review E*, 76(5):056119, 2007.

[43] Liam Roditty and Uri Zwick. Replacement paths and k simple shortest paths in unweighted directed graphs. In *International Colloquium on Automata, Languages, and Programming*, pages 249–260. Springer, 2005.

[44] Stefan Savage, Andy Collins, Eric Hoffman, John Snell, and Thomas Anderson. The end-to-end effects of internet path selection. In *ACM SIGCOMM Computer Communication Review*, volume 29, pages 289–299. ACM, 1999.

[45] Yu-Keng Shih and Srinivasan Parthasarathy. A single source k-shortest paths algorithm to infer regulatory pathways in a gene network. *Bioinformatics*, 28(12):i49–i58, 2012.

[46] Piet Van Mieghem. *Graph spectra for complex networks*. Cambridge University Press, 2010.

[47] Vibhav Vineet, Pawan Harish, Suryakant Patidar, and PJ Narayanan. Fast minimum spanning tree for large graphs on the gpu. In *Proceedings of the Conference on High Performance Graphics 2009*, pages 167–171. ACM, 2009.
[48] Xiao Fan Wang and Guanrong Chen. Complex networks: small-world, scale-free and beyond. *IEEE circuits and systems magazine*, 3(1):6–20, 2003.

[49] Jin Y Yen. Finding the k shortest loopless paths in a network. *management Science*, 17(11):712–716, 1971.