Exploring Greedy Routing in Small-World Networks: A Simulation-Based Approach

Chenkai Wang

Email: wangck2022@mail.sustech.edu.cn

Abstract

The small-world phenomenon has significant implications for decentralized navigation in large-scale networks. This work examines greedy routing on two-dimensional augmented lattices, in which long-range links are assigned based on a power-law distribution. A custom simulator is constructed to estimate the expected delivery time T as a function of the clustering exponent r. Simulation results are presented to analyze how variations in r affect routing efficiency, offering empirical insights into the relationship between network structure and navigability.

Keywords: Small World Network, Greedy Algorithm

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Contents

1 Description of Homework 3

- In one dimension, set the minimum value of the chain size to 100,000.
- In two dimensions, set the toroidal lattice to 20,000 x 20,000.

Select one of these options and plot the output clearly. Pseudocode is not required.

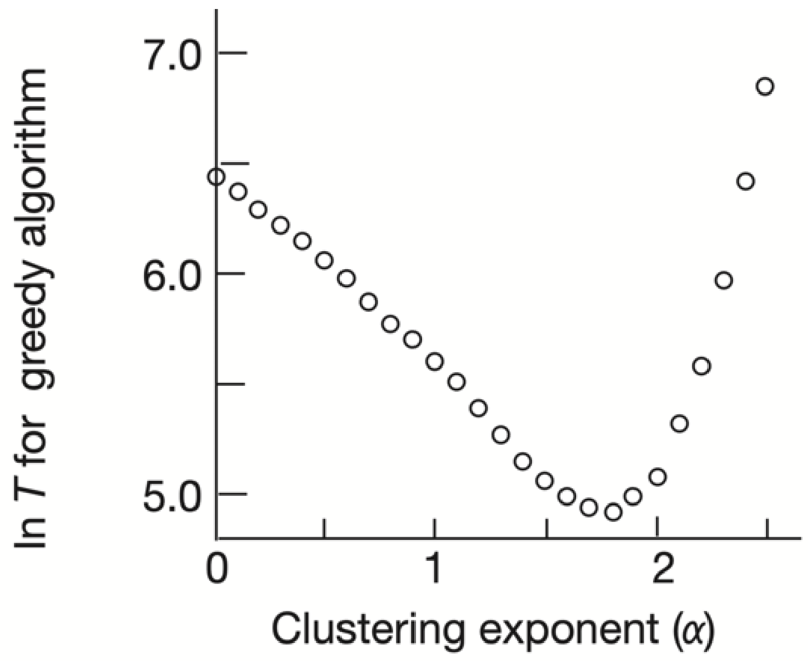


Figure 1: Homework Description

2 Introduction

2.1 Small-World Phenomenon

In 1967, Stanley Milgram conducted a landmark sociological experiment that demonstrated the surprising proximity between any two individuals in a social network [3]. The results revealed that, on average, approximately 5.5 intermediaries were required to connect two randomly selected individuals. This phenomenon is now known as the *small-world effect*. Figure 2 illustrates a hypothetical path connecting such individuals through a minimal sequence of acquaintances.

This empirical observation suggests that short paths are prevalent in real-world networks. An important question then arises: how can such paths be identified algorithmically, especially under decentralized conditions?

2.2 From Real-World Networks to Model Abstractions

A social network can be abstracted as a graph in which each node represents an individual and each edge denotes a mutual acquaintance. In principle, computing the distances between all pairs of individuals would allow for the identification of optimal connection paths. However, constructing and storing such a complete social graph at scale is infeasible due to several factors:

- The immense size and density of real-world networks impose significant storage and computational costs.
- Social relationships are dynamic, rendering static network representations quickly outdated.
- In many applications, the existence and efficiency of a viable connection path are more relevant than the precise distance between any two nodes.

These challenges motivate the use of simplified network models that retain key characteristics of real-world systems while enabling analytical tractability. Ideally, such models should (i) be governed by construction rules that are simple enough to allow for analytical or computational scaling, and (ii) preserve essential small-world properties that support efficient navigation using only local information.

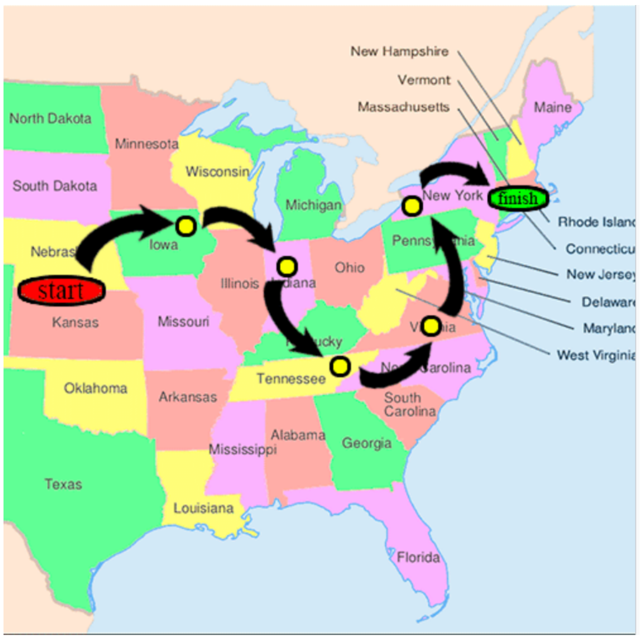


Figure 2: A possible path in the small-world experiment.

2.3 Research Motivation and Objectives

This study focuses on decentralized conditions, where each node makes routing decisions based solely on local information, namely the destination's location and its immediate neighbors. Without access to global knowledge, routing relies on local heuristics, making it scalable and realistic for large-scale networks.

Within this setting, the project investigates how structural parameters, particularly the clustering exponent r, influence routing efficiency in Kleinberg's augmented grid model. The main objectives are twofold: (1) to simulate greedy routing across a range of r values, and (2) to evaluate the expected delivery time (EDT) under varying configurations.

3 Network Construction and Greedy Routing

3.1 Network Model

To enable the analytical study of decentralized routing behavior, we consider a structured network based on a two-dimensional $n \times n$ lattice. Each node is assigned two types of directed edges, as illustrated in Figure 3.

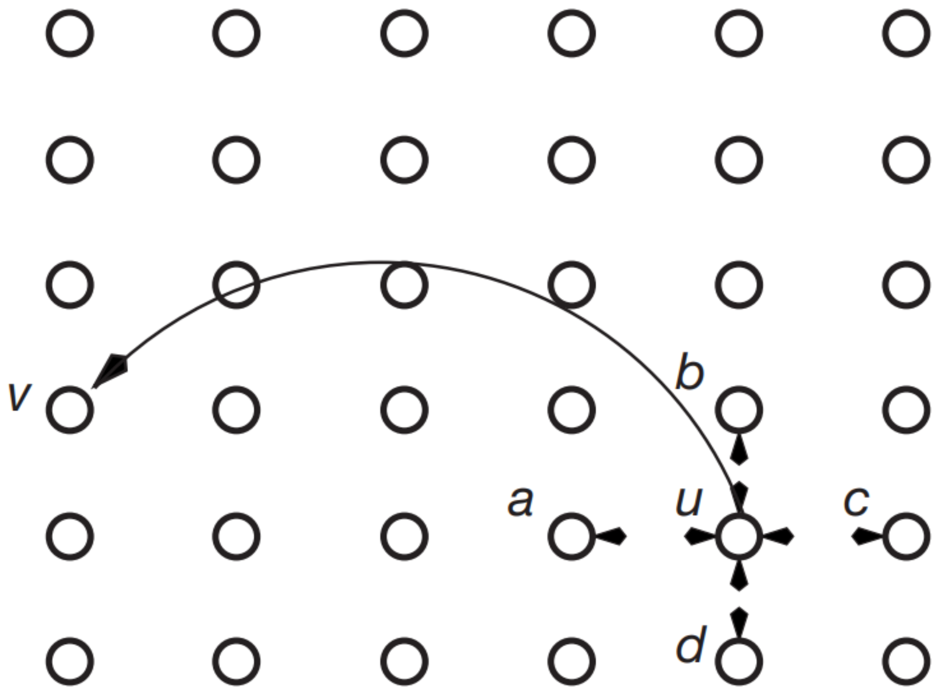


Figure 3: Illustration of the network structure with short-range and long-range links [2].

- **Short-range edges**: Each node u is connected to all lattice neighbors within a fixed Manhattan distance p. In Figure 3, $p = 1$, corresponding to immediate grid neighbors.

- **Long-range edges**: Each node u establishes q additional directed connections to nodes $v \neq u$, sampled independently with probability proportional to $d(u, v)^{-r}$, where $d(u, v)$ is the Manhattan distance and r is the clustering exponent.

This model, originally proposed by Kleinberg [2], incorporates both local and global connectivity, balancing regular structure with random shortcuts.

Formally, the network consists of nodes indexed by coordinates (i, j) for $i, j \in \{1, 2, \ldots, n\}$, forming a toroidal lattice. The Manhattan distance between nodes $u = (i, j)$ and $v = (k, l)$ is given by

$$d(u, v) = |k - i| + |l - j|.$$

The parameters p and q control the local and global connectivity, respectively. As r varies, the distribution of long-range connections transitions from uniform $(r = 0)$ to highly localized $(r \gg 1)$, which significantly affects navigability.

3.2 Greedy Routing Strategy

Given a source node u and a target node v, the greedy routing algorithm operates by iteratively forwarding the message to the neighbor closest to v in Manhattan distance. At each step, only local information is used to make routing decisions.

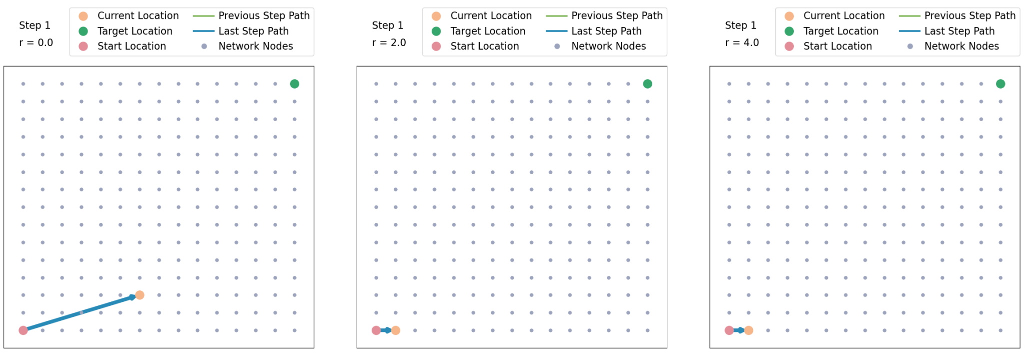
Each move reduces the distance to the target, so the algorithm is guaranteed to terminate either upon reaching v or becoming trapped in a local minimum (in the absence of cycles, this guarantees convergence). While greedy strategies do not always yield globally optimal paths, they are computationally efficient and suitable for large networks where global path information is unavailable.

There are several motivations for employing greedy routing in this context:

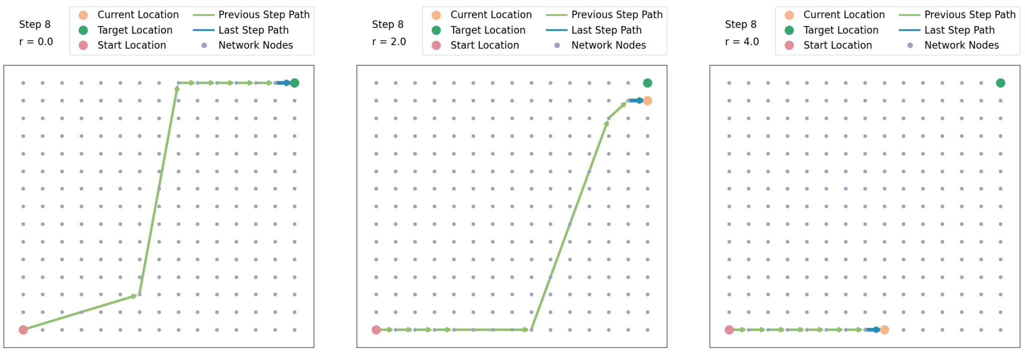
- Computing global shortest paths is infeasible at scale due to memory and time constraints.
- Small-world networks are hypothesized to support efficient decentralized routing using only local heuristics.
- Approximate solutions are acceptable in many real-world scenarios, particularly when the trade-off in performance yields substantial efficiency gains.

3.3 Illustration of Greedy Path Trajectories

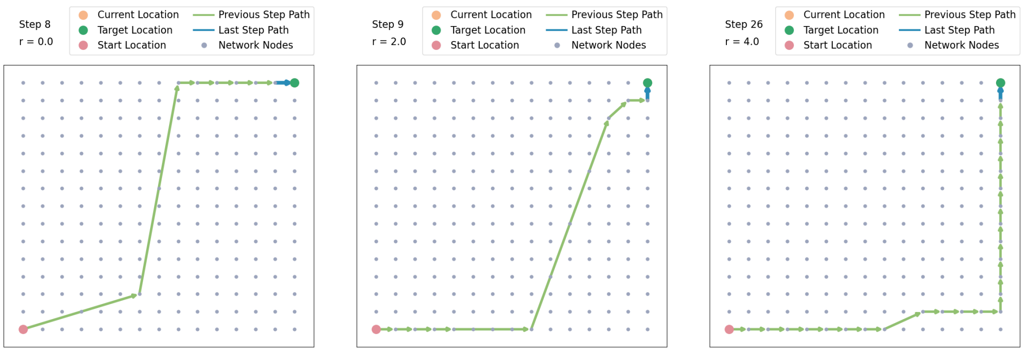
To visually demonstrate the progress of greedy routing through the network, we present a sequence of trajectory snapshots under different clustering exponents. These images track the movement from the source node to the destination over time. The trajectory exhibits local optimization behavior, where each move attempts to reduce the distance to the target, though not necessarily along a globally optimal path.



(a) Greedy path at step 1.



(b) Greedy path at step 8.



(c) Greedy path at step 26.

Figure 4: Progress of greedy routing under different values of r, with $n = 15$, $p = 1$, $q = 1$.

In the following section, we analyze the relationship between the expected delivery time cost of the algorithm and the clustering exponent.

4 Simulation of the Greedy Algorithm

In this section, we investigate how the clustering exponent r influences the expected delivery time T of greedy routing on a two-dimensional lattice. To accomplish this, we utilize two different implementations of the greedy algorithm:

- A custom implementation developed from scratch with optimizations.
- A publicly available simulator based on Kleinberg's model.

For both approaches, simulations are conducted with $n = 20{,}000$, short-range parameter $p = 2$, and long-range connections $q = 2$. For each trial, a source node u and a target node v are randomly selected such that $u \neq v$. The clustering exponent r is varied from 0 to 3 in increments of 0.1, and the corresponding expected delivery time T is recorded. This setup approximates the average-case delivery time across the network.

4.1 Approach 1: Custom Implementation

Here we describe our optimized algorithm for estimating the greedy navigation time in Kleinberg's two-dimensional small-world model, parameterized by p, q, and r.

4.1.1 Key Components

- **Precomputed CDF:** Long-range link distances are sampled from a distribution proportional to d^{1-r}, where d is the Manhattan distance and r is the clustering exponent. We precompute the cumulative distribution function (CDF) to enable efficient inverse transform sampling.
- **Offset Dictionary:** To avoid enumerating all n^2 nodes when sampling at distance d, we construct a lookup table $\mathcal{O}[d]$ containing all integer offsets (dx, dy) such that $|dx| + |dy| = d$. This allows efficient uniform sampling of valid coordinate offsets at each distance.

- **Toroidal Manhattan Distance:** The toroidal distance accounts for wrap-around effects and is defined as

  $$d_{\text{tor}}(\omega, v) = \min(|\omega_x - v_x|, n - |\omega_x - v_x|) + \min(|\omega_y - v_y|, n - |\omega_y - v_y|).$$

- **Cycle Detection:** A 2D boolean array V tracks whether a node has been visited. If the current node is about to revisit an already visited node, the process terminates early to avoid infinite loops.

4.1.2 Algorithm Steps

a) Initialize an empty list $\mathcal{L}$ to store step counts.

b) Precompute the CDF over distances and the offset dictionary $\mathcal{O}[d]$.

c) For each of K trials:

   (a) Randomly select source $u = (u_x, u_y)$ and target $v = (v_x, v_y)$.

   (b) Initialize step counter $s = 0$ and visited grid V to all false.

   (c) While $u \neq v$ and $s < M$:

      i. Add all short-range neighbors within distance p to set N.

      ii. Sample q long-range neighbors using the CDF and offset dictionary, and add to N.

      iii. Select neighbor $\omega \in N$ minimizing $d_{\text{tor}}(\omega, v)$.

      iv. If ω is already visited, break (cycle detected).

      v. Set $u \leftarrow \omega$, increment s.

   (d) Append s to $\mathcal{L}$.

d) Return $\hat{T} = \frac{1}{K} \sum_{s \in \mathcal{L}} s$ as the estimated delivery time.

The details of the algorithm are shown in Algorithm 2. Unless otherwise specified, the number of trials is set to 100.

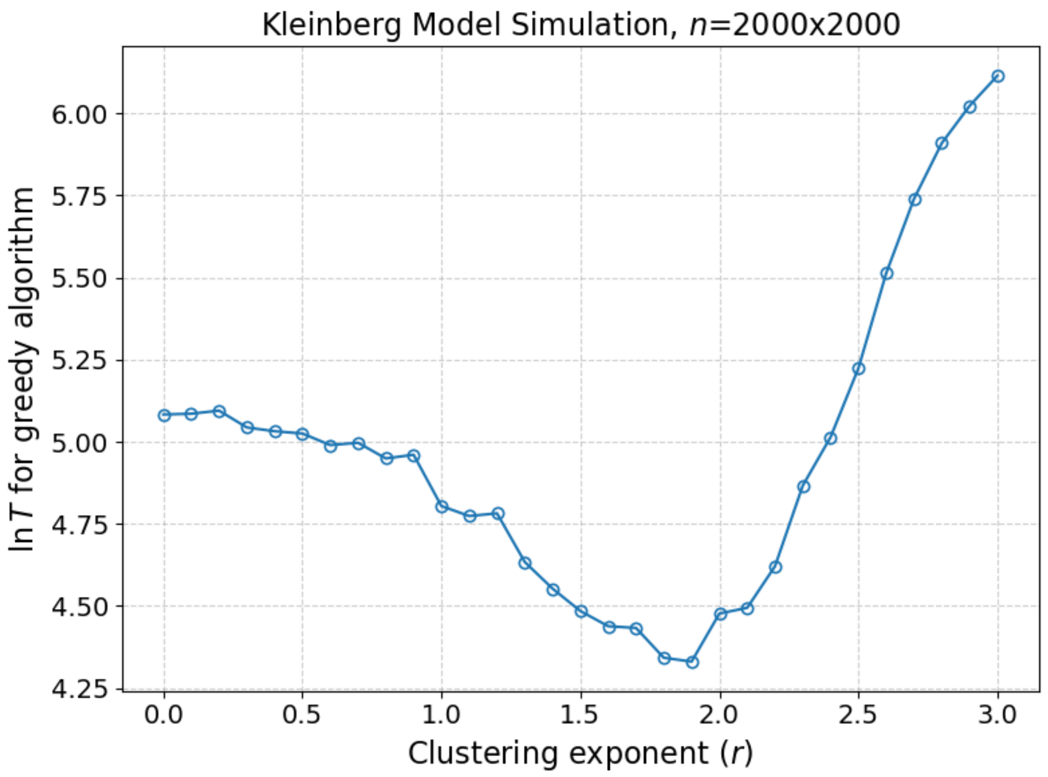


Figure 5: Relationship between clustering exponent r and $\ln T$ using the custom implementation.

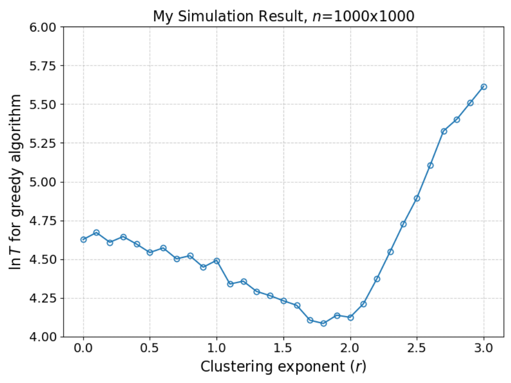
However, the algorithm is not fully efficient. The space complexity is $\mathcal{O}(n^2)$ due to the visited grid used for cycle detection. The time complexity is approximately $\mathcal{O}(K \cdot M \cdot (p^2 + q))$, where K is the number of trials, M is the maximum number of steps per trial, p^2 accounts for the enumeration of short-range neighbors, and q accounts for sampling long-range neighbors.

4.2 Approach 2: GitHub Open Source Implementation

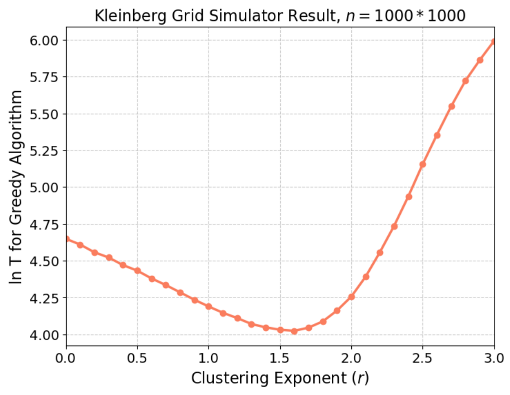
To complement our own implementation, we experimented with the open-source **Kleinberg Grid Simulator** [1], a tool designed to simulate greedy routing on augmented grid networks. The simulator supports both Python and Julia backends and includes several engineering features such as parallelism and efficient long-range link sampling.

Specifically, it adopts *double rejection sampling* to draw long-range links from a power-law distribution governed by exponent r, and uses fixed-size big-integer types in its Julia backend to accelerate computation. These optimizations result in both high computational speed and low memory overhead, making the simulator well-suited for large-scale experiments.

To verify the correctness of our implementation and explore consistency across methods, we compare the expected delivery time $\ln T$ as a function of clustering exponent r across the two approaches. As shown in Figure 6, both implementations exhibit a U-shaped pattern with minimum delivery time near $r \approx 2$, in agreement with Kleinberg's theoretical prediction.



(a) Custom implementation.



(b) GitHub implementation.

Figure 6: Comparison of $\ln T$ versus clustering exponent r for two implementations.

To further evaluate scalability, we also ran the **Kleinberg Grid Simulator** on a 20000×20000 grid. The result is shown in Figure 7. The curve confirms that even at this large scale, the expected delivery time remains minimized near $r \approx 2$, reinforcing the theoretical insight.

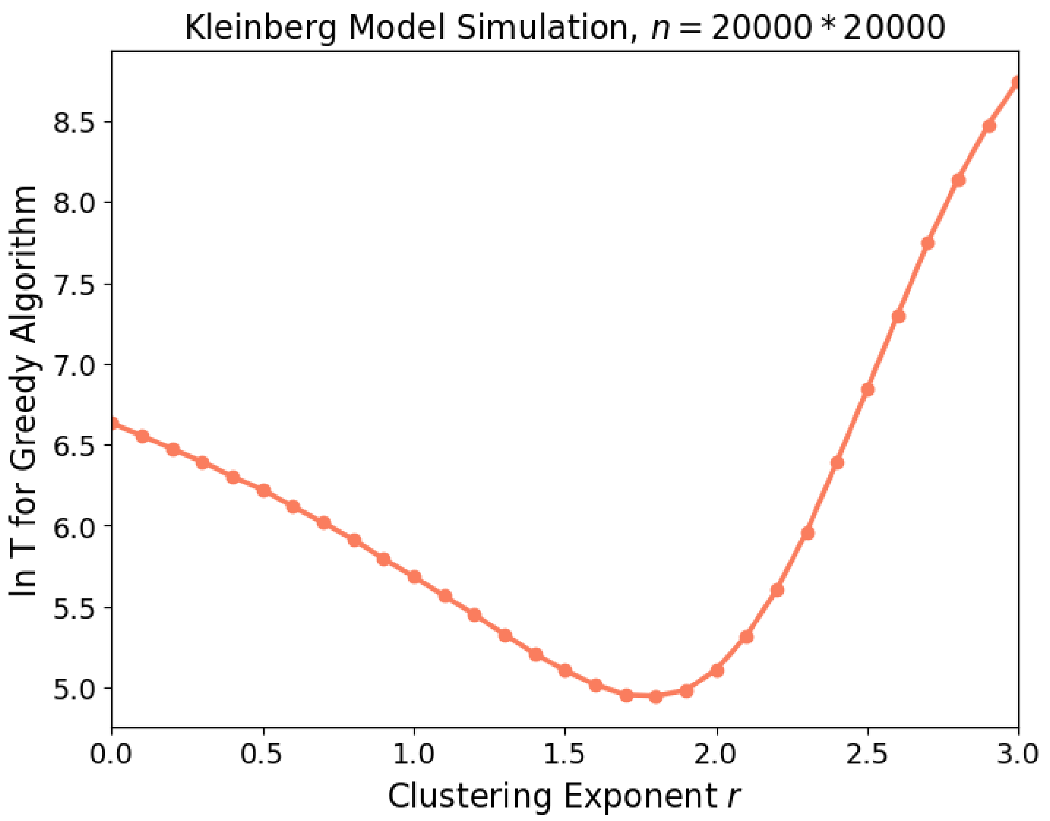


Figure 7: Relationship between clustering exponent r and $\ln T$ using the Kleinberg Grid Simulator on a 20000×20000 grid.

These experiments suggest that the simulator is not only efficient but also faithful to Kleinberg's model. Its strong performance at large scales offers insights into practical design considerations, from which we also draw inspiration to optimize our own simulator.

5 Conclusion

This project investigates the algorithmic implications of the small-world phenomenon through simulations of greedy routing on augmented two-dimensional grids. A custom implementation was developed to estimate the expected delivery time $\ln T$ as a function of clustering exponent r, and validated against the **Kleinberg Grid Simulator**.

Both implementations exhibit the theoretically predicted U-shaped delivery time curve, with minimum routing cost near $r = 2$. The close match between results confirms the correctness of our implementation, while minor differences reveal the impact of sampling methods and computational strategies.

The **Kleinberg Grid Simulator** further demonstrates excellent scalability and efficiency on large grids (e.g., 20000×20000), making it a valuable reference for future optimization work. Overall, this project highlights the critical role of the clustering exponent in decentralized navigation and provides a reproducible simulation pipeline for further exploration of routing dynamics in small-world networks.

References

[1] Fabien Mathieu et. al. https://github.com/balouf/kleinberg-grid-simulator/.

[2] Jon M Kleinberg. “Navigation in a small world”. In: *Nature* 406.6798 (2000), pp. 845–845.

[3] Stanley Milgram. “The small world problem”. In: *Psychology today* 2.1 (1967), pp. 60–67.

Appendix

The Greedy Algorithm for trajectory

Algorithm 1 Greedy navigation with multi-range connections

```
Input: Grid size $n$, short-range parameter $p$, long-range parameter $q$, clustering exponent $r$, source node $u$, target node $v$
 1: Initialize path $S \leftarrow \{u\}$ and temporary set $S_{\text{temp}} \leftarrow \emptyset$
 2: while $u \neq v$ do
 3:     $S_{\text{temp}} \leftarrow \emptyset$
 4:     // Add short-range neighbors within distance $p$
 5:     for each node $w$ such that $1 \leq d(u, w) \leq p$ do
 6:         Add $w$ to $S_{\text{temp}}$
 7:     end for
 8:     // Add $q$ long-range neighbors sampled by distance
 9:     for $i = 1$ to $q$ do
10:         Sample a node $z$ from all nodes $\neq u$, with probability proportional to $d(u, z)^{-r}$
11:         Add $z$ to $S_{\text{temp}}$
12:     end for
13:     $\omega \leftarrow$ the closest node to $v$ in $S_{\text{temp}}$
14:     Append $\omega$ to $S$
15:     $u \leftarrow \omega$
16: end while
Output: Navigation path $S$ from $u$ to $v$
```

Greedy Algorithm for Estimating Navigation Time

Algorithm 2 Estimation of greedy navigation time

```
Input: Grid size $n$, number of trials $K$, short-range distance $p$, long-range
    count $q$, clustering exponent $r$, maximum steps $M$
 1: Initialize empty list $\mathcal{L} \leftarrow \emptyset$
 2: Precompute CDF $\mathcal{C}$ over distance $d$ with probability $\propto d^{1-r}$
 3: Precompute offset dictionary $\mathcal{O}[d] \leftarrow \{(dx, dy) : |dx| + |dy| = d\}$
 4: for $t = 1$ to $K$ do
 5:   Randomly sample source node $u = (u_x, u_y)$ and target node $v = (v_x, v_y)$
 6:   Initialize step count $s \leftarrow 0$, visited grid $V[0 \dots n-1][0 \dots n-1] \leftarrow$ false
 7:   while $u \neq v$ and $s < M$ do
 8:     Set $V[u_x][u_y] \leftarrow$ true, initialize neighbor set $N \leftarrow \emptyset$
 9:     for all $(dx, dy)$ such that $|dx| + |dy| \leq p$ and $(dx, dy) \neq (0, 0)$ do
10:       Add $((u_x + dx) \bmod n,\ (u_y + dy) \bmod n)$ to $N$
11:     end for
12:     Initialize long-range set $\mathcal{Z} \leftarrow \emptyset$
13:     while $|\mathcal{Z}| < q$ do
14:       Sample distance $d$ from CDF $\mathcal{C}$
15:       Uniformly sample $(dx, dy) \in \mathcal{O}[d]$
16:       $z \leftarrow ((u_x + dx) \bmod n,\ (u_y + dy) \bmod n)$
17:       if $z \neq u$ then
18:         $\mathcal{Z} \leftarrow \mathcal{Z} \cup \{z\}$
19:       end if
20:     end while
21:     $N \leftarrow N \cup \mathcal{Z}$
22:     Select $\omega \in N$ that minimizes toroidal Manhattan distance to $v$:
23:       $d_{\text{tor}}(\omega, v) = \min(|\omega_x - v_x|, n - |\omega_x - v_x|) + \min(|\omega_y - v_y|, n - |\omega_y - v_y|)$
24:     if $V[\omega_x][\omega_y] =$ true then
25:       break {Cycle detected}
26:     end if
27:     $u \leftarrow \omega$, $s \leftarrow s + 1$
28:   end while
29:   Append $s$ to $\mathcal{L}$
30: end for
31: return Estimated time $\hat{T} = \frac{1}{K} \sum_{s \in \mathcal{L}} s$
```