Kleinberg's Grid Reloaded

Fabien Mathieu

Kleinberg's Grid Reloaded

Fabien Mathieu[1]

1   Nokia Bell Labs, route de Villejust, 91620 Nozay
   `fabien.mathieu@nokia-bell-labs.com`

—— **Abstract** ————————————————————————————

One of the key features of small-worlds is the ability to route messages with few hops only using local knowledge of the topology. In 2000, Kleinberg proposed a model based on an augmented grid that asymptotically exhibits such property.

In this paper, we propose to revisit the original model from a simulation-based perspective. Our approach is fueled by a new algorithm that uses dynamic rejection sampling to draw augmenting links. The speed gain offered by the algorithm enables a detailed numerical evaluation. We show for example that in practice, the augmented scheme proposed by Kleinberg is more robust than predicted by the asymptotic behavior, even for very large finite grids. We also propose tighter bounds on the performance of Kleinberg's routing algorithm. At last, we show that fed with realistic parameters, the model gives results in line with real-life experiments.

1   Introduction

In a prescient 1929 novel called *Láncszemek* (in English, *Chains*), Karinthy imagines that any two people can be connected by a small chain of personal links, using no more than *five* intermediaries [10].

Years later, Milgram validates the concept by conducting real-life experiments. He asks volunteers to transmit a letter to an acquaintance with the objective to reach a target destination across the United States [17, 20]. While not all messages arrive, successful attempts reach destination after six hops in average, popularizing the notion of *six degrees of separation*.

Yet, for a long time, no theoretical model could explain why and how this kind of *small-world* routing works. One of the first and most famous attempts to provide such a model is due to Jon Kleinberg [12]. He proposes to abstract the social network by a grid augmented with *shortcuts*. If the shortcuts follow a heavy tail distribution with a specific exponent, then a simple greedy routing can reach any destination in a short time ($O(\log^2(n))$ hops). On the other hand, if the exponent is wrong, then the time to reach destination becomes $\Omega(n^\alpha)$ for some α. This seminal work has led to multiple studies from both the theoretical and empirical social systems communities.

Contribution

In this paper, we propose a new way to numerically benchmark the greedy routing algorithm in the original model introduced by Kleinberg. Our approach uses dynamic rejection sampling, which gives a substantial speed improvement compared to previous attempts, without

making any concession about the assumptions made in the original model, which is kept untouched.

Fueled by the capacity to obtain quick and accurate results even for very large grids, we give a fresh look on Kleinberg's grid, through three independent small studies. First, we show that the model is in practice more robust than expected: for grids of given size there is quite a large range of exponents that grant short routing paths. Then we observe that the lower bounds proposed by Kleinberg in [12] are not tight and suggest new bounds. Finally, we compare Kleinberg's grid to Milgram's experiment, and observe that when the grid parameters are correctly tuned, the performance of greedy routing is consistent with the *six degrees of separation* phenomenon.

Roadmap

Section 2 presents the original augmented grid model introduced by Kleinberg and the greedy routing algorithm. A brief overview of the main existing theoretical and experimental studies is provided, with a strong emphasis on the techniques that can be used for the numerical evaluation of Kleinberg's model.

In Section 3, we give our algorithm for estimating the performance of greedy routing. We explain the principle of dynamic rejection sampling and detail why it allows to perfectly emulate Kleinberg's grid with the same speed that can be achieved by toroidal approximations. We also give a performance evaluation of the simulator based on our solution. For readers interested in looking under the hood, a fully working code (written in Julia) is given in Appendix A.

To show the algorithm benefits, we propose in Section 4 three small studies that investigate Kleinberg's model from three distinct perspectives: robustness of greedy routing with respect to the shortcut distribution (Section 4.1); tightness of the existing theoretical bounds (Section 4.2); emulation of Milgram's experiment within Kleinberg's model (Section 4.3).

2   Model and Related Work

We present here the model and notation introduced by Kleinberg in [12, 11], some key results, and a brief overview of the subsequent work on the matter.

2.1   Kleinberg's Grid

In [12], Kleinberg considers a model of directed random graph $G(n, r, p, q)$, where n, p, q are positive integers and r is a non-negative real number. A graph instance is built from a square lattice of $n \times n$ nodes with Manhattan distance d: if $u = (i, j)$ and $v = (k, l)$, then $d(u, v) = |i - j| + |k - l|$. d represents some natural proximity (geographic, social, ...) between nodes. Each node has some *local* neighbors and q *long range* neighbors. The local neighbors of a node u are the nodes v such that $d(u, v) \leq p$. The q long range neighbors of u, also called *shortcuts*, are drawn independently and identically as follows: the probability that a given long edge starting from u arrives in v is proportional to $(d(u, v))^{-r}$.

The problem of decentralized routing in a $G(n, r, p, q)$ instance consists in delivering a message from node u to node v in a hop-by-hop basis. At each step, the message bearer needs to choose the next hop among its neighbors. The decision can only use the lattice coordinates of the neighbors and destination. The main example of decentralized algorithm is the *greedy routing*, where at each step, the current node chooses the neighbor that is closest to destination based on d (in case of ties, an arbitrary breaking rule is used).

The main metric to analyze the performance of a decentralized algorithm is the *expected delivery time*, which is the expected number of hops to transmit a message between two nodes chosen uniformly at random in the graph.

This paper focuses on studying the performance of the greedy algorithm. Unless stated otherwise, we assume $p = q = 1$ (each node has up to four local neighbors and one shortcut). Let $e_r(n)$ be the expected delivery time of the greedy algorithm in $G(n, r, 1, 1)$.

2.2 Theoretical Results

The main theoretical results for the genuine model are provided in the original papers [11, 12], where Kleinberg proves the following:

- $e_2(n) = O(\log^2(n))$;
- for $0 \le r < 2$, the expected delivery time of any decentralized algorithm is $\Omega(n^{(2-r)/3})$;
- for $r > 2$, the expected delivery time of any decentralized algorithm is $\Omega(n^{(r-2)/(r-1)})$.

Kleinberg's results are often interpreted as follows: short paths are easy to find only in the case $r = 2$. The fact that only one value of r asymptotically works is sometimes seen as the sign that Kleinberg's model is not robust enough to explain the small-world routing proposed by Karinthy and experimented by Milgram. However, as briefly discussed by Kleinberg in [5], there is in fact some margin if one considers a grid of given n. This tolerance will be investigated in more details in Section 4.1.

While we focus here on the original model, let us give a brief, non-exhaustive, overview of the subsequent extensions that have been proposed since. Most proposals refine the model by considering other graph models or other decentralized routing algorithms. New graph models are for example variants of the original model (studying grid dimension or the number of shortcuts per node [2, 5, 9]), graphs inspired by peer-to-peer overlay networks [14], or arbitrary graphs augmented with shortcuts [6, 8]. Other proposals of routing algorithms usually try to enhance the performance of the greedy one by granting the current node additional knowledge of the topology [7, 14, 15].

A large part of the work above aims at improving the $O(\log^2(n))$ bound of the greedy routing. For example, in the small-world percolation model, a variant of Kleinberg's grid with $O(\log(n))$ shortcuts per node, greedy routing performs in $O(\log(n))$ [14].

2.3 Experimental Studies

Many empirical studies have been made to study how routing works in real-life social networks and the possible relation with Kleinberg's model (see for example [13, 5] and the references within). On the other hand, numerical evaluations of the theoretical models are more limited to the best of our knowledge. Such evaluations are usually performed by averaging R runs of the routing algorithm considered.

In [11], Kleinberg computes $e_r(n)$ for $n = 20,000$ and $r \in [0, 2.5]$, using 1,000 runs per estimate. However, he uses a torus instead of a regular grid (this will be discussed later in the paper). In [14], networks of size up to 2^{24} (corresponding to $n = 2^{12}$ in the grid) are investigated using 150 runs per estimate.

Closer to our work, Athanassopoulos *et al.* propose a study centered on numerical evaluation that looks on Kleinberg's model and some variants [1]. For the former, they differ from the original model by having fixed source and destination nodes. They compute $e_r(n)$ for values of n up to 3,000 and $r \in \{0, 1, 2, 3\}$, using 900 runs per estimate.

To compare with, in the present paper, we consider values of n up to $2^{24} \approx 16,000,000$ and $r \in [0, 3]$, with at least 10,000 runs per estimate. To explain such a gain, we first need

to introduce the issue of shortcuts computation.

2.3.1   Drawing shortcuts

As stated in [1], the main computational bottleneck for simulating Kleinberg's model comes from the shortcuts.

- There are n^2 shortcuts in the grid (assuming $q = 1$);
- When one wants to a shortcut, any of the $n^2 - 1$ other nodes can be chosen with non-null probability. This can be made by inverse transform sampling, with a cost $\Omega(n^2)$;
- The shortcut distribution depends on the node u considered, even if one uses relative coordinates. For example, a corner node will have $i + 1$ neighbors at distance i for $1 \leq i < n$, against $4i$ neighbors for inner nodes (as long as the ball of radius i stays inside the grid). This means that, up to symmetry, each node has a unique shortcut distribution[1]. This prevents from mutualising shortcuts drawings between nodes.

In the end, building shortcuts as described above for each of the R runs has a time complexity $\Omega(Rn^4)$, which is unacceptable if one wants to evaluate $e_r(n)$ on large grids.

The first issue is easy to address: as observed in [12, 14, 1], we can use the *Principle of deferred decision* [18] and compute the shortcuts on-the-fly as the path is built, because they are drawn independently and a node is never used twice in a given path. This reduces the complexity to $\Omega(Rn^2 e_r(n))$.

2.3.2   Torus approximation

To lower the complexity even more, one can approximate the grid by the torus. This is the approach adopted in [11, 14]. The toroidal topology brings two major features compared to a flat grid:

- The distribution of the relative position of the shortcut does not depend on the originating node. This enables to draw shortcuts in advance (in bulk);
- There is a strong radial symmetry, allowing to draw a "radius" and an "angle" separately.

To illustrate the gain of using a torus instead of a grid, consider the drawing of k shortcuts from k distinct nodes. In a grid, if one uses inverse transform sampling for each corresponding distribution, the cost is $\Omega(n^2 k)$. In the torus, one can compute the probabilities to be at distance i for i between 1 and n (the maximal distance in the torus), draw k radii, then choose for each drawn radius i a node uniformly chosen among those at distance i. Assuming drawing a float uniformly distributed over $[0, 1)$ can be made in $O(1)$, the main bottleneck is the drawing of radii. Using bulk inverse transform sampling, it can be performed in $O(n + k \log(k))$, by sorting k random floats, matching then against the cumulative distribution of radii and reverse sorting the result.

3   Fast Estimation of Expected Delivery Time

We now describe our approach for computing $e_r(n)$ in the flat grid with the same complexity than for the torus approximation.

---

[1] To take advantage of symmetry, one can consider the isometric group of a square grid, which can be built with the quarter-turn and flip operations. However, its size is 8, so even using symmetry, there are at least $\frac{n^2}{8}$ distinct (non-isomorphic) distributions.
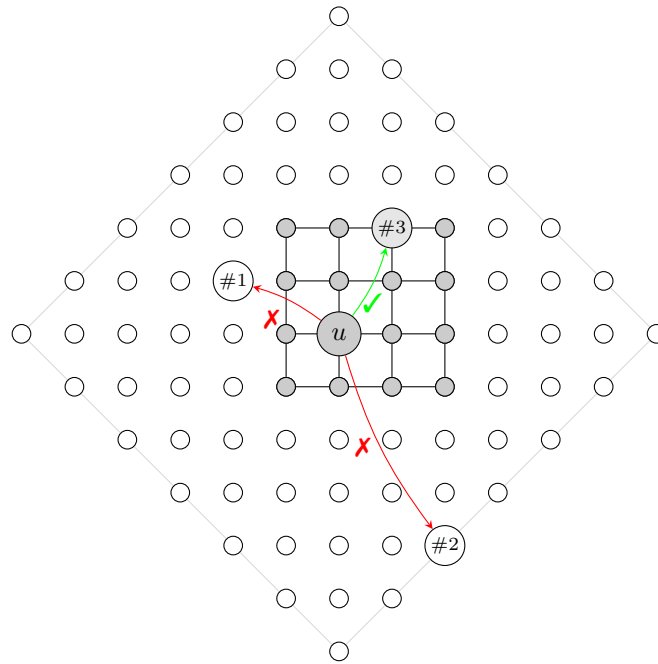
Figure 1 Main idea of the dynamic rejection sampling approach ($n = 4$).

## 3.1 Dynamic rejection sampling for drawing shortcuts

In order to keep low computational complexity without making any approximation of the model, we propose to draw a shortcut of a node u as follows:

1. We embed the actual grid G (we use here G to refer to the lattice nodes of $G(n, r, p, q)$ in a virtual lattice B_u made of points inside a ball of radius $2(n-1)$. Note that the radius chosen ensures that G is included in B_u no matter the location of u.
2. We draw a node inside B_u such that the probability to pick up a node v is proportional to $(d(u, v))^{-r}$. This can be done in two steps (radius and angle):
   - For the radius, we notice that the probability to draw a node at distance i is proportional to i^{1-r}, so we pick an integer between 1 and $2(n-1)$ such that the probability to draw i is to $i^{1-r} / \sum_{k=1}^{2(n-2)} k^{1-r}$.
   - For the angle, pick an integer uniformly chosen between 1 and $4i$
3. This determines a unique point v among the $4i$ points at distance i from u in the virtual lattice, chosen with a probability proportional to $(d(u, v))^{-r}$. If v belongs to the actual grid, it becomes the shortcut, otherwise we try again (back to step #2).

This technique, illustrated in Figure 1, is inspired by the *rejection sampling* method [21]. By construction, it gives the correct distribution: the node v that it eventually returns is in the actual grid and has been drawn with a probability proportional to $(d(u, v))^{-r}$.

We call this *dynamic* rejection sampling because the sampled distribution changes with the current node u. Considering u as a relative center, the actual grid G moves with u and acts like an acceptance mask. On the other hand, the distribution over the virtual lattice B_u remains constant. This enables to draw batches of relative shortcuts that can be used over multiple runs, exactly like for the torus approximation.

The only possible drawback of this approach is the number of attempts required to draw a correct shortcut. Luckily, this number is contained.
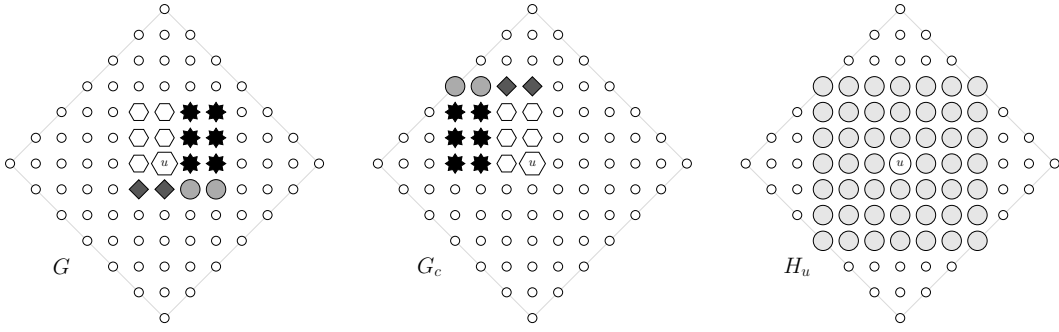
Figure 2 Graphical representation of G, G_c and H_u ($n = 4$). The sub-lattices used to built a bijection between G and G_c (cf proof of Lemma 1) are represented with distinct shapes and gray levels.

▶ **Lemma 1.** *The probability that a node drawn in B_u belongs to G is at least $\frac{1}{8}$.*

Proof. We will prove that

$$\frac{\sum_{v \in G \setminus \{u\}} (d(u, v))^{-r}}{\sum_{v \in B_u \setminus \{u\}} (d(u, v))^{-r}} > \frac{1}{8}.$$

We use the fact that the probability decreases with the distance combined with some geometric arguments. Let G_c be a $n \times n$ lattice that has u as one of its corner. Let H_u the $(2n - 1) \times (2n - 1)$ lattice centered in u.

In terms of probability of drawing a node in $G \setminus \{u\}$, the worst case is when u is at some corner: there is a bijection f from G to G_c such that for all $v \in G \setminus \{u\}, d(u, f(v)) \geq d(u, v)$. Such a bijection can be obtained by splitting G into $G \cap G_c$ and three other sub-lattices that are flipped over $G \cap G_c$ (see Figure 2). This gives

$$\sum_{v \in G \setminus \{u\}} (d(u, v))^{-r} \geq \sum_{v \in G \setminus \{u\}} (d(u, f(v)))^{-r} = \sum_{v \in G_c \setminus \{u\}} (d(u, v))^{-r}.$$

Then we observe that the four possible lattices G_c obtained depending on the corner occupied by u fully cover H_u. In fact, axis nodes are covered redundantly. This gives

$$\sum_{v \in H_u \setminus \{u\}} (d(u, v))^{-r} < 4 \sum_{v \in G_c \setminus \{u\}} (d(u, v))^{-r}.$$
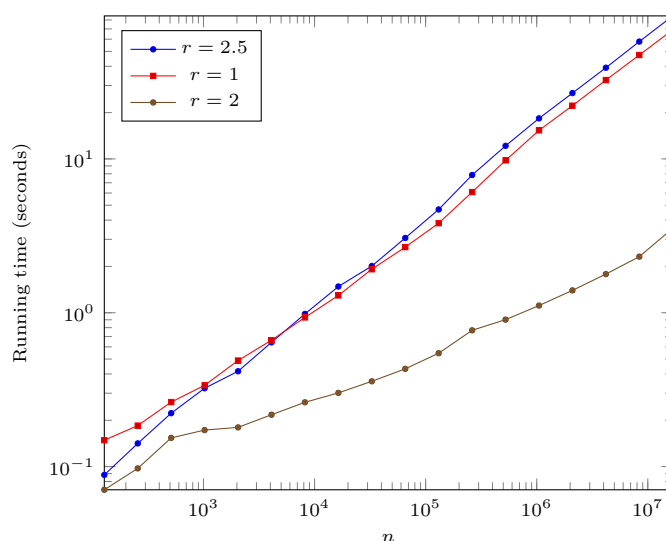
Lastly, if one folds $B_u \setminus H_u$ back into H_u like the corners of a sheet of paper, we get a strict injection g from $B_u \setminus H_u$ to $H_u \setminus \{u\}$ (the diagonal nodes of H_u are not covered). Moreover, for all $v \in B_u \setminus H_u, d(u, v) \geq d(u, h(v))$. This gives

$$\sum_{v \in B_u \setminus H_u} (d(u, v))^{-r} \leq \sum_{v \in B_u \setminus H_u} (d(u, h(v)))^{-r} < \sum_{v \in H_u \setminus \{u\}} (d(u, v))^{-r}.$$

This concludes the proof, as we get

$$\frac{\sum_{v \in G \setminus \{u\}} (d(u, v))^{-r}}{\sum_{v \in B_u \setminus \{u\}} (d(u, v))^{-r}} \geq \frac{\sum_{v \in G_c \setminus \{u\}} (d(u, v))^{-r}}{\sum_{v \in H_u \setminus \{u\}} (d(u, v))^{-r} + \sum_{v \in B_u \setminus H_u} (d(u, v))^{-r}} > \frac{1}{8}.$$

◀

**Figure 3** Computing $e_r(n)$ using $R = 10,000$ runs

Remarks

- When $r = 0$ (uniform shortcut distribution), the bound $\frac{1}{8}$ is asymptotically tight: the success probability is exactly the ratio between the number of nodes in $G \setminus \{u\}$ and $B_u \setminus \{u\}$, which is $\frac{n^2-1}{4(n-1)(2n-1)} \xrightarrow[n \to +\infty]{} \frac{1}{8}$. On the other hand, as r grows, the probability mass gets more and more concentrated around u (hence in G), so we should expect better performance (cf Section 3.2).
- The dynamic rejection sampling approach can be used in other variants of Kleinberg's model, like for other dimensions or when the number of shortcuts per node is a random variable (like in [9]). The only requirement is the existence of some *root* distribution (like the distribution over B_u here) that can be carved to match any of the possible distributions with a simple acceptance test.
- Only the nodes from H_u may belong to G, so nodes from $B_u \setminus G$ are always sampled for nothing. For example, in Figure 1, this represents 36 nodes over 84. By drawing shortcuts in $H_u \setminus \{u\}$ instead of $B_u \setminus \{u\}$, we could increase the success rate lower bound to 1/4. However, this would make the algorithm implementation more complex (the number of nodes at distance i is not always $4i$), which is in practice not worth the factor 2 improvement of the lower bound. This may not be true for a higher dimension β. Adapting the proof from Lemma 1, we observe that using a ball of radius $\beta(n-1)$ will lead to a bound $\beta!(2\beta)^{-\beta}$, while a grid of side $2n-1$ will lead to $2^{-\beta}$. The two bounds are asymptotically tight for $r = 0$. In that case the grid approach is $\frac{\beta^\beta}{\beta!}$ more efficient than the ball approach (this grows exponentially with β).

3.2 Performance Evaluation

We implemented our algorithm in Julia 0.4.5. A working code example is provided in Appendix A. Simulations were executed on a low end device (a Dell tablet with 4 Gb RAM and Intel M-5Y10 processor), which was largely sufficient for getting fast and accurate results on very large grids, thanks to the dynamic rejection sampling.

Unless said otherwise, $e_r(n)$ is obtained by averaging $R = 10,000$ runs. For n, we mainly
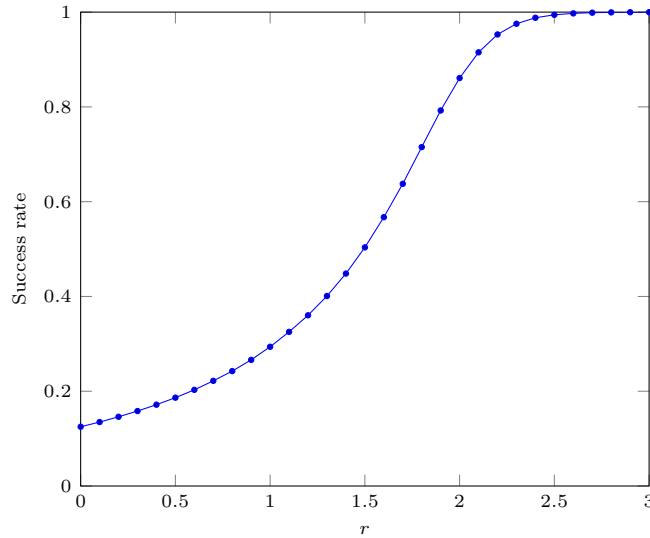
Figure 4 Frequency of shortcuts drawn in B_u that belongs to G observed during a computation of $e_r(n)$ $(n = 2^{14})$

consider powers of two ranging from 2^7 (about 16,000 nodes) to 2^{24} (about 280 trillions nodes). We focus on $r \in [0, 3]$.

Regarding the choice of the bulk size k (we assume here the use of bulk inverse transform sampling), the average number of shortcuts to draw over the R runs is $Re_r(n)$. This leads to an average total cost for drawing shortcuts in $O(\lceil \frac{Re_r(n)}{k} \rceil (n + k \log(n)))$. k can be optimized if $e_r(n)$ is known, but this requires bootstrapping the estimation of $e_r(n)$. Yet, we can remark that for n fixed and R large enough, we should choose k of the same order of magnitude than n, which ensures an average cost per shortcut in $O(\log(n))$. This is the choice we made in our code, which gives a complexity in $O(\max(Re_r(n), n) \log(n))$. This is not efficient for $n \gg Re_r(n)$ (the bulk is then over-sized, so lot of unused shortcuts are drawn), but this seldom happens with our settings.

Figure 3 presents the time needed to compute $e_r(n)$ as a function of n for $r \in \{1, 2, \frac{5}{2}\}$. We observe running times ranging from seconds to a few minutes. To compare with, in [1], which is to the best of our knowledge the only work disclosing computation times, one single run takes about 4 seconds for $n = 400$. With a similar time budget, our implementation averages 10,000 runs for $n = 2^{17} \approx 130,000$ ($r = 1$ or $r = 2.5$), or up to $n = 2^{24} \approx 16,000,000$ for the optimal exponent $r = 2$. In other words, we are several orders of magnitude faster.

We also looked at the cost of the dynamic rejection sampling approach in terms of shortcuts drawn outside of G. Figure 4 shows the success frequency of the sampling as a function of r for $n = 2^{14}$ (the actual value of n has no significant impact as long as it is large enough). We verify Lemma 1: the success rate is always at least $1/8 = 0.125$, which is a tight bound for $r = 0$. For $r = 1$, the rate is about 0.29, and it climbs to 0.86 for $r = 2$. Failed shortcuts become negligible (rate greater than 0.99) for $r \geq 2.5$. Overall, Figure 4 shows that the cost of drawing some shortcuts outside G is a small compared to the benefits offered by drawing shortcuts from a unique distribution.
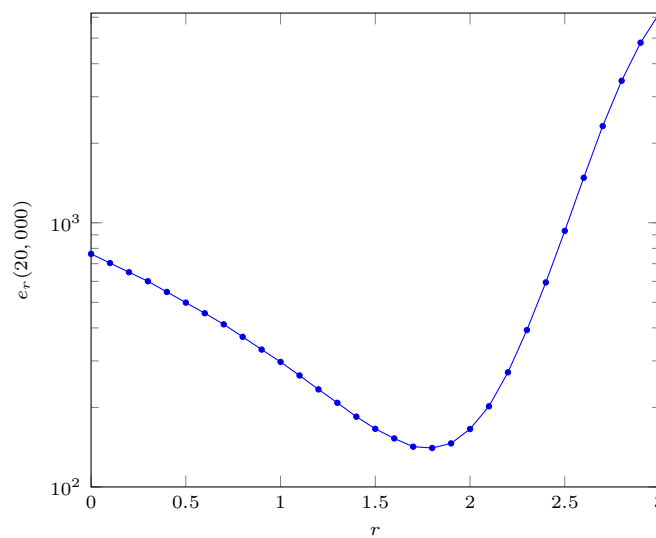
Figure 5 Expected delivery time for $n = 20,000$

Remark

In terms of success rate, Figure 4 shows that $r = 2.5$ is much more efficient than $r = 1$, but running times tend to be slightly longer for $r = 2.5$ (cf Figure 3). The reason is that $e_1(n)$ is lower than $e_{2.5}(n)$, which overcompensates the success rate difference.

4 Applications

Given the tremendous amount of strong theoretical results on small-world routing, one can question the interest of proposing a simulator (even a fast one!).

In this Section, we prove the interest of numerical evaluation through three (almost) independent small studies.

4.1 Efficient enough exponents

In [11], Kleinberg gives an estimation of $e_r(20,000)$ using a torus approximation (cf Section 2.3.2). A few years later, he discussed in more details the results, observing that [5]:

- $e_r(n)$ stays quite similar for $r \in [1.5, 2]$;
- the best value of r is actually slightly lower than 2.

We believe that these observations are very important as they show that routing in Kleinberg's grid is more robust than predicted by theory: it is efficient as long as r is *close enough* to 2.

Using our simulator, we can perform the same experiment. As shown by Figure 5, the results are quite similar to the ones observed in [11].

Yet, there is a small but *essential* difference between the two experiments: Kleinberg approximated his grid by a torus while we stay true to the original model. Why do we use the word *essential*? Both shapes have the same asymptotic behavior (proofs in [11] are straightforward to adapt), so why should we care? It seems to us that practical robustness is an essential feature if one wants to accept Kleinberg's grid as a reasonable model for routing in social networks. To the best of our knowledge, no theoretical work gives quantitative
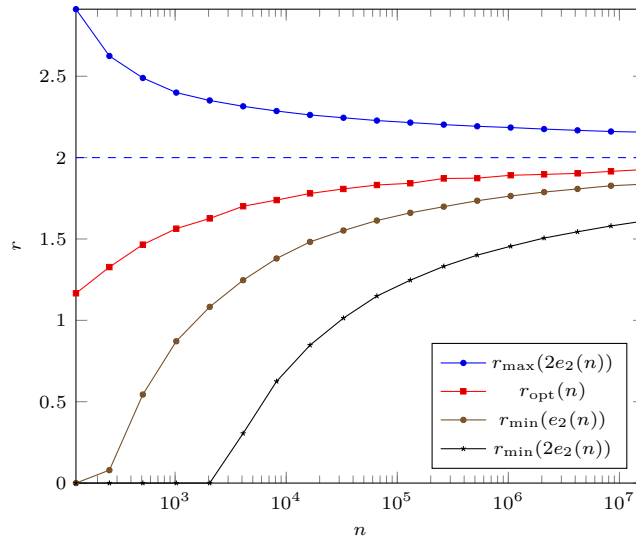
Figure 6 Reasonable values of r

results on this robustness, so we need to rely on numerical evaluation. But when Kleinberg uses a torus approximation, we can not rule out that the observed robustness is a by-side effect of the toroidal topology. The observation of a similar phenomenon on a flat grid discards this hypothesis. In fact, it suggests (without proving) that the robustness with respect to the exponent for grids of finite size may be a general phenomenon.

We propose now to investigate this robustness in deeper details. We have evaluated the following values, which outline, for a given n, the values of r that can be considered reasonable for performing greedy routing:

- The value of r that minimizes $e_r(n)$, denoted r_{opt};
- The smallest value of r such that $e_r(n) \leq e_2(n)$, denoted $r_{\min}(e_2(n))$;
- The smallest and largest values of r such that $e_r(n) \leq 2e_2(n)$, denoted $r_{\min}(2e_2(n))$ and $r_{\max}(2e_2(n))$ respectively.

The results are displayed in Figure 6. All values but r_{opt} are computed by bisection. For r_{opt}, we use a Golden section search [19]. Finding a minimum requires more accuracy, so the search of r_{opt} is set to use $R = 1,000,000$ runs per estimation. Luckily, as the computation operates by design through near-optimal values, we can increase the accuracy with reasonable running times.

Besides confirming that $r = 2$ is asymptotically the optimal value, Figure 6 shows that the range of reasonable values for finite grids is quite comfortable. For example, considering the range of values where $e_r(n)$ is less than twice $e_2(n)$, we observe that:

- For $n \leq 2^{11}$ (less than four million nodes), any r between 0 and 2.35 works;
- For $n \leq 2^{14}$ (less than 270 million nodes), the range is between 0.85 and 2.26.
- Even for $n = 2^{24}$ (about 280 trillions nodes), all values of r between 1.58 and 2.16 can still be considered *efficient enough*.

4.2 Asymptotic Behavior

Our simulator can be used to verify the theoretical bounds proposed in [12]. For example, Figure 7 shows $e_r(n)$ for r equal to 1, 2, and $\frac{5}{2}$.
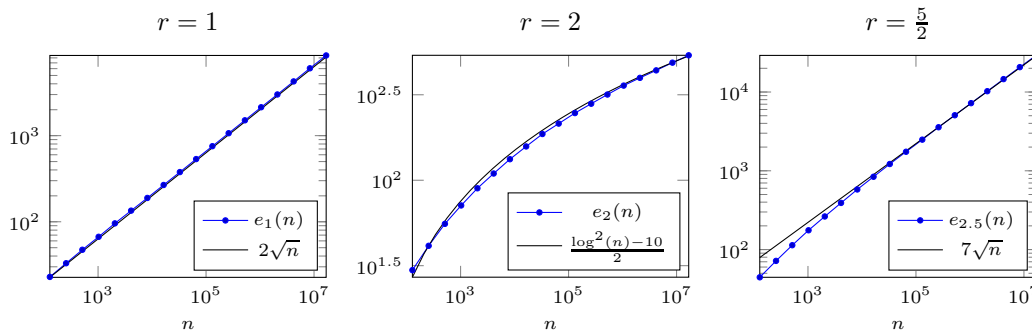
Figure 7 Expected delivery time for different values of r
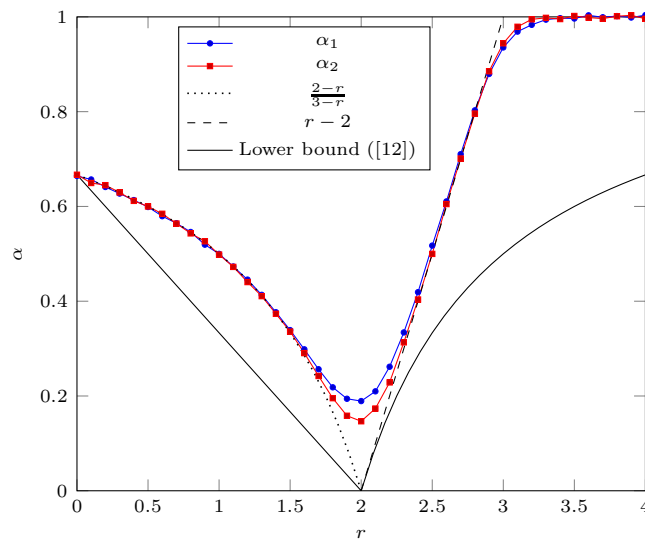
Figure 8 Estimates of the exponent α

As predicted, $e_r(n)$ seems to behave like $\log^2(n)$ for $r = 2$ and like n^α for the two other cases. Yet, the exponents found differ from the ones proposed in [12]. For both $r = 1$ and $r = 2.5$, we observe $\alpha = \frac{1}{2}$, while the lower bound is $\frac{1}{3}$. Intrigued by the difference, we want to compute α as a function of r.

However, we see in Figure 7 that a $\log^2(n)$ curve appears to have a positive slope in a logarithmic scale, even for large values of n. This may distort our estimations. To control the possible impact of this distortion, we estimate the exponent at two distinct scales:

- $n \in [2^{15}, 2^{20}]$, using the estimation $\alpha_1 := \frac{\log_2(e_r(2^{20})) - \log_2(e_r(2^{15}))}{5}$;
- $n \in [2^{20}, 2^{24}]$, using the estimation $\alpha_2 := \frac{\log_2(e_r(2^{24})) - \log_2(e_r(2^{20}))}{4}$.

The Results are displayed in Figure 8. The range of r was extended to $[0, 4]$ due to the observation of a new transition for $r = 3$

As feared, our estimations do not indicate 0 for $r = 2$, but the fact that α_2 is closer to 0 than α_1 confirms that this is likely caused by the $\log^2(n)$ factor. Moreover, we observe that α_1 and α_2 only differ for $r \approx 2$ and $r \approx 3$, suggesting that both estimates should be accurate except for these critical values.

Based on Figures 7 and 8, it seems that the bounds proposed by Kleinberg in [12] are only tight for $r = 0$ and $r = 2$. Formally proving more accurate bounds is beyond the scope

and spirit of this paper, but we can conjecture new bounds hinted by our simulations:

- For $0 \leq r < 2$, we have $e_r(n) = \Theta(n^{\frac{2-r}{3-r}})$;
- For $2 < r < 3$, we have $e_r(n) = \Theta(n^{r-2})$;
- For $r > 3$, we have $e_r(n) = \Theta(n)$.

This conjectured bounds are consistent with the one proposed in [2], where it is proved that for the 1-dimensional ring, we have:

- For $0 \leq r < 1$, $e_r(n) = \Omega(n^{\frac{1-r}{2-r}})$;
- For $r = 1$, $e_r(n) = \Theta(\log^2(n))$
- For $1 < r < 2$, $e_r(n) = O(n^{r-1})$;
- For $r = 2$, $e_r(n) = O(n\frac{\log(\log(n))}{\log(n)})$
- For $r > 2$, $e_r(n) = O(n)$.

It is likely that the proofs in [2] can be adapted to the 2-dimensional grid, but some additional work may be necessary to demonstrate the bounds' tightness. Moreover, our estimates may have missed some slower-than-polynomial variations. For example, the logarithms in the bounds for $r = 2$ in [2] may have a counterpart in the grid for $r = 3$. This would explain why the estimates are not sharp around that critical value (like for the case $r = 2$ and the term in $\log^2(n)$).

Remark

For $r \geq 3.5$, we have observed that $e_r(n) \approx \frac{2}{3}n$, which is the expected delivery path in absence of shortcuts: for these high values of r, almost all shortcuts link to an immediate neighbor of the current node, which make them useless, and the rare exceptions are so short that they make no noticeable difference.
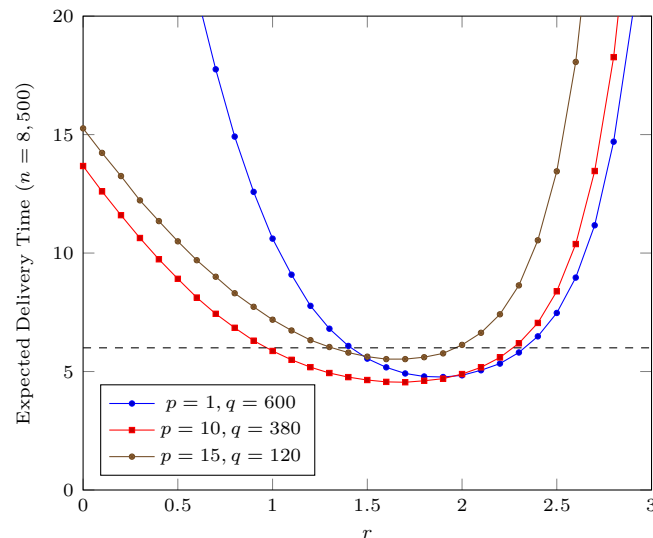
4.3   Six degrees of separation

What makes Milgram's experiments [17, 20] so famous is the surprising observation that only a few hops are required to transmit messages in social networks, a phenomenon called *six degrees of separation* in popular culture.

Yet, the values of $e_r(n)$ observed until now are quite far for the magic number six. For example, in Figure 5, the lowest value is 140. In addition to the asymptotic lack of robustness with respect to the exponent (discussed in Section 4.1), this may partially motivate the amount of work accomplished to increase the realism of the model and the performance of the routing algorithm.

In our opinion, a good model should be as simple as possible while being able to accurately predict the phenomenon that needs to be explained. We propose here to answer a simple question: is the genuine model of greedy routing in $G(n, r, p, q)$ a good model? Sure, it is quite simple. To discuss its accuracy, we need to tune the four parameters n, r, p and q to fit the conditions of Milgram's experiments as honestly as we can.

Size The experiments of Milgram were conducted in the United States, with a population of about 200,000,000 at the late sixties. All inhabitants were not susceptible to participate to the experiments: under-aged, undergraduate or disadvantaged people may be considered as *de facto* excluded from the experiments. Taking that into consideration, the correct n is probably somewhere in the range $[5000, 14000]$. We propose to set $n = 8,500$, which corresponds to about 72,000,000 potential subjects.

Exponent In [5], Kleinberg investigates how to relate the r-harmonic distribution with real-life observations. He surveys multiple social experiments and discusses the correspondence with the exponent of his model, which gives estimates of r between 1.75 and 2.2.

Neighborhood The default value $p = q = 1$ means that there are no more than five "acquaintances" per node. This is quite small compared to what is observed in real-life social networks. For example, the famous Dunbar's number, which estimates the number of *active* relationships, is 150 [4]. More recent studies seem to indicate that the average number of acquaintances is larger, ranging from 250 to 1500 (see [3, 16, 22] and references within). We propose to set p and q so that the neighborhood size $2p(p+1)+q$ is about 600, the value reported in [16]. Regarding the partition between local links (p) and shortcuts (q), we consider three typical scenarios:

- $p = 1$, $q = 600$ (shortcut scenario: the neighborhood is almost exclusively made of shortcuts, and local links are only here to ensure the termination of greedy routing).
- $p = 10$, $q = 380$ (balanced scenario).
- $p = 15$, $q = 120$ (local scenario, with a value of q not too far from Dunbar's number).

Having set all parameters, we can evaluate the performance of greedy routing. The results are displayed in Figure 9. We observe that the expected delivery time roughly stands between five and six for a wide range of exponents.

- $r \in [1.4, 2.3]$ for the shortcut scenario.
- $r \in [1.3, 2.3]$ for the balanced scenario.
- $r \in [1.3, 2]$ for the local scenario.

Except for the local scenario, which leads to slightly higher routing times for $r > 2$, the six degrees of separation are achieved for all values of r that are consistent with the observations surveyed in [5]. This allows to answer our question: the augmented grid proposed by Kleinberg is indeed a good model to explain the *six degrees of separation* phenomenon.

5   Conclusion

We proposed an algorithm to evaluate the performance of greedy routing in Kleinberg's grid. Fueled by a dynamic rejection sampling approach, the simulator based on our solution performs several orders of magnitude faster than previous attempts. It allowed us to investigate greedy routing under multiple perspective.

- We noted that the performance of greedy routing is less sensitive to the choice of the exponent r than predicted by the asymptotic behavior, even for very large grids.
- We observed that the bounds proposed in [12] are not tight except for $r = 0$ and $r = 2$. We conjectured that the tight bounds are the 2-dimensional equivalent of bounds proposed in [2] for the 1-dimensional ring.
- We claimed that the model proposed by Kleinberg in [12, 11] is a good model for the *six degrees of separation*, in the sense that it is very simple *and* accurate.

Our simulator is intended as a tool to suggest and evaluate theoretical results, and possibly to build another bridge between theoretical and empirical study of social systems. We hope that it will be useful for researchers from both communities. In a future work, we plan to make our simulator more generic so it can handle other types of graph and augmenting schemes.

References

1   Stavros Athanassopoulos, Christos Kaklamanis, Ilias Laftsidis, and Evi Papaioannou. An experimental study of greedy routing algorithms. In *High Performance Computing and Simulation (HPCS), 2010 International Conference on*, pages 150–156, June 2010.

2   Lali Barrière, Pierre Fraigniaud, Evangelos Kranakis, and Danny Krizanc. Efficient routing in networks with long range contacts. In *Proceedings of the 15th International Conference on Distributed Computing*, DISC '01, pages 270–284, London, UK, 2001.

3   Ithiel de Sola Pool and Manfred Kochen. Contacts and influence. *Social Networks*, 1:5–51, 1978.

4   R. I. M. Dunbar. Neocortex size as a constraint on group size in primates. *Journal of Human Evolution*, 22(6):469–493, June 1992.

5   David Easley and Jon Kleinberg. The small-world phenomenon. In *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*, chapter 20, pages 611–644. Cambridge University Press, 2010.

6   Pierre Fraigniaud, Cyril Gavoille, Adrian Kosowski, Emmanuelle Lebhar, and Zvi Lotker. Universal Augmentation Schemes for Network Navigability: Overcoming the $\sqrt{}(n)$-Barrier. *Theoretical Computer Science*, 410(21-23):1970–1981, 2009.

7   Pierre Fraigniaud, Cyril Gavoille, and Christophe Paul. Eclecticism shrinks even small worlds. *Distributed Computing*, 18(4):279–291, 2006.

8   Pierre Fraigniaud and George Giakkoupis. On the searchability of small-world networks with arbitrary underlying structure. In *Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC)*, pages 389–398, June 6–8 2010.

9   Pierre Fraigniaud and George Giakkoupis. Greedy routing in small-world networks with power-law degrees. *Distributed Computing*, 27(4):231–253, 2014.

10   Frigyes Karinthy. Láncszemek, 1929.

11   Jon Kleinberg. Navigation in a small world. *Nature*, August 2000.

12   Jon Kleinberg. The small-world phenomenon: An algorithmic perspective. In *in Proceedings of the 32nd ACM Symposium on Theory of Computing*, pages 163–170, 2000.

13 David Liben-Nowell, Jasmine Novak, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins. Geographic routing in social networks. *Proceedings of the National Academy of Sciences of the United States of America*, 102(33):11623–11628, 2005.

14 Gurmeet Singh Manku, Moni Naor, and Udi Wieder. Know thy neighbor's neighbor: The power of lookahead in randomized P2P networks. In *Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing (STOC)*, pages 54–63. ACM, 2004.

15 Chip Martel and Van Nguyen. Analyzing kleinberg's (and other) small-world models. In *Proceedings of the Twenty-third Annual ACM Symposium on Principles of Distributed Computing*, PODC '04, pages 179–188, New York, NY, USA, 2004. ACM.

16 Tyler H. McCormick, Matthew J. Salganik, and Tian Zheng. How many people do you know?: Efficiently estimating personal network size. *Journal of the American Statistical Association*, 105(489):59–70, 2010.

17 Stanley Milgram. The small world problem. *Psychology Today*, 67(1):61–67, 1967.

18 Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York, NY, USA, 2005.

19 William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. Minimization or maximization of functions. In *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, chapter 10. Cambridge University Press, New York, NY, USA, 3 edition, 2007.

20 Jeffrey Travers and Stanley Milgram. An experimental study of the small world problem. *Sociometry*, 32:425–443, 1969.

21 John von Neumann. Various techniques used in connection with random digits. *Nat. Bureau Standards*, 12:36–38, 1951.

22 Barry Wellman. Is Dunbar's number up? *British Journal of Psychology*, 2011.

A   Code for Expected Delivery Time (tested on Julia Version 0.4.5)

```julia
using StatsBase # For using Julia built-in sample function

function shortcuts_bulk(n, probas, bulk_size)
  radii = sample(1:(2*n-2), probas, bulk_size)
  shortcuts = Tuple{Int, Int}[]
  for i = 1:bulk_size
    radius = radii[i]
    angle = floor(4*radius*rand())-2*radius
    push!(shortcuts, ((radius-abs(angle)),
      (sign(angle)*(radius-abs(radius-abs(angle))))))
  end
  return shortcuts
end

# Estimates the Expected Delivery Time for G(n, r, p, q) over R runs
function edt(n, r, p, q, R)
  bulk_size = n
  probas = weights(1./(1:(2*n-2)).^(r-1))
  shortcuts = shortcuts_bulk(n, probas, bulk_size)
  steps = 0
  for i = 1:R
    # s: start/current node; a: target node; d: distance to target
    s_x, s_y, a_x, a_y = tuple(rand(0:(n-1), 4)...)
    d = abs(s_x - a_x) + abs(s_y - a_y)
    while d>0
      sh_x, sh_y = -1, -1 # sh will be best shortcut node
      d_s = 2*n # d_s will be distance from sh to a
      for j = 1:q # Draw q shortcuts
        ch_x, ch_y = -1, -1 # ch will be current shortcut
        c_s = 2*n # c_s will be distance from ch to a
        # Dynamic rejection sampling
        while (ch_x < 0 || ch_x >= n || ch_y < 0 || ch_y >= n)
          r_x, r_y = pop!(shortcuts)
          ch_x, ch_y = s_x + r_x,  s_y + r_y
          if isempty(shortcuts)
            shortcuts = shortcuts_bulk(n, probas, bulk_size)
          end
        end
        c_s = abs(a_x - ch_x) + abs(a_y - ch_y)
        if c_s < d_s # maintain best shortcut found
          d_s = c_s
          sh_x, sh_y = ch_x, ch_y
        end
      end
      if d_s < d-p # Follow shortcut if efficient
        s_x, s_y  = sh_x, sh_y
        d = d_s
      else # Follow local links
        d = d - p
        delta_x = min(p, abs(a_x - s_x))
        delta_y = p - delta_x
        s_x += delta_x*sign(a_x - s_x)
        s_y += delta_y*sign(a_y - s_y)
      end
      steps += 1
    end
  end
  steps /= R
  return steps
end
```