The Mathematica® Journal

Two Basic Results Concerning Random Walks on Graphs

Greg Markowsky

Two problems involving random walks on graphs are studied. First, the starting point from which a random walk on $\mathbb{Z}^2$ is most likely to hit a given point before another given point is determined. Second, the slowest mixing initial distribution under a random walk on a given finite graph is found.

Introduction

This article describes my investigation into several basic problems regarding random walks on graphs. On several occasions, I asked myself questions which my intuition failed to answer. I guessed at an answer, and spent some time in a fruitless attempt at proving that it was correct. Out of frustration I turned to computer simulations, only to discover that my guesses were faulty. Once I had the correct answer, I was able to supply the proofs. As every mathematician knows, it is much easier to solve a problem when you know the right answer ahead of time. This presentation is deliberately informal, as it represents the record of an actual investigation that took place, rather than a crafted paper. In fact, the notebook that I used to run my experiments has become the paper, with explanatory text added and unnecessary debris removed.

Probability That a Random Walk Hits One Point before Another

A graph G is a set of vertices V and a set of pairs of vertices E, known as edges. We will assume throughout that G is an undirected, simple graph. That is, no edges are of the form $\{i, i\}$, the edge $\{i, j\}$ is the same as the edge $\{j, i\}$, and the set of edges E contains no repeated elements. The degree d of a vertex v of a graph is the number of edges containing v. A random walk on G is the random movement of a particle from vertex to vertex, where at each step a particle at any vertex moves to an adjacent vertex, with the probability of moving to each adjacent vertex given by the reciprocal of the degree of the vertex. This movement is memoryless, in the sense that each step is independent of earlier steps.

One of the most interesting and fundamental questions in this field is the question of recurrence and transience in the integer lattice of dimension n. The basic and well-known result is that a random walk on $\mathbb{Z}^n$ is recurrent (returns infinitely often to any point) if and only if $n < 3$. See [1] for a beautiful proof of this involving resistance in electric circuits, as well as a more elementary proof. I have recently asked myself a related question. Given two fixed points a, b and a variable point x, let us find $P_x(b \text{ before } a)$, the probability that a random walk starting at x hits b before hitting a. In one dimension this question is quite easy, but in two dimensions, as far as I have been able to ascertain, no closed form for the solution is known. To circumvent the difficulty of this question, however, we can ask a simpler one. Namely, given a and b, let us determine the point x that maximizes $P_x(b \text{ before } a)$. I spent some time working in vain on this problem, before I realized that it would help to know what the answer is before attempting a proof. I wrote the following program.

```
q[0] = {{0, 0}};
q[n_] := Module[{tmp},
  tmp = {};
  Do[tmp = Join[tmp, {{n, 0} + i {-1, 1}}], {i, 0, n}];
  tmp = Join[tmp, {1, -1} * # & /@ tmp,
    {-1, 1} * # & /@ (Reverse /@ tmp),
    {-1, -1} * # & /@ (Reverse /@ tmp)];
  DeleteDuplicates[tmp]]

f[n_Integer, zeropt_List, onept_List] :=
 Module[{range, fun, tmprange, j, t},
  range = Join @@ (q /@ Range[0, n]);
  fun[{_, _}] := .5;
  (fun[#] = 0) & /@ zeropt;
  (fun[#] = 1) & /@ onept;
  j = 0;
  While[j < 300,
   tmprange = range;
   While[Length[tmprange] > 0,
    t = First[tmprange];
    If[FreeQ[zeropt, t] && FreeQ[onept, t],
     fun[t] =
      .25 (fun[t + {1, 0}] + fun[t + {-1, 0}] + fun[t + {0, 1}] +
         fun[t + {0, -1}])
    ];
    tmprange = Rest[tmprange];
   ];
   j = j + 1
  ];
```

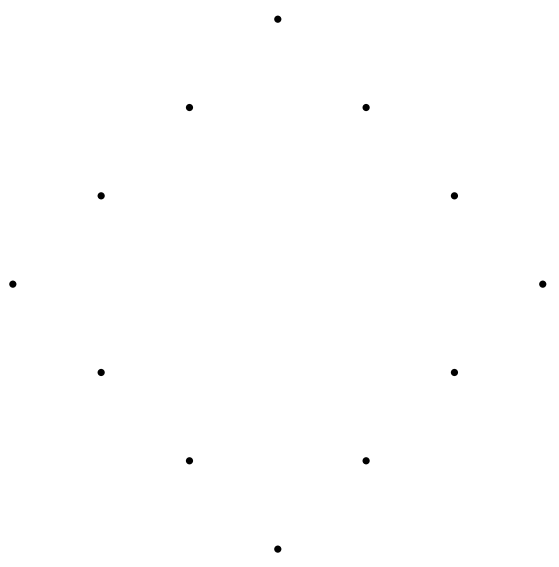
```
range = Complement[range, Join[onept, zeropt]];
Graphics[
 Join[Tooltip[Point[#], Text[ToString[fun[#]]]] & /@ range,
  Tooltip[{PointSize -> Large, Blue, Point[#]},
     Text[ToString[fun[#]]]] & /@ zeropt,
  Tooltip[{PointSize -> Large, Red, Point[#]},
     Text[ToString[fun[#]]]] & /@ onept]]
]
```

The function `q` is used to generate the correct range of points, a diamond-shaped region of radius `n`, and place the points in the right order. Here is an example.

```
q[3]
```

```
{{3, 0}, {2, 1}, {1, 2}, {0, 3}, {2, -1}, {1, -2},
 {0, -3}, {-1, 2}, {-2, 1}, {-3, 0}, {-1, -2}, {-2, -1}}
```

```
Graphics[Point@q[3]]
```

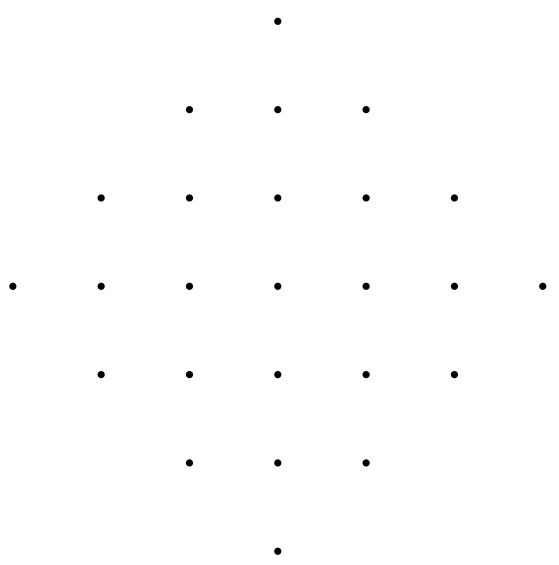


This nests the diamonds to create a square lattice.

```
Join @@ (q /@ Range[0, 3])
```

```
{{0, 0}, {1, 0}, {0, 1}, {0, -1}, {-1, 0}, {2, 0}, {1, 1},
 {0, 2}, {1, -1}, {0, -2}, {-1, 1}, {-2, 0}, {-1, -1},
 {3, 0}, {2, 1}, {1, 2}, {0, 3}, {2, -1}, {1, -2},
 {0, -3}, {-1, 2}, {-2, 1}, {-3, 0}, {-1, -2}, {-2, -1}}
```

```
Graphics[Point@%]
```

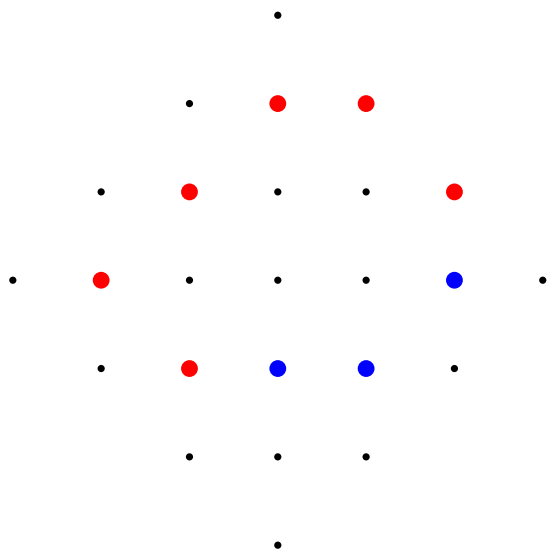


To understand the function `f` one must understand a bit of the mathematics of random walks. Due to the memoryless property of random walks, $g(x) = P_x(b \text{ before } a)$ is a *harmonic function*. That is, g is the average of its surrounding values,

$$g(x) = \frac{1}{4}\sum g(y) = \frac{1}{4}\left(g(x+(0,1)) + g(x+(0,-1)) + g(x+(-1,0)) + g(x+(1,0))\right),$$

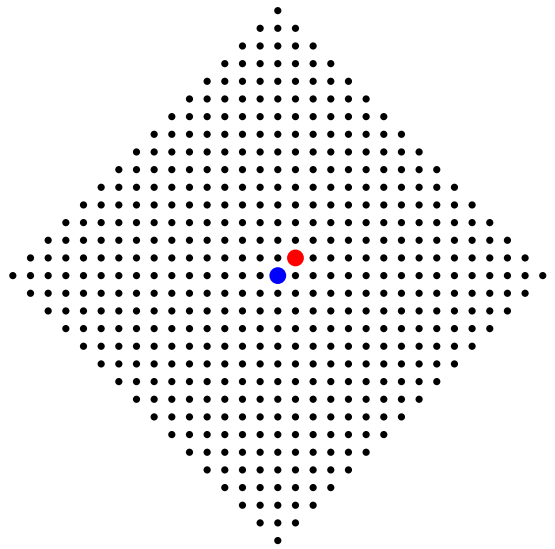
for all x in $\mathbb{Z}^2 - \{a, b\}$ and the sum is over all y adjacent to x. In fact, it is easy to see that g is the unique function harmonic on $\mathbb{Z}^2 - \{a, b\}$, such that $g(b) = 1$, $g(a) = 0$, and $g(x) \to 1/2$ as $x \to \infty$ (see [1]). The program `f` begins by setting $fun(b) = 1$, $fun(a) = 0$, and $fun(x) = 1/2$ for all other x. In each iteration through the `While` loop, the value of the function at each point is set to the average of the neighboring values. In this way the function `fun` becomes more and more nearly harmonic, and as a result closer to the values of our desired function $g(x)$. This process is described in [1], and just to check that `f` is working correctly I tested it on an example that appears in [1].

```
f[3, {{0, -1}, {1, -1}, {2, 0}},
 {{2, 1}, {1, 2}, {0, 2}, {-1, 1}, {-2, 0}, {-1, -1}}]
```



This exactly matches the values given in [1]. Confident that my function works well, I tested it on an example.

```
f[15, {{0, 0}}, {{1, 1}}]
```



And here we see the value of numerical experimentation. Lacking what I now know to be the proper intuition, I was foolishly confident that the maximum would occur at the point (2, 2), since this seems to be the point most "separated" from *a* by *b*. However, the simulation above clearly shows that this is not the case, and that in fact (2, 1) and (1, 2) are the maximal values. After playing with several other examples, which readers are encouraged

to do on their own, I realized that the maximal point is always one adjacent to b. Once that is realized, it is clear that it must be the point or points "farthest" from a, in some sense. One can define farthest in terms of the Euclidean metric, but it is actually more natural to define it in the way given in the statement of theorem 1.

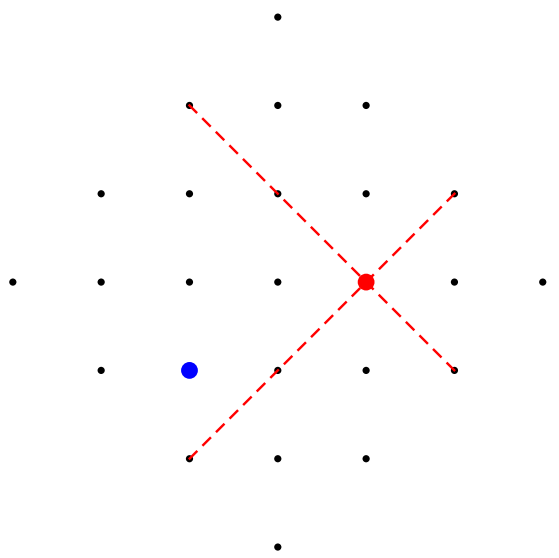
Theorem 1

The function g is maximized at one of the points adjacent to b. In order to determine the correct point, draw two lines l and m of slopes 1 *and* −1 *through b. This creates four half-planes. The point a lies in two of these half-planes, unless it lies on one of l or m, in which case a lies in only one half-plane. In case a lies in two half-planes, g is maximized at the unique point adjacent to b that does not share a half-plane with a. In case a lies in only one half-plane, g is maximized at the two points that do not lie in the half-plane containing a.*

Proof

Here is a sample situation.

```
qrange = Join @@ (q /@ Range[0, 3]);
Graphics[Join[Point[#] & /@ qrange,
  {PointSize -> Large, Blue, Point[{-1, -1}]},
  {PointSize -> Large, Red, Point[{1, 0}]},
  {Dashed, Thickness[Medium], Line[{{-1, 2}, {2, -1}}]},
  {Dashed, Line[{{2, 1}, {-1, -2}}]}]]
```



The statement of the theorem is that the maximum occurs at $(2, 0)$, since that is the point adjacent to $(1, 0)$ that does not lie in a half-plane with $(-1, -1)$. First let us prove that the maximum occurs at a point adjacent to b. Let U be the set of points adjacent to b. Then, for any x not in U,

$$P_x(b \text{ before } a) = \sum_{y \text{ in } U} P_x(y \text{ before } a \text{ and } U - \{y\})\, P_y(b \text{ before } a).$$

Thus, $P_x(b \text{ before } a)$ is seen to be a weighted average over U of the quantities $\{P_y(a \text{ before } b)\}_{y \in U}$ with weights $\{P_x(y \text{ before } a \text{ and } U - \{y\})\}_{y \in U}$. Since

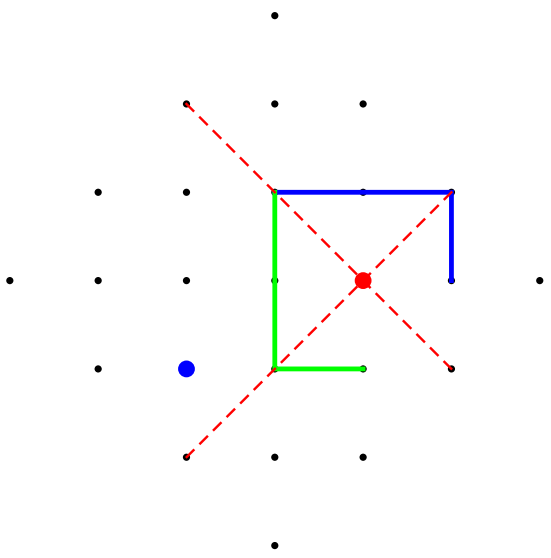
$$\sum_{y \text{ in } U} P_x(y \text{ before } a \text{ and } U - \{y\}) < 1,$$

it follows that

$$P_x(b \text{ before } a) < max_{y \in U}\, P_y(b \text{ before } a),$$

so that the maximum is obtained on U. Now, to determine at which point it is obtained, let us first define a *reflection* in this context. The reflection of a point x in the line l is the unique point x' such that $x\,x'$ is a line perpendicular to l with its midpoint on l. So, for instance, in the graphic above the reflection of $(1, 2)$ over l is the point $(-1, 0)$. If $P = p_1\, p_2 \ldots p_n$ is a sequence of points in $\mathbb{Z}^2$ such that p_i is adjacent to p_{i+1} for all i, then we call P a *path*. The reflection of a path $p_1\, p_2 \ldots p_n$ in l is the path $p'_1\, p'_2 \ldots p'_n$, where each p'_i is the reflection of p_i. So, in the graphic below, the blue path is a reflection in l of the green path.

```
qrange = Join @@ (q /@ Range[0, 3]);
Graphics[Join[Point[#] & /@ qrange,
  {PointSize -> Large, Blue, Point[{-1, -1}]},
  {PointSize -> Large, Red, Point[{1, 0}]},
  {Thick, Blue, Line[{{2, 0}, {2, 1}, {1, 1}, {0, 1}}]},
  {Thick, Green, Line[{{1, -1}, {0, -1}, {0, 0}, {0, 1}}]},
  {Red, Thickness[Medium], Dashed, Line[{{-1, 2}, {2, -1}}]},
  {Dashed, Line[{{2, 1}, {-1, -2}}]}]]
```



Now,

$$P_y(b \text{ before } a) = \lim_{n \to \infty} \frac{\{\text{number of distinct paths starting at } y \text{ of length } n \text{ that hit } b \text{ before } a\}}{\{\text{number of distinct paths starting at } y \text{ of length } n\}}.$$

Suppose that a lies in one of the half-planes formed by the line l and that y is a point adjacent to b that also lies in this half-plane. Let $P = p_1 \ldots p_n$ be any path starting at y that hits b before a. This path must hit l at some point, possibly at b. Let i be the smallest value such that $p_i \in l$. Then we set $P^o = p'_1 \ldots p'_{i-1}\, p_i\, p_{i+1} \ldots p_n$. That is, P^o is the reflection of P up until P and P' both hit l, from which point P and P^o coincide. P^o is then a path, beginning at y', that must hit b before a (note that for $j < i$, p_j lies in the half-plane not containing a). The map $P \to P^o$ is an injective map from the set of paths starting from y that hit b before a to the set of paths starting from y' that hit b before a. Thus, the number of distinct paths starting at y of length n that hit b before a is less than or equal to the number of distinct paths starting at y' of length n that hit b before a. It is in fact easy to see that this inequality is strict, since there are paths starting at y' that hit b before a, but whose reflections hit a before b. It follows that $P_y(b \text{ before } a) < P_{y'}(b \text{ before } a)$. Thus, the maximum occurs at the point or points adjacent to x that do not share a half-plane with a, and the theorem follows. ∎

Slowest Mixing Distribution

Suppose now that G is a finite graph. The question of hitting probabilities of points can be handled in a similar way to the above argument, where we still must look for harmonic functions on the graph. However, there is a related question that is a bit different, but quite interesting. Suppose, rather than a sole walker, we have a large number of random walkers, all moving at the same time. The object of interest will be the fraction of walkers that are at each vertex at any given time. Suppose G has n vertices, numbered from 1 to n. A *distribution vector* is a vector in $[0, 1]^n$ whose components sum to 1, representing a distribution of random walkers on the graph. Form an $n \times n$ matrix whose elements satisfy $a_{ij} = 0$ unless vertices i and j are adjacent, in which case $a_{ij} = 1$. This is known as the *adjacency matrix*. We will suppose below that the matrix is *regular* (every vertex has the same degree) and not *bipartite* (bipartite means that the vertices of the graph can be divided into two sets A and B, such that every vertex in A is adjacent only to vertices in B, and vice versa). Let P denote the *probability matrix* obtained by dividing all elements in the adjacency matrix by the degree of each vertex. If at time 0 the walkers have distribution v_0, then $v_1 = P\, v_0$ is the distribution at time 1, followed by $v_2 = P\, v_1 = P^2\, v_0$, etc. The distribution vector $(1/n, 1/n, \ldots, 1/n)$ gives the *stationary distribution*, the distribution that is unchanged (since each row of P sums to 1) when the random walk undergoes a step. The key question is, when we start with an arbitrary distribution, what happens? Do the walkers somehow approach the stationary distribution, or do they stay

disordered forever? It is a consequence of the Perron–Frobenius theorem that all eigenvalues of P are real and lie in the interval $[-1, 1]$, and that the eigenvectors $v_1, \ldots, v_n$ form an orthogonal basis of R^n. The largest eigenvalue λ_n is simple and equal to 1, with the stationary distribution $v_n = (1/n, 1/n, \ldots, 1/n)$ as an eigenvector. Any function v on the vertices of the graph can be expressed uniquely as $v = a_1 v_1 + \ldots + a_n v_n$, and $P v = \lambda_1 a_1 v_1 + \ldots + \lambda_{n-1} a_{n-1} v_{n-1} + a_n v_n$, $P^2 v = {\lambda_1}^2 v_1 + \ldots + {\lambda_{n-1}}^2 a_{n-1} v_{n-1} + a_n v_n$, and so on. If v is a distribution vector, then $a_n = 1$, since the basis of eigenvectors is orthogonal and the sum of the components of any vector orthogonal to v_n must be 0. We know that $v_1, \ldots, v_{n-1} \in [-1, 1]$. Let us assume that none of $v_1, \ldots, v_{n-1}$ are equal to -1 (in case one of them is, it can be shown that the graph is bipartite). In that case, an arbitrary distribution will converge geometrically to the stationary distribution, since ${\lambda_i}^m \to 0$ as $m \to \infty$. Thus, we see that the largest magnitude of the eigenvectors other than v_n determines the slowest possible rate of convergence of any distribution to the stationary one. This is all standard and well known in this field (see [2]). I was interested in using *Mathematica* to understand these theorems better, so I decided to try to determine the "slowest mixing" distribution, given any graph. Again, my intuition failed me, as I imagined that the slowest mixing distribution was likely to be complicated and difficult to obtain. The slowest mixing distribution will be the distribution vector $v = a_1 v_1 + \ldots + a_{n-1} v_{n-1} + v_n$ where a_1 is maximized, where λ_1 is chosen to be the largest eigenvalue (in magnitude) other than λ_n. If we define $f : R^{n-1} \to R^n$ by $(a_1, \ldots, a_{n-1}) \to a_1 v_1 + \ldots + a_{n-1} v_{n-1} + v_n$, then we seek to find the maximal value of a_1 that lies in $f^{-1}([0, 1]^n)$. Let us use `GraphData` to find interesting examples with which to work.

We only want regular graphs, and let us consider a number of vertices that is small enough with which to calculate but still large enough to be interesting; nine seems reasonable. Let us also take only graphs of degree four.

```
m = Select[GraphData["Regular"],
  GraphData[#, "VertexCount"] == 9 &&
    GraphData[#, "EdgeCount"] == 18 &]
```

```
{{Circulant, {9, {1, 2}}}, {Circulant, {9, {1, 3}}},
 {GeneralizedQuadrangle, {2, 1}}, {Quartic, {9, 1}},
 {Quartic, {9, 2}}, {Quartic, {9, 3}}, {Quartic, {9, 4}},
 {Quartic, {9, 5}}, {Quartic, {9, 7}}, {Quartic, {9, 8}},
 {Quartic, {9, 9}}, {Quartic, {9, 10}}, {Quartic, {9, 11}},
 {Quartic, {9, 12}}, {Quartic, {9, 13}}, {Quartic, {9, 15}}}
```

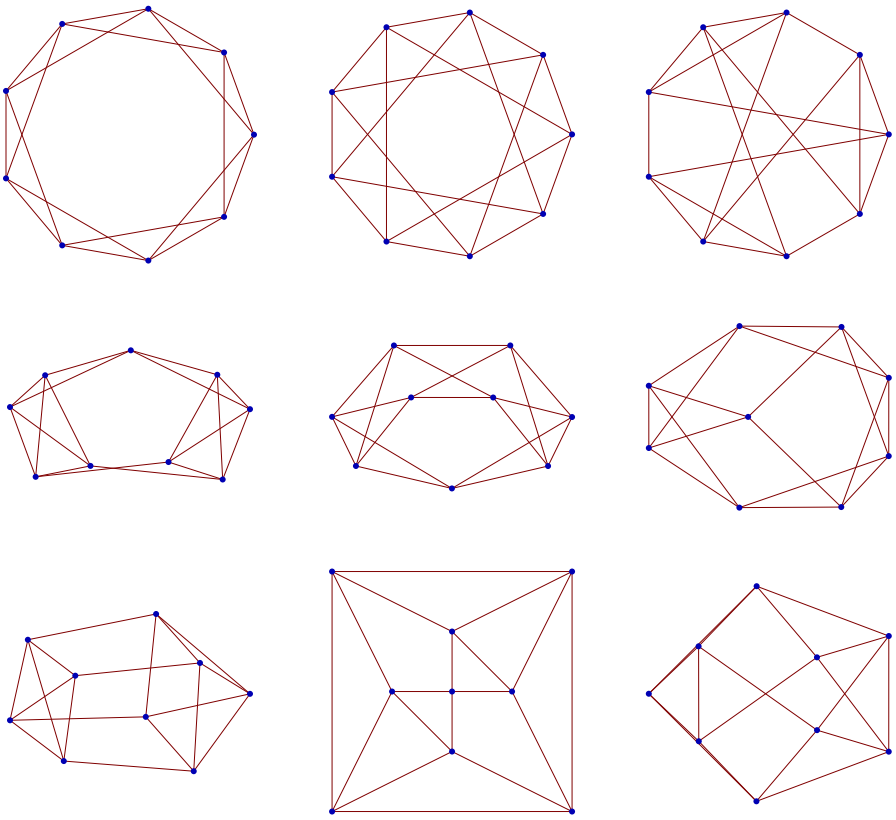
We get rid of the bipartite and disconnected graphs.

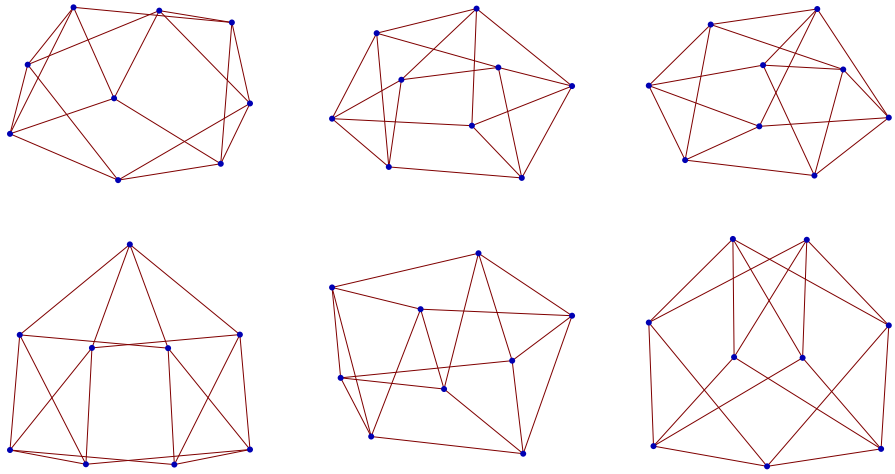
```
m =
 Select[m,
  ! GraphData[#, "Bipartite"] && GraphData[#, "Connected"] &]
```

```
{{Circulant, {9, {1, 2}}}, {Circulant, {9, {1, 3}}},
 {GeneralizedQuadrangle, {2, 1}}, {Quartic, {9, 1}},
 {Quartic, {9, 2}}, {Quartic, {9, 3}}, {Quartic, {9, 4}},
 {Quartic, {9, 5}}, {Quartic, {9, 7}}, {Quartic, {9, 8}},
 {Quartic, {9, 9}}, {Quartic, {9, 10}}, {Quartic, {9, 11}},
 {Quartic, {9, 12}}, {Quartic, {9, 13}}, {Quartic, {9, 15}}}
```

Here are the graphs.

```
Partition[GraphData[#, "Image"] & /@ m, 3] // Grid
```





Here are the spectrums, the sets of eigenvalues of adjacency matrices. They all have degree four as their largest eigenvalue, as they should, and all other eigenvalues lie in $(-4, 4)$.

```
spec = N /@ GraphData[#, "Spectrum"] & /@ m
```

```
{{-2., -2., -1.53209, -1.53209, -0.347296,
  -0.347296, 1.87939, 1.87939, 4.}, {-2.87939, -2.87939,
  -0.652704, -0.652704, 0.532089, 0.532089, 1., 1., 4.},
 {-2., -2., -2., -2., 1., 1., 1., 1., 4.},
 {-2., -2., -1.56155, -1., -1., 0., 1., 2.56155, 4.},
 {-3., -2., -1., -1., 0., 0., 1., 2., 4.},
 {-2.74796, -1.80194, -1.57018, -1., -0.445042, 0.353984,
  1.24698, 1.96416, 4.}, {-2.61803, -1.61803, -1.61803,
  -1.30278, -0.381966, 0.618034, 0.618034, 2.30278, 4.},
 {-2., -2., -2., -1., -1., 1., 1., 2., 4.},
 {-2.59615, -2., -1.53209, -1.18264, -0.347296,
  0.515722, 1.26308, 1.87939, 4.}, {-2.84224, -2.,
  -1.50694, -1., 0., 0.506942, 1., 1.84224, 4.},
 {-2.96416, -2.24698, -1.35398, -0.554958, 0., 0.570181,
  0.801938, 1.74796, 4.}, {-2.87939, -2.26308, -1.51572,
  -0.652704, 0.182644, 0.532089, 1., 1.59615, 4.},
 {-2.56155, -2., -2., -1., 0., 1., 1., 1.56155, 4.},
 {-3.30278, -1.61803, -1.61803, -0.618034,
  0.302776, 0.618034, 0.618034, 1.61803, 4.},
 {-3.56155, -2., -1., 0., 0., 0.561553, 1., 1., 4.},
 {-3., -2., -2., 0., 0., 1., 1., 1., 4.}}
```

Just as a check, let us make sure that all spectrums sum to 0. This is because the sum of the eigenvalues of a matrix is equal to the trace, and the trace of the adjacency matrix is 0.

```
Chop[Total /@ spec]
```

```
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

The sum of the squares of the eigenvalues gives twice the number of edges.

```
(Total[#^2] & /@ spec) / 2
```

```
{18., 18., 18., 18., 18., 18., 18.,
 18., 18., 18., 18., 18., 18., 18., 18., 18.}
```

The sum of the cubes of the eigenvalues gives six times the number of triangles in the graph.

```
Chop@(Total[#^3] & /@ spec) / 6
```

```
{9., 3., 6., 10., 6., 7., 8., 8., 7., 6., 5., 5., 6., 4., 2., 4.}
```

See [3] for a proof of the preceding facts. We see that the fourth graph in our list has many triangles, while the next to last has hardly any. It may be guessed that this property has something to do with how rapidly a distribution gets mixed on a graph, so let us closely consider these two graphs.

```
tri = m[[4]]
```

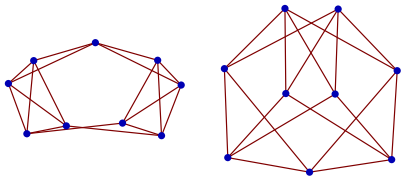
```
{Quartic, {9, 1}}
```

```
notri = m[[-2]]
```

```
{Quartic, {9, 13}}
```

Here is what they look like.

```
Row[GraphData[#, "Image"] & /@ {tri, notri}]
```



Recall that we are trying to find the distribution that converges most slowly to the uniform one. The *probability matrix* is defined to be the adjacency matrix divided by the degree, in this case four. In the context of random walks on regular graphs, the probability matrix is

more relevant than the adjacency matrix, as it represents the transition probabilities to the neighboring states of each step of the walk. Here are the all-important eigenvalues and eigenvectors for the probability matrix of the graph with many triangles.

```
Transpose[N@Eigensystem[GraphData[tri, "AdjacencyMatrix"]] /
   4] // Grid
```

```
    1.              {0.25, 0.25, 0.25, 0.25,
                     0.25, 0.25, 0.25, 0.25, 0.25}
 0.640388         {-0.25, -0.25, -0.195194, -0.195194,
                   0., 0.195194, 0.195194, 0.25, 0.25}
   -0.5              {0.25, 0.25, 0., -0.25,
                      -0.5, -0.25, 0., 0.25, 0.25}
   -0.5       {0., 0., 0.25, -0.25, 0., -0.25, 0.25, 0., 0.}
-0.390388        {-0.25, -0.25, 0.320194, 0.320194,
                  0., -0.320194, -0.320194, 0.25, 0.25}
  -0.25          {0., 0., 0., 0., 0., 0., 0., -0.25, 0.25}
  -0.25          {-0.25, 0.25, 0., 0., 0., 0., 0., 0., 0.}
   0.25               {0.25, 0.25, -0.5, -0.5,
                       1., -0.5, -0.5, 0.25, 0.25}
    0.        {0., 0., -0.25, 0.25, 0., -0.25, 0.25, 0., 0.}
```

We define `v[1]`, …, `v[9]` to be the eigenvectors.

```
{v[1], v[2], v[3], v[4], v[5], v[6], v[7], v[8], v[9]} =
 (N@Eigensystem[GraphData[tri, "AdjacencyMatrix"]] / 4)[[2]]
```

```
{{0.25, 0.25, 0.25, 0.25, 0.25, 0.25, 0.25, 0.25, 0.25},
 {-0.25, -0.25, -0.195194, -0.195194,
  0., 0.195194, 0.195194, 0.25, 0.25},
 {0.25, 0.25, 0., -0.25, -0.5, -0.25, 0., 0.25, 0.25},
 {0., 0., 0.25, -0.25, 0., -0.25, 0.25, 0., 0.},
 {-0.25, -0.25, 0.320194, 0.320194, 0., -0.320194, -0.320194,
  0.25, 0.25}, {0., 0., 0., 0., 0., 0., 0., -0.25, 0.25},
 {-0.25, 0.25, 0., 0., 0., 0., 0., 0., 0.},
 {0.25, 0.25, -0.5, -0.5, 1., -0.5, -0.5, 0.25, 0.25},
 {0., 0., -0.25, 0.25, 0., -0.25, 0.25, 0., 0.}}
```

Now, if the theory is correct, this should be an orthogonal basis of R^9, so that for any vector w we should have

$$w = \sum_{i=1}^{9} \frac{w \cdot v_i}{|v_i|^2} v_i.$$

This is a standard result from linear algebra, and holds in the much more general context of Hilbert spaces (see [4]). Let us verify it with the vector $w = (1, 2, 3, 4, 5, 6, 7, 8, 9)$.

```
w = {1, 2, 3, 4, 5, 6, 7, 8, 9};
Sum[(Dot[w, v[i]] / Norm[v[i]]^2) v[i], {i, 1, 9}]
```

```
{1., 2., 3., 4., 5., 6., 7., 8., 9.}
```

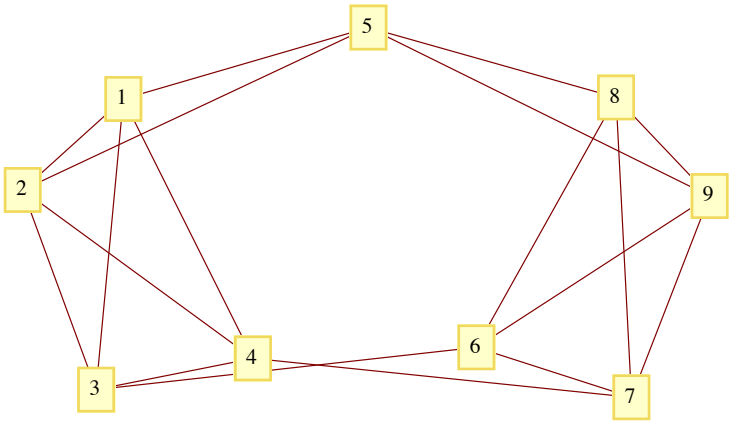
In order to find the slowest mixing distribution, then, we need to maximize the coefficient of v_2 in the representation of any distribution vector w in $[0, 1]^9$. A distribution vector is a vector w such that $w \cdot (1, 1, 1, 1, 1, 1, 1, 1, 1) = 1$. Thus, we are led to ask *Mathematica* the following.

```
NMaximize[{Dot[{v1, v2, v3, v4, v5, v6, v7, v8, v9}, v[2]],
  Dot[{v1, v2, v3, v4, v5, v6, v7, v8, v9},
    {1, 1, 1, 1, 1, 1, 1, 1, 1}] == 1, 0 ≤ v1 ≤ 1, 0 ≤ v2 ≤ 1,
  0 ≤ v3 ≤ 1, 0 ≤ v4 ≤ 1, 0 ≤ v5 ≤ 1, 0 ≤ v6 ≤ 1, 0 ≤ v7 ≤ 1,
  0 ≤ v8 ≤ 1, 0 ≤ v9 ≤ 1}, {v1, v2, v3, v4, v5, v6, v7, v8, v9}]
```

```
{0.25, {v1 → 0., v2 → 0., v3 → 0., v4 → 0.,
  v5 → 0., v6 → 0., v7 → 0., v8 → 1., v9 → 0.}}
```

Interestingly enough, the slowest mixing distribution occurs when we concentrate the entire mass of walkers on vertex 8. But which is vertex 8? Here is the graph with the vertices labeled.

```
GraphData[tri, "LabeledImage"]
```



There does not seem to be anything too special about vertex 8, and considering that $v_2 = \{-0.25, -0.25, -0.195194, -0.195194, 0., 0.195194, 0.195194, 0.25, 0.25\}$, we see that vertices 1, 2, and 9 would have fared equally well.

```
v[2]
```

```
{-0.25, -0.25, -0.195194, -0.195194,
 0., 0.195194, 0.195194, 0.25, 0.25}
```

Let us now see how the other graph works.

```
Transpose[
  N@Eigensystem[GraphData[notri, "AdjacencyMatrix"] / 4]] //
 Grid
```

```
    1.              {1., 1., 1., 1., 1., 1., 1., 1., 1.}
-0.890388          {0., 0.780776, -0.780776,
                    0.780776, -0.780776, -1., -1., 1., 1.}
  -0.5             {4., -2., -2., -2., -2., 1., 1., 1., 1.}
  -0.25             {0., 1., -1., -1., 1., 0., 0., 0., 0.}
   0.25            {-2., -1., -1., 0., 0., 1., 1., 1., 1.}
   0.25             {0., -1., -1., 1., 1., 0., 0., 0., 0.}
 0.140388           {0., -1.28078, 1.28078,
                     -1.28078, 1.28078, -1., -1., 1., 1.}
    0.              {0., 0., 0., 0., 0., 0., 0., -1., 1.}
    0.              {0., 0., 0., 0., 0., -1., 1., 0., 0.}
```

```
{v[1], v[2], v[3], v[4], v[5], v[6], v[7], v[8], v[9]} =
 (N@Eigensystem[GraphData[notri, "AdjacencyMatrix"]] / 4)[[
  2]]
```

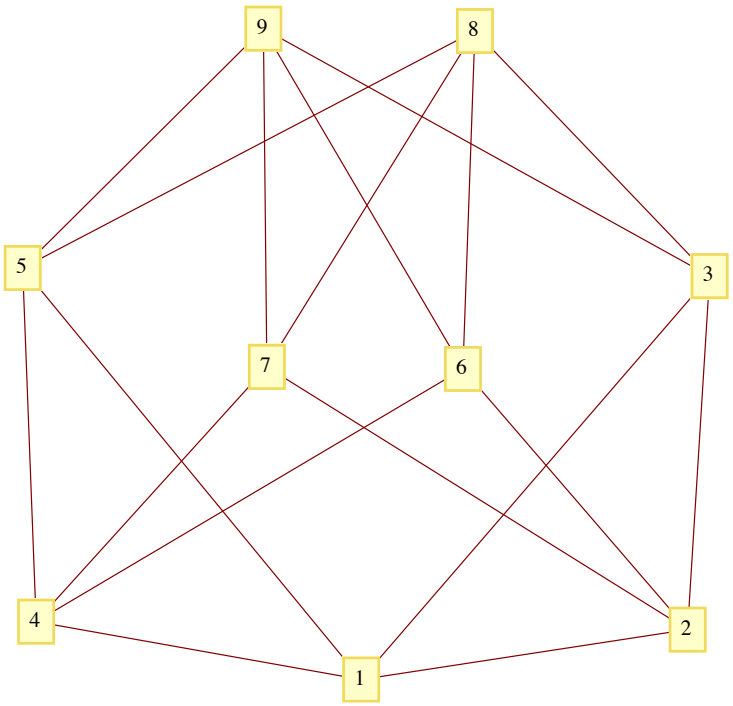
```
{{0.25, 0.25, 0.25, 0.25, 0.25, 0.25, 0.25, 0.25, 0.25},
 {0., 0.195194, -0.195194, 0.195194,
  -0.195194, -0.25, -0.25, 0.25, 0.25},
 {1., -0.5, -0.5, -0.5, -0.5, 0.25, 0.25, 0.25, 0.25},
 {0., 0.25, -0.25, -0.25, 0.25, 0., 0., 0., 0.},
 {-0.5, -0.25, -0.25, 0., 0., 0.25, 0.25, 0.25, 0.25},
 {0., -0.25, -0.25, 0.25, 0.25, 0., 0., 0., 0.},
 {0., -0.320194, 0.320194, -0.320194, 0.320194, -0.25, -0.25,
  0.25, 0.25}, {0., 0., 0., 0., 0., 0., 0., -0.25, 0.25},
 {0., 0., 0., 0., 0., -0.25, 0.25, 0., 0.}}
```

```
NMaximize[{Dot[{v1, v2, v3, v4, v5, v6, v7, v8, v9}, v[2]],
  Dot[{v1, v2, v3, v4, v5, v6, v7, v8, v9},
    {1, 1, 1, 1, 1, 1, 1, 1, 1}] == 1, 0 ≤ v1 ≤ 1, 0 ≤ v2 ≤ 1,
  0 ≤ v3 ≤ 1, 0 ≤ v4 ≤ 1, 0 ≤ v5 ≤ 1, 0 ≤ v6 ≤ 1, 0 ≤ v7 ≤ 1,
  0 ≤ v8 ≤ 1, 0 ≤ v9 ≤ 1}, {v1, v2, v3, v4, v5, v6, v7, v8, v9}]
```

```
{0.25, {v1 → 0., v2 → 0., v3 → 0., v4 → 0.,
  v5 → 0., v6 → 0., v7 → 0., v8 → 1., v9 → 0.}}
```

Here, $v(2) = \{0., 0.195194, -0.195194, 0.195194, -0.195194, -0.25, -0.25, 0.25, 0.25\}$, so we see that vertices 6, 7, and 9 would have worked equally well. Here is the labeled graph.

```
GraphData[notri, "LabeledImage"]
```



Having worked through all of this, it is now clear which is the slowest mixing distribution. One can simply take the maximal component of v_2 and place all of the mass on that vertex. One could also distribute the mass over several vertices with maximal size and the same sign, for instance over vertices 8 and 9 in the `notri` graph. We are led once again to the correct theorem.

Theorem 2

Given the conditions as above, the slowest mixing distribution on the vertices of a graph is the one obtained by putting a unit mass on one of the vertices that corresponds to a maximal component of the eigenvector associated with the second largest eigenvalue (in absolute value) of the probability matrix of the graph. This distribution is not necessarily unique, as there may be more than one maximal distribution, and an equally slow distribution can be obtained by distributing the unit mass in any way over a set of maximal components of the same sign.

Acknowledgments

This work was supported by the Priority Research Centers Program through the National Research Foundation of Korea (NRF), funded by the Ministry of Education, Science and Technology (Grant #2009-0094070). I would like to thank Professor Sangu Lee of Sungkyunkwan University for the invitation that led to the writing of this paper.

References

[1] P. Doyle and L. Snell, *Random Walks and Electric Networks,* Washington, D.C.: Mathematical Association of America, 1984.

[2] J. Doob, *Stochastic Processes*, New York: Wiley, 1953.

[3] D. West, *Introduction to Graph Theory*, Upper Saddle River, NJ: Prentice Hall Publishing, 2001.

[4] G. Folland, *Real Analysis: Modern Techniques and Their Applications*, 2nd ed., New York: Wiley, 1999.

G. Markowsky, "Two Basic Results Concerning Random Walks on Graphs," *The Mathematica Journal*, 2011. dx.doi.org/doi:10.3888/tmj.13–16.

About the Author

Greg Markowsky is currently a lecturer in mathematics at Monash University. He does research on probability theory, complex analysis, and geometry. He also worked for two years at Wolfram Research on the WolframAlpha project.

Greg Markowsky
School of Mathematical Sciences
Monash University
Building 28
Wellington Road
Clayton
Victoria 3800, Australia 9905-4487
gmarkowsky@gmail.com