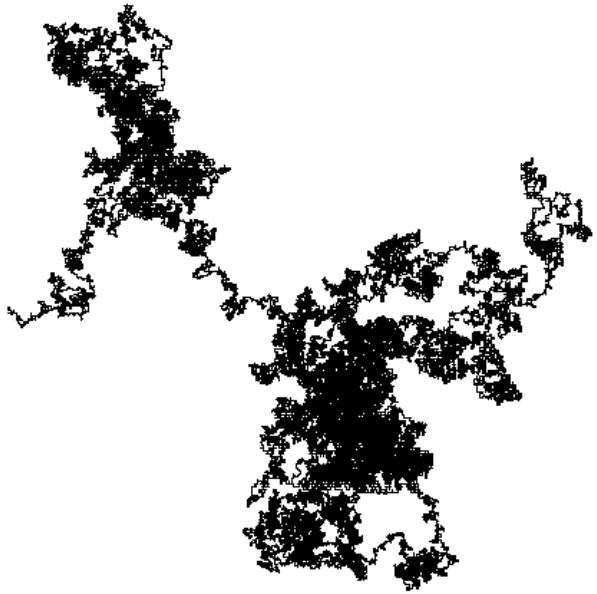
Random Walks

Dieter W. Heermann

Monte Carlo Methods

2015

- Perhaps the most straight-forward application of probability and Monte Carlo simulation can be seen in the *random walk model*.
- Problem: A man starts from a point O and walks l yards in a straight line; he then turns through any angle whatever and walks another l yards in a second straight line. He repeats this process N times. I require the probability that after N of these stretches he is at a distance between r and $r + \delta r$. (Nature, 27 July 1905, Karl Pearson).



- Lord Rayleigh pointed out the connection between this problem and an earlier paper of his (Rayleigh) published in 1880 concerned with sound vibrations. Rayleigh pointed out that, for large values of N, the answer is given by

$$\frac{2}{Nl^2} e^{-r^2/Nl^2} r\delta r \quad .$$

- 1919-21 the lattice random walk or Polya walk was introduced by George Polya.

There are numerous applications:

Applications

The following are the applications of random walk:

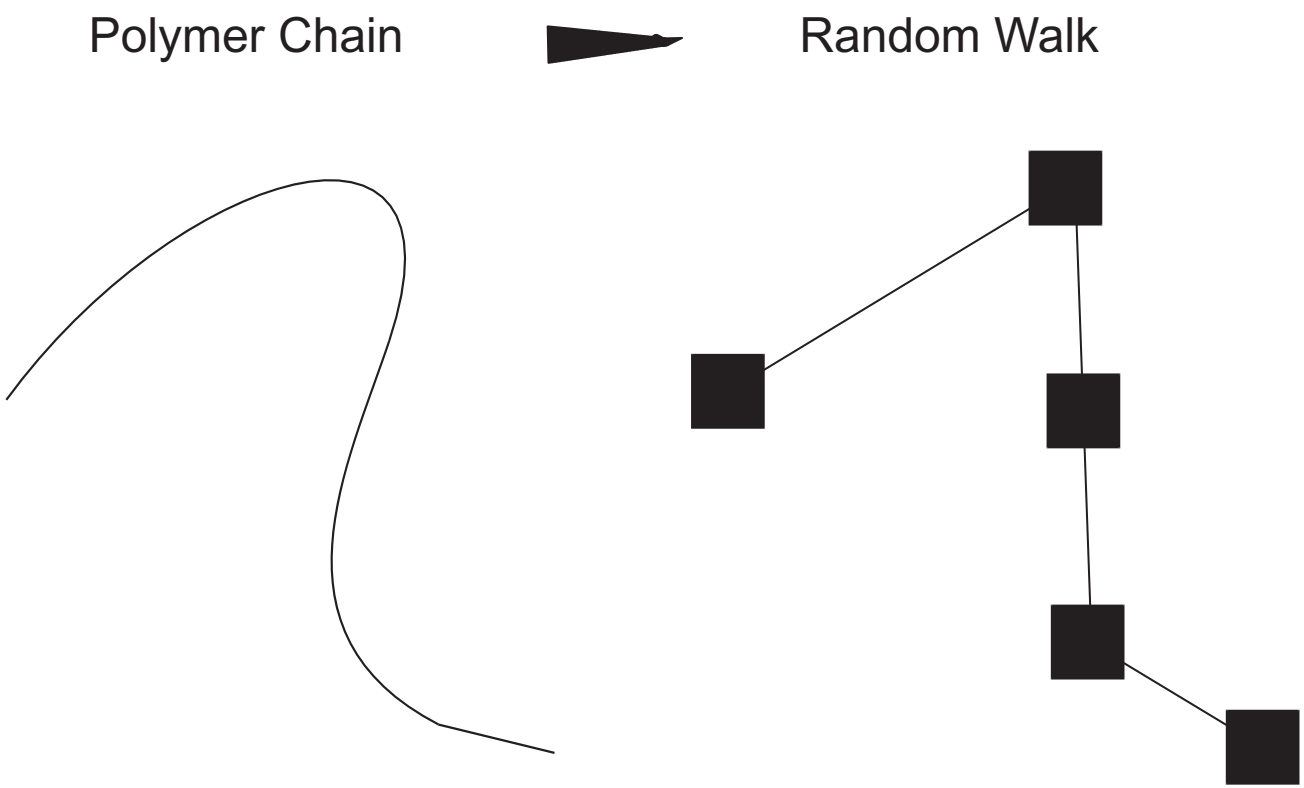
- In economics, the "random walk hypothesis" is used to model shares prices and other factors. Empirical studies found some deviations from this theoretical model, especially in short term and long term correlations. See share prices.
- In population genetics, random walk describes the statistical properties of genetic drift
- In physics, random walks are used as simplified models of physical Brownian motion and the random movement of molecules in liquids and gases. See for example diffusion-limited aggregation.
- In mathematical ecology, random walks are used to describe individual animal movements, to empirically support processes of biodiffusion, and occasionally to model population dynamics.
- Also in physics, random walks and some of the self interacting walks play a role in quantum field theory.
- In polymer physics, random walk describes an ideal chain. It is the simplest model to study polymers.
- In other fields of mathematics, random walk is used to calculate solutions to Laplace's equation, to estimate the harmonic measure, and for various constructions in analysis and combinatorics.
- In computer science, random walks are used to estimate the size of the Web. In the World Wide Web conference-2006, bar-yossef et al. published their findings and algorithms for the same. (This was awarded the best paper for the year 2006).

In all these cases, random walk is often substituted for Brownian motion.

- In brain research, random walks and reinforced random walks are used to model cascades of neuron firing in the brain.
- In vision science, fixational eye movements are well described by a random walk.
- In psychology, random walks explain accurately the relation between the time needed to make a decision and the probability that a certain decision will be made. (Nosofsky, 1997)
- Random walk can be used to sample from a state space which is unknown or very large, for example to pick a random page off the internet or, for research of working conditions, a random illegal worker in a given country.
- When this last approach is used in computer science it is known as Markov Chain Monte Carlo or MCMC for short. Often, sampling from some complicated state space also allows one to get a probabilistic estimate of the space's size. The estimate of the permanent of a large matrix of zeros and ones was the first major problem tackled using this approach.
- In wireless networking, random walk is used to model node movement.
- Bacteria engage in a biased random walk.
- Random walk is used to model gambling.
- In physics, random walks underlying the method of Fermi estimation.
- During World War II a random walk was used to model the distance that an escaped prisoner of war would travel in a given time.

Click here to view the wikipedia list

One interpretation and use is for polymers:



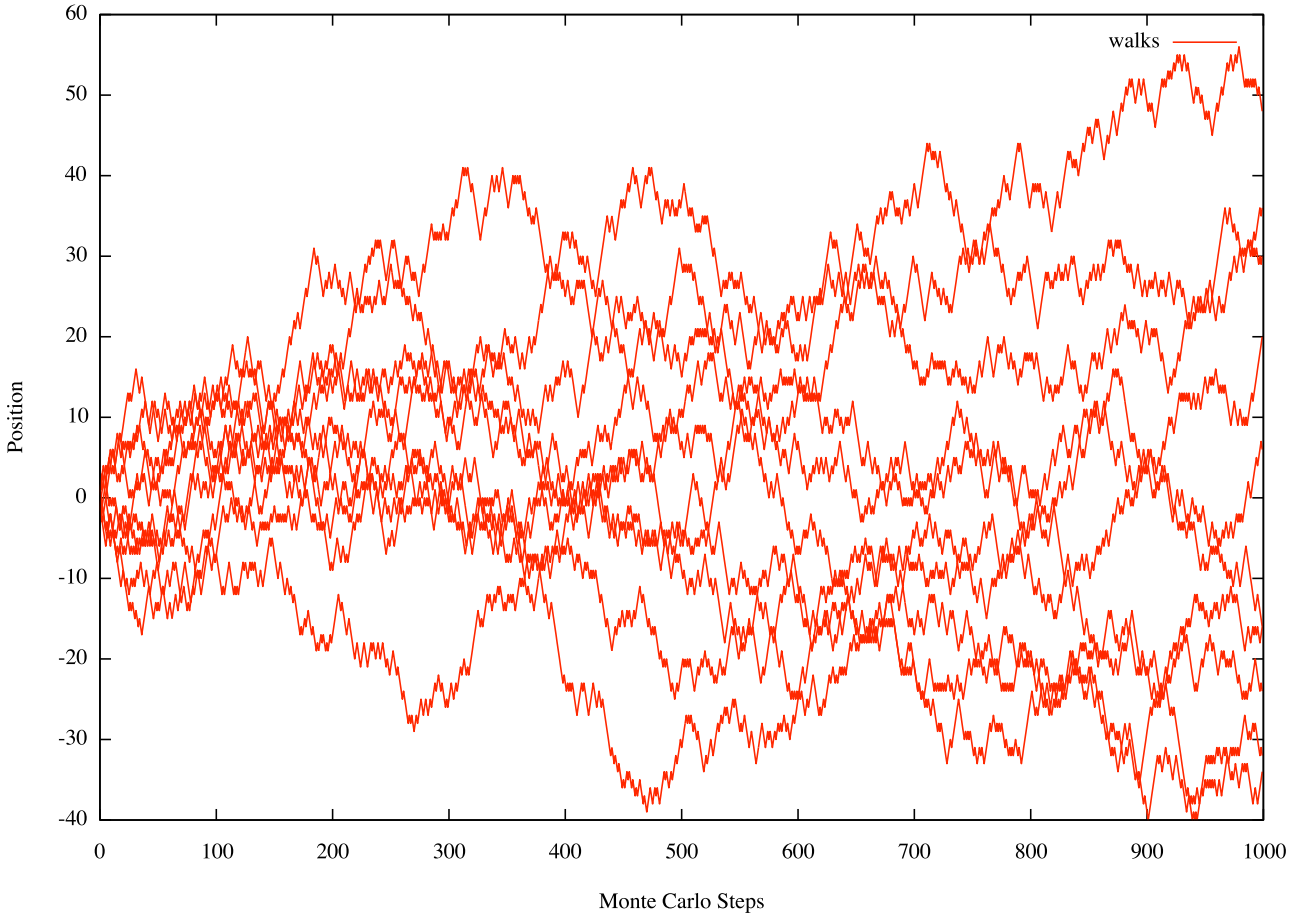
Random Walk

- We assume a lattice. For simplicity we take a simple square lattice.
- On this lattice a particle or walker is placed. The walker regards this initial position as the origin.
- The walker draws a random number and decides, according to the drawn random number, to go to a new position on the lattice.
- The new position must be one of the nearest neighbours, and each of the neighbours has the same probability to be visited.
- Once he is at the new position, the walker regards this position as his new origin. In other words, he immediately forgets where he came from.
- Every step is made as if it is the first step.
- All steps are then independent of each other.

- It is assumed that in the array random are stored numbers which are uniformly distributed in the interval $(0, 1)$.
- A random number from the array is then multiplied by 4 and converted to an integer value. This integer value can either be $0, 1, 2$ or 3 labeling the four possible directions or nearest neighbours on the square lattice.
- The numbers $0, 1, 2$ and 3 are uniformly distributed as long as the numbers in the array random are so distributed. Depending on the direction the random number points to, the walker occupies the appropriate position on the lattice by increasing or decreasing the x or y variable.
- The variables xn and yn hold the new position of the random walker.

```
/* ---- Choose a new nn site ---- */
i = floor(random[index++]* 4.0);
switch (i) {
   case 0: xn = x-1;
           yn = y;
           break;
   case 1: yn = y-1;
           xn = x;
           break;
   case 2: yn = y+1;
           xn = x;
           break;
   case 3: xn = x+1;
           yn = y;
           break;
} /* ---- switch i ---- */
```

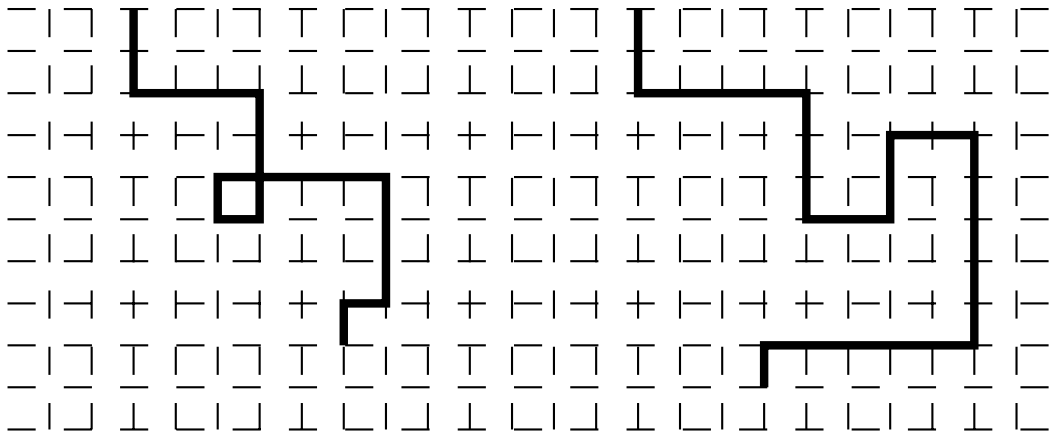
1D Random Walks



- Let us assume that the walker performed N steps. This constitutes one realization of a random walk.
- We may now be interested in computing properties of such a walk. From just one realization we cannot draw any conclusion since the walk may be atypical. We need to generate many walks, calculate for every walk the desired property and then average over the results.
- The point which we want to make is that the generation of the samples, i.e., all the realizations of random walks are generated independently. Let A_i be the observable property computed for the i-th realization of a random walk. We define the average, or expectation value for the observable A, denoted by $<A>$, as the arithmetic mean over all A_i

$$<A> \quad = \frac{1}{n} \sum_{i=1}^{n} A_i \quad . \tag{1}$$

Self-Avoiding Random Walks



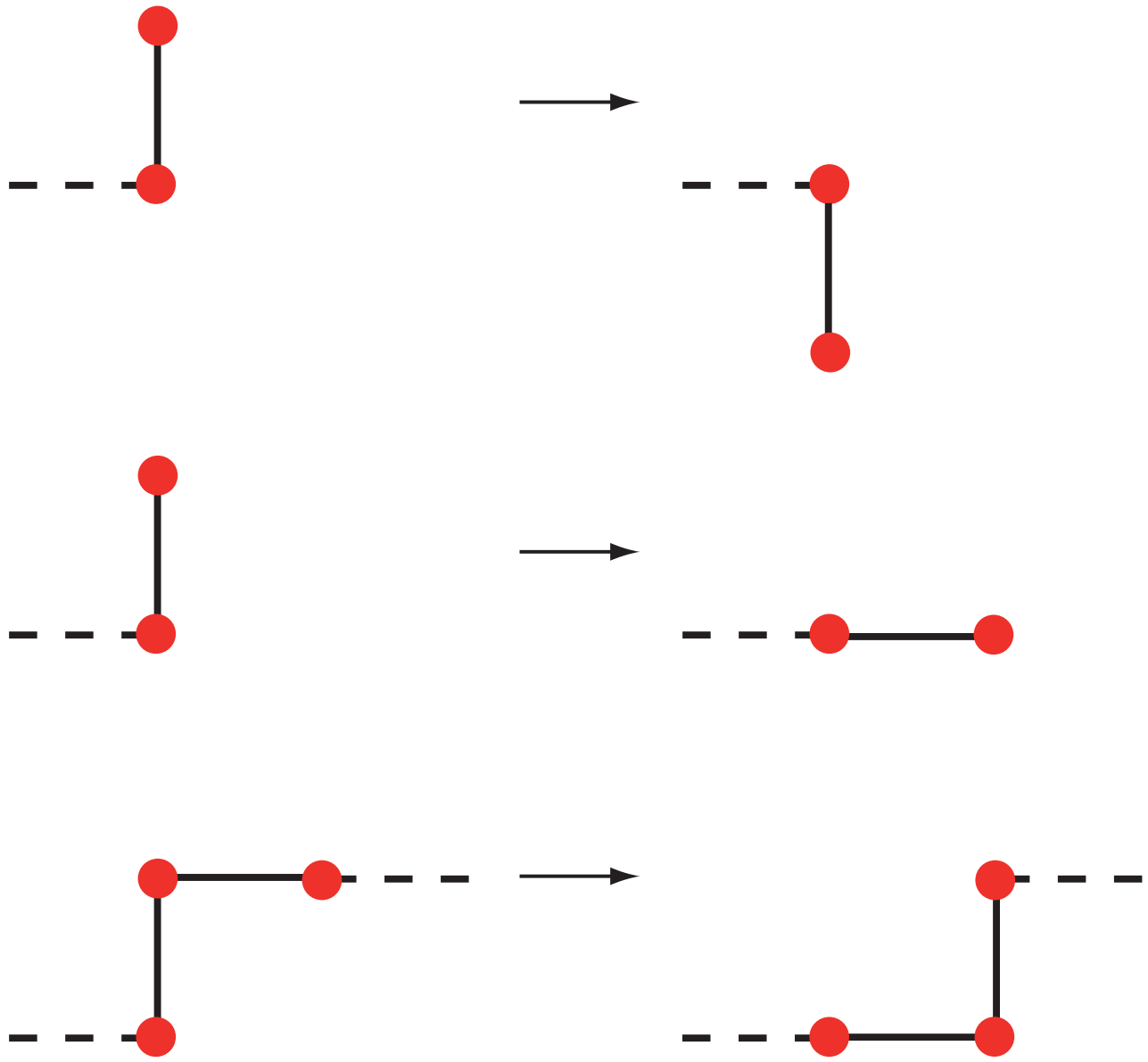
Not a SAW SAW

Simple Sampling of Self-Avoiding Random Walks

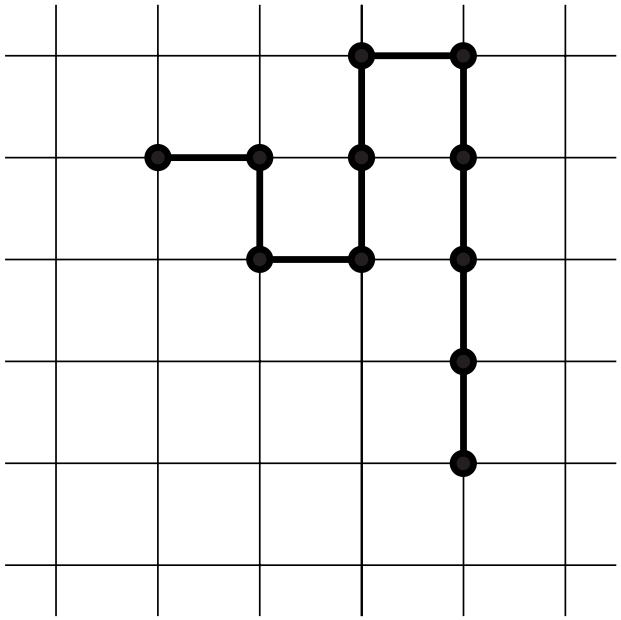
```
while ( sample < sample_size ) {

  /* ==== Reset the walker to the origin ==== */

  w[0][0]   = xc;
  w[0][1]   = yc;
  x         = xc;
  y         = yc;
  l         = 0;
  occupancy = 0;
  walk++;

  return_code = r250( N,ran,mf);

  while ( (l < N)  &&  (occupancy == 0) )  {
    d  = ran[l] * 4;
    switch (d) {
        case 0:  x++;
                 break;

        case 1:  y++;
                 break;

        case 2:  x--;
                 break;

        case 3:  y--;
                 break;

    }

    if ( ( x < 0 ) || ( x == L ) || ( y < 0 ) || ( y == L ) ) {
        /* Random walker not on the lattice */
        exit(-1);
    }

    if ( g[x][y] < walk ) {
        g[x][y] = walk;
        l++;
        w[l][0] = x;
        w[l][1] = y;
        occupancy = 0;
    }
    else {
        occupancy = 1;
    }
  }

  /* ==== Now check if a SAW was generated. If yes, then ==== */
  /* ==== we do the analysis, else we must try again     ==== */

  if ( l == N ) {
    /* ---- we can compute the end-to-end distance etc. ---- */

    x = xc - w[N-1][0];
    y = yc - w[N-1][1];
    end_to_end += x*x + y*y;
```

The code can be found here

- Performing the simple sampling simulation it becomes immediately evident that we have a problem with the simple sampling technique for the self-avoiding random walk model.
- As we increase the number of steps the walker should travel, it becomes harder and harder to find a walk. In almost all cases the walk terminates earlier because there is a violation of the self-avoiding condition! (attrition problem).
- This shows that the simple sampling, even though being the simplest and perhaps even most powerful method has clear limitations.



Reptation Algorithm



1: Assume that we have generated a random walk.
2: Choose one of the end points at random and delete this point.
3: Choose one the end points at random.
4: Add the delete point to the choosen end with a random direction.

Pivot Algorithm

- Let W denote the set of self-avoiding walks of length N on a lattice λ.
- Further let $G(\lambda)$ be the group of lattice symmetries.
- The pivot algorithm [1] takes a self-avoiding random walk and pivots the walk to generate a new walk from the set W such the sequence of generated walks yields a Markov chain which is aperiodic and irreducible with uniform stationary distribution π.

- Let W denote the set of self-avoiding walks of length N on a lattice λ.
- Further let $G(\lambda)$ be the group of lattice symmetries.
- The pivot algorithm [1] takes a self-avoiding random walk and pivots the walk to generate a new walk from the set W such the sequence of generated walks yields a Markov chain which is aperiodic and irreducible with uniform stationary distribution π.

Pivot Algorithm

1: Start with a self-avoiding walk $\omega_0 \in W$.
2: Next choose an integer i uniformly from the set $\{0, 1, 2, ..., N-1\}$. The site connected with this index is the pivot site $x = \omega_t(i)$.
3: Select a lattice symmetry g uniformly from the symmetry group G.
4: Set $\bar{\omega}(k) = \omega_t(k)$ for $k \leq i$, and $\bar{\omega}(k) = g(\omega_t(k))$ for $k \rangle i$.
5: **if** $\bar{\omega}$ is self-avoiding **then**
6: $\quad \omega_{t+1} = \bar{\omega}$.
7: **else**
8: $\quad$ let $\omega_{t+1} = \omega_t$.
9: $\quad$ Goto 2. for the next generation $t := t + 1$.
10: **end if**

- The sequence $\{\omega_t\}$ is aperiodic and irreducible with uniform stationary distribution π.
- The sequence further is reversible

$$\pi(\omega_i)P(\omega_i,\omega_j) = \pi(\omega_j)P(\omega_j,\omega_i) \quad . \tag{2}$$

Since π is uniform, we need to show that P is symmetric. Suppose there are m ways to move, with one pivot, from a self-avoiding walk ω to another self-avoiding walk $\bar{\omega}$. For $i = 1, 2, ..., m$, consider the pairs (x_i, g_i). Each pair gives a transition, using the pivot algorithm from ω to $\bar{\omega}$.

Thus,

$$P(\omega, \bar{\omega}) = \sum_{i=1}^{m} P(g = g_i) \cdot P(x = x_i) \quad . \tag{3}$$

Notice that the pairs (xi, g_i^{-1}), for $i = 1, 2, ..., m$ give one-step transitions from $\bar{\omega}$ and that $P(g = g_i) = P(g = g_i^{-1})$ because g is chosen uniformly. Therefore

$$P(\omega, \bar{\omega}) = \sum_{i=1}^{m} P(g = g_i) \cdot P(x = x_i) = \sum_{i=1}^{m} P(g = g_i^{-1}) \cdot P(x = x_i) = P(\bar{\omega}, \omega) \tag{4}$$

 N. Madras and A.D Sokal, *The pivot algorithm: a highly ecient Monte Carlo method for the self-avoiding walk*, J. Stat. Phys. 50: 109-186, (1988)