Hand-in 1

By Mark Emilson (ma2757em-s)
In collaboration with Douglas Ihre

1. Single-particle random walk

A. Average return time from simulations

To calculate the average time it takes for a particle that starts in node a to leave the node and return, simulations with 10 000 particles were performed. These particles each started in node a and from there the next node was chosen at random, in accordance with the transition rate matrix Λ. This was done until the particle returned to node a. Before each step, the time spent in the current node was recorded by adding the time tNext to the total time. tNext is calculated by,

$$tNext = -ln(R)/w^*$$

were, R is a uniformly distributed random variable between 0 and 1 and w^* is the maximum out-degree of Λ.

After doing this for 10 000 particles, the average return time was calculated to 6.7466.

B. Theoretical return time

To calculate the return time theoretically, the following formula was used,

$$E[T_i] = \frac{1}{w_i \pi_i}$$

where, π_i is the marginal distribution and w_i is the out-degree of the i-th node of Λ. The theoretical result was calculated to 6.75, which means that the simulations performed in 1a gave an answer close to the correct value.

C. Average hitting-time from simulations

The average hitting time was simulated the same way as in 1a, but with the difference that the particle did not return to the same node but was to reach a different specific node. When the particle reached this node the simulation was done. The average time it took for the different particles to move from node o to node d was 8.7847.

D. Theoretical hitting-time

The theoretical hitting-time was calculated with the following formula,

$$\tau_i^S = \frac{1}{\omega_i} + \sum_{j \in \chi} P_{ij} \tau_j^S$$

where, ω_i is the out-degree of the i-th node of Λ, P_{ij} is the normalized weight matrix and τ^S_j is the hitting time from j to S. Since τ^d_d is 0, this leads to four equations with four unknowns. The result from the equation system gave τ^d_o = 8.7857 which again is close to the simulated value.

2. Graph coloring and network games

A.

During this exercise the coloring of a line graph with 10 nodes was studied to see if it could reach a stationary value. The initial state of the graph was that all nodes were red, and from that stage they could update their color to green or have the same color. Different nodes were chosen randomly with a uniform distribution to update color. The probability of a certain node getting a certain color was calculated with the following equation,

$$P(X_i(t+1) = a \mid X(t), I(t) = i) = \frac{e^{-\eta(t)\sum_j W_{ij} c(a, X_j(t))}}{\sum_{s \in \mathcal{C}} e^{-\eta(t)\sum_j W_{ij} c(s, X_j(t))}},$$

where, the cost function c, is given by,

$$c(s, X_j(t)) = \begin{cases} 1 & \text{if } X_j(t) = s \\ 0 & \text{otherwise.} \end{cases}$$

and ηis given by

$$\eta(t) = \frac{t}{100}.$$

where, t is the amount of iteration performed. This means that every node will get a higher probability of updating its color to a color that its neighbors do not have. The graph will reach a stationary point if each node does not have any neighbours with the same color. This can be evaluated with the following formula,

$$U(t) = \frac{1}{2} \sum_{i,j \in \mathcal{V}} W_{ij}\, c(X_i(t), X_j(t)),$$

When U(t)=0, the graph has reached a stationary point where no node will change its color. The results from these iterations can be seen in Figure 1. The time it takes to converge depends both on the randomness in the choice of node and in the choice of color. For the simulation in Figure 1 it took around 80 iterations before the graph reached a stationary point.

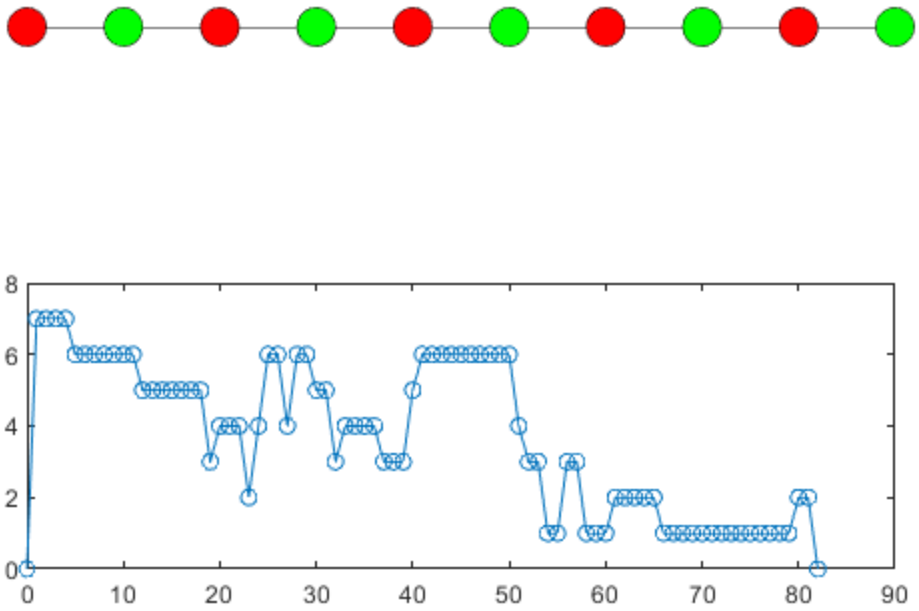

Figure 1. Upper: The graph and color of the nodes at the stationary point. Lower: The utility function at different iterations.

B.

In this exercise the same type of coloring was used as in exercise a, but now the graph is much bigger with 100 nodes with different links. Each graph now also has the option of choosing 8 different colors. The probability of updating to a certain color is calculated in the same way as i b, except that the cost function now is,

$$c(s, X_j(t)) = \begin{cases} 2 & \text{if } X_j(t) = s, \\ 1 & \text{if } |X_j(t) - s| = 1, \\ 0 & \text{otherwise.} \end{cases}$$

This means that the cost function punishes nodes that have the same color or similar color as its neighbours. The result from 1000 iterations can be seen in Figure 2. It can be seen that the utility function does not reach zero but after 1000 iterations it has the value 4. This means that the solution is not perfect but since it is a much bigger network than i a, it might be reasonable to be satisfied with this solution. To get a better result perhaps a more sophisticated method needs to be used, where the colors and nodes are not chosen at random the same way. There is also the possibility that there is not any better solution. It should also be noted that even after 1000 iterations the graph still regularly changes color. This is because different colors can still give the same utility as long as they are at least two steps away from the color of their neighbours.

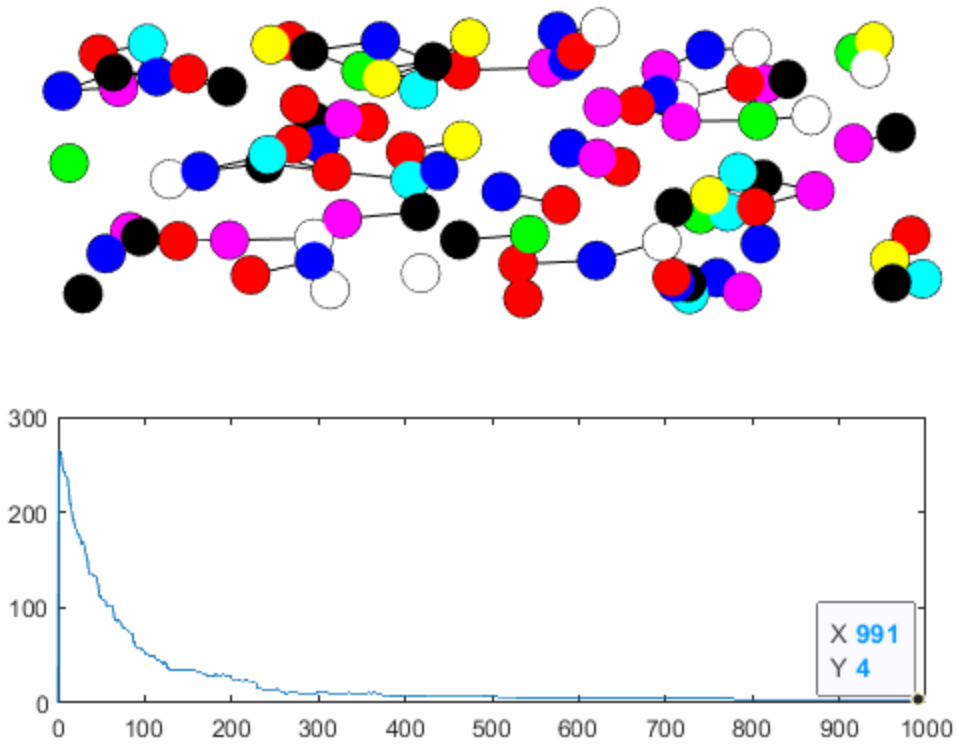

Figure 1. Upper: The graph and color of the nodes after 1000 iterations. Lower: The utility function at different iterations.