A Q-learner is a basic concept in machine learning, introduced by Chris Watkins in his 1989 PhD thesis (http://www.cs.rhul.ac.uk/home/chrisw/). It is defined as follows: a Q-learner has a finite set of *states*, and for each state it has (1) a finite set of *actions* (which are transitions to other states), and (2) a time-varying discrete probability distribution on the set of actions (initialized to uniform). Thus, the full set of distributions can be thought of as a probability transition matrix. The Q-learner starts in an initial state. After making a transition (selected according to the appropriate discrete distribution), it gets a reward, and this reward is used to update the distribution associated with the starting state. This is done repeatedly. The update is by a specific rule (with two free parameters) defined by Watkins, and typically results in the actions tending to cause transitions to "good" states. For example, a Q-learner performing a random walk on a graph can be "taught" to prefer a specific route by being given larger rewards when it encounters the preferred route. Watkins proved a convergence theorem for this process.

The previous ideas are now classical. For a novelty, I wish to start with a (small, connected) graph and place an independent Q-learner on each node. The Q-learners only have knowledge of their neighbours (and, where necessary, their neighbours' states). We now wish to study the global behaviour of this network. Initial study would be by simulation (this is quite easy with modern software with multi-threading ability); it is not clear whether any theory is possible. Two specific problems are:

1. Graph colouring. The state is a positive integer (thought of as a colour) with a specified upper bound. Two neighbours are said to *conflict* if they have the same colour. We want to eliminate conflicts. Actions are transitions to other colours. Rewards could be defined as (1) if all conflicts with neighbours are eliminated by the new colour, a large positive reward is given; (2) if it is impossible to eliminate all conflicts (because not enough colours are available), then the reward is the number of conflict eliminated. Question: Does this process have a global equilibrium state? If so, how fast is it reached, and how are the fluctuations around the equilibrium distributed?

2. Shortest-path routing. A source and destination node are specified. Packets are forwarded by nodes. There are two types of packets: data and acknowledgement. Both keep an internal count of how many hops they have made (edges traversed). The input state is the neighbour from which a packet is received; the output state is the neighbour to which it is forwarded. Upon receiving a packet, a node selects a neighbour to forward it to (from its own distribution), but the reward is delayed – when the packet reached the destination node, and acknowledgement packet is generated and directed back to the source. When (and if) this acknowledgement packet passes a node which previously routed the data packet, the reward is dependent on the total number of hops (higher reward for smaller number). Does this process converge to the globally optimum shortest path?