Cable Cost and Modularity

Thomas McColgan

June 30, 2008

1 Methods

1.1 Modularity

The Modularity Q of a given net is calculated using the spectral algorithm described in [**?**]. This uses the same definition of modularity as Alon in [**?**], though he normalizes it for net size. This does not have any effect on the results described here, because the normalization is done by a linear function.

The calculation of eigenvectors and -values is done by a Jacobi algorithm taken from Numerical Recipies. This is far from optimal because it seems to be unstable at times and calculates all eigenvectors, but only one is needed.

1.2 Genetic Algorithm

The evolution of network populations is achieved by the algorithm described in [**?**], with a few minor modifications. An integer representation is used instead of the binary one. Numbers are replaced by new random numbers instead of flipping random bits.

Additional fitness penalties were introduced for networks that did not use all inputs, accelerating evolution.(a complete solution to the task must always use all inputs, unless it contains a tautology)

Optionally networks were penalized for having a number of gates different from a nominal amount, thus effectively fixing the number of gates across the population.

In order to measure the effects of wiring cost optimization on modularity, wiring penalization can also be activated when desired.

1.3 Wire Cost

The Wiring Cost for nets was calculated by fixing the input and output gates in certain positions and then optimizing the locations of the remaining gates using the algorithm described in [**?**]. All connections were given equal weight, $\alpha = 1$. The algorithm was applied twice in order to calculate optimal x and y coordinates of the gates, and the wiring cost was calculated using the same metric that was being optimized for.

1.4 Logical Circuits

The nets are structured in the same way as the NAND gates in [**?**]. Since feedback loops were allowed in the networks, instability may occur and evaluation is not trivial. [**?**] does not specify how he handled this. All gates were initially set to a false, except for the input gates, which contained the values corresponding to the goal that was to be evaluated. the gates were then adjusted to take on the NAND value of their inputs. This process was repeated until either the network reached a stable state, or a given number of iterations was exceeded. If the network does not stabilize after this transient period, the result is assumed to be wrong. It is necessary that the entire net be stable, not just the output gate.

2 Results

2.1 Delayed Wire Cost

Turning on wiring optimization from the start lets the genetic algorithm get stuck on non-optimal nets with low wiring cost, preventing solutions that completely satisfy the logic goal from being found.

Therefor we use the following strategy: First a standard evolution is allowed to take place, without optimizing for wire. This creates a diverse population of nets that all solve the given task perfectly. Once this state is reached an additional selection pressure for wiring cost is introduced, with a b_{wire} low enough to prevent the networks from loosing their functionality.

This method produced a sharp increase in the average modularity of the population when the wiring optimization is introduced. Further jumps may occur if the optimization is left running. If the wiring cost was deactivated again, the modularity mostly reverts to a state similar to that beforehand.

This was tested extensively for small networks of 4 inputs, with several goasl, and in 2 runs for larger networks with 8 inputs.

The size of the jumps varies from run to run, and in very rare cases it does not seem to take place at all. Why is this?

When we initially observed this we suspected that it might be due to a decrease in gates, which would mean that the normalization would have to be changed, possibly neutralizing the effect. However it turned out that it works for fixed numbers of gates as well.

This seems to indicate that there is a correlation between modularity and wiring optimality.

PLOTS

2.2 Scatterplots

Another way we intended to show the correlation between wiring and modularity was by randomly generating nets, measuring their modularity and wire length, and calculating the covariance of these two variables.

This method also shows a significant correlation, however the correlation is opposite to that of the one expected, more wire corresponding to higher modularity.

This might be due to the fact that this method ignores the functions of the networks. Nets sharing a common logic function might constitute subgroups in which the correlation is as expected. It is hard to show this, because there is a very large number of possible goals, and it would take a long time to generate a sufficiently large random population for any non-trivial goal.

SCATTERPLOT

2.3 Stochastic Wire Cost

The effect of wiring optimization on modularity was also shown in a third manner. The wiring cost was introduced as a selection pressure from the start, but individuals were only penalized for wiring cost with a certain probability. This was tested on the small networks for a few runs, and seemed to yield nets with higher modularity, coherent with the result from the delayed wire cost. It increased the number of generations needed to find a perfect logical solution though.

3 Todo

- Implement better numerical algorithms for the Eigensystems and Matrix inversion. LAPACK might be more suitable than the generic NR algorithms.

- Make process fully parallel, probably with MPI.
- Run larger number of trials to determine the exact characteristics of the process.
- Figure out what really causes the different signs in the correlations with and without fixed goal.
- ...