Useful Network Theory??

Brent Follin

July 24, 2015

1 Problem Summary

The general scope of the problem is network complexity in the internet of things (IoT). In particular, I'm looking to investigave how complex a system one can develop before the IoT starts to 'fail', a concept that will take some definition later.

1.1 Definitions

The IoT consists of a set of connected devices $d_i \in \{D\}$ that each can be in one of a multiplicity of states $s_i^\alpha \in \{1, ..., m_i\}$, such that the total state space of the IoT is

$$M \equiv \prod_i m_i,$$

e.g. the state S^α of the IoT ranges in labels from $\{1, ..., M\}$.

The interesting thing is interactions between devices. A particular, important, example is *if-then interactions*, where if some element d_i is in state s_i^α then some other element d_j changes from inital state s_j^α to s_j^β. For example, if a motion sensor detects movement, it might tell a smart lamp to turn on. There are other interactions possible: global interactions, where the system as a whole reacts to the state of some component (say, a shutdown signal from the router); for simplicity I'll stick to *if-then*.

2 Hamiltonian Formalism

One approach to characterizing the IoT system comes from lattice quantum theory. We consider the tensor space M with elements

$$|S\rangle = s_i^\alpha \otimes s_j^\beta \otimes ... \otimes s_k^\gamma,$$

which represents the product state space with size $m_i \times m_j \times ... \times m_k$ and dimension equal to the number of objects in the IoT system. Interactions are then modeled by some discrete time evolution tensor $U(t)$, that is

$$|S(t_n)\rangle = U(t_{n-1})|S(t_{n-1})\rangle.$$

For *if-then interactions*, the form of $U(t_{n-1})$ is that of a symmetric matrix U, with the effect of the i^{th} device on the j^{th} device encoded in the element U_{ij}. For instance, in a 2-device system with devices A and B with binary states ($0 =$ off and $1 =$ on), we can represent the interaction *if A is on, then turn B off* with

$$U = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix},$$

since

$$\begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

and

$$\begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

and so on.

The benefits of this formalism is that it makes the devices in the system transparent, since each dimension of the state $|S\rangle$ corresponds to a device in the IoT. However, it's unclear if the formalism can be generally extended to devices with more than two states; in particular, it's not immediately clear if the formalism can support more than one *if-then statement* between the same two devices. Of course, one solution is to treat each device as its bitwise representation (a device with $2^{m-1} < n < 2^m$ potential states is represented by m light switches), but that leads to complications in interpretation. The bitwise representation is nice, however, because then the formalism maps pretty clearly onto Ising spin models, of which there's a mountain of literature in condensed matter physics. In summary, two bits connected with a *if-then interaction*[1] would be equivalent to 'neighboring' particles, whose spin orientations influence their neighbors, with probability one.

2.1 Complexity measures in the Hamiltonian Formalism

The interactions are encoded in the transition matrix, so a clear measure of the system no longer allowing for the actions of some 'actors' would be if

$$|S(t_{n+1}\rangle = |S(t_n)\rangle$$

for some time step t_n. That is, if the system gets stuck in some state, then clearly the system is no longer dynamic. This could be extended to circumstances where the states are cyclic; that is we reach a situation where

$$|S(t_{n+m})\rangle = |S(t_n)\rangle$$

for some sufficiently small m.

[1] A symptom of the interpretational complexity is how to model *if-then interactions* between devices to *if-then interactions* between bits. Not impossible, I think, but not obvious, and not something I've found previous work on.

3 Graph Theory Formalism

A completely different picture can come from moving from the space with dimension given by the number of devices m (or, perhaps more realistically, the number of bits required to represent those devices, n) to the space whose dimension is given by the total number of states the IoT system, of dimension 2^n. In this space, a vector represents a (mixed) state of the IoT system as a whole; to enforce a determined state on the system we'd require realized states to belong to the unit basis, with a 1 in some row and 0's in all others. Any interaction is just a transition between states, which we might also represent in a (now much bigger) transition matrix. That's probably not the most useful way to think about it, though.

Since we're in the case where we only care about definitive states, and the rules that tell us which state(s) we can evolve to from the state we're currently in (rules that are set by interactions between devices), it makes sense to think of this space as a graph. Here, nodes represent states of the IoT, and edges represent allowed transitions. It's a very clear visual, the technical issue is generating the edges that appropriately represent an *if-then interaction*. It's more tedious than hard: an algorithm isn't too hard to come up with, but because of the potentially large dimension of the graph, I probably need to be smart in implementing such an algorithm if I want it to actually finish.

3.1 Complexity measures in the Graph Theory formalism

Here, the equivalent to being stuck in the same state is entering a node in which there is no way out, and the cycles would just be a loop in the graph without a way out. This is well-studied, and is known as graph complexity: there are even Python packages to calculate this node-by-node, including what happens to the complexity when you start removing nodes (an interesting and well-studied question in graph theory, which would be related to robustness of the IoT, is how many objects you can remove from a graph before nodes (states) become inaccessible).

4 Dead end investigations

I looked into cellular automata; there wasn't an apparent way to implement local (state dependent) rules in such a formalism. I also checked out work on so-called epsilon machines, which are a way of expressing the complexity of a chain of events by constructing the simplest possible system that can lead to that output. It's interesting, and I can construct epsilon machines for a given IoT system, but I don't see how to use that machine to answer questions about the ability of devices in the system to act.