What is the correct semantic basis for adaptive systems?

Simon Dobson
Lero@UCD
simon.dobson@ucd.ie

Lero foundations series, Trinity College Dublin, 23Jan2009

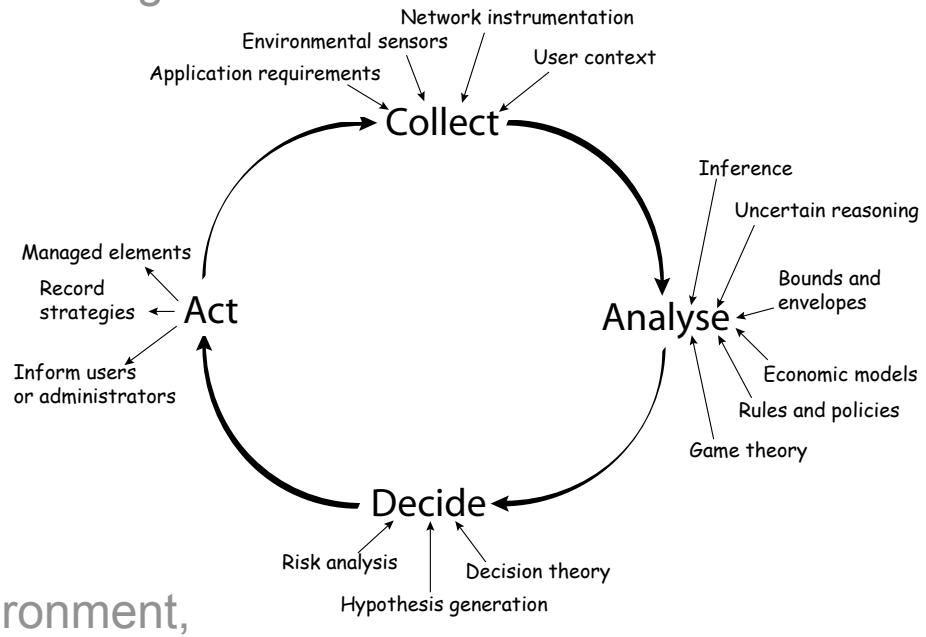
Overview

- What does it mean to be an *adaptive* system?
  - What does it mean to be “correct”, when correctness changes?
  - Match changes against their expected causes
  - Identify a behavioural “envelope” and stay within it
- Without these sorts of notions it’s hard to see how we can effectively develop complex adaptive systems

- My goal
  - Identify the challenges for semantics and analysis in this area
  - Review some of the approaches we’ve thought about
  - Solicit comments, suggestions and collaborations

Driving forces

- Complexity kills innovation and maintenance
  - Systems become too hard to change
- Autonomic systems
  - Reduce TCO by increasing management automation
  - "Self-*" properties
- Adaptive systems
  - Respond to changes in environment, requirements, constraints, goals, …



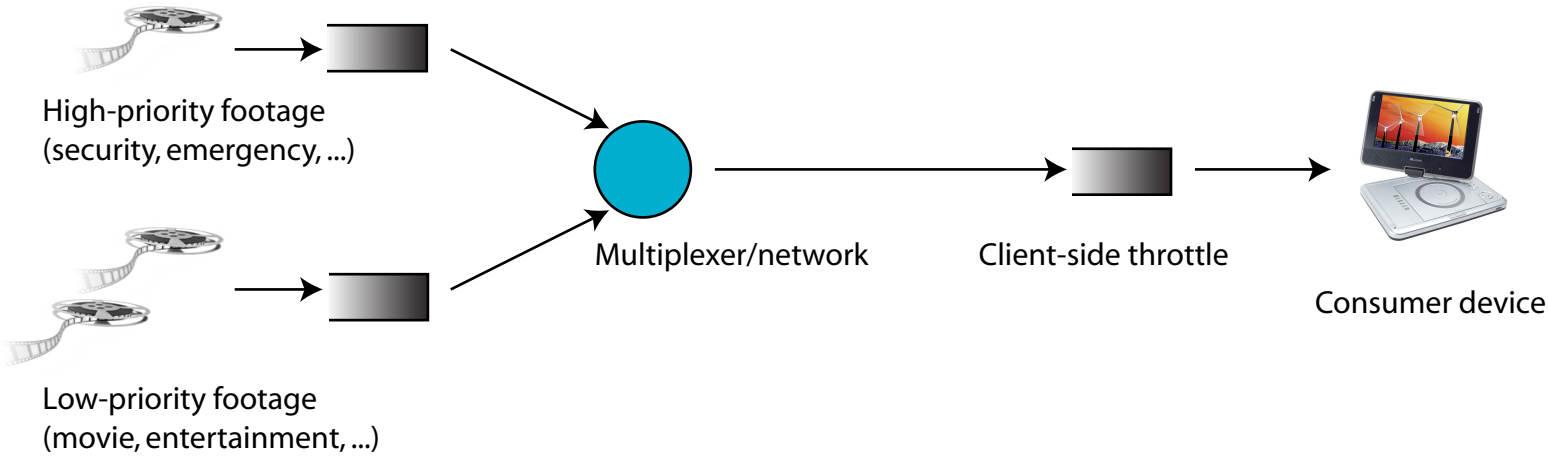
From Dobson **et alia**. A survey of autonomic communications. ACM Trans. Autonomous and Adaptive Systems **1**(2). 2006.

Traditional approaches

- We have a good ideas about how we describe the "correct" working of classical systems
  - …even if we're less good at achieving it…
  - Weakest pre-conditions, Hoare triples, process algebras, …
- The common theme is that there is a (single) "correct" behaviour – and our goal is to find and implement it
  - Requirements gathering, specification, verification, validation, …
- Quite a "closed universe", in that the system, once defined, isn't supposed to change
  - Or, at least, handling such changes is an off-line process

What changes for *adaptive* systems?

- Consider a network with multimedia data



- What is the "meaning" of this system?
  - Deliver the different streams "well"
  - …prioritising some over others
  - …in the presence of noise, congestion, failiure, …

An informal view

- "Correctness" here is rather more tenuous than we might normally expect
  - High-level goals: "show me my movie unless something important happens, in which case show me that – quickly!"
  - Low-level goals: control windows, frame rates, resolution ad continuity of streams
  - Low-level constraints: constrained (possibly variable) bandwidth
  - Environmental factors: fire, flood, pestilence, and other things that can be sensed
- We *expect* different behaviours under different circumstances

Central problem

The reason we have the system in the first place, defined externally to it

The way in which we pull the knobs and levers, and the effect on the dials

The dials, knobs and levers we can use to affect its operation

Given a system with a set of **high-level goals**, a particular set of **monitoring and management facilities**, and a changing environment, what is the **appropriate control strategy** that the system should take in **response to changes** in its environment in order to maintain, **as far as possible**, its goals?

This whole process is a dynamic, during the system's "live" operation

It may not **be** possible, and so we need a fallback strategy in case we fail

Pause for thought

- In some sense, nothing's changed: we define the outputs we see in response to *all* inputs, explicit, and implicit

- But that neglects the needs of engineering
  - Functional design: the algorithms etc we want to use
  - Non-functional or adaptive design: how the algorithms are changed in response to changing circumstances
  - We really need to separate these concerns
- Two notions of correctness
  - Point: do the right thing at any given time
  - Process: do the right things over time

Coutax **et alia**. Context is key. Comm. ACM **48**(3). 2005.

Abstractly – 1

- We can view almost any system S as a mapping from inputs to outputs whilst maintaining an internal state

$$N_S : State_S \times Input_S \rightarrow State_S \times Output_S$$

- What adaptivity does is to “lift” this “next state” function to include the external context

$$N_S : Context_S \times State_S \times Input_S \rightarrow State_S \times Output_S$$

- However, it doesn’t do this arbitrarily

Abstractly – 2

- We don't end up with just *any* function of three variables
- We want there to be some relationship between the original behaviour(s) and the contextualised ones
  - There's a "shape" to the context that's "reflected" in the lifted, contextualised function
- So we need some way of capturing this "shape", and of *generating* the lifted function
  - Make sure the repeated use of N_S leads to a system that preserves the properties we want

Dobson and Nixon. More principled design of pervasive computing systems, LNCS 3425. 2004.

Which brings us to…

- What is the correct semantic basis for *engineering* adaptive systems?

- Desiderata
  - Focus on adaptation, perhaps at the expense of the details of exactly what behaviour will be exhibited
  - Provide a clear mapping between causes (in the environment) and effects (in the system)
  - Can handle the uncertainty of sensor data and other information sources cleanly
  - Support design tasks – how do I get what I want?
  - Support analysis tasks – does what I've built do (only) what I want?

Less interested in **what happens** than in how what happens **changes over time**

Don't over-react to changes that may simply happen because of low-resolution sensing or noise

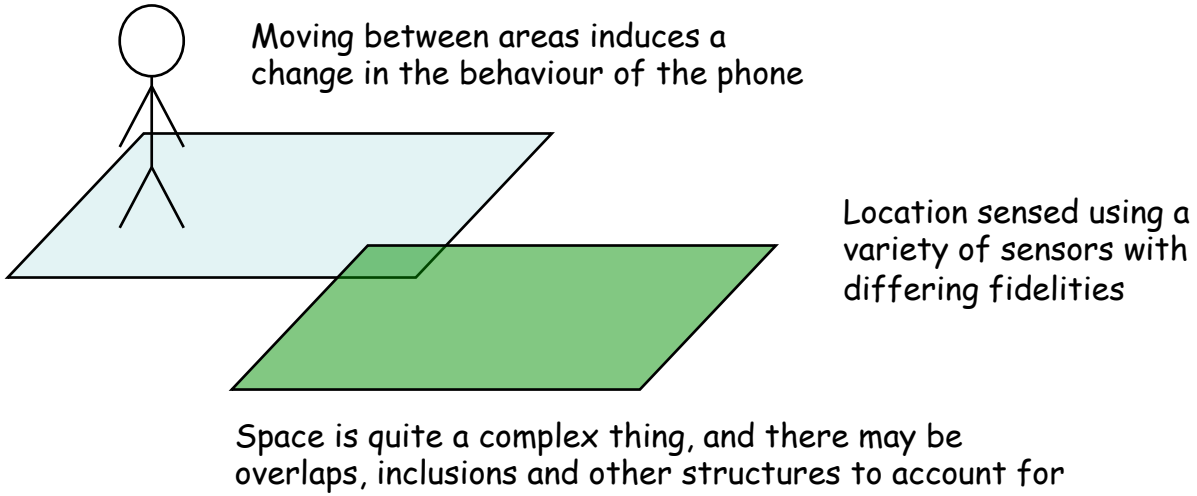
…and I wish I had the answer, but…

- We haven't yet found a good model that answers these challenges

- However, we *have* explored a number of possibilities, and it's the two most promising of these that I'd like to share with you for the remainder of this talk
  - Fibre structures and categorical models
  - Dynamical systems and topology

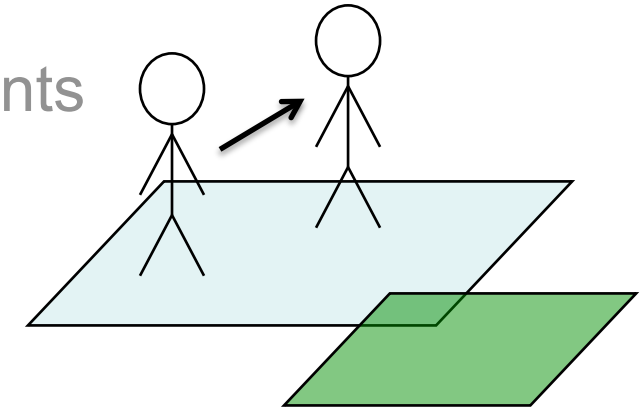
Not all changes matter – 1

- A classical example of an adaptive system is a location-based application, for example changing the behaviour of your cellphone depending on where you are
  - Behaviours: different ringtones, forwarding, …
  - Context: location from GPS, Ubisense, Bluetooth proximity, …



Not all changes matter – 2

- Only some of the user's movements will change the system: the rest are "noise", in a sense
  - Didn't cross a "significant" threshold
  - May often be the result of sensor noise

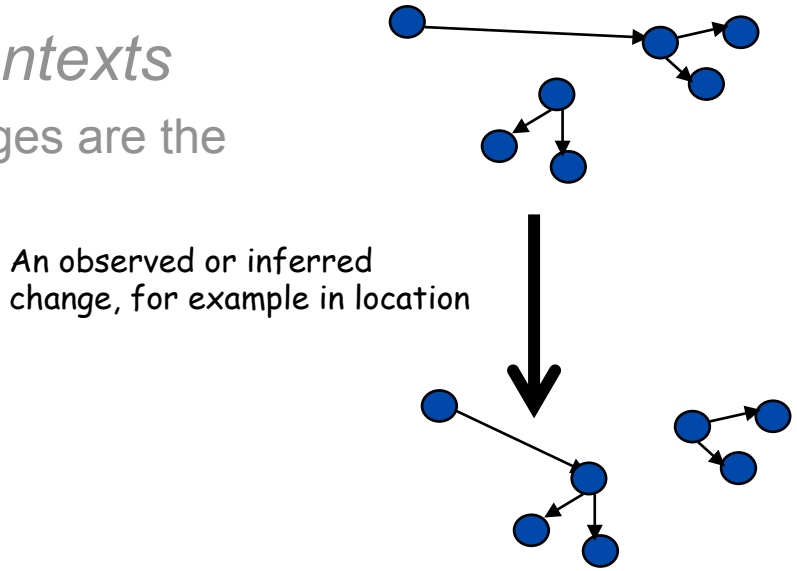


- How can we distinguish between the "real" and "imaginary" movements?

- Observe that there are *many* motions that result in the *same* behavioural change, and so are in some senses equivalent

Graphs as models

- We can define a graph of *contexts*
  - Nodes are complete contexts, edges are the single-act changes between them

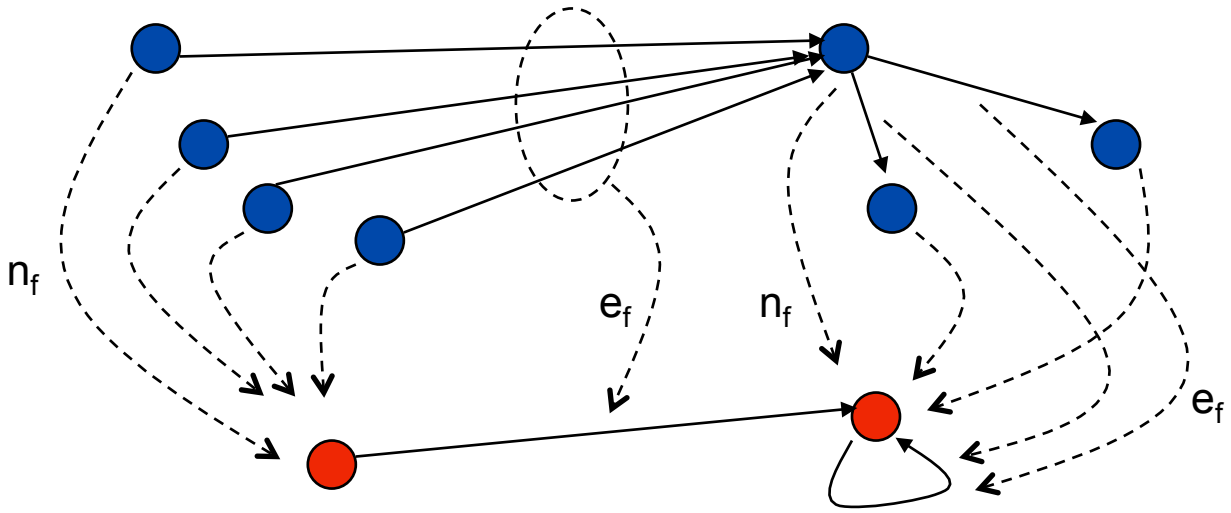


- Similarly there is a graph of *behaviours*
  - Nodes are behaviours or scripts, edges the ability to stop one and start another directly

Essentially the **workflow** of the application being provided

Graph homomorphisms

- A graph homomorphism is a mapping between graphs that preserves the adjacency structure
  - $f = (n_f, e_f) : A \rightarrow B$
  - $source_B(e_f(e)) = n_f(source_A(e))$ ...and similarly for edge targets

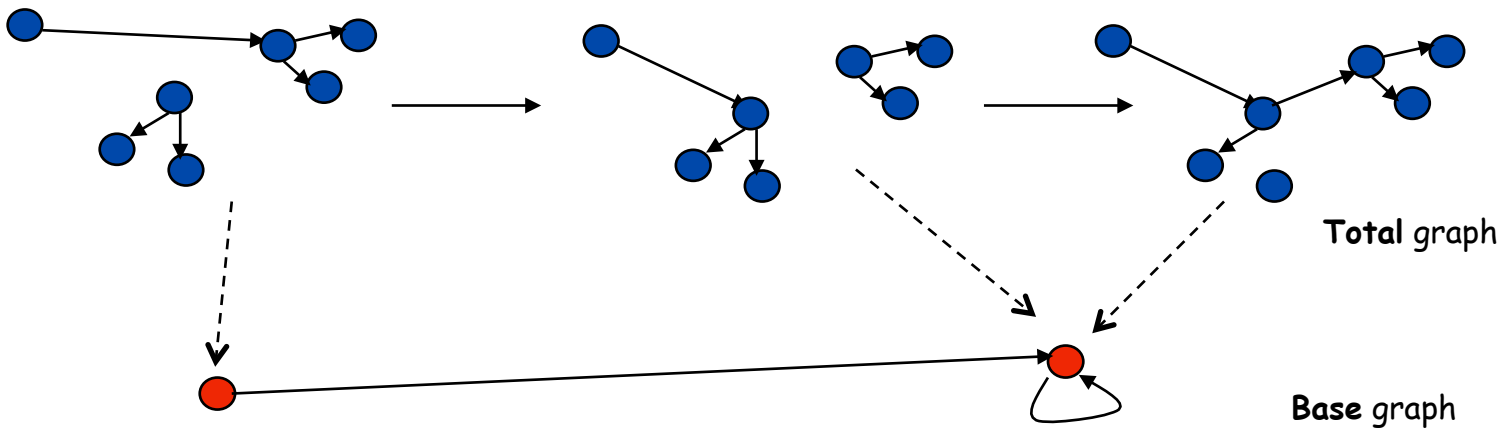


Fibre structures

- Given two graphs A and B and a homomorphism f between them we can define the *fibre above a node* b
  - The nodes of A that map to b under f
  - The edges between those nodes, which must map to a reflexive edge from b to b

$$n_f(a) = b$$

$$e_f(e) = id_b$$



Fibrations

- A homomorphism $f : A \rightarrow B$ is a *fibration* if, for each edge e in B such that $n_f(a) = target_B(e)$, there is a unique edge e^a (the *lifting of* e *at* a) such that $e_f(e^a) = a$ and $target_A(e^a) = a$

Boldi and Vigna. Fibrations of graphs. Discrete mathematics **243**. 2002

- This is a powerful connection between two graphs
  - Put another way, there is at most one edge from one fibre to a *particular* node in another (although there may be many such nodes)
  - In the case of a context evolution graph, the mapping reduces the transitions that are observable in behaviour
  - Preserves some aspects of the path through the context in the changes of behaviour

Isolating “real” changes

- If we consider a context evolution graph fibred over a behaviour graph
  - Nodes (context graphs) = those contexts select a behaviour
  - Edges within fibres = those transitions (observations, inference) that *do not change* the behaviour
  - So the contexts within a fibre are *equivalent* but not *equal*: it is still worth observing the differences
  - …but only a transition between fibres will lead to a change in behaviour

Dobson and Ye. Using fibrations for situation identification. Proc. Workshop on Combining theory and system-building at PERVASIVE 2006.

Discussion

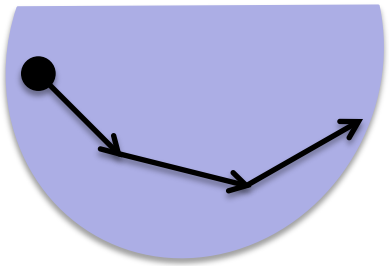
- This work has led to a better understanding of adaptation, especially in pervasive systems
  - Situation identification is a major area of research for us
  - Fibre structures like this help to add rigour to the way in which we map context to situation
  - Especially well-developed for managing location

Ye et alia. Using situation lattices in sensor analysis. Proc. PERCOM 2009. (To appear.)

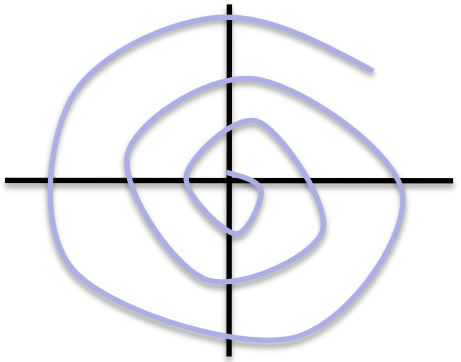
- However, it doesn't capture more continuous aspects, and doesn't emphasise the dynamics of the problem

There's science…

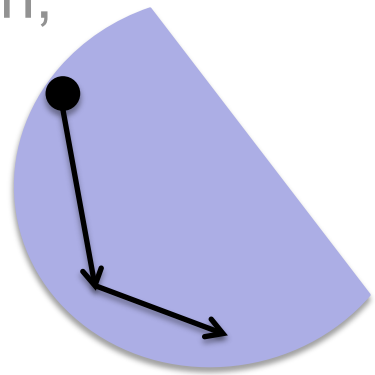
- Suppose we take a bowl and roll a ball around it

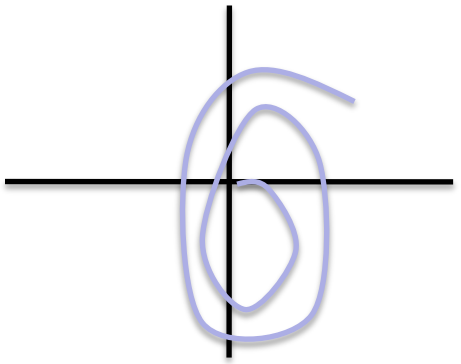


We can plot the ball's x-y position in the bowl and describe how it'll move, eventually coming to rest at the origin



- If we change the orientation of the bowl and do the *same* action, we'll see a *different* dynamics
  - But still determined by the initial conditions and the action





…and then there's engineering…

- This situation is well-understood as a *dynamical system*, and can be applied to a wide range of situations
  - Describe the *phase space* of the system – the valid tuples of all the variables – and the way one configuration changes into another
  - From a given environment we can predict what a behaviour will result from a given action
- However, what we want is pretty much the opposite
  - A way to *engineer* the selection of a particular behaviour given the constraints of an environment
  - Also want to know how the action selected changes as the environment changes

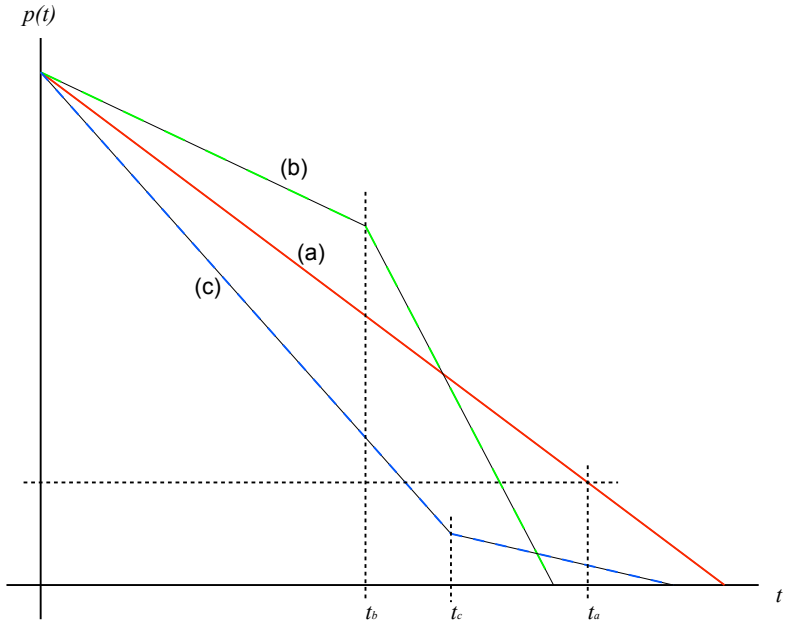
Neither of these problems is discussed, as far as we know, in the dynamical systems literature

Behavioural envelopes

- What does it mean to be the phase space of an adaptive system?
  - Select all the variables in the system
  - Identify all the legal tuples, a sub-space of the space of all values
  - Define a dynamics within this sub-space: how the system will evolve from point to point
  - Each control action changes the sub-space, or the dynamics, or both
- In a physical system we *observe* the valid sub-space
- In an engineered system we may want to *construct* it to meet the high-level goals of the system

A simple example

- In a wireless sensor network we might see graphs like this relating power to time
  a) Linear use over time
  b) Lower use, and then a more precipitate drop
  c) A steep drop where we intervene as time t_c to reduce load



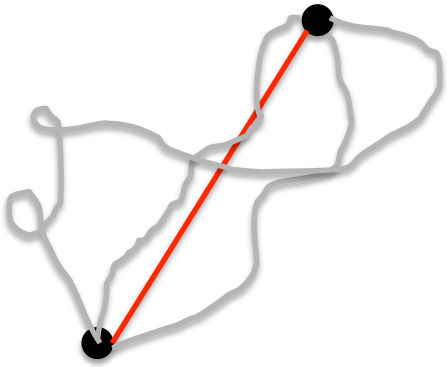
From Dobson. An adaptive systems perspective on network calculus, with applications in autonomic control. Int. J. Autonomous and Adaptive Communications Systems **1**(3). 2008.

Changing the load changes the dynamics of the power behaviour

Similar approaches work for data centre power consumption etc

Trading-off alternatives

- In many systems there may be several different *acceptable* dynamics we could pick
  - In this previous example we might reduce communications *or* reduce sensing frequency *or* reduce accuracy – each of which saves power
- The autonomic problem is to decide *which* dynamics we want from the set of possibilities
- Techniques such as calculus of variations may help



Each path f_i between the two points is a function on $[0, 1]$, to which we can associate a path length $F(f_i)$

We then need to minimise $F(f_i)$ to find the shortest path

A larger example-in-progress

- Consider an area being observed by a collection of mobile environmental sensor nodes
  - The state space is the position of the nodes, their observational and communications radii etc
  - The dynamics is the way in which they move
  - The goal is to keep the area “as observed as possible”
- This is a tractable problem *if* there are enough sensors to find a static coverage
- More difficult if the sensors must move continuously
  - Test the different strategies for moving nodes
  - Evaluate against changing environments, i.e. if we introduce a pollutant and want to balance detailed sensing against overall surveillance

Bartolini et alia. Autonomous deployment of self-organizing mobile sensors for a complete coverage. Proc. IWSOS. 2008.

Cellai and Dobson. Generating a dynamic sensor coverage of an area. Work-in-progress for submission to ACM Trans. Sensor Networks.

Discussion

- A more complex mathematics, but perhaps richer
- Focus on the way in which changing the environment changes the way a system will behave
  - Phase space “morphs” over time, changing the results of actions
- Identify actions that can “right” the system
- Well-developed tools with lots of interesting concepts
  - Stability, periodicity, chaos, …
  - Intended for *science* rather than *engineering*
  - …but at least we know it works at *some* level

Where does this leave us?

- We believe that the dynamical systems approach works well as an *analogy* to gaining insight, and can be made to work as a *semantics* for describing adaptive systems

- However, there's a lot of ground to cover
  - How to we describe the individual behaviours?
  - What's the best way to program dynamics?
  - What can we prove about such systems

- There's an entire programme of research here that'll take a lot of effort to explore effectively