Automating a Solution for Optimum PTP Deployment

ITSF 2015

David O'Connor

Bridge Worx in Sync

Sync Architect V4: Sync planning & diagnostic tool. Evaluates physical layer synchronisation distribution by behaviour simulation. (2000).

Sync Architect V5: Sync planning & diagnostic tool. Evaluates physical layer and PTP synchronisation distribution by behaviour simulation. (2016).

Sync Architect V6: Sync planning, diagnostic and management platform. Predicts optimum resource deployment. Monitors actual synchronisation behaviour against ideal simulated behaviour. (2017).

The Optimisation Elephant

It’s a question we constantly get asked by *Architect* users, “Can you provide a button that just plans it for you?” It’s the *elephant* in our room.

We would like to be able to say, *yes*.

Network optimisation (because that’s really what planning is) is extremely complex. Is it based on, “if…then…else” algorithms? We don’t think so.

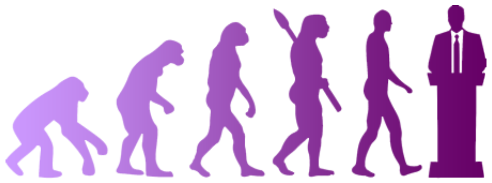
I would like to share with you some research and development we are currently working on…

PTP Resource Optimisation

We are developing a generic network optimisation engine, its first target application is the optimisation of Precision Time Protocol resource deployment:

- What's the minimum resources I need to hit my quality target?
- Where should I deploy these resources?
- How should I route PTP streams?
- How can I efficiently implement protection and load sharing?
- Oh – and don't forget I've got some existing constraints so how can I minimise change?

An Evolutionary Approach



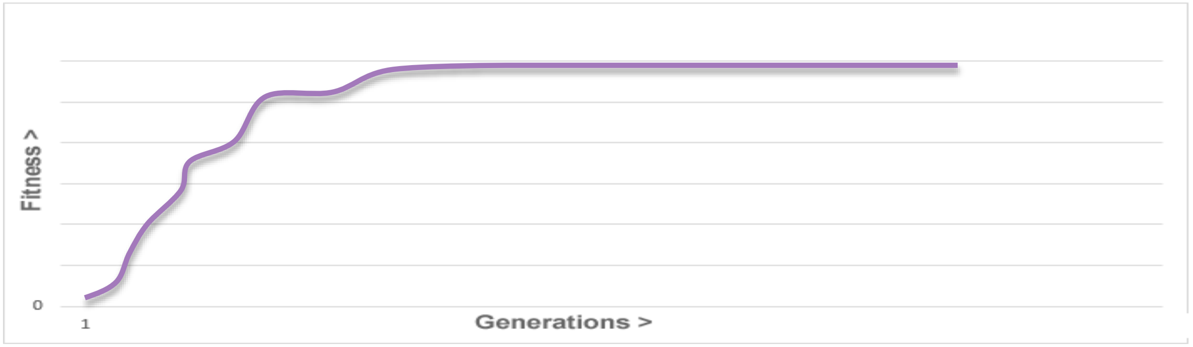
- Our research is currently based on the concept of Evolutionary Algorithms (EAs).
- EAs are a subset of evolutionary computation within the wider field of artificial intelligence.
- They are roughly based upon the basic mechanisms of biological evolution: reproduction, mutation and selection.
- Candidate solutions are *individuals* within a *population*. Each are evaluated for their suitability (*fitness*) and are selected in pairs, based on their *fitness*, to play the role of parents to two new offspring solutions based upon their parent's *genes*. A touch of random *mutation* is thrown in to spice things up. The process is repeated through many *generations* of the *population* until an ideal solution emerges.

How do they Work?

A simplistic example: suggest the most attractive face?..

When do we Stop Hunting?

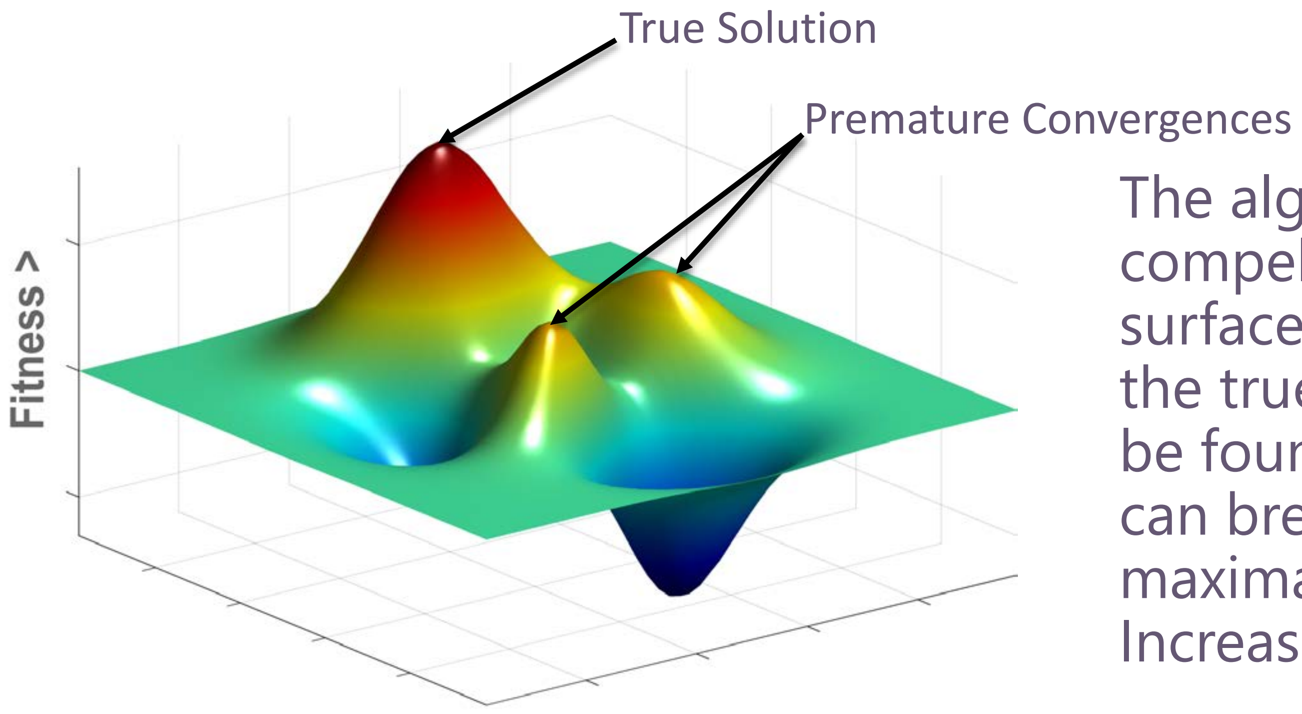
If we observe the fitness of the best solution per generation (population)…



The fitness plateaus, suggesting the algorithm can not find a better solution.

Premature Convergence

Imagine we have a two dimensional problem and we map the entire solution space…



The algorithm is compelled to follow the surface to a maxima but the true solution can only be found if the algorithm can break out of a false maxima…
Increase mutation…

The true solution…

Real World Applications

- Evolutionary Algorithms perform well maximising solutions to all types of problems because they are not prescriptive they meekly follow fitness for purpose.
- They are widely used in; engineering design, art, biology, economics, marketing, genetics, operations research, robotics, social sciences, physics, politics and chemistry.
- They tend to be applied to well constrained problems because they don't scale well to large issues, fitness can be difficult to quantify, and as we have seen, stop criteria is a problem.

...and now PTP deployment...

Challenge 1

How to build a solution *genome* based on network resources and topology?

- Most EA solutions are based upon simple string or tree representations of solution operators. Ours must be far more complex – we thought, but it didn’t turn out that bad...
- Central single network object model as context reference. Our simulation models are ideally suited (given a little pruning).
- Solutions are just differential models based on the central object model.

Challenge 2

How can we determine degrees of fitness for a solution with enough resolution to drive the reproductive process?

- Fitness can not yield a simple, yes/no result if it is to efficiently drive the reproductive process.
- Our network behaviour simulations are ideal measures of fitness providing both blatant and subtle indicators of solution suitability.
- They allow us to provide a feature almost unique in the field of EAs: the user can simply modify the fitness evaluation by electing network behavioural properties without having to modify the simulation itself.

Challenge 3

How do we optimise the processing power required?

- Our experience of network simulation has taught us that simulation as an evaluation technique is not cheap in terms of processing power. Even an evaluation time of just 10 seconds could represent a convergence time of 18 days!
- There are many techniques we use to minimise convergence time:
  - Parallel processing of solution fitness – ideally every individual simultaneously.
  - Incest Prevention: Not allowing identical individuals to be selected for reproduction.
  - Elitism: Automatically promoting the best individuals to the next generation without reproduction or mutation.
  - Steered Mutation: Not allowing invalid mutations.
  - Exaggerated Mutation: Increasing the extent of mutation as individuals converge on a fitness maxima.

Challenge 4

How do we identify the stop criteria?

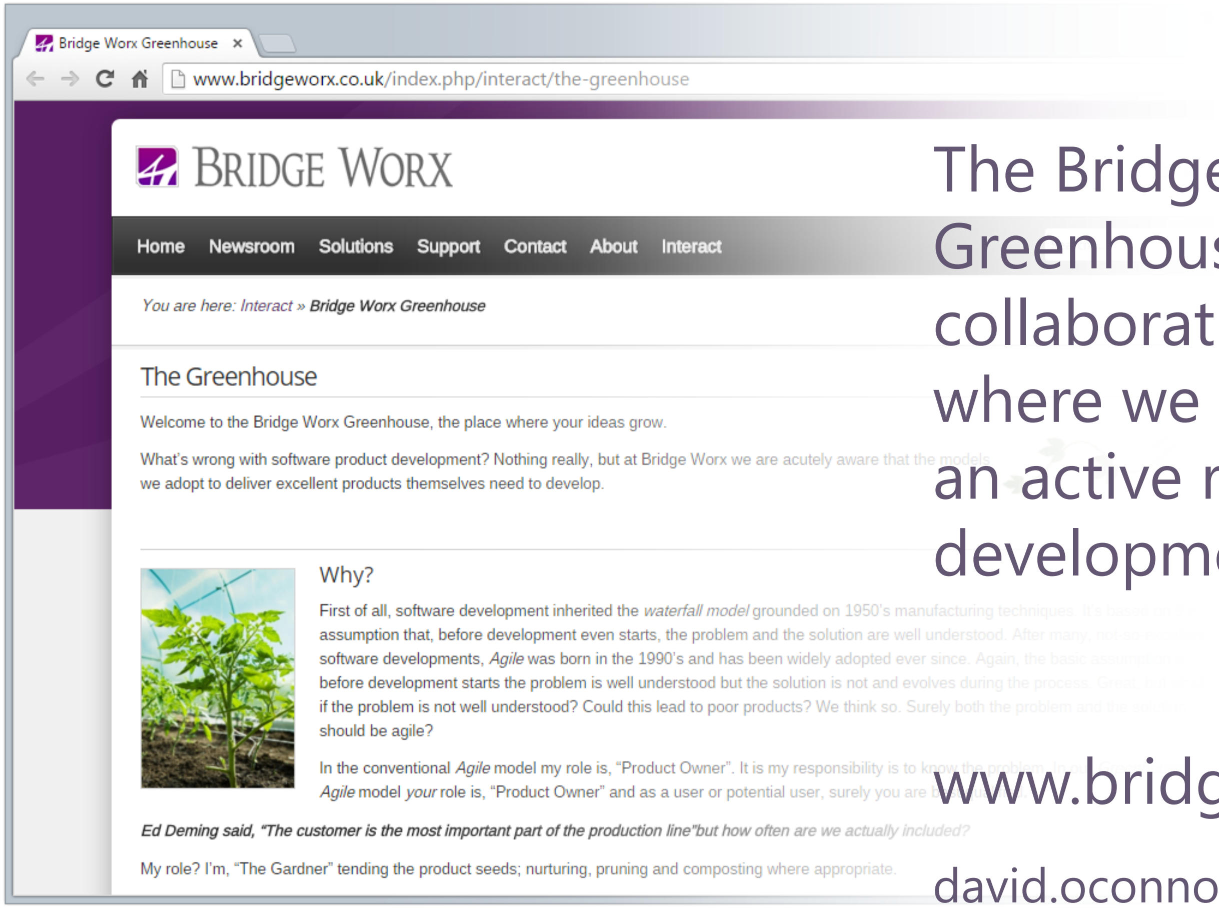
- The bugbear of all EAs. With an increase in solution complexity comes another possibility, *just good enough* solutions.
- We are using the simulation based evaluation to drive to solutions where some subtleties become irrelevant and once a solution achieves a just good enough criteria the search can stop.

How is it Going?

- Our primary application of this engine, Sync Architect V6 is currently in development and is scheduled for release in late 2017. Our network models and behaviour simulations have lent themselves well to the EA Engine. We are currently working distributing solution evaluation in the Cloud.
- Our ultimate goal is to build a generic network optimisation engine designed to allow all manner of optimisations to be performed by deploying behavioural simulations…

…But we need your help…

Our Greenhouse



Bridge Worx Greenhouse

www.bridgeworx.co.uk/index.php/interact/the-greenhouse

BRIDGE WORX

Home Newsroom Solutions Support Contact About Interact

*You are here: Interact » **Bridge Worx Greenhouse***

The Greenhouse

Welcome to the Bridge Worx Greenhouse, the place where your ideas grow.

What's wrong with software product development? Nothing really, but at Bridge Worx we are acutely aware that the models we adopt to deliver excellent products themselves need to develop.

Why?

First of all, software development inherited the *waterfall model* grounded on 1950's manufacturing techniques. It's based on the assumption that, before development even starts, the problem and the solution are well understood. After many, not-so-excellent software developments, *Agile* was born in the 1990's and has been widely adopted ever since. Again, the basic assumption is before development starts the problem is well understood but the solution is not and evolves during the process. Great. But what if the problem is not well understood? Could this lead to poor products? We think so. Surely both the problem and the solution should be agile?

In the conventional *Agile* model my role is, "Product Owner". It is my responsibility is to know the problem. In our Greenhouse *Agile* model *your* role is, "Product Owner" and as a user or potential user, surely you are best qualified.

Ed Deming said, "The customer is the most important part of the production line"but how often are we actually included?

My role? I'm, "The Gardner" tending the product seeds; nurturing, pruning and composting where appropriate.

The Bridge Worx Greenhouse is a collaborative online forum where we invite you to take an active role in our development projects.

www.bridgeworx.co.uk

david.oconnor@bridgeworx.co.uk

Thank you