

Re: emergent/swarm/evolutionary systems etc

Source: <http://coding.derkeiler.com/Archive/Python/comp.lang.python/2004-03/4945.html>

From: Josiah Carlson (jcarlson_at_uci.edu)

Date: 03/30/04

Date: Tue, 30 Mar 2004 10:01:35 -0800

> *From this, and other reading on genetic algorithms, I've found that nearly all of the applications being developed and used are far 'stronger' than what I have in mind. Most critically, the great preponderance of uses involve a program that simply modulates a set of given variables in search for an answer, rather than modifying the source code itself. Understandably, this is because, as stated above, such 'weak' methods are of limited use for most problems (I wouldn't want to fly in a plane that had been designed by one).*

> *Since, at the moment, I'm more interested in a 'pet cyber slime' than a tool for designing efficient circuit layouts, such 'strong' applications are of limited use. Even the 'state space' and 'random search' examples above are considerably 'stronger' than I would desire to use. For me, the goal is not only to create a program that can find an optimal solution to a problem, but which can also find better ways of learning – something disallowed by a fixed-algorithm engine.*

Take this with a grain of salt, but generally, expecting computers to learn is a high expectation. From neural networks (aka connectionist networks), genetic algorithms, naive bayesian statistics, etc., the desire in all of these cases is for a system that responds to a certain environment, produces some reasonable solutions in this environment, and works reasonably well in other environments.

The one limitation of all these approaches is that the algorithms and methods for doing this have limited knowledge (usually a few hundred bytes of state), no higher-order insight, etc. The algorithms, when given proper input and design, can solve certain problems. You may get lucky and your algorithm/data is applicable to another problem, but those situations are the rare exception, rather than the rule.

These 'learning' methods do the best when applied to what you have shown is called 'strong' problems. The 'weak' problems I've seen worked on, generally had weak (read poor) results. The stronger your problem, the better results you will likely have.

> *There's a lot of other interesting stuff on this site, but it'll take a*

- > *while to examine it all, and it'll be quite a while more before I have the*
- > *time to start on Stephen Wolfram's "A New Kind of Science". One question*
- > *though – can I, indeed, create, modify, run and save files using just a*
- > *python script?*

Don't think of Python as a scripting language. It can be used that way, but Python has better built-in data structures and language features than the most used languages out there (C, C++ and Java).

Can you create a file? Certainly, open a file for writing that didn't previously exist...

```
file_for_writing = open('filename', 'wb')
```

Can you modify a file? Certainly, open a file for reading and updating that exists...

```
file_for_updating_in_place = open('filename', 'r+b')
```

Want to run a file? (be careful though, executing the contents of a file can be dangerous)...

```
fil = open('filename', 'rb')
contents = fil.read()
fil.close()
exec(contents)
```

Want to save a file?

```
fil = open('filename', 'wb')
fil.write("some string of what you want to write")
fil.close()
```

– Josiah