Journal of
Computer Science & Systems Biology

Teahan, J Comput Sci Syst Biol 2013, 6:3
DOI: 10.4172/jcsb.1000e104

Editorial Open Access

Multiple Worlds and Branching Time for Bio-inspired Computation

William J Teahan*

School of Computer Science, Bangor University, Bangor, Wales, UK

Editorial

Bio-inspired computation takes inspiration from biological systems in order to perform some form of computation (such as finding a solution to a problem). Examples include evolutionary algorithms; swarm intelligence (for example, ant colony optimisation and particle swarm optimisation); artificial life; and artificial neural networks. Usually for these systems the underlying assumption is that time is linear (i.e., non-branching, where 'branching' time is a term used by concurrent systems meaning that two or more different courses of actions diverge). For example, the possibility of jumping ahead in time, or returning to a past computation, is usually not considered desirable. In addition, the environment and task in which the computational task takes place is usually considered within a single world rather than multiple worlds where computation takes place either in some pre-determined order or in parallel.

The purpose of this article is to draw attention to the further improvements that can arise if one considers alternative scenarios that make use of multiple worlds and systems where branching time is actively employed i.e. either the agents are transported from one world to another, or information is communicated from an agent in one world to another agent in a different world. This article will also draw attention to a new multiple world scenarios that leads to a significant improvement over existing approaches in the quality of the solution found.

Multiple world simulations are a standard requirement for many types of problems. These are problems where the fitness of an agent (or agents) needs to be evaluated by running a simulation. For example, for the Santa Fe Trail artificial ant problem, a standard benchmark used to evaluate evolutionary programming algorithms, each agent in the population being evolved has to have its behaviour evaluated by running a simulation in a world containing the Santa Fe Trail and the agent by itself, since the process of the agent eating food modifies the world. The evaluation would be invalidated if further agents were added to the world since the food counts and/or which food has been eaten would no longer necessarily be correct for all agents.

Figure 1 illustrates the way a typical evolutionary algorithm works for a problem that requires multiple world simulations to be processed. At the top of Figure 1 is the typical scenario for a sequential implementation and at the bottom the scenario for a parallel implementation. Both show the first two generations for a small population of seven agents. The square boxes represent the simulated worlds where the agent's performance at traversing the Santa Fe Trail is evaluated (these are labelled, for example, as W1:1 for the first world in Generation 1). Each box has a single agent depicted with a bug shape; in the figure, these have traversed along the trail with varying degrees of success. In between the two sets of worlds is shown the time lines that represent the order that events occur as witnessed by an observer external to the worlds. A circle represents an event (when a change occurs where a simulation might start, say, or an agent might transfer to another world). A vertical line represents standard linear time where the agents in the world continue their processing without transferring to or communicating with another world. The solid lines that cross the vertical time lines represent when an agent directly transfers between worlds. The dashed lines indicate when information is being transferred rather than the agent itself.

The figure depicts the case of a typical evolutionary algorithm where crossover and cloning operations occur (mutation can be considered an operation that, although changing the behaviour of an agent, has no effect on the time lines). A cloning operation, on the other hand, causes an agent to directly transfer from one world to another (e.g. in the figure, this occurs from world W1:4 to worlds W2:1 and W2:6). A crossover operation causes information to be transferred from

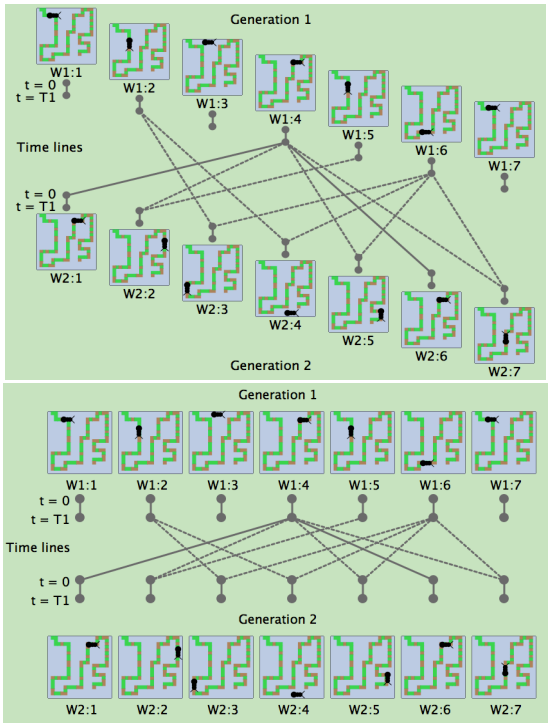

Figure 1: Time line diagrams for sequential (at the top) and parallel (at the bottom) evolutionary algorithms that require multiple world simulations in order to solve the Santa Fe Trail problem.

***Corresponding author:** William J Teahan, School of Computer Science, Bangor University, Bangor, Wales, UK, E-mail: w.j.teahan@bangor.ac.uk

Received June 06, 2013; **Accepted** June 10, 2013; **Published** June 14, 2013

Citation: Teahan WJ (2013) Multiple Worlds and Branching Time for Bio-inspired Computation. J Comput Sci Syst Biol 6: e104. doi:10.4172/jcsb.1000e104

agents in two parent worlds to create a new child agent–this occurs, for example, for parent worlds W1:2 and W1:6 to child world W2:4.

For the sequential scenario, each simulation needs to be run (from time t=0 to some termination time t=T1). To the external observer, the processing occurs in a series of downward steps since it only ever occurs in a single world at any one time. The parallel scenario, on the other hand is able to process the simulated worlds simultaneously (although in reality the times to run the simulations may vary, some taking longer than others to complete requiring more complex management of available processors than the theoretical case shown in the figure).

Clearly, an effective means for managing multiple worlds and branching time for evolution of simulation type problems can potentially lead to reductions in resources (such as time) required to find satisfactory solutions. But recent research has brought to light how the use of multiple worlds and branching time can also substantially improve the quality of the solutions found.

Headleand and Teahan have recently described three new evolutionary algorithms–Grammatical Herding ('GH'), a swarm based method that takes inspiration from the herding behaviour of horses to generate program-based solutions to problems [2]; seeded Grammatical Evolution ('GH seeded GE'), a method that uses GE [3,4] to seed the initial population prior to GE, again in order to generate a program-based solution [1]; and Template Based Evolution (TBE), a genetic evolution algorithm that evolves behaviour (rather than programs) for embodied situated agents whose fitness is tested implicitly through repeated trials in a world [5]. The scenario for GH seeded GE is shown on the top in Figure 2. Two worlds are shown–the left world where the GH algorithm is used to evolve the population of agents, and the right world where GE is used.

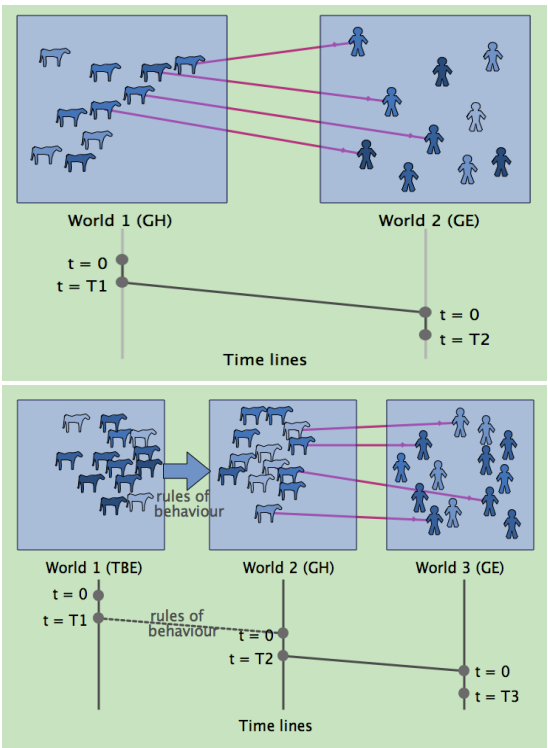

Figure 2: Time line diagrams for two new evolutionary algorithms, GH seeded GE on the left [1], and TBE-GH seeded GE on the right. (GH=Grammatical Herding [2]; GE=Grammatical Evolution [3,4]; TBE=Template Based Evolution [5]).

For simulation type problems such as the Santa Fe Trail problem, the scenarios for these algorithms are similar to that shown in Figure 1, but this has been simplified for visualisation purposes in Figure 2 as a single square. (We can abstractly consider this 'superset' of worlds as consisting of agents that are traversing a landscape (world) of possible program solutions; however, in reality, the implementation must treat each agent-based simulation separately in its own world for reasons stated above).

For the GH seeded GE algorithm shown at the top of Figure 2, there are two separate time lines for the two worlds, one for the world where GH is controlling the behaviour of the agents (represented by the horse shapes in the diagram), and one for the GE world (where the agents are represented by person shapes). The GH simulation starts at time t=0, then at some termination time t=T1 the best performing agents will then transfer directly without change to start the GE simulation at time t=0. (This is shown by the magenta lines at the top of the diagram). These agents will run until some termination time t=T2. Note that the notion of branching time is clear in this scenario. Both the GH and GE worlds could continue with their processing and there is nothing stopping a later transferral of agents between worlds, although this has not been implemented for the method described in [1]. It is important to note that the GH seeded GE algorithm was able to find a solution for the Santa Fe Trail problem significantly better than the current state of the art published in [4].

A further scenario we will also investigate is shown at the bottom of Figure 2. In this case, we have added a further world where we propose first using TBE to evolve the herding behaviour of the GH agents. Once this has run and well-performing rules of behaviour have been found, this information will be transferred to the GH world to hopefully improve the effectiveness of the swarm-based herding agents. These agents will then be transmitted to the GE world by seeding the initial population for that world.

Conclusion

Scenarios involving multiple worlds and different branching time configurations provide a virtually untapped resource for future research into bio-inspired computation. These scenarios can be useful for helping to design improved parallel algorithms, and have the potential to produce more effective solutions to problems.

References

1. Headleand C, Teahan WJ (2013) "Swarm based population seeding of Grammatical Evolution". J Comput Sci Syst Biol.
2. Headleand C, Teahan WJ (2013) "Grammatical Herding". J Comput Sci Syst Biol 6: 43-47.
3. O'Neill M, Ryan C (2003) Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language. Kluwer.
4. Georgiou L, Teahan WJ (2011) "Constituent Grammatical Evolution". Proc. Int. Joint Conf on AI (IJCAI): Barcelona, Spain.
5. Headleand C, Teahan WJ (2013) "Template Based Evolution". EcoMASS Workshop, Genetic and Evolutionary Computation Conference (GECCO): Amsterdam, Netherlands.