EvoMachina: a novel evolutionary algorithm inspired by bacterial genome reorganisation

Tim Hoverd and Susan Stepney

Department of Computer Science, University of York, UK
York Centre for Complex Systems Analysis, University of York, UK
tim.hoverd@york.ac.uk, susan.stepney@york.ac.uk

EvoMachina is a novel natural computation algorithm, inspired by recent understandings of the processes of genome reorganisation in bacteria and viruses. The basics concepts of EvoMachina are Machines, Templates, Repositories, Domains, and Spaces [1].

- **Machine.** A machine is an active component, performing the various operations in the system, such as mutation, replication, expression, and domain-specific activities. It is the analogue of the protein in a cell.
- **Template.** Machines are described by Templates. Translator machines build a specific machine from its template description. A template is the analogue of mRNA in cells.
- **Repository.** Templates are stored in a Repository. Transcriber machines extract individual templates from the repository. The repository is the analogue of a DNA chromosome in cells.
- **Domain.** The Domain captures the 'physics' of a particular Repository and its associated Machines. There may be multiple domains, in particular, domains related to evolution, and domains related to problem-specific features.
- **Space.** An Individual is a Space that contains machines, templates, and repositories. It is the analogue of a cell. A Site is a Space that contains individuals; it is the implementation of physical location.

An Individual is a candidate solution. Individuals replicate when some criterion is met, such as a clock tick and winning a tournament for generational algorithms, or a suitable energy level or machine concentration being reached in other approaches. The tournaments, or other processes that trigger replication, are mediated by the sites, through Environment Orientation [4].

Replication is executed by a specific machine, which copies the repositories, with mutation, and clones or shares relevant machines, depending on the particular parameter settings. This replication, although executed by machines, is not itself an implementation of a biological process; rather it is a *shortcut* mechanism [2], allowing the algorithm to focus on the relevant biological concepts.

Example: Travelling Salesperson Problem (TSP)

The TSP Domain encodes the cities, the distances between each pair, and the calculation of fitness in terms of journey length. The TSP Repository comprises

a single template, the list of cities, which is a direct encoding of the candidate solution.

The Evo Domain encodes details of k-opt mutation operations on permutations. The Evo Repository is a list of integers, comprising the available values of k for k-opt mutations. The Evo Template is a single integer value to be used in a k-opt machine. The Evo Transcriber Machine build an Evo Replicator Machine from an Evo Template, embedding the specific value of k. The Evo Replicator Machine makes a copy of its individual; it uses the specific value of k built into it by the Translator Machine to mutate the TSP repository, building a new candidate solution; it also uses a hard-wired 'physics' to mutate the Evo Repository, affecting the k values available in the new individual.

An Individual initially comprises a TSP Repository, an Evo Repository, and Transcriber and Translator Machines. The Translator then builds a particular Replicator when needed.

The overall Space can either be a single site, containing multiple individuals, corresponding to a well-mixed system, or it can be an n-D toroidal lattice of sites, for a spatially varying system, with mobile individuals.

When run, the fittest individual tens to the shortest pathlength; the value of k used to produce fit individuals drops from an initial high value to 2.

Summary

The relatively complex architecture of EvoMachina incorporates many of the concepts of modern biological evolutionary knowledge: genome reordering, translation and transcription expression, evolving mutability, metabolic networks (through further machines encoded in repositories, or hard-wired, not covered here), and more. The aim is to develop a system that can evolve its evolvability, to adapt to online data streams, and support an open-ended reflective evolutionary system [3].

Acknowledgements. This work is funded by the EU FP7 project EvoEvo, grant number 610427.

References

1. Paul Andrews and Susan Stepney. A metamodel for the evolution of evolution. In *ECAL 2015, York, UK, July 2015*, pages 621–628. MIT Press, 2015.
2. Wolfgang Banzhaf, Bert Baumgaertner, Guillaume Beslon, Ren Doursat, James A. Foster, Barry McMullin, Vinicius Veloso de Melo, Thomas Miconi, Lee Spector, Susan Stepney, and Roger White. Defining and simulating open-ended novelty: Requirements, guidelines, and challenges. *Theory in Biosciences*, 2016. doi:10.1007/s12064-016-0229-7.
3. Simon Hickinbotham and Susan Stepney. Bio-reflective architectures. In *ALife 2016, Cancun, Mexico, July 2016*. MIT Press, 2016.
4. Tim Hoverd and Susan Stepney. Environment orientation: a structured simulation approach for agent-based complex system. *Natural Computing*, 14(1):83–97, 2015.