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

Introduction

EvoMachina is a novel natural computation algorithm, inspired by recent understandings of the processes of genome reorganisation in bacteria and viruses. It has been developed as part of the EU FP7 project EvoEvo, taking inspiration from its biological experiments, and developed to support Living Technology applications.

This abstract outlines the conceptual model underlying EvoMachina, its implementation, and a reference application.

The conceptual model

The basics concepts of EvoMachina are Structures (Machines, Templates, Repositories), Domains, and Spaces (individuals and Sites) [1], see figure 1.

- **Structure**. A structure is a sequence of objects of a particular type, that stores information, and may additionally have behaviour.
  - **Machine**. A machine is an active structure; different machines perform 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. Machines may degrade, and so need continual replenishment.
  - **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. Kloner machines build mutated repositories on replication. Any information incorporated in associated machines can therefore evolve. The repository is the analogue of a DNA chromosome in cells.
- **Domain**. A domain captures the ‘physics’ of a particular collection of structures. There may be multiple domains, in particular, domains related to evolution, and domains related to problem-specific features.

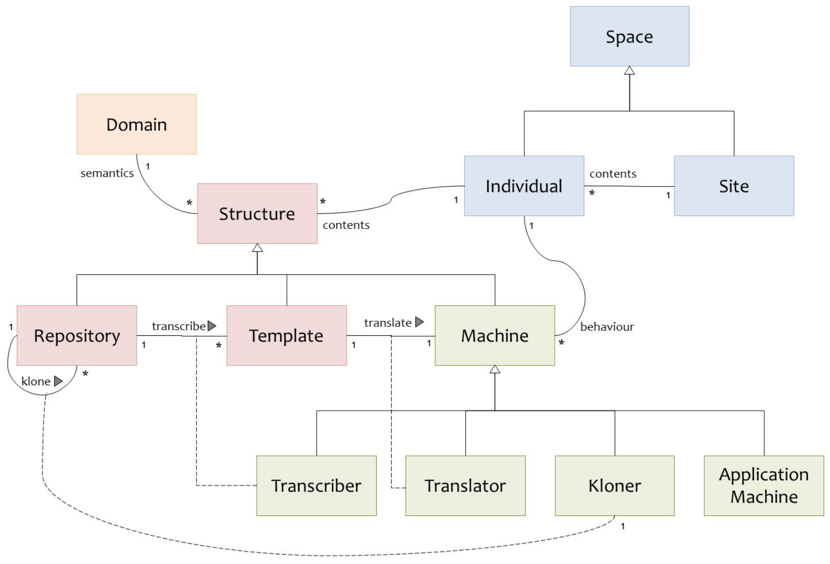

Fig. 1. Conceptual model of EvoMachina: see text for details.

– **Space**.
  - **Individual**. An individual is a space that contains a collection of Structures, comprising various machines, templates, and repositories. Its behaviour is given by the activity of its various machines. It is the analogue of a cell. Individuals may contain other individuals, analogues of cells with compartments.
  - **Site**. A site is a space that can contain individuals; it is the implementation of physical location. A site may also contain others sites, allowing hierarchical spatial structures.

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.

Implementation

The conceptual model outlined above has been translated into a 'Platform model' [2, 7], making design decisions suitable for implementation in a concurrent object-oriented language.

The EvoMachina framework [5] is implemented in Java8, using its concurrency management to allow machines and spaces to execute concurrently, and exploiting lambda expressions for succinct customisation of application-specific machines and domains.

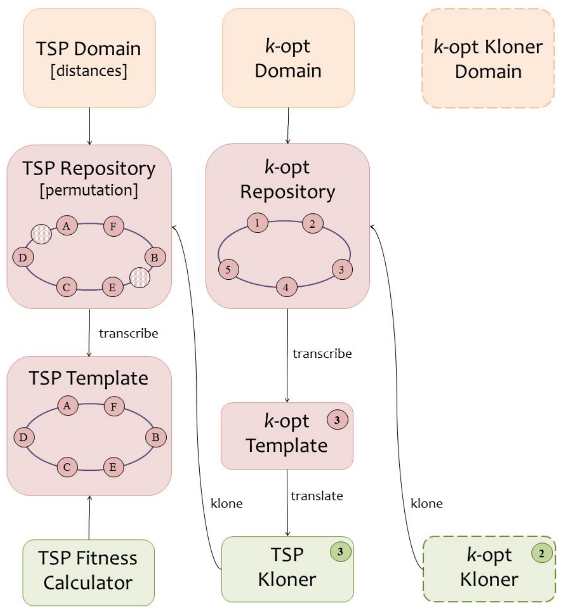

Fig. 2. Instantiation of an EvoMachina Individual for the TSP: see text for details. (Some machines are omitted for clarity.)

Replication uses a specific machine, the Kloner, which copies the repositories with mutation (including genome reorganisation operations). Relevant machines are cloned or shared, 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 [3], allowing the algorithm to focus on the relevant biological concepts.

The tournaments, or other processes that trigger replication, are mediated by the sites, through Environment Orientation [6].

Example: Travelling Salesperson Problem (TSP)

A reference implementation of TSP is provided with the EvoMachina framework. It comprises three domains with the following structures (figure 2):

- TSP Domain
  - The 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, a particular permuted list of cities, along with some non-coding items.
  - The (hard-coded) TSP Transcriber machine builds a TSP template from the TSP repository. This template comprises the permuted list of cities

  with the non-coding items removed, which is a direct encoding of the candidate solution.
  - The (hard-coded) application-specific TSP fitness calculator machine is used by the domain to calculate the fitness (path-length) of a candidate solution template.

- k-opt Domain
  - The k-opt domain encodes details of k-opt mutation operations on TSP repository permutations.
  - The k-opt repository is a list of integers, comprising the available values of k for k-opt mutations.
  - The (hard-coded) k-opt Transcriber machine builds a k-opt template from the k-opt repository. The k-opt template is a single integer value.
  - The (hard-coded) k-opt Translator machine builds a TSP Kloner machine from a k-opt template, embedding the specific value of k along with some hard-coded behaviour.
  - The TSP Kloner machine is used in the making of a copy of its individual; it uses its specific value of k built in by the Translator machine to mutate the TSP repository, building a new candidate solution.

- k-opt Kloner Domain
  - The k-opt kloner domain encodes details of mutation operations on k-opt repositories.
  - The (hard-wired) k-opt kloner machine is used in the making of a copy of its individual; to copy and mutate the k-opt Repository in a way defined in the k-opt Kloner Domain, changing the k values available in the new individual.

An Individual initially comprises a TSP Repository, a k-opt Repository, and the relevant Transcriber and Translator Machines. These express the particular Kloners 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 on 48 USA State capitals data, the fittest individuals tend to the shortest pathlength (the optimum value of 33551). See figure 3.

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 [4].

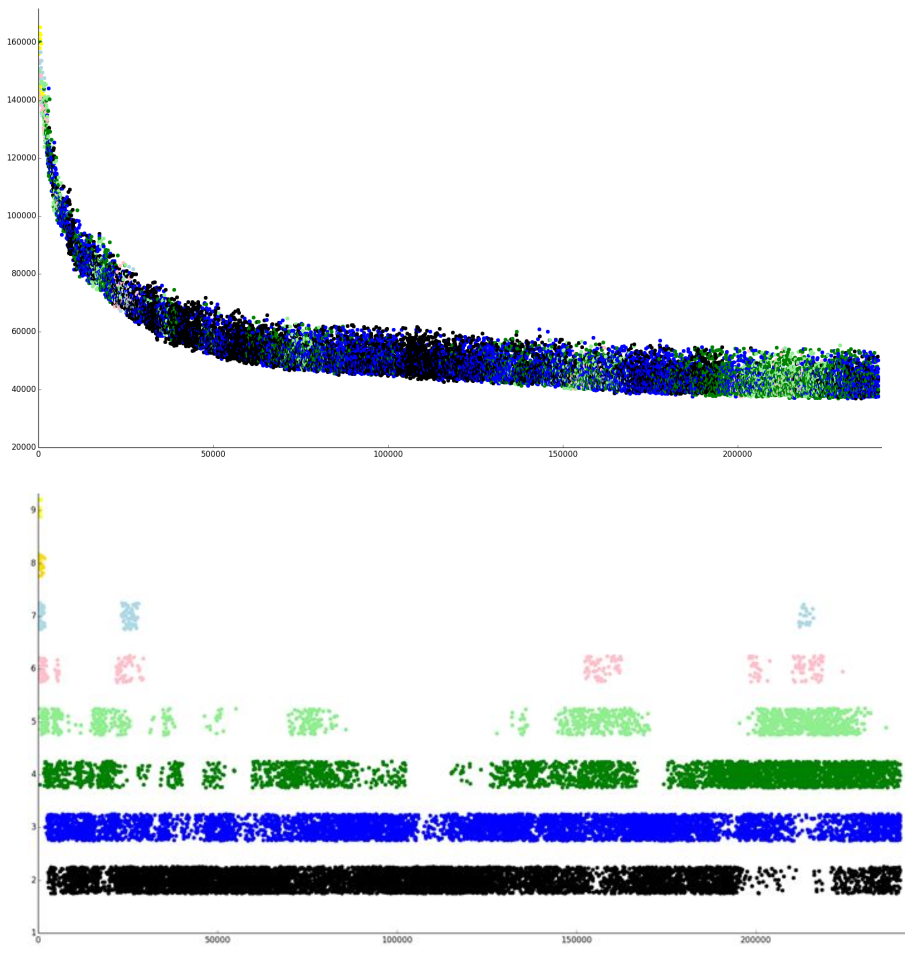

Fig. 3. Results from a typical run of EvoMachina on the 48 US state capitals TSP: see text for details. Each dot represents the birth of an individual; the colour represents the k-value of the parent used to create this individual: darker colours represent lower k values. (top) fitness plotted against time of birth; (bottom) k-value of parent plotted against time of birth.

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. Paul S. Andrews, Susan Stepney, and Jon Timmis. Simulation as a scientific instrument. In *Proceedings of the 2012 Workshop on Complex Systems Modelling and Simulation, Orleans, France, September 2012*, pages 1–10. Luniver Press, 2012.
3. 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.
4. Simon Hickinbotham and Susan Stepney. Bio-reflective architectures. In *ALife 2016, Cancun, Mexico, July 2016*. MIT Press, 2016.
5. Tim Hoverd. EvoMachina user documentation. EvoEvo deliverable 4.3, Department of Computer Science, University of York, 2016. Available with EvoMachina software from github/evoevo-york/evomachina.
6. Tim Hoverd and Susan Stepney. Environment orientation: a structured simulation approach for agent-based complex system. *Natural Computing*, 14(1):83–97, 2015.
7. Susan Stepney and Paul S. Andrews. CoSMoS special issue editorial. *Natural Computing*, 14(1):1–6, 2015.