The Journal of Open Source Software

hstrat: a Python Package for phylogenetic inference on distributed digital evolution populations

Matthew Andres Moreno [1], **Emily Dolson** [1], **and Charles Ofria** [1]

1 Michigan State University

DOI: 10.21105/joss.04866

Software

- Review
- Repository
- Archive

Editor: Øystein Sørensen

Reviewers:

- @GeekLogan
- @JJ
- @kgd-al

Submitted: 16 October 2022
Published: 11 December 2022

Summary

Digital evolution systems instantiate evolutionary processes over populations of virtual agents *in silico*. These programs can serve as rich experimental model systems. Insights from digital evolution experiments expand evolutionary theory, and can often directly improve heuristic optimization techniques (Hernandez, Lalejini, & Dolson, 2022a; Hernandez, Lalejini, & Ofria, 2022). Perfect observability, in particular, enables *in silico* experiments that would be otherwise impossible *in vitro* or *in vivo*. Notably, availability of the full evolutionary history (phylogeny) of a given population enables very powerful analyses (Dolson & Ofria, 2018; Hernandez, Lalejini, & Dolson, 2022b; Shahbandegan et al., 2022).

As a slow but highly parallelizable process, digital evolution will benefit greatly by continuing to capitalize on profound advances in parallel and distributed computing (D. Ackley & Small, 2014; Moreno & Ofria, 2020), particularly emerging unconventional computing architectures (D. H. Ackley & Williams, 2011; Furber et al., 2014; Lauterbach, 2021). However, scaling up digital evolution presents many challenges. Among these is the existing centralized perfect-tracking phylogenetic data collection model (Bohm et al., 2017; De Rainville et al., 2012; Godin-Dubois et al., 2019; Ofria et al., 2020; Ofria & Wilke, 2004), which is inefficient and difficult to realize in parallel and distributed contexts. Here, we implement an alternative approach to tracking phylogenies across vast and potentially unreliable hardware networks.

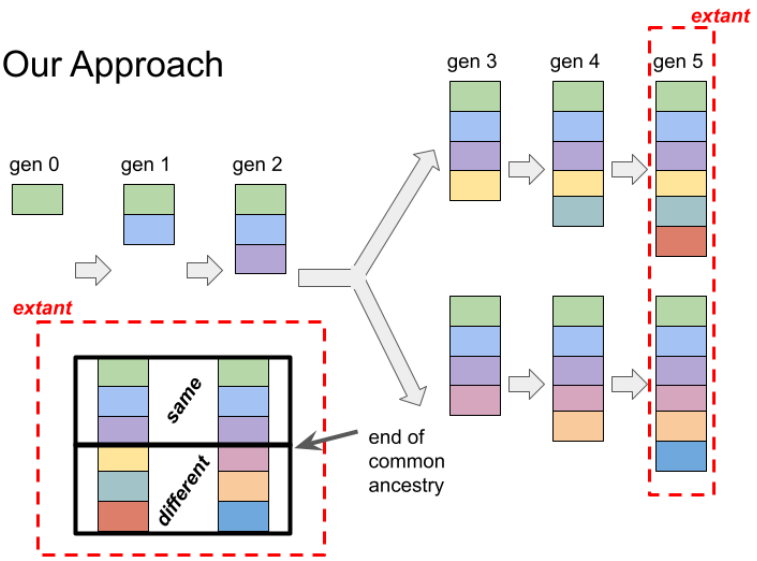

Figure 1: Example scenario of hereditary stratigraph annotation inheritance. Each generation, a new randomly-generated "fingerprint" (drawn as a solid-colored rectangle) is appended to the genome annotation. Genomes' annotations will share identical fingerprints for the generations they experienced shared ancestry. Estimation of the most recent common ancestor (MRCA) generation is straightforward: the first mismatching fingerprints denote the end of common ancestry.

The `hstrat` Python library exists to facilitate application of hereditary stratigraphy, a cutting-edge technique to enable phylogenetic inference over distributed digital evolution populations (Moreno et al., 2022). This technique departs from the traditional perfect-tracking approach to phylogenetic record-keeping. Instead, hereditary stratigraphy enables phylogenetic history to be inferred from heritable annotations attached to evolving digital agents. This approach aligns with phylogenetic reconstruction methodologies in evolutionary biology (Gaffney, 1979; Horner & Pesole, 2004). Hereditary stratigraphy attaches a set of immutable historical "checkpoints" — referred to as *strata* — as an annotation on evolving genomes. Figure 1 illustrates how these strata accumulate over generations and how they can be used to infer the phylogenetic relationships.

Checkpoints can be strategically discarded to reduce annotation size at the cost of increasing inference uncertainty. A particular strategy for which checkpoints to discard when is referred to as a *stratum retention policy*. We refer to the set of retained strata as a *hereditary stratigraphic column*.

Appropriate stratum retention policy choice varies by application. For example, if annotation size is not a concern it may be best to preserve all strata. In other situations, it may be necessary to constrain annotation size to remain within a fixed memory budget.

Key features of the library include:

- object-oriented hereditary stratigraphic column implementation to annotate arbitrary genomes,
- modular interchangeability and user extensibility of stratum retention policies,
- programmatic interface to query guarantees and behavior of stratum retention policy,
- modular interchangeability and user extensibility of back-end data structure used to store annotation data,
- a suite of visualization tools to elucidate stratum retention policies,
- support for automatic parameterization of stratum retention policies to meet user size complexity or inference precision specifications,

- tools to compare two columns and extract information about the phylogenetic relationship between them,
- extensive documentation hosted on ReadTheDocs,
- a comprehensive test suite to ensure stability and reliability,
- convenient availability as a Python package via the PyPI repository, and
- pure Python implementation to ensure universal portability.

Statement of Need

The `hstrat` software exists to equip parallel and distributed evolution digital systems — simulations that instantiate the process of evolution in an agent-based framework — with phylogenetic tracking capabilities. Parallel and distributed computation exponentiates the power of digital evolution by allowing for larger populations, more generations, more sophisticated genotype-phenotype mappings, and more robust fitness functions (Channon, 2019; Harding & Banzhaf, 2007; Langdon & Banzhaf, 2019; Miikkulainen et al., 2019). Indeed, several notable projects within the field have successfully exploited massively parallel and distributed computational resources (Bennett III et al., 1999; Blondeau et al., 2009; Sims, 1994). Further development of methodology and software such as this work will position the field to continue leveraging ongoing advances in computing hardware.

The capability to detect phylogenetic cues within digital evolution has become increasingly necessary in both applied and scientific contexts. These cues unlock *post hoc* insight into evolutionary history — particularly with respect to ecology and selection pressure — but also can be harnessed to drive digital evolution algorithms as they unfold (Burke et al., 2003; Murphy & Ryan, 2008). However, parallel and distributed evaluation complicates, among other concerns, maintenance of an evolutionary record. Existing phylogenetic record keeping requires inerrant and complete collation of birth and death reports within a centralized data structure. Such perfect tracking approaches are brittle to data loss or corruption and impose communication overhead.

Hereditary stratigraphy methodology, and the implementing `hstrat` software library, meets the demand for efficient, tractable, and robust phylogenetic inference at scale. This approach exchanges a centralized perfect record of history for a process where history is estimated from comparison of available extant genomes, aligning with the paradigm of phylogenetic inference in wet biology.

Although targeted to digital evolution use cases, impact of our work extends beyond to the various applications of digital evolution. Evolutionary biology poses uniquely abstract, nuanced, and nebulous questions, such as the origins of life and the evolution of complexity (McShea & Simpson, 2011; Pross, 2016). Computational modeling provides one foothold for such inquiry, particularly with respect to phenomena that typically unfold on geological timescales or hypotheses that invoke counterfactuals outside the bounds of physical reality (Clune et al., 2011; McKinley et al., 2008; O'Neill, 2003). Elsewhere, in the realm of computational optimization, heuristic algorithms provide a foothold to explore high-dimensional and deceptive search spaces (Eiben & Smith, 2015). Improvements to distributed phylogenetic tracking can benefit both of these niches, providing means to test more sophisticated evolutionary hypotheses and means to discover better solutions to hard problems.

In addition to its founding purview within digital evolution, we anticipate hereditary stratigraphy — and this software implementation — may find applications within other domains of distributed computing. For example, hereditary stratigraphy could equip a population protocol system with the capability for on-the-fly estimation of the relationship between descendants of a forking message chain or a forking process tree (Angluin et al., 2006; Aspnes & Ruppert, 2009).

Across all these domains, free availability of the `hstrat` software will play a central role in bringing hereditary stratigraphy methodology into practice. Library development incorporates intentional design choices to promote successful outside use, including

- modular, hierarchical, well-named, and consistent API,
- comprehensive and application-oriented documentation,
- "batteries included" provision of many stratum retention policy options covering a wide variety of use cases,
- declarative configuration interfaces (i.e., automatic parameterization), and
- emphasis on user extensibility without modification of core library code.

Projects Using the Software

A pre-release version of this software was used to perform experiments, validate derivations, and create visualizations for the conference article introducing the method of hereditary stratigraphy (Moreno et al., 2022).

A native version of this software is being incorporated into the next version of DISHTINY, a digital framework for studying evolution of multicellularity (Moreno et al., 2021; Moreno & Ofria, 2022). We also anticipate making this software available through the Modular Agent Based Evolution framework as a community-contributed component (Bohm et al., 2019).

Related Software

To our knowledge, no existing software is available to facilitate phylogenetic analyses of fully-distributed digital evolution populations. Centralized phylogenetic tracking, however, is a common practice in digital evolution systems. Many rely on custom "hand-rolled" solutions. However, several general-purpose libraries and frameworks exist to facilitate centralized phylogenetic tracking. These include,

- Automated Phylogeny Over Geological Timescales (APOGeT) (Godin-Dubois et al., 2019),
- Distributed Evolutionary Algorithms in Python (DEAP) (De Rainville et al., 2012),
- Empirical (Ofria et al., 2020), and
- Modular Agent-Based Evolution Framework (MABE) (Bohm et al., 2017).

Empirical, MABE, and APOGeT are C++ libraries. DEAP, eponymously, is a Python library.

Empirical and MABE's phylogenetic tracking tools cater to asexual lineages, while DEAP and APOGeT focus on sexual lineages. Both MABE and Empirical incorporate options for on-the-fly pruning, where records for extinct lineages are deleted in order to temper memory use. APOGeT focuses on species-level tracking rather than individual-level genealogical tracking.

Note that the DEAP's distributing computing features are organized around a centralized controller-worker model. Phylogenetic tracking takes place on the controller process. If any selection and reproduction takes place on worker processes (i.e., island model), migration is managed by the controller process. In this scenario, any phylogenetic tracking requires all phylogenetic history occuring on islands in between migrations to be reported back to the controller process.

In prepublication work, we performed phylogenetic inference using fixed-length bitstring components of digital genomes subjected to bitwise mutation. (This work motivated development of the more efficient and robust hereditary stratigraph annotation scheme.) A huge number of biology-oriented phylogenetic reconstruction tools have been developed, but we found the design of BioPython's `Phylo` module particularly well suited to non-DNA data (Cock et al., 2009).

Future Work

The `hstrat` project remains under active development.

A major next objective for the `hstrat` project will be development of a header-only C++ library to complement the Python library presented here. This implementation will improve memory and CPU efficiency as well as better integrating with many scientific computing or embedded systems applications, which often employ native code to meet performance requirements.

This direction opens the possibility of adding support for other high-level languages in the future (Beazley, 2003). Indeed, at a minimum, we plan to update the Python library to include support for wrapping these native components (Jakob et al., 2017). However, the pure Python implementation will remain as a fallback for platforms lacking native support.

As released in version 1.0.1, the `hstrat` library contains a comprehensive set of tools to perform pairwise comparisons between extant hereditary stratigraphic columns. However, a key use case for the library will be phylogenetic reconstruction over entire populations. We plan to expand the library to add tools for population-level phylogenetic reconstruction and visualization in future releases.

Although, as currently released, `hstrat` does support serialization and de-serialization via Python's pickle protocol, which suffices for data storage and transfer within and between Python contexts, human-readable and binary serialization formats intercompatible outside of Python will be crucial.

Acknowledgements

Thank you to reviewers Kevin Godin-Dubois, Juan Julián Merelo Guervós, and Logan Walker. Their comments and contributions improved the maintainability of the software, particularly with respect to improving usability for outside developers.

This research was supported in part by NSF grants DEB-1655715 and DBI-0939454 as well as by Michigan State University through the computational resources provided by the Institute for Cyber-Enabled Research. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1424871. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Ackley, D. H., & Williams, L. R. (2011). Homeostatic architectures for robust spatial computing. *2011 Fifth IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops*, 91–96. https://doi.org/10.1109/sasow.2011.18

Ackley, D., & Small, T. (2014). *Indefinitely scalable computing = artificial life engineering*. *ALIFE 14: The Fourteenth International Conference on the Synthesis and Simulation of Living Systems*, 606–613. https://direct.mit.edu/isal/proceedings/alife2014/606/99046

Angluin, D., Aspnes, J., Eisenstat, D., & Ruppert, E. (2006). On the power of anonymous one-way communication. In J. H. Anderson, G. Prencipe, & R. Wattenhofer (Eds.), *Principles of distributed systems* (pp. 396–411). Springer Berlin Heidelberg. https://doi.org/10.1007/11795490_30

Aspnes, J., & Ruppert, E. (2009). *An introduction to population protocols*. 97–120. https://doi.org/10.1007/978-3-540-89707-1_5

Beazley, D. M. (2003). Automated scientific software scripting with SWIG. *Future Generation Computer Systems*, *19*(5), 599–609. https://doi.org/10.1016/s0167-739x(02)00171-1

Bennett III, F. H., Koza, J. R., Shipman, J., & Stiffelman, O. (1999). Building a parallel computer system for $18,000 that performs a half peta-flop per day. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, & R. E. Smith (Eds.), *Proceedings of*

the genetic and evolutionary computation conference (Vol. 2, pp. 1484–1490). Morgan Kaufmann. ISBN: 1-55860-611-4

Blondeau, A., Cheyer, A., Hodjat, B., & Harrigan, P. (2009). *Distributed network for performing complex algorithms*. Google Patents.

Bohm, C., G., N. C., & Hintze, A. (2017). *MABE (modular agent based evolver): A framework for digital evolution research. ECAL 2017, the Fourteenth European Conference on Artificial Life*, 76–83. https://direct.mit.edu/isal/proceedings/ecal2017/76/99520

Bohm, C., Lalejini, A., Schossau, J., & Ofria, C. (2019). MABE 2.0: An introduction to MABE and a road map for the future of MABE development. *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 1349–1356. https://doi.org/10.1145/3319619.3326825

Burke, E. K., Gustafson, S., Kendall, G., & Krasnogor, N. (2003). Is increased diversity in genetic programming beneficial? An analysis of lineage selection. *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.*, *2*, 1398–1405 Vol.2. https://doi.org/10.1109/cec.2003.1299834

Channon, A. (2019). Maximum individual complexity is indefinitely scalable in Geb. *Artificial Life*, *25*(2), 134–144. https://doi.org/10.1162/artl_a_00285

Clune, J., Goldsby, H. J., Ofria, C., & Pennock, R. T. (2011). Selective pressures for accurate altruism targeting: Evidence from digital evolution for difficult-to-test aspects of inclusive fitness theory. *Proceedings of the Royal Society B: Biological Sciences*, *278*(1706), 666–674. https://doi.org/10.1098/rspb.2010.1557

Cock, P. J. A., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B., & Hoon, M. J. L. de. (2009). Biopython: Freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, *25*(11), 1422–1423. https://doi.org/10.1093/bioinformatics/btp163

De Rainville, F.-M., Fortin, F.-A., Gardner, M.-A., Parizeau, M., & Gagné, C. (2012). DEAP: A Python framework for evolutionary algorithms. *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation*, 85–92. https://doi.org/10.1145/2330784.2330799

Dolson, E., & Ofria, C. (2018). *Ecological theory provides insights about evolutionary computation*. 105–106. https://doi.org/10.1145/3205651.3205780

Eiben, A. E., & Smith, J. E. (2015). *Introduction to evolutionary computing* (2nd ed.). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-44874-8

Furber, S. B., Galluppi, F., Temple, S., & Plana, L. A. (2014). The SpiNNaker project. *Proceedings of the IEEE*, *102*(5), 652–665. https://doi.org/10.1109/jproc.2014.2304638

Gaffney, E. S. (1979). An introduction to the logic of phylogeny reconstruction. In *Phylogenetic analysis and paleontology* (pp. 79–112). Columbia University Press. https://doi.org/10.7312/crac92306-005

Godin-Dubois, K., Cussat-Blanc, S., & Duthen, Y. (2019, August). *APOGeT: Automated phylogeny over geological timescales*. https://doi.org/10.13140/rg.2.2.33781.93921

Harding, S., & Banzhaf, W. (2007). Fast genetic programming and artificial developmental systems on GPUs. *21st International Symposium on High Performance Computing Systems and Applications (HPCS'07)*, 2–2. https://doi.org/10.1109/hpcs.2007.17

Hernandez, J. G., Lalejini, A., & Dolson, E. (2022a). Phylogenetic diversity predicts future success in evolutionary computation. *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 23–24. https://doi.org/10.1145/3520304.3534079

Hernandez, J. G., Lalejini, A., & Dolson, E. (2022b). What can phylogenetic metrics tell us about useful diversity in evolutionary algorithms? In W. Banzhaf, L. Trujillo, S. Winkler, & B. Worzel (Eds.), *Genetic programming theory and practice XVIII* (pp. 63–82). Springer Nature Singapore. https://doi.org/10.1007/978-981-16-8113-4_4

Hernandez, J. G., Lalejini, A., & Ofria, C. (2022). A suite of diagnostic metrics for characterizing selection schemes. *arXiv Preprint arXiv:2204.13839*. https://doi.org/10.48550/arxiv.2204.13839

Horner, D. S., & Pesole, G. (2004). Phylogenetic analyses: A brief introduction to methods and their application. *Expert Review of Molecular Diagnostics*, *4*(3), 339–350. https://doi.org/10.1586/14737159.4.3.339

Jakob, W., Rhinelander, J., & Moldovan, D. (2017). *pybind11 – seamless operability between C++11 and Python*.

Langdon, W. B., & Banzhaf, W. (2019). *Continuous long-term evolution of genetic programming*. *ALIFE 2019: The 2019 Conference on Artificial Life*, 388–395. https://doi.org/10.1162/isal_a_00191

Lauterbach, G. (2021). The path to successful wafer-scale integration: The Cerebras story. *IEEE Micro*, *41*(6), 52–57. https://doi.org/10.1109/mm.2021.3112025

McKinley, P., Cheng, B. H. C., Ofria, C., Knoester, D., Beckmann, B., & Goldsby, H. (2008). Harnessing digital evolution. *Computer*, *41*(1), 54–63. https://doi.org/10.1109/mc.2008.17

McShea, D. W., & Simpson, C. (2011). The miscellaneous transitions in evolution. In *The major transitions in evolution revisited*. The MIT Press. https://doi.org/10.7551/mitpress/9780262015240.003.0002

Miikkulainen, R., Liang, J., Meyerson, E., Rawal, A., Fink, D., Francon, O., Raju, B., Shahrzad, H., Navruzyan, A., Duffy, N., & Hodjat, B. (2019). Evolving deep neural networks. In R. Kozma, C. Alippi, Y. Choe, & F. C. Morabito (Eds.), *Artificial intelligence in the age of neural networks and brain computing* (pp. 293–312). Academic Press. https://doi.org/10.1016/b978-0-12-815480-9.00015-3

Moreno, M. A., Dolson, E., & Ofria, C. (2022). Hereditary stratigraphy: Genome annotations to enable phylogenetic inference over distributed populations. *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, *ALIFE 2022: The 2022 Conference on Artificial Life*, 65–66. https://doi.org/10.1162/isal_a_00550

Moreno, M. A., & Ofria, C. (2020). *Practical steps toward indefinite scalability: In pursuit of robust computational substrates for open-ended evolution*. OSF. https://doi.org/10.17605/osf.io/53vgh

Moreno, M. A., & Ofria, C. (2022). Exploring evolved multicellular life histories in a open-ended digital evolution system. In *Frontiers in Ecology and Evolution* (Vol. 10). https://doi.org/10.3389/fevo.2022.750837

Moreno, M. A., Rodriguez Papa, S., & Ofria, C. (2021). Case study of novelty, complexity, and adaptation in a multicellular system. *The 2021 Conference on Artificial Life*.

Murphy, G., & Ryan, C. (2008). A simple powerful constraint for genetic programming. In M. O'Neill, L. Vanneschi, S. Gustafson, A. I. Esparcia Alcázar, I. De Falco, A. Della Cioppa, & E. Tarantino (Eds.), *Genetic programming* (pp. 146–157). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-78671-9_13

O'Neill, B. (2003). Digital evolution. *PLOS Biology*. https://doi.org/10.1371/journal.pbio.0000018

Ofria, C., Moreno, M. A., Dolson, E., Lalejini, A., Rodriguez Papa, S., Fenton, J., Perry, K., Jorgensen, S., hoffmanriley, grenewode, Baldwin Edwards, O., Stredwick, J., cgnitash,

theycallmeHeem, Vostinar, A., Moreno, R., Schossau, J., Zaman, L., & djrain. (2020). *Empirical: C++ library for efficient, reliable, and accessible scientific software* (Version 0.0.4) [Computer software]. https://doi.org/10.5281/zenodo.4141943

Ofria, C., & Wilke, C. O. (2004). Avida: A software platform for research in computational evolutionary biology. *Artificial Life*, *10*(2), 191–229. https://doi.org/10.1162/106454604773563612

Pross, A. (2016). *What is life?: How chemistry becomes biology*. Oxford University Press. ISBN: 9780198784791

Shahbandegan, S., Hernandez, J. G., Lalejini, A., & Dolson, E. (2022). Untangling phylogenetic diversity's role in evolutionary computation using a suite of diagnostic fitness landscapes. *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2322–2325. https://doi.org/10.1145/3520304.3534028

Sims, K. (1994). Evolving 3D morphology and behavior by competition. *Artificial Life*, *1*(4), 353–372. https://doi.org/10.1162/artl.1994.1.4.353