WIKIPEDIA

Evolutionary algorithm

In computational intelligence (CI), an **evolutionary algorithm** (**EA**) is a subset of evolutionary computation,[1] a generic population-based metaheuristic optimization algorithm. An EA uses mechanisms inspired by biological evolution, such as reproduction, mutation, recombination, and selection. Candidate solutions to the optimization problem play the role of individuals in a population, and the fitness function determines the quality of the solutions (see also loss function). Evolution of the population then takes place after the repeated application of the above operators.

Evolutionary algorithms often perform well approximating solutions to all types of problems because they ideally do not make any assumption about the underlying fitness landscape. Techniques from evolutionary algorithms applied to the modeling of biological evolution are generally limited to explorations of microevolutionary processes and planning models based upon cellular processes. In most real applications of EAs, computational complexity is a prohibiting factor.[2] In fact, this computational complexity is due to fitness function evaluation. Fitness approximation is one of the solutions to overcome this difficulty. However, seemingly simple EA can solve often complex problems; therefore, there may be no direct link between algorithm complexity and problem complexity.

Contents

Implementation

The following is an example of a generic single-objective genetic algorithm.

Step One: Generate the initial population of individuals randomly. (First generation)

Step Two: Repeat the following regenerational steps until termination:

1. Evaluate the fitness of each individual in the population (time limit, sufficient fitness achieved, etc.)
2. Select the fittest individuals for reproduction. (Parents)
3. Breed new individuals through crossover and mutation operations to give birth to offspring.

4. Replace the least-fit individuals of the population with new individuals.

Types

Similar techniques differ in genetic representation and other implementation details, and the nature of the particular applied problem.

- Genetic algorithm – This is the most popular type of EA. One seeks the solution of a problem in the form of strings of numbers (traditionally binary, although the best representations are usually those that reflect something about the problem being solved),[2] by applying operators such as recombination and mutation (sometimes one, sometimes both). This type of EA is often used in optimization problems.
- Genetic programming – Here the solutions are in the form of computer programs, and their fitness is determined by their ability to solve a computational problem. There are many variants of Genetic Programming, including Cartesian genetic programming, Gene expression programming, Grammatical Evolution, Linear genetic programming, Multi expression programming etc.
- Evolutionary programming – Similar to genetic programming, but the structure of the program is fixed and its numerical parameters are allowed to evolve.
- Evolution strategy – Works with vectors of real numbers as representations of solutions, and typically uses self-adaptive mutation rates.
- Differential evolution – Based on vector differences and is therefore primarily suited for numerical optimization problems.
- Neuroevolution – Similar to genetic programming but the genomes represent artificial neural networks by describing structure and connection weights. The genome encoding can be direct or indirect.
- Learning classifier system – Here the solution is a set of classifiers (rules or conditions). A Michigan-LCS evolves at the level of individual classifiers whereas a Pittsburgh-LCS uses populations of classifier-sets. Initially, classifiers were only binary, but now include real, neural net, or S-expression types. Fitness is typically determined with either a strength or accuracy based reinforcement learning or supervised learning approach.

Comparison to biological processes

A possible limitation of many evolutionary algorithms is their lack of a clear genotype–phenotype distinction. In nature, the fertilized egg cell undergoes a complex process known as embryogenesis to become a mature phenotype. This indirect encoding is believed to make the genetic search more robust (i.e. reduce the probability of fatal mutations), and also may improve the evolvability of the organism.[3][4] Such indirect (also known as generative or developmental) encodings also enable evolution to exploit the regularity in the environment.[5] Recent work in the field of artificial embryogeny, or artificial developmental systems, seeks to address these concerns. And gene expression programming successfully explores a genotype–phenotype system, where the genotype consists of linear multigenic chromosomes of fixed length and the phenotype consists of multiple expression trees or computer programs of different sizes and shapes.[6]

Related techniques

Swarm algorithms include:

- Ant colony optimization is based on the ideas of ant foraging by pheromone communication to form paths.[7] Primarily suited for combinatorial optimization and graph problems.
- The runner-root algorithm (RRA) is inspired by the function of runners and roots of plants in nature.[8]
- Artificial bee colony algorithm is based on the honey bee foraging behaviour. Primarily proposed for numerical optimization and extended to solve combinatorial, constrained and multi-objective optimization problems.
- Bees algorithm is based on the foraging behaviour of honey bees. It has been applied in many applications such as routing and scheduling.
- Cuckoo search is inspired by the brooding parasitism of the cuckoo species. It also uses Lévy flights, and thus it suits for global optimization problems.
- Particle swarm optimization is based on the ideas of animal flocking behaviour.[7] Also primarily suited for numerical optimization problems.

Other population-based metaheuristic methods

- Hunting Search – A method inspired by the group hunting of some animals such as wolves that organize their position to surround the prey, each of them relative to the position of the others and especially that of their leader. It is a continuous optimization method[9] adapted as a combinatorial optimization method.[10]
- Adaptive dimensional search – Unlike nature-inspired metaheuristic techniques, an adaptive dimensional search algorithm does not implement any metaphor as an underlying principle. Rather it uses a simple performance-oriented method, based on the update of the search dimensionality ratio (SDR) parameter at each iteration.[11]
- Firefly algorithm is inspired by the behavior of fireflies, attracting each other by flashing light. This is especially useful for multimodal optimization.
- Harmony search – Based on the ideas of musicians' behavior in searching for better harmonies. This algorithm is suitable for combinatorial optimization as well as parameter optimization.
- Gaussian adaptation – Based on information theory. Used for maximization of manufacturing yield, mean fitness or average information. See for instance Entropy in thermodynamics and information theory.
- Memetic algorithm – A hybrid method, inspired by Richard Dawkins's notion of a meme, it commonly takes the form of a population-based algorithm coupled with individual learning procedures capable of performing local refinements. Emphasizes the exploitation of problem-specific knowledge, and tries to orchestrate local and global search in a synergistic way.

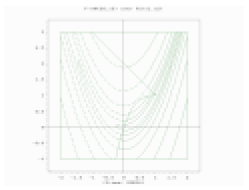
Examples

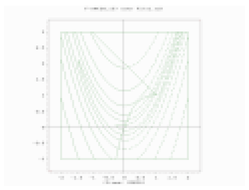
In 2020, Google stated that their AutoML-Zero can successfully rediscover classic algorithms such as the concept of neural networks.[12]

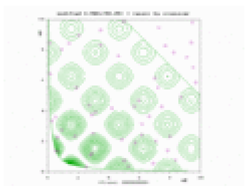
The computer simulations *Tierra* and *Avida* attempt to model macroevolutionary dynamics.

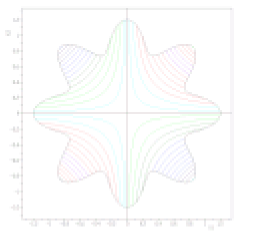
Gallery

[13] [14] [15]


A two-population EA search over a constrained Rosenbrock function with bounded global optimum.


A two-population EA search over a constrained Rosenbrock function. Global optimum is not bounded.


Estimation of distribution algorithm over Keane's function


A two-population EA search of a bounded optima of Simionescu's function.

References

1. Vikhar, P. A. (2016). "Evolutionary algorithms: A critical review and its future prospects". *Proceedings of the 2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*. Jalgaon: 261–265. doi:10.1109/ICGTSPICC.2016.7955308 (https://doi.org/10.1109%2FICGTSPICC.2016.7955308). ISBN 978-1-5090-0467-6. S2CID 22100336 (https://api.semanticscholar.org/CorpusID:22100336).
2. Cohoon, J; et al. (2002-11-26). *Evolutionary algorithms for the physical design of VLSI circuits* (https://www.ifte.de/mitarbeiter/lienig/cohoon.pdf) (PDF). *Advances in Evolutionary Computing: Theory and Applications*. Springer, pp. 683-712, 2003. ISBN 978-3-540-43330-9.
3. G.S. Hornby and J.B. Pollack. "Creating high-level components with a generative representation for body-brain evolution". *Artificial Life*, 8(3):223–246, 2002.
4. Jeff Clune, Benjamin Beckmann, Charles Ofria, and Robert Pennock. "Evolving Coordinated Quadruped Gaits with the HyperNEAT Generative Encoding" (http://www.ofria.com/pubs/2009CluneEtAl.pdf) Archived (https://web.archive.org/web/20160603205252/http://www.ofria.com/pubs/2009CluneEtAl.pdf) 2016-06-03 at the Wayback Machine. *Proceedings of the IEEE Congress on Evolutionary Computing Special Section on Evolutionary Robotics*, 2009. Trondheim, Norway.
5. J. Clune, C. Ofria, and R. T. Pennock, "How a generative encoding fares as problem-regularity decreases" (http://jeffclune.com/publications/Clune-HyperNEATandRegularity.pdf), in *PPSN* (G. Rudolph, T. Jansen, S. M. Lucas, C. Poloni, and N. Beume, eds.), vol. 5199 of *Lecture Notes in Computer Science*, pp. 358–367, Springer, 2008.
6. Ferreira, C., 2001. "Gene Expression Programming: A New Adaptive Algorithm for Solving Problems" (http://www.gene-expression-programming.com/webpapers/GEP.pdf). *Complex Systems*, Vol. 13, issue 2: 87–129.
7. Slowik, Adam; Kwasnicka, Halina (2018). "Nature Inspired Methods and Their Industry Applications—Swarm Intelligence Algorithms". *IEEE Transactions on Industrial Informatics*. Institute of Electrical and Electronics Engineers (IEEE). **14** (3): 1004–1015. doi:10.1109/tii.2017.2786782 (https://doi.org/10.1109%2Ftii.2017.2786782). ISSN 1551-3203 (https://www.worldcat.org/issn/1551-3203). S2CID 3707290 (https://api.semanticscholar.org/CorpusID:3707290).

8. F. Merrikh-Bayat, "The runner-root algorithm: A metaheuristic for solving unimodal and multimodal optimization problems inspired by runners and roots of plants in nature", *Applied Soft Computing*, Vol. 33, pp. 292–303, 2015
9. Oftadeh, R.; Mahjoob, M.J.; Shariatpanahi, M. (October 2010). "A novel meta-heuristic optimization algorithm inspired by group hunting of animals: Hunting search" (https://doi.org/10.1016%2Fj.camwa.2010.07.049). *Computers & Mathematics with Applications*. **60** (7): 2087–2098. doi:10.1016/j.camwa.2010.07.049 (https://doi.org/10.1016%2Fj.camwa.2010.07.049).
10. Amine Agharghor; Mohammed Essaid Riffi (2017). "First Adaptation of Hunting Search Algorithm for the Quadratic Assignment Problem". *Europe and MENA Cooperation Advances in Information and Communication Technologies*. Advances in Intelligent Systems and Computing. **520**: 263–267. doi:10.1007/978-3-319-46568-5_27 (https://doi.org/10.1007%2F978-3-319-46568-5_27). ISBN 978-3-319-46567-8.
11. Hasançebi, O., Kazemzadeh Azad, S. (2015), "Adaptive Dimensional Search: A New Metaheuristic Algorithm for Discrete Truss Sizing Optimization", *Computers and Structures*, 154, 1–16.
12. Gent, Edd (13 April 2020). "Artificial intelligence is evolving all by itself" (https://www.science.org/content/article/artificial-intelligence-evolving-all-itself). *Science | AAAS*. Archived (https://web.archive.org/web/20200416222954/https://www.sciencemag.org/news/2020/04/artificial-intelligence-evolving-all-itself) from the original on 16 April 2020. Retrieved 16 April 2020.
13. Simionescu, P.A.; Beale, D.G.; Dozier, G.V. (2004). "Constrained optimization problem solving using estimation of distribution algorithms" (http://faculty.tamucc.edu/psimionescu/PDFs/WCCI2004-Paper1361.pdf) (PDF). Proc. of the 2004 Congress on Evolutionary Computation - CEC2004. Portland, OR: 1647–1653. doi:10.1109/CEC.2006.1688506 (https://doi.org/10.1109%2FCEC.2006.1688506). S2CID 1717817 (https://api.semanticscholar.org/CorpusID:1717817). Retrieved 7 January 2017.
14. Simionescu, P.A.; Dozier, G.V.; Wainwright, R.L. (2006). "A Two-Population Evolutionary Algorithm for Constrained Optimization Problems" (http://faculty.tamucc.edu/psimionescu/PDFs/WCCI2006-Paper7204(1).pdf) (PDF). *2006 IEEE International Conference on Evolutionary Computation*. Proc 2006 IEEE International Conference on Evolutionary Computation. Vancouver, Canada. pp. 1647–1653. doi:10.1109/CEC.2006.1688506 (https://doi.org/10.1109%2FCEC.2006.1688506). ISBN 0-7803-9487-9. S2CID 1717817 (https://api.semanticscholar.org/CorpusID:1717817). Retrieved 7 January 2017.
15. Simionescu, P.A. (2014). *Computer Aided Graphing and Simulation Tools for AutoCAD Users* (1st ed.). Boca Raton, FL: CRC Press. ISBN 978-1-4822-5290-3.

External links

- An Overview of the History and Flavors of Evolutionary Algorithms (https://www.staracle.com/general/evolutionaryAlgorithms.php)

Bibliography

- Ashlock, D. (2006), *Evolutionary Computation for Modeling and Optimization*, Springer, ISBN 0-387-22196-4.
- Bäck, T. (1996), *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms (https://books.google.com/books?id=htJHI1UrL7IC&printsec=frontcover#v=onepage&q&f=false)*, Oxford Univ. Press.
- Bäck, T., Fogel, D., Michalewicz, Z. (1997), *Handbook of Evolutionary Computation*, Oxford Univ. Press.

- Banzhaf, W., Nordin, P., Keller, R., Francone, F. (1998), *Genetic Programming - An Introduction*, Morgan Kaufmann, San Francisco
- Eiben, A.E., Smith, J.E. (2003), *Introduction to Evolutionary Computing*, Springer.
- Holland, J. H. (1992), *Adaptation in Natural and Artificial Systems (https://books.google.com/books?id=5EgGaBkwvWcC&printsec=frontcover#v=onepage&q&f=false)*, The University of Michigan Press, Ann Arbor
- Michalewicz Z., Fogel D.B. (2004). How To Solve It: Modern Heuristics, Springer.
- Benko, Attila; Dosa, Gyorgy; Tuza, Zsolt (2010). "Bin Packing/Covering with Delivery, solved with the evolution of algorithms". *2010 IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA)*. pp. 298–302. doi:10.1109/BICTA.2010.5645312 (https://doi.org/10.1109%2FBICTA.2010.5645312). ISBN 978-1-4244-6437-1. S2CID 16875144 (https://api.semanticscholar.org/CorpusID:16875144).
- Poli, R.; Langdon, W. B.; McPhee, N. F. (2008). *A Field Guide to Genetic Programming* (https://web.archive.org/web/20160527142933/http://cswww.essex.ac.uk/staff/rpoli/gp-field-guide/). Lulu.com, freely available from the internet. ISBN 978-1-4092-0073-4. Archived from the original (http://cswww.essex.ac.uk/staff/rpoli/gp-field-guide/) on 2016-05-27. Retrieved 2011-03-05.
- Price, K., Storn, R.M., Lampinen, J.A., (2005). "Differential Evolution: A Practical Approach to Global Optimization" (https://books.google.com/books?id=hakXI-dEhTkC&printsec=frontcover#v=onepage&q&f=false), Springer.
- Ingo Rechenberg (1971): Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution (PhD thesis). Reprinted by Fromman-Holzboog (1973).
- Hans-Paul Schwefel (1974): Numerische Optimierung von Computer-Modellen (PhD thesis). Reprinted by Birkhäuser (1977).
- Simon, D. (2013): Evolutionary Optimization Algorithms (http://academic.csuohio.edu/simond/EvolutionaryOptimization), Wiley.
- *Computational Intelligence: A Methodological Introduction (https://books.google.com/books?id=yQVGAAAAQBAJ&printsec=frontcover#v=onepage&q&f=false)* by Kruse, Borgelt, Klawonn, Moewes, Steinbrecher, Held, 2013, Springer, ISBN 978-1-4471-5012-1
- Rahman, Rosshairy Abd.; Kendall, Graham; Ramli, Razamin; Jamari, Zainoddin; Ku-Mahamud, Ku Ruhana (2017). "Shrimp Feed Formulation via Evolutionary Algorithm with Power Heuristics for Handling Constraints" (https://doi.org/10.1155%2F2017%2F7053710). *Complexity*. **2017**: 1–12. doi:10.1155/2017/7053710 (https://doi.org/10.1155%2F2017%2F7053710).