Debdutta Burman *et al*, International Journal of Computer Science and Mobile Applications,
Vol.10 Issue. 9, September- 2022, pg. 20-25 **ISSN: 2321-8363**
Impact Factor: 6.308
(An Open Accessible, Fully Refereed and Peer Reviewed Journal)

Analysing and Integrating Evolutionary Algorithms towards Optimum Solution

Debdutta Burman[1]; Sandip Sarkar[2]; Anirban Bhar[3]; Sujata Kundu[4]
[1,2]B. Tech student, Department of Information Technology, Narula Institute of Technology, Kolkata, India
[3,4]Assistant Professor, Department of Information Technology, Narula Institute of Technology, Kolkata, India

[1] debdutta0401@gmail.com
[2] ss7405150@gmail.com
[3] anirban.bhar@nit.ac.in
[4] sujata.kundu@nit.ac.in

DOI: 10.5281/zenodo.7088929

Abstract

In the theory of evolutionary algorithms (EAs), computational time complexity is an essential problem. This study reports on the average time complexity of EAs. Researchers have been studying the computational time complexity of evolutionary algorithms since the mid-1990s (EAs). The earliest findings were based on toy problems using extremely simple algorithms like the (1+1)-EA. This work resulted in a better knowledge of how EAs perform on various types of fitness landscapes, as well as generic mathematical methods that could be used to analyse increasingly complex EAs on more realistic challenges. In reality, it has been able to study the (1+1)-EA on combinatorial optimization problems with real-world applications, as well as more realistic population-based EAs on structured toy problems in recent years. The results of these two study lines have been surveyed in this report. The most prevalent mathematical procedures are presented, as well as the essential ideas that underpin them and their various applications.

Keywords: Evolutionary Algorithms (EAs), Computational Time Complexity, Combinatorial Optimization Problems.

1. Introduction

There is currently a lot of demand on researchers working in the field of soft computing to find novel optimization methods inspired by nature. Evolutionary computation (EC) algorithms are the modern name for the entire family of evolutionary optimization techniques. The genetic algorithm (GA) [1], genetic programming (GP) [2], differential evolution (DE) [3], evolution strategy (ES) [4], and evolutionary programming (EP) [5] are the main algorithms that fall within the category of evolutionary computation. Each of these methods comes in a wide range of variations and has a wide range of industrial uses.

Evolutionary algorithms (EAs) generate a population of potential solutions over a number of generations using the concepts of fitness and natural selection. Numerous optimization issues have successfully used these techniques in their solution. For instance, the authors of [6] optimised air traffic control procedures using an evolutionary algorithm. In [7], the authors developed control plans for robots with intricate locomotor systems by combining two evolutionary methods. Open-loop and closed-loop solutions are the two main types of solutions that can be created.

Early studies focused more on describing the behaviour of EAs than they did on evaluating their effectiveness. The most common instrument in these early attempts is undoubtedly the schema theory [8]. It was initially suggested to specifically investigate the behaviour of the straightforward genetic algorithm (sGA) [9]. Even though the schema theory was regarded as essential for comprehending GAs up until the early 1990s, it "cannot

explain the dynamical or limit behaviour of EAs," as Eiben and Rudolph noted when examining these pioneering tools [10].
The first convergence findings about the time-limit behaviour of EAs for optimization issues originally surfaced in the 1990s with the introduction of Markov Chain theory in the analysis of EAs. The deterministic concept of convergence was evidently incorrect because an EA's state transitions are probabilistic in nature. Therefore, stochastic convergence definitions had to be used [11]. No of how it is initialised, the ideal EA should be able to solve the issue at hand with probability 1 after a finite number of steps. The method is claimed to visit the global optimum in this situation in a finite amount of time. The method is said to have converged to the optimum if it keeps the solution in the population indefinitely. Rudolph demonstrated using Markov Chains that elitist variants converge to the global optimum but standard GAs using mutation, crossover, and proportional selection do not [12]. Following that, he expanded his research by outlining broad requirements that, when met by an EA, ensure its convergence [13]. The fact that "building a quantitatively accurate Markov model for each variant of an evolutionary algorithm in order to examine the limit behaviour" was not required for the work's motivation. Also provided were the conditions for non-convergence. Therefore, a particular analysis is only required when an EA does not meet the above conditions. Another novelty was that "simple proofs that no longer require Markov chain theory" were finally produced as a result of the intensive study on the convergence of EAs.
The investigation of the time limit behaviour, even if an algorithm does converge, does not, however, provide any specific information about the estimated time for the solution to be found. Aytug and Kohler conducted an investigation of the number of generations required, independent of the optimization function, to ensure convergence under a fixed probability with a fixed level of confidence [14]. Greenhalgh and Marshall [15] have enhanced the results that were achieved. The analysis's best upper bound, however, is equivalent to the one provided by a random algorithm (RA) that selects random candidates independently for each generation [16].
The inability of these methods to produce useful bounds supported the notion that the function to be optimised must be taken into account when assessing the time complexity of problem-independent search heuristics.

2. Evolutionary Algorithm Presentation

The presentation of the main EAs can be categorized as: genetic algorithm, genetic programming, differential evolution, evolution strategies, and evolutionary programming.

2.1 Generic Algorithm

One of the earliest and best-known natural-based optimization algorithms is the genetic algorithm (GA). The Darwinian theory of species evolution is taken into account in the GA, where the search for solution space mimics a natural process that occurs in the environment. In GAs, the population of individuals, or chromosomes, each stand for a potential answer to the issue. The objective function identifies the issue being resolved.

2.2 Generic Programming

Genetic programming (GP) [2] is a relatively new technique that uses modified genetic operators to work on very particular kinds of solution. In an effort to discover a method for the automatic production of programme codes when the assessment criteria for their appropriate operation are known, Koza [2] created the GP. The evolved possible solutions are coded as trees rather than the linear chromosomes (of bits or numbers) common in GAs since the searched answer is a programme. The core loop of GP is identical to the algorithm because GP does not employ the same coding structure as GA.

2.3 Differential Evolution

An evolutionary approach known as differential evolution (DE) is particularly helpful for function optimization in continuous search spaces. The primary form of the DE algorithm was discussed by Storn and Price, despite the fact that a variant of the DE algorithm for combinatorial issues has also been discussed. The key benefits of

DE over a conventional GA are that it is simple to use, efficiently utilises memory, has a reduced computational effort, and has a lower computational complexity (which allows it to scale better when tackling huge issues) (faster convergence).

2.4 Evolution Strategies
When compared to GAs, the evolution strategies (ESs) differ primarily in the selection process. By selecting individuals based on their fitness value and maintaining a steady population size, the GA produces the offspring from the parental population.

2.5 Evolutionary Programming
Evolutionary programming (EP) was created as a method for figuring out a language's grammar. However, EP gained more traction after being suggested as a method for numerical optimization.

3. Cooperative Evolutionary Algorithm
In order to address the large-scale complicated optimization issues (Rentsen, Zhou, & Teo, 2016), Potter and de Jong (1994) first proposed the cooperative co-evolutionary approach. They used a divide-and-conquer approach and evolved the interacting co-adapted sub-problems. In many real-world optimization problems, such as function optimization (Potter & de Jong, 1994), designing artificial neural networks (Potter & de Jong, 1995), the occurrence of Red Queen dynamics (Pagie & Hogeweg, 2000), and machine learning applications (Juillé & Pollack, 1996), the cooperative coevolution exhibits promising performance.

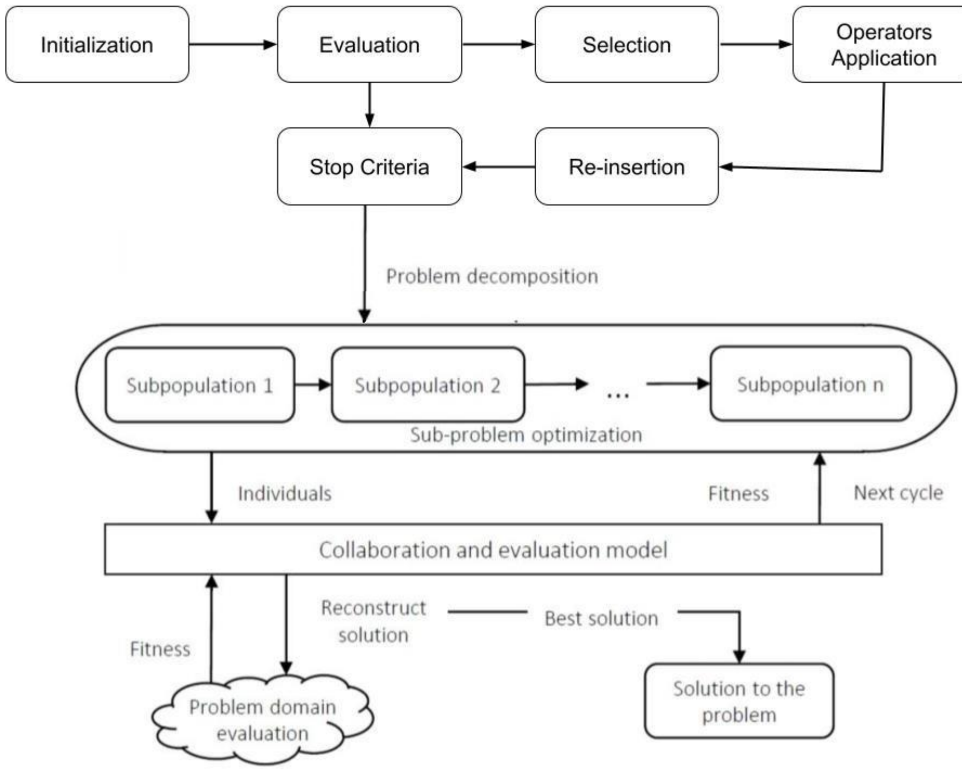

Figure 1: General architecture of cooperative evolutionary algorithm

4. Analysis of Evolutionary Algorithm's Computational Complexity

Numerous combinatorial optimisation issues have been solved using evolutionary algorithms (also known as EAs). Few theoretical analyses of the typical computing time or time complexity of EAs for combinatorial optimisation issues do, however, exist.

Early in the 1990s, a (1+1) EA on a toy problem called the ONE-MAX issue was the only EA that was researched as part of an investigation of the time complexity of EAs. Researchers in theoretical computer science and evolutionary computation have recently become more interested in this topic. The first efforts toward comprehending the time complexity of EAs have been taken. However, the study is only the beginning, and the field still has a lot of unanswered concerns.

4.1 Mathematical models and analysis tools of EAs

Most EAs are analysed using the Markov chain model, which is a popular approach. If an EA's Markov chain is homogeneous, a linear system will be satisfied by the predicted computation times (mean first hitting times) to the optimal set. It is possible to find an explicit solution to the linear equations for particular circumstances, such as when the probability transition matrix of the Markov chain is a tridiagonal matrix or a lower triangular matrix, but only for some straightforward evolutionary algorithms and problems.

The aforementioned linear system is typically too complex to have an explicit solution. Estimating the limits of the typical calculation time is crucial.

An alternative model for analysing EAs is the Martingale model, which was used to examine the convergence of non-elitism EAs. It is a good model for assessing the temporal complexity of EAs even if it is less popular than the Markov chain model. The drift analysis has been set up with superior drift circumstances based on this super martingale.

Building a consistent mathematical model of EAs to analyse the typical computation time is the key issue at hand.

4.2 Fitness landscape classification based on the average computation time of EAs

Both the theory and practise of evolutionary computation revolve around the classification of fitness landscapes. Despite numerous attempts to describe fitness landscapes, such as deception, multimodality, fitness distance correlation, and epistasis variance, none of them are especially good at describing simple or complex issues. An initial landscape classification can be offered based on the drift analysis and the average computation time of EAs. It conceptually distinguishes between simple and complex landscapes.

Studying the features of simple and difficult landscapes using the average calculation time is the major intriguing subject.

4.3 Impact of population on the average computation of EAs

Most assessments of the time complexity of EAs undertaken now have only included (1+1) EAs. There aren't many theoretical findings regarding the typical computation time of population-based EAs. However, a population size greater than one is used in the vast majority of EA applications. When population is used, EAs become more reliable and effective. Theoretically contrasting (1+1) EAs with population-based EAs is crucial.

The major issue is determining which scenario the population will shorten the average computation time by comparing the average computation times of (1+1) EAs and (N+N) EAs.

4.4 Impact of genetic operators on the average computation times of EAs

Many EAs incorporate crossover, mutation, and selection as enhanced search operators, but little theoretical attention is given to how these factors may affect the average computation time of EAs. There are several issues here that are worth researching.

The average computation times of EAs should be compared utilising various mutations because there are numerous alternative mutation operators.

It was necessary to determine which types of problems the crossover would speed up the average calculation time for by comparing the average computation times of EAs with and without crossover.
Additionally, it was required to analyse the various selection methods applied in the EAs and demonstrate how they affect the average calculation time of the EAs.

4.5 Analysis of time complexity of EAs for P-class problem

A problem-specific algorithm can easily solve a P-class problem, but it is still unclear if an EA can do it more quickly than the competitor approach. On these P-class problems, it is necessary to analyse the time complexity of the EAs.
One of the more challenging simple combinatorial optimization issues is maximum matching. Therefore, testing the effectiveness of EAs on simple issues is a smart idea. In order to find the precise maximum matching for a family of bipartite graphs, an EA is shown to require at least exponential average time. The EA can, however, locate a close match for any graph in polynomial time.

4.6 Analysis of time complexity of EAs for NP-class problem

Although there are a few experimental studies showing that EAs perform well on a few challenging situations, it will be exceedingly challenging to properly verify these results. Theoretically, it was required to analyse the time complexity of EAs and determine whether they are capable of effectively resolving any NP-class issues.

5. Conclusion

In conclusion, it is clear that, when employed correctly, evolutionary algorithms can produce close to ideal solutions to a wide range of path planning issues. The EA can create a solution that enables the agent to navigate to the intended goal in a close to optimal fashion for a variety of issues, from straightforward ones like the open-loop turn circle intercept problem to more complicated ones like closed-loop single pursuer single evader. We were able to describe solutions in a form that was straightforward to comprehend, implement, and change by using unique controller representations. We were able to confirm EA and demonstrate that it is in fact convergent on results that are close to ideal by comparing the evolved solutions to known optimal solutions. The evolutionary technique is strengthened by this validation, which can also be used as justification to utilise it on more challenging issues for which the ideal solution is unknown.
Future research will try to create solutions to more complicated issues that haven't yet been analytically resolved using the evolutionary framework that is presented in this thesis. This includes situations where there are several targets or multiple attackers who must work together to accomplish their objective. We also intend to start investigating coevolutionary scenarios, where both the attacker and the target can develop new strategies over time. We also want to try and improve our own controller representations.

References

[1] Holland JH (1975) Adaptation in natural and artificial systems. MIT Press, Cambridge.
[2] Koza J (1992) Genetic programming: on the programming of computers by means of natural selection. MIT Press, Cambridge.
[3] Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11:341–359.
[4] Rechenberg I (1973) Evolutionstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution. Frommann-Holzboog Verlag, Stuttgart.
[5] Fogel LJ, Owens AJ, Walsh MJ (1966) Artificial intelligence thorough simulated evolution. Wiley, New York.
[6] S. Oussedik, D. Delahaye, and M. Schoenauer, "Dynamic air traffic planning by generic algorithms," Proceedings of the Congress on Evolutionary Computation, vol. 2, pp. 1110–1118, 1999.

[7] M. Duarte, J. Gomes, S. Oliveira, and A. Christensen, "Evolution of repertoire-based control for robots with comple locomotor systems," Transactions on Evolutionary Computation, accepted for publication in future issue of this journal.
[8] D. E. Goldberg. Genetic Algorithms for Search, Optimization, and Machine Learning, Addison-Wesley, USA, 1989.
[9] J. H. Holland. Adaptation in Natural and Artificial Systems, 2nd ed., MIT Press, Cambridge, MA, 1992.
[10] A. E. Eiben, G. Rudolph. Theory of Evolutionary Algorithms: A Bird0s Eye View. Theoretical Computer Science, vol. 229, no.1{2, pp. 3{9, 1999.
[11] T. BÄack, D. B. Fogel, Z. Michalewicz. Handbook of Evolutionary Computation, IOP Publishing Ltd, Bristol, UK, 1997.
[12] G. Rudolph. Convergence Analysis of Canonical Genetic Algorithms. IEEE Transactions on Neural Networks, vol.5, no.1, pp. 96{101, 1994.
[13] G. Rudolph. Finite Markov Chain Results in Evolutionary Computation: A Tour d0Horizon. Fundamenta Informaticae, vol. 35, no.1{4, pp. 67{89, 1998.
[14] H. Aytug, G. J. Koehler. Stopping Criteria for Finite Length Genetic Algorithms. ORSA Journal on Computing, vol. 8, no.2, pp. 183{191, 1996.
[15] D. Greenhalgh, S. Marshall. Convergence Criteria for Genetic Algorithms. SIAM Journal on Computing, vol. 30, no.1, pp. 269{282, 2000.
[16] M. Safe, J. A. Carballido, I. Ponzoni, N. B. Brignole. On Stopping Criteria for Genetic Algorithms, In 17th Brazilian Symposium on Arti¯cial Intelligence, Lecture Notes in Artificial Intelligence, Springe-Verlag, vol. 3171, pp. 405{413, 2004.