ANALYSIS OF PROBLEMS AND CHALLENGES OF EVOLUTIONARY ALGORITHMS IN DATA MINING

T.Kanimozhi[1], Assistant professor, Faculty of Science and Humanities, Department of Computer Science, SRM Institute of Science and Technology, Ramapuram, Chennai – 600089.

R.Priya[2], Assistant professor, Faculty of Science and Humanities, Department of Computer Science, SRM Institute of Science and Technology, Ramapuram, Chennai – 600089.

Abstract

The major aim of this work deals with the study about evolutionary algorithms and their applications in real-world scenario. Several categories of evolutionary algorithms such as genetic algorithms, genetic programming, differential evolution and evolution strategies are reviewed. The pseudo-code form of each of these techniques is also analysed. The pseudo-code form can be converted into any programming language for easy implementation. The working principles of these algorithms are summarized.

Keywords: Genetic algorithm – Genetic programming – Differential evolution – Evolution strategy

1. Introduction

Genetics and natural selection have influenced evolutionary algorithms, which are randomised search processes. Several data mining applications use evolutionary algorithms as optimization algorithms. Evolutionary algorithms operate on a population of individuals which present potential solutions for chromosomal problems. Every individual can be just a series of zeros and ones or as complicated as a computer program. Individuals in the initial population can be produced randomly, or they might be seeded with knowledge of previously solved problems. The algorithm assesses individuals based on how well they address the current problem using an objective function that is distinct for every problem and should be provided by the user. Better performing individuals are chosen to be the parents for the next generation. Evolutionary algorithms generate new individuals by employing simple randomised operators, which are analogous to reproductive mechanism and mutation in living organisms [1]. All new solutions are assessed, and the process of selection and generation of new individuals is continued till a good solution is identified or a predefined time period has passed. Of late, in the research area of soft computing optimization techniques are playing a big part. Figure 1 presents different evolutionary approaches that focus on optimization techniques. The entire set of optimization algorithms using evolution processes are defined as evolutionary computational algorithms. In this field, these are the main algorithms: genetic algorithm [2], genetic programming [3], differential evolution [4], evolution strategy [5], and evolutionary programming [6]. There are different variations in each of these algorithms and they are used in various industrial applications. The study is arranged as follows: In section 2, the important evolutionary algorithms such as genetic algorithm, genetic programming, differential evolution and evolution strategies are discussed in brief and in section 3, different evolution algorithms and their description are reviewed.

Figure 1: Evolutionary algorithm Classifications

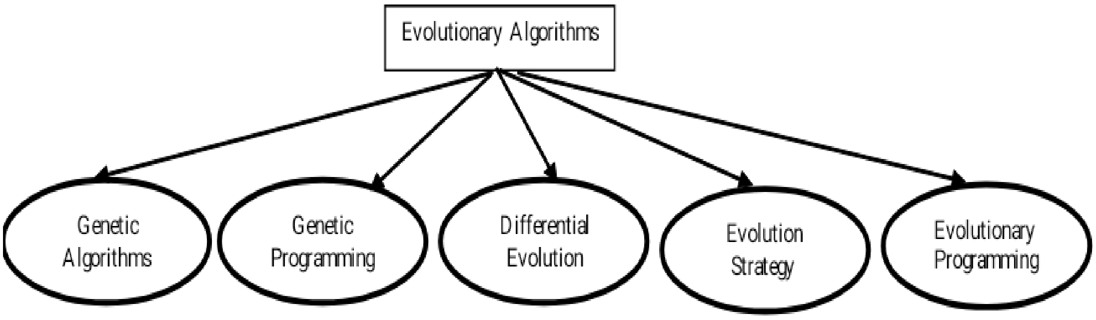
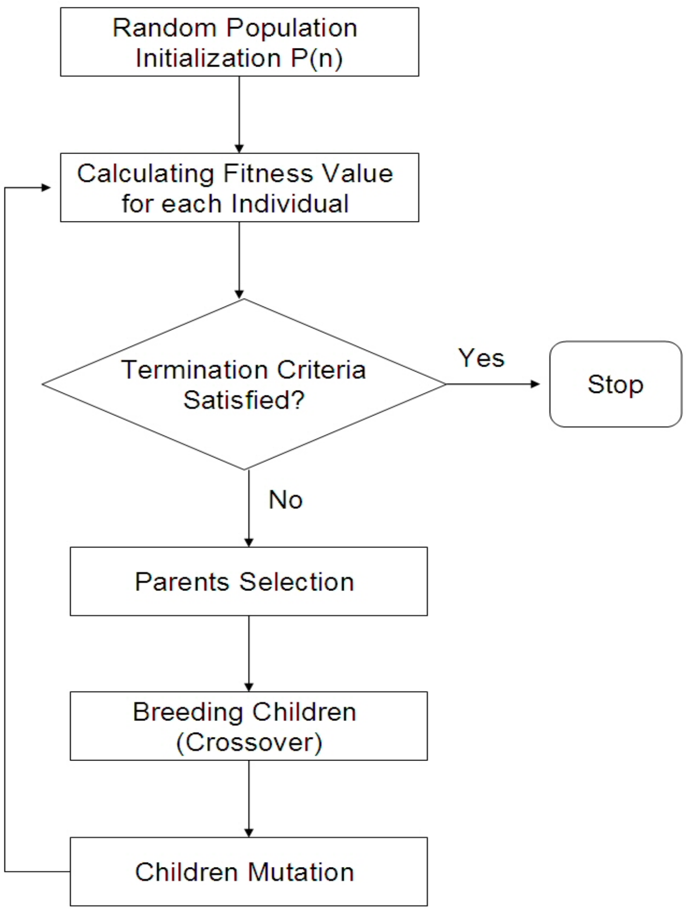


Figure 2: General working process of Evolutionary algorithms



In between upper and lower bounds, an initial random population is formed. Afterwards, at random, choose three more vectors out of each parameter vector. And to the third vector, add the weighted difference of two of the vectors. The target vector and the donor vector are combined to create a trial vector. The target and trial vectors are compared, and, one that has the lowest function value is chosen to go to the next generation. Check to see if the termination criteria has been met. If yes, the best solution is found. If not, mutation is used to create a next generation of the population.

2. Description of Evolutionary Algorithms

2.1 Genetic Algorithms

In genetic algorithms, we begin with a group of individuals termed as a population. Each individual has a set of attributes called genes. All these genes are grouped together to form a chromosome. Each chromosome is actually an expected solution to the problem [10]. To solve the problem, we determine an objective function. We check whether an individual fits in to the objective function or not. This also shows the capacity of an individual to compete with other individuals. This function is called as the fitness function and it actually evaluates an individual based on a fitness value or score. There are three operations in genetic algorithms: Selection, where the fittest individuals are selected and their genes are sent to the new generation. Crossover, where individual groups are swapped among two individuals to crossover in order to produce a superior offspring. Mutation that is used to retain genetic diversity from one generation of a population to the next and prevent premature convergence.

Algorithm 1: Genetic Algorithm

1. define objective function (OFun)
2. allocate number of generation to 0 (g=0)
3. introduce random individuals in initial population Pop(g)
4. evaluate individuals in population Pop(g) using OFun
5. while termination condition is not met do
6. g=g+1
7. pick the individuals to population Pop(g) from Pop(g-1)
8. modify individuals of Pop(g) using crossover and mutation
9. evaluate individuals in population Pop(g) using OFun
10. end while
11. return the fittest selected individual during the evolution process

2.2 Genetic Programming

Genetic programming [3] is an evolutionary process that encompasses genetic algorithms to perform the study of a vast complex computerized processes. As the solution searched is a program, all the possible solutions are programmed as trees instead of linear chromosomes (of bits or numbers) as is uses in genetic algorithms.

2.3 Differential evolution algorithm

In the search space, the differential evolution method generates random initial population. By adding the vector difference between two randomly recognised individuals to a third individual in the population, it creates new individuals. If the new individual's fitness function has a higher value than the previous one, it will take the place of the old one [8]. The CRV parameter reflects the crossover value, whereas parameter F scales the values added to the specific choice variables (mutation) [9]. The general method of differential evolution is presented. Within search space, the differential evolution method generates an initial population at random. By summing the vector difference of two randomly selected individuals and a third individual within the population, it creates new individuals. If the fitness function of the new individual has a higher value, it will take the place of the old one [23]. The F parameter adjusts the values supplied to the specific decision variables (called mutation), while the CRV parameter specifies the crossover value [24].

Algorithm 2: Differential Evolution

define objective function (OFun)
allocate number of generation to 0 (g=0)
create random individuals in initial population Pop(g)
while termination condition is not met do
g=g+1
for each i-th individual in the population Pop(g) do
randomly pick three integer numbers:
$x1, x2, x3 \in [1; \text{population size}]$, where $x1 \neq x2 \neq x3 \neq i$
for each j-th gene in i-th individual ($j \in [1:n]$) do
$a_{i,j} = b_{x1,j} + F . (b_{x2,j} - b_{x3,j})$
randomly choose one real number $random_j \in [0:1]$
if $random_j$ < CRV then $c_{i,j} = a_{i,j}$ else $c_{i,j} = b_{i,j}$ end if
end for
if an individual c_i is stronger than individual b then
substitute child c_i individual for individual b_i
end if
end for
end while
return the fittest selected individual in population Pop(g)

2.4 Evolution strategies

In evolutionary strategy, population is temporarily constructed. The size of this population is different from the original parental population. The assumed parameters are taken to be λ and μ. Here, importance is not given to fitness value. The individuals in the temporary population complete out crossover and mutation. Evolution strategy algorithms work on floating point vectors, whereas the genetic algorithms work on binary vectors. We shall discuss the algorithm for ES(μ + λ), where μ are random individuals created in the initial population and λ individuals are generated by reproduction from the original population [8].

Algorithm 3: ES($\mu + \lambda$) Evolution Strategy

1. define objective function (OFun)
2. allocate number of generation to 0 (g = 0)
3. create μ random individuals in the initial population Pop(g)
4. evaluate individuals in population Pop(g) using OFun
5. while termination condition is not met do
6. g = g + 1
7. create the population S(g) by reproduction λ individuals from population Pop(g-1)
8. create the population T(g) using mutation and crossover of individuals from population S(g)
9. evaluate the individuals in population T(g)
10. choose the fittest μ individuals to population Pop(g) from the population Pop(g-1) and T(g)
11. end while
12. return the fittest selected individual in population Pop(g)

2.5 Evolutionary programming

The evolutionary programming was introduced as an approach for numerical optimization. Evolutionary programming differs from evolutionary strategy in that it creates a new population of individuals by mutating every individual from parent population. In ES($\mu + \lambda$), on the other hand, all the individuals get an equal chance of being chosen for the temporary population upon which genetic operations are performed. The newly generated population and parent populations have equal size in evolutionary programming, i.e., $\mu = \lambda$). Evolutionary programming was proposed as a numerical optimization technique. The difference between evolutionary strategy and evolutionary programming is that, in evolutionary programming, through mutating each individual from the parent population, the new population of individuals is created. Whereas in ES($\mu + \lambda$), all individuals have equal probability of being selected to the temporary population on which the genetic operations are carried out. In evolutionary programming, both the newly created and the parent populations are of the same size, i.e., $\mu = \lambda$). Finally, a ranking selection mechanism is applied to develop a new generation of population, with individuals from parent as well as mutant populations being employed.

Algorithm 4: Evolutionary Programming

1. define objective function (OFun)
2. allocate number of generation to 0 (g = 0)
3. create random individuals in the initial population Pop(g)
4. evaluate individuals in population Pop(g) using OFun
5. while termination condition is not met do
6. g = g + 1
7. create the population T(g) using mutation of each individual from population Pop(g-1)
8. evaluate the individuals in population T(g) using OFun
9. choose the individuals to population Pop(g) from the total of individuals in Pop(g-1) and T(g) using ranking selection method
10. end while
11. return the fittest selected individual in population Pop(g)

3. Comparisons of different evolution algorithms

The significant Evolutionary Strategy parameters are shown in Table 1. These parameters are used to create an algorithm design.

3.1 Table 1: Evolutionary Strategy parameters

Symbol	Parameter	Range
μ	Number of parent individuals	$\mathbb{N}$
$\nu = \lambda/\mu$	Offspring-parent ratio	$\mathbb{R}_+$
$\sigma_i^{(0)}$	Initial standard deviations	$\mathbb{R}_+$
n_σ	Number of standard deviations. N denotes the problem dimension	$\{1, N\}$
τ_0, τ	Multiplier for mutation parameters	$\mathbb{R}_+$
ρ	Mixing number	$\{1, \mu\}$
r_x	Recombination operator for object variables	{intermediary, discrete}
r_σ	Recombination operator for strategy variables	{intermediary, discrete}
κ	Maximum age	$\mathbb{R}_+$

3.2 Table 2: Different evolution algorithms researched and their descriptions

Algorithm, references	Author	Brief description	Year
Genetic Algorithm			
Fluid genetic algorithm [12]	Jafari-Marandi et al.	A modified form of genetic algorithm that is fluid genetic algorithm was introduced	2017
Block-based genetic algorithm [13]	Tseng et al.	An approach to generate feasible Assembly/Disassembly Sequence Planning (ASP/DSP) using genetic algorithm	2018
Tribe competition-based genetic algorithm [14]	Ma et al.	Individual populations are grouped into several tribes	2017
Genetic Programming			
Statistical genetic programming [15]	Haeri et al.	Uses statistical data to produce some well-structured subtrees which are then used to create the initial population	2017
Multi-dimensional genetic programming [16]	La Cava et al.	Analyses a novel program representation with genetic programming for representing multidimensional features	2018
Surrogate genetic programming [17]	Kattan et al.	Employs meta-models to generate one of the two populations after dividing the population into two segments	2015
Differential Evolution			
Stochastic Quasi-Gradient-differential evolution [18]	Sala et. al.	Introduces a hybrid variant of differential evolution that combines Stochastic Quasi-Gradient methods with in differential evolution	2017
Opposition-based Compound Sinusoidal differential evolution [19]	Draa et. al.	To adjust the scaling factor and crossover rate values, a compound sinusoidal formula was used.	2019

Colonial competitive differential evolution [20]	Ghasemi et al.	The Differential evolution algorithm depends upon socio-political evolution for the economic load dispatch problem	2016
Memory-based differential evolution [21]	Parouha et al.	The algorithm employs swarm mutation as well as swarm crossover to get unbounded optimization	2016
Evolution Strategy			
Covariant matrix self-adaptation evolution strategy with repelling subpopulations [22]	Ahrari et. al.	Multiple subpopulations concurrently find the search space	2017
Matrix adaptation evolution strategy [23]	Beyer et. al.	The covariance update and square root functions on the covariance matrix are not required	2017
Weighted recombination evolution strategy [24]	Akimoto et. al.	With general convex quadratic functions, weighted recombination is used to investigate evolution techniques	2018
Evolutionary Programming			
Fast convergence evolutionary programming [25]	Basu	Fast convergence evolutionary programming was created to speed up convergence and enhance solution quality	2017
Immune log-normal evolutionary programming [26]	Mansor et. al.	Proposed to overcome the issue of economic dispatch with forbidden operation zones	2017
Automated generation of mutation operators for evolutionary programming [27]	Hong et. al.	Generates mutation operators for the evolutionary programming system using genetic programming	2018

3.3 Evolutionary algorithms: problems and challenges

Genetic algorithms are computer programs that replicate natural development, and are becoming more widely used in a variety of fields. They are used to deal with challenges ranging from neural network architecture search to strategic games, as well as to explain adaptation and learning processes. Competence on the benefits and limitations of this approach is widely discussed in literature or on the internet, necessitating a unification of such expertise in view of recent advancements in this domain. In this study, a discussion on the features, drawbacks, constraints and future research recommendations of genetic algorithms is presented. Genetic algorithms can explore enormous and complex areas of potential solutions, efficiently discovering items of interest, and providing an effective modelling approach to characterise evolutionary systems, ranging from games to economies. But, they include high computation costs, complicated parameter setup, and critical solution representation. Latest trends like GPU, parallel, and quantum computing, as well as the development of sophisticated parameter control techniques and creative representation schemes, could be important to overcoming these limitations. This compilation review intends to familiarize users and novices in the area about genetic algorithm study, as well as to outline prospective research directions for future work. It emphasises the possibilities for interdisciplinary study combining genetic algorithms with innovative breakthroughs in social sciences, open-ended evolution, artificial life, and artificial intelligence. Newer and better constraint management strategies are required in evolutionary algorithms. Furthermore, greater research into the use of EAs in dynamic optimization problems, optimization in noisy and non-stationary environments, and multi-objective optimization problems (particularly with high number of decision variables) is needed. In addition, more research into population size adaptability in various optimization contexts is required. To handle long term issues more effectively, new approaches should be created. The issue of constraint handling in evolutionary algorithms is one of many topics that is yet to be resolved. Penalty methods, methods evolving in the feasible region, methods using parallel population approaches, methods based on the assumption of feasible individual superiority, methods using multi-objective optimization techniques, and hybrid methods are the six key categories of constraint handling methods. All these categories are further subdivided. To properly apply constraint handling methods in evolutionary algorithms for real-world applications, it is imperative to address numerous queries, like whether the objective function is identified in the unfeasible domain (else, penalization methods can't be employed); whether any optimal active constraints available (else, all methods that depend on searching for feasible region bounds are meaningless); What are the features of constraints? (The techniques for linear constraints are eliminated if any one constraint is nonlinearly uneven.) Furthermore, in real-world use of evolutionary algorithms with constraint handling approaches, decision criterion like complexity and difficulty of implementation usually dominate the method's effectiveness. Because of sheer convenience, penalty methods, feasibility rules, and stochastic ranking algorithms are widely applied in real-world applications [11]. As such, there is no universal solution for dealing with constraints in evolutionary algorithms that can handle real-world problems, hence research on constraint management strategies in evolutionary algorithms for real-world applications is nevertheless a domain of interest.

Conclusion and future trends

In this work, all working principles of evolutionary algorithms such as genetic algorithms, genetic programming, differential evolution and evolutionary strategy are explained. These evolutionary computational algorithmic models apply evolutionary processes to solve complex problems. A brief survey about the study and utilization of these evolutionary algorithms is also reviewed. The discussed algorithms offer better approximation results to problems that would be difficult to solve using other methods. Better results are achieved when these algorithms are used with different data sets in various data mining applications.

Of late, hybridization is becoming an evolving phenomenon in which two or more algorithms are joined to achieve better outcomes. Numerous studies are in the process, with researchers continuously modifying evolutionary algorithms to improve system performance. These improvements are described for genetic algorithms in [12, 14], [15, 28] for genetic programming, [18, 29] for differential evolution, [22, 23] for evolutionary strategies, and [25, 26] for evolutionary programming. By nature, evolutionary algorithms are stochastic. For each run of the evolutionary algorithm, a new result is generated. As a result, the primary goal is to verify that the outcomes created by evolutionary algorithmic procedures are repeatable. Evolutionary algorithms could become a major topic for researchers in the future, and expectations are becoming high for novel research problems to emerge as a result of new evolutionary algorithms.

References:

1. Erick Cantú-Paz, Chandrika Kamath (2001), On the Use of Evolutionary Algorithms. DataMining: A Heuristic Approach, Eds., Idea Group Publishing

2. Holland JH (1975) Adaptation in natural and artificial systems. MIT Press, Cambridge.

3. Koza J (1992) Genetic programming: on the programming of computers by means of natural selection. MIT Press, Cambridge.

4. Storn R, Price K (1997) Differential evolution – a simple and efficient heuristic for globaloptimization over continuous spaces. J Glob Optim 11:341-359

5. Rechenberg I (1973) Evolutionstrategie: optimierungtechnischer systeme nach prinzipiender biologischen evolution. Frommann-Holzboog Verlag, Stuttgart

6. Fogel LJ, Owens AJ, Walsh MJ (1966) Artificial intelligence thorough simulated evolution. Wiley, New York.

7. Rutkowski L (2008) Computational intelligence: methods and techniques. Springer, Berlin.

8. Libiao Zhang, Xiangli u, Chunguang Zhou, Ming Ma, Zhezhou Yu (2011) An improved differential evolution algorithm for optimization problems. Advances in Computer Science,Intelligent System and Environment pp 233-238.

9. Slowik A (2011) Application of adaptive differential evolution algorithm with multiple trial vectors to artificial neural networks training. IEEE Transactions on Industrial Electronics 58(8):3160-3167.

10. Adam Slovik, Halina Kwasnicka (2018) Evolutionary algorithms and their applications to engineering problems. Neural Computing and Applications (2020)32:12363-12379.

11. Petrowski A, Ben-Hamida S (2017) Constrained continuous evolutionary optimization. Evol Algorithms 1:93–133.

12. Jafari-Marandi R, Smith BK (2017) Fluid genetic algorithm (FGA). J Comput Des Eng 4:158–167.

13. Tseng H-E, Chang C-C, Lee S-C, Huang Y-M (2018) A blockbased genetic algorithm for disassembly sequence planning.Expert Syst Appl 96:492–505.

14. Ma B, Xia Y (2017) A tribe competition-based genetic algorithm for feature selection in pattern classification. Appl Soft Comput 58:328–338.

15. Haeri MA, Ebadzadeh MM, Folino G (2017) Statistical genetic programming for symbolic regression. Appl Soft Comput 60:447–469.

16. La Cava W, Silva S, Danai K, Spector L, Vanneschi L, Moore JH (2019 Multidimensional genetic programming for multiclass classification. Swarm Evol Comput 44:260–272.

17. Kattan A, Ong Y (2015) Surrogate genetic programming: a semantic aware evolutionary search. Inf Sci 296:345–359.

18. Sala R, Baldanzini N, Pierini M (2017) SQG-differential evolution for difficult optimization problems under a tight function evaluation budget. In: Proceedings of the international workshop on machine learning, optimization, and big data, pp 322–336.

19. Draa A, Chettah K, Talbi H (2019) A compound sinusoidal differential evolution algorithm for continuous optimization. Swarm and Evol Comput 50:100450.

20. Ghasemi M, Taghizadeh M, Ghavidel S, Abbasian A (2016) Colonial competitive differential evolution: an experimental study for optimal economic load dispatch. Appl Soft Comput 40:342–363.

21. Parouha RP, Das KN (2016) A memory based differential evolution algorithm for unconstrained optimization. Appl Soft Comput 38:501–517.

22. Ahrari A, Deb K, Preuss M (2017) Multimodal optimization by covariance matrix self-adaptation evolution strategy with repelling subpopulations. Evol Comput 25(3):439–471.

23. Beyer HG, Sendhoff B (2017) Simplify your covariance matrix adaptation evolution.

24. Akimoto Y, Auger A, Hansen N (2018) Quality gain analysis of the weighted recombination evolution strategy on general convex quadratic functions. Theor Comput Sci. https://doi.org/10.1016/j.tcs.2018.05.015 strategy. IEEE Trans Evol Comput 21(5):746–759

25. Basu M (2017) Fast convergence evolutionary programming for multi-area economic dispatch. Electr Power Compon Syst 45(15):1629–1637.

26. Mansor MH, Musirin I, Othman MM (2017) Immune log-normal evolutionary programming (ILNEP) for solving economic dispatch problem with prohibited operating zones. In: 4th international conference on industrial engineering and applications, ICIEA, pp 163–167.

27. Hong L, Drake JH, Woodward JR, Ozcan E (2018) A hyperheuristic approach to automated generation of mutation operators for evolutionary programming. Appl Soft Comput 62:162–175

28. Ffrancon R, Schoenauer M (2015) Memetic semantic genetic programming. In: Proceedings of the annual conference on genetic and evolutionary computation, GECCO, pp 1023–1030.

29. Trivedi A, Sanyal K, Verma P, Srinivasan D (2017) A unified differential evolution algorithm for constrained optimization problems. In: Proceedings of the IEEE Congress on evolutionary computation (CEC), pp 1231–1238