Volume 4, No. 4, March-April 2013

ISSN No. 0976-5697

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Coevolution Evolutionary Algorithm: A Survey

Jeniefer Kavetha. M
Department of Computer Science
Pondicherry University Pondicherry
jenikavi.mtech@gmail.com

Abstract: Evolutionary Computing techniques have become one of the most powerful tools for solving optimization problems and is based on the mechanisms of natural selection and genetics. In Evolutionary Algorithm, Co-evolution is a natural choice for learning in problem domains where one agent's behaviour is directly related to the behaviour of other agents. Co-evolution provides a framework to implement search heuristics that are more elaborate than those driving the exploration of the state space in canonical evolutionary systems. This paper presents the concept of Co-evolutionary learning and explains a search procedure which successfully addresses the underlying impediments in Co-evolutionary search. Co-evolution employs evolutionary algorithms to solve a high-dimensional search problem by decomposing it into low-dimensional subcomponents. The objective of this survey is to discuss about the various existing Co-evolutionary algorithm and their successful implementation in real world optimization problem. Hence the outcome of the study is to bring out the various research opportunities in implementing the concept of Co-evolution for many optimization problems in different application

Keywords: Evolutionary Computation, Evolutionary Algorithm, Co-evolution Computation and Co-evolutionary Algorithm.

I. INTRODUCTION

Optimization problem is the problem of finding the *best* solution from all feasible solutions. Nature inspired algorithms are meta-heuristics that imitate the nature for solving optimization problems. Nature is the perfect example for optimization because each and every feature or phenomenon in nature always finds the optimal strategy, like balancing the ecosystem, maintaining diversity, adaptation in migrating environment etc [1]. Evolutionary computation is a process which gradually evolves a population of solutions, in a manner resembling the Darwin's Theory of survival of the fittest. Each solution is evaluated individually for finding its fitness. Based on the fitness, a selection strategy decides which solutions should survive and move to the next generation.

Evolutionary algorithms (EAs) have been applied to a variety of problems, from static optimization to job-shop scheduling. EAs frequently have an advantage over many traditional local search heuristic methods where search spaces are highly modal, discontinuous, or highly constrained. In 1960s, EAs were the most well known, classical and established algorithms among nature inspired algorithms [2]. EAs employ a powerful design philosophy to find solutions to hard problems. EAs are non-deterministic algorithms or cost based optimization algorithms. Therefore evolutionary algorithms (EAs) are often used as an optimization tool for finding best optimal solution.

Since 1990, Co-evolutionary interactions have been added to the standard evolutionary algorithm. Initially introduced as a method of improving the ability of EAs to optimize, Co-evolutionary algorithms also facilitate the evolutionary modeling of biological inter-species interactions. Its potential not only as a powerful problem-solving tool capable of outperforming standard evolutionary algorithms, but also (uniquely) as a tool for sophisticated individual-based modeling of Co-evolutionary interactions, Co-evolutionary algorithms are an exciting innovation.

Co-evolutionary algorithms are an extension of evolutionary algorithms. The behaviour of a Co-evolutionary algorithm depends on the interaction between individuals [4]. In this case, the individuals are evaluated not in terms of an assumed and explicit fitness function, but rather by direct interactions between individuals taken from the co-evolving populations. Traditional evolutionary algorithms (EAs) assess the fitness of an individual objectively, that is, independent of the population context in which the individual is placed. Co-evolutionary algorithms (CEAs) operate much like traditional EAs except that fitness assessment is not objective, but subjective: an individual is evaluated through its interaction with other individuals in the evolutionary system [2]. Since fitness is subjective in CEAs, it is not clear under what conditions a CEA would be expected to optimize in a fashion like a traditional EA would in solving a static problem [1]. In this paper, the existing algorithms in Co-evolution are briefly explained.

II. OVERVIEW OF COEVOLUTIONARY COMPUTATION

A. *Co-evolutionary mode:*

The concept of co-evolution mode, which was first proposed by Hillis in 1992, is to improve premature convergence in traditional genetic algorithms via inter-evolution between living creatures and environments, which correspond to chromosomes and multi-criteria, respectively [12]. The objective is constant improvement in chromosome survival. The under-lying concept is that both chromosomes and multi-criteria should evolve. In others words, co-evolution is a scheme where living being (chromosome) and environment (multiple criteria) interact and co-evolve. The genes are improved continuously for survival and the environment changes with the living being. For instance, when an eagle hunts a rabbit, the rabbit must run fast to survive, and the eagle must also fly fast to catch the rabbit [15]. This means that both the chromosome and multiple criteria constraint conditions must evolve. The evaluation

criteria for next generation are selected and based on the degree of fitness of criteria. In this way, search speed is faster than traditional evolutionary algorithm. Thus, the near optimum solution can be acquired rapidly and convergent immaturity can be avoided.

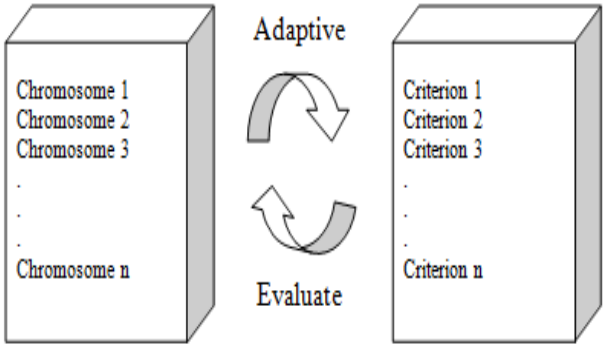


Figure 1: Co-evolutionary Model

B. Coevolutionary algorithms:

A Co-evolutionary algorithm (CoEA) is an Evolutionary Algorithm that is able to manage two or more populations simultaneously. Co-evolution can be defined as the co-existence of some interacting populations, evolving simultaneously. In this manner, evolutionary biologist Price defined "Co-evolution as reciprocally induced evolutionary change between two or more species or populations" [4]. It works by managing two or more populations (also called species) simultaneously, allowing interactions among its own individuals. This approach allows splitting the problem into different parts, employing a population to handle each one separately, but joining its own individuals to evaluate the solutions obtained.

C. Evolutionary Algorithm Vs Co-evolutionary Algorithm:

The difference between co-evolutionary algorithm and the general evolutionary algorithm is that the traditional approach only adopts a simplified function to evaluate whether the chromosomes are good or bad. Restated, whether the chromosomes are good or bad depends on the values computed by single fitness function. Thus, their weakness is their lack of capability to obtain the optimal solution area by adaptively adjusting to environmental change. Co-evolution concept is analogous to human thinking patterns because it considers more criteria [22]. Through interactive evaluation between chromosomes and criteria, the evaluation criteria change dynamically and immediately. Thus the solution-seeking speed can be expedited.

In evolutionary algorithm, a population is generated on a random probability basis, then for each chromosome evaluates the fitness using an appropriate fitness function suitable for the problem. Based on this, the best chromosomes are selected into the mating pool; where they undergo genetic operator like cross over and mutation thus produce new set of solution [14]. In Co-evolutionary, a population is randomly generated and each chromosome is evaluated using an appropriate fitness function with corresponding criterion, then best chromosomes are selected and they undergo genetic operator like cross over and mutation thus produce new set of solution. Finally, selection operation is used to select better chromosomes for the evolution population in the next generation. Before the termination condition is reached, the evolution continues through reproduction operation and selection operation. Fig.2 shows the working flow of co-evolutionary algorithm and evolutionary algorithm.

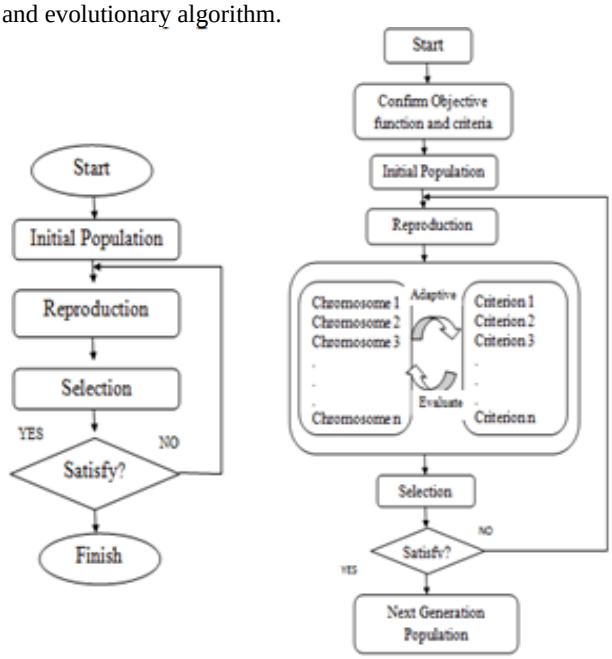
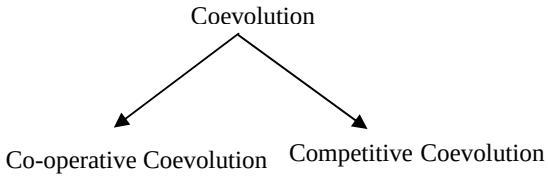


Figure 2: Flow chart of Coevolutionary Algorithm and Evolutionary algorithm

III. RELATED WORK

Indeed, with varied success, nature-inspired heuristic EAs have been applied too many types of difficult problem domains, such as parameter optimization and machine learning. The inspiration for Co-evolutionary algorithms (CoEAs) is the same as for traditional evolutionary algorithms (EAs): attempt to harness the Darwinian notions of heredity and survival of the fittest for problem-solving purposes. In general, Co-evolutionary algorithms can be divided into two ways: cooperative and competitive (or antagonistic) Co-evolution. In co-evolutionary algorithms, one or more populations co-evolve influencing each other, cooperative co-evolution in which the populations work together to accomplish the same task and competitive co-evolution as predators and preys in nature [14].



A. Co-operative Co-evolution:

Co-operative Co-evolutionary algorithms are often used in situations where a problem can be naturally decomposed into sub-components. Individuals represent such sub-components and are assessed in a series of collaborations

with other individuals in order to form complete solutions, for example consider an optimization problem with N parameters (variables), a natural decomposition is chosen to maintain N sub-populations. The fitness of a sub-population is an estimate of how well it "cooperates" with other species to produce good solutions [17]. Cooperative Co-evolutionary algorithms have had success in a variety of domains, for example, manufacturing scheduling, function optimization, designing artificial neural networks and room painting.

B. Competitive Co-evolution:

Competitive Co-evolution either occurs within one population engaged in self-play, or between multiple populations [18]. Competitive co-evolution has several interactional species. In order to obtain the common resources and space, they compete with each other. Single population competitive Co-evolution has been successfully applied to the Iterated Prisoner's Dilemma, pursuit and evasion, and to finding robust game strategies in, for example, Tic-T ac-Toe and backgammon game etc.

C. Cooperative versus Competitive CEAs

Most popularly competitive Co-evolution has been applied to game playing strategies and additionally it demonstrates the effectiveness of competition for evolving better solutions by developing a concept of competitive fitness to provide a more robust training environment than independent fitness functions. Competition played a vital part in attempts to coevolve complex agent behaviors. Finally, competitive approaches have been applied to a variety of machine learning problems.

In this [5], author opened the door for research on cooperative CEAs by developing a relatively general framework for such models and applying it, first, to static function optimization and later to neural net-work learning. In Potter's model, each population contains individuals representing a component of a larger solution, and evolution of these populations occurs almost independently, in tandem with one another, interacting only to obtain fitness.

In this [6], take a different, somewhat more adaptive approach to cooperative co-evolution of neural networks. In this case a parent population represents potential network plans, while an offspring population is used to acquire node information. Plans are evaluated based on how well they solve a problem with their collaborating nodes, and the nodes receive a share of this fitness. Thus a node is rewarded for participating more with successful plans, and thus receives fitness only indirectly.

Potter's methods have also been used or extended by other researchers. Eriksson and Olsson use a cooperative Co-evolutionary algorithm for inventory control optimization. Wiegand attempts to make the algorithm more adaptively allocate resources by allowing migrations of individuals from one population to another in a method similar to the Schlierkamp-Voosen and Muhlenbein competitive mechanisms.

The difference between two types is the individual in the competitive co-evolution represents a completed solution, but the cooperative co-evolution is not. In the case of cooperative co-evolution, an individual in a population works with other individuals to accomplish sane task. However, in the case of competitive co-evolution, individuals in population interact or compete with each other.

IV. CO-EVOLUTION IN VARIOUS APPLICATION

A. Co-operative Coevolution in various application:

The algorithms below are implemented in various applications, which are based on co-operative Co-evolutionary algorithm:

- IFS-CoCo
- CoCo MOPSO
- CoCo Genetic Programming

a. Instance and Feature selection based on the Co-operative Coevolution (IFS- CoCo):

Instance selection and feature selection are two orthogonal methods for reducing the amount and complexity of data. Feature selection aims at the reduction of redundant features in a data set whereas instance selection aims at the reduction of the number of instances. In this, instance and feature selection based on the cooperative Co-evolution (IFS-CoCo), both processes are applied simultaneously to the initial data set, aiming to obtain a suitable training set to perform the classification process. IFS-CoCo is composed of three populations. The individuals of each one defines a different type of baseline classifier, depending on each population's characteristics. Thus, each population is focused on performing a basic data reduction task. The first population performs an IS process, the second population performs a FS process and the third population performs simultaneously both IS and FS processes. With the employment of Co-evolution, this approach is intended to improve the results of data reduction techniques when applied to classification tasks [12].

b. Cooperative Coevolutionary MOPSO (CoCo-MOPSO):

The cooperative co-evolutionary PSO proposed by van den Bergh and Engelbrecht [9], the problem is decomposed in the search space and the decision variables are evolved by different species, or called sub-swarms. The main difference is that the assignment of decision variables to different sub-swarms is adapted by the competitive mechanism.

At the start of the optimization process, sub-swarms are randomly initialized and the ith variable is assigned to the ith sub-swarm. In order to evaluate a particle in a sub-swarm, the particle under evaluation is combined with the representative of every other species to form a complete solution. In this, the best particle in the sub-swarm is defined as the representative of the sub-swarm. The archive is updated after each particle evaluation. The sub-swarms are evaluated in an iterative manner. Before proceeding to the evaluation of the next sub-swarm, the representative of previous sub-swarm will be updated. This updating process is based on a partial order such that Pareto ranks will be considered first followed by niche count in order to break the tie of ranks. For any two particles, the particle with lower rank is selected. In the case of a tie in rank, the particle with lower niche count is selected. The rationale of selecting a non-dominated representative with the lowest niche count is to promote the diversity of the solutions [9].

c. Co-operative Coevolutionary Genetic Programming (CoCo GP):

In Multi-robot path planning, the problem of multi-robot motion planning deals with computation of paths of various

robots such that each robot has an optimal or near optimal path, but the overall path of all the robots combined is optimal [8]. This is a more complex task as compared to a single-robot motion planning, where the factor of coordination among the various robots is not applicable, and the single robot can use its own means to compute the path.

The basic working of the algorithm is a cooperative genetic programming. This planner operates in two levels: master and slave. The slave genetic programming is basically a decentralized path planning for all the various robots in the system. Using this level the system tries to generate better and better paths for the individual motion of the various robots. The second level is the master level. Each robot in the slave has numerous genetic programming individual over which it tries to carry forward the optimization. The master is simply a genetic algorithm that tries to select the best combination of paths for the various robots, such that the overall path of all robots combined is optimal. In simple terms it may be regarded as the slaves optimize the individual robot paths, and the master optimizes the network plan of all robots. However it needs to be noted that the slaves are conscious of cooperation amongst each other to generate collision free paths, and to help each other to optimize [8].

B. Competitive Coevolution in various applications:

The algorithms below are implemented in various applications which are based on co-operative Co-evolutionary algorithm:

- CCQGA
- ComCo MPSO

a. Competitive Co-evolutionary Quantum Genetic Algorithm (CCQGA):

Quantum-inspired evolutionary algorithm has been introduced recently and gained much attention and wide applications for both function and combinatorial problem. The competitive co-evolutionary quantum genetic algorithm (CCQGA) begins by randomly generate two initial populations. With corresponding competitive co-evolutionary strategy calculate the fitness values of two populations, then calculate the competitive degree of Population1 and Population2. Perform selection, crossover, mutation and quantum rotation operation for population1 and population2 with corresponding criteria until termination condition is satisfied, finally the best scheduling result is produced [10].

b. Competitive Coevolutionary MOPSO (ComCo MPSO):

Ideally, all particles from the competing sub-swarms should compete with all other particles from the other sub-swarms in order to determine the extent of its suitability. However, such an exhaustive approach requires extensive computational effort and it is practically infeasible. In this, each sub-swarm will be assigned a probability of representing a particular variable and only two sub-swarms, the current sub-swarm and competitor sub-swarm will compete for the right to represent any variable at any one time. The selection probability is initialized, so that all sub-swarms have equal probability of being selected [9]. This probability will be updated depending on outcome of the competition process.

In the competitive co-evolutionary MOPSO, in first cycle ith variable is assigned to the ith sub-swarm. In the subsequent cycles, a competing sub-swarm is selected using roulette wheel selection based on the selection probability to compete against the current species for the right to represent the, say, jth decision variable. During the competition process, the representatives of the current and competitor sub-swarms will combine with representatives of the other sub-swarms to form two complete solutions. The sub-swarm is providing the better solution is the winner and will represent the jth decision variable in the next round of cooperation. Therefore, the selection probability of a sub-swarm will increase as it becomes increasingly adapted to the decision variable. Vice versa, the probability of an unsuitable sub-swarm will be reduced [9].

V. SIMPLE COEVOLUTIONARY ALGORITHMS PROCEDURE

Simple Coevolutionary Algorithm is basically divided into two based on population, that is single population coevolutionary algorithm and multi population coevolutionary algorithm. After all, biologists use the term coevolution to refer to genetic influence of two species over each other. Since a population refers to a collection of members of a species, it seems that we must have two populations to have coevolution.

In a single population CoEA (Algorithm 1), individuals are evaluated by interacting with other individuals from that population.

Algorithm 1**: SINGLE POPULATION CoEA**

a. Initialize *population*
b. Select *evaluators* from population
c. Evaluate *individuals* from *population* by interacting with *evaluators*
d. **while not** done **do**
    a) Select *parents* from *population*
    b) Produce *children* from *parents* via variation
    c) Select *evaluators* from(*children, parents*)
    d) Evaluate *individuals* from *children* by interacting with *evaluators*
    e) Select survivors for next generation
    end while
e. **return** solution

In a multi-population CoEA (Algorithm 2), individuals in one population interact with individuals in one or several other populations.

Algorithm 2: **MULTI-POPULATION CoEA**

a. **for** each *pop* ∈ populations **do**
    a) Initialize *pop*
    b) Select evaluators from (*populations* − *pop*)
    c) Evaluate *individuals* from *pop* by interacting with *evaluators*
    end for
b. **while not** done **do**
c. **for** each *pop* ∈ populations **do**
    a) Select *parents* from *pop*
    b) Produce *children* from *parents* via variation
    c) Select *evaluators* from (*populations* – *pop*)

d) Evaluate *individuals* from *children* by interacting with *evaluators*
e) Select *survivors* for next generation

end for
end while

d. return solution

VI. CONCLUSION

This observation is relatively new and unexplored, but represents an exciting direction in coevolutionary algorithms research. It offers the possibility that a coevolutionary algorithm could not only find interesting or capable entities, but do so faster than random search would. The introduction to co-optimization and coevolutionary computation given in this study lays a strong foundation for more research in coevolutionary computation.

VII. REFERENCES

[1] Axlerod R, "The Evolution of Cooperation, Basic Books".

[2] Baeck T,"Evolutionary Algorithms in Theory and Practice", Oxford University Press.

[3] Bull L & Fogarty T C, "Evolving Cooperative Communicating Classifier Systems" A V Sebald & L J Fogel, Proceedings of the Third Annual Conference on Evolutionary Programming, World Scientific, pp308-315.

[4] Floreano D & Nolfi S (1997), "Adaptive Behaviour in Competing Coevolving Species", in P Husbands & I Harvey (eds.) Proceedings of the 4th European Conference on Artificial LIfe, MIT Press, pp378-387.

[5] Holland J H (1975)(ed.) Adaptation in Natural and Artificial Systems, University of Michigan Press.

[6] Holland J H (1975)(ed.) Adaptation in Natural and Artificial Systems, University of Michigan Press.

[7] Bull L & Fogarty T C (1996a), "Evolutionary Computing in Cooperative Multi-Agent Systems", in S Sen (ed.) Proceedings of the 1996 AAAI Symposia on Adaptation, Coevolution and Learning in Multi-Agent Systems, AAAI, pp22-27.

[8] Rahul Kala, "Multi-robot path planning using co-evolutionary genetic programming"

[9] C.K. Goh, K.C. Tan, D.S. Liu, S.C. Chiam, " A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design".

[10] Jinwei Gu, Manzhan Gu, CuiwenCao, Xingsheng Gu, "A novel competitive co-evolutionary quantum genetic algorithm for stochastic job shop scheduling problem".

[11] Chang Ying-Hua* , "Adopting co-evolution and constraint-satisfaction concept on genetic algorithms to solve supply chain network design problems".

[12] Joaquı´n Derrac, Salvador Garcı , Francisco Herrera, "IFS-CoCo: Instance and feature selection based on cooperative coevolution with nearest neighbor rule"

[13] Ying-Hua Chang , Tz-Ting Wu, "Dynamic multi-criteria evaluation of co-evolution strategies for solving stock trading problems".

[14] W. D. Hillis, "Co-evolving parasites improve simulated evolution as an optimization procedure,"Artif. Life II, vol. X, pp. 313–324, 1992.

[15] S. G. Ficici, "Solution concepts in coevolutionary algorithms," in Comput. Sci.. Waltham, MA: Brandeis Univ., 2004.S. G. Ficici, "Solution concepts in coevolutionary algorithms," in Comput. Sci.. Waltham, MA: Brandeis Univ., 2004.

[16] S. G. Ficici and J. B. Pollack, "Pareto optimality in coevolutionary learning," in Proc. 6th Eur. Conf. Advances in Artif. Life, 2001, pp. 316–325.

[17] M. A. Potter and K. A. D. Jong, "Cooperative coevolution: An architecture for evolving coadapted subcomponents,"Evol. Comput., vol. 8, pp. 1–29, 2000

[18] K. O. Stanley and R. Miikkulainen, "Competitive coevolution through evolutionary complexification," J. Artif. Intell. Res., vol. 21, pp. 63–100, 2004

[19] A. Bucci and J. B. Pollack, "On identifying global optima in cooperative coevolution," inProc. Genetic and Evol. Comput. Conf., Wash-ington, DC, 2005, pp. 539–544

[20] L. Pagie and P. Hogeweg, "Evolutionary consequences of coevolving targets,"Evol. Comput., vol. 5, pp. 401–418, 1997

[21] R. A. Watson and J. B. Pollack, "Coevolutionary dynamics in a minimal substrate," inProc. Genetic Evol. Comput. Conf., 2001, pp. 702–709

[22] B. Dolin, F. H. Bennett, III, and E. G. Rieffel, "Co-evolving an effective fitness sample: Experiments in symbolic regression and distributed robot control," inProc. 2002 ACM Symp. Appl. Comput., Madrid, Spain, 2002, pp. 553–559

[23] J. R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection. Cambridge, MA: MIT Press, 1992

[24] C. D. Rosin, Coevolutionary Search Among Adversaries. San Diego, CA: Univ. California, 1997

[25] E. D. D. Jong and J. B. Pollack, "Ideal evaluation from coevolution," Evol. Comput., vol. 12, pp. 159–192, 2004