Why Evolutionary Ontologies are a completely different field than Genetic Algorithms

Oliviu Matei, Diana Contraş, Alina Pintescu
Technical University of Cluj-Napoca
Baia Mare, Romania

***Abstract* - Evolutionary ontologies (EO) are a field of evolutionary computation as genetic algorithms (GA). Although there are commonalities between the two concepts, we will demonstrate by means of this article that there are significant differences, which makes them completely distinct.**

Keywords: evolutionary ontologies, genetic algorithms.

I. INTRODUCTION

Evolutionary computation is a field of computer research inspired by the natural evolution. The algorithms that appear in this area are called evolutionary algorithms and they include subdomains such as genetic algorithms, evolutionary strategies and genetic programming [26].

Genetic algorithms are adaptive heuristic search techniques based on the principles of genetics and natural selection, according to [6]. The individuals of GA are named "chromosomes" and they are usually encoded as binary bit strings. Initial population is chosen randomly. Each individual is evaluated by fitness function according to the objective problem. A series of steps are performed in a loop: parents are chosen by the selection mechanism, offspring are generated by crossing parents, offspring are modified through mutation operator, offspring are evaluated by fitness function, thereupon are selected the survivors that will form the population of the new generation and the loop resumes [30].

In [19] Matei et al. have introduced the term of "evolutionary ontologies" (EO). They are evolutionary algorithms which manipulate ontologies as individuals.

This article shows several reasons for which EO's are a different field of genetic computation than GA, although they share some commons aspects.

In section 2 we will present an overview of genetic algorithms, then, in section 3 we will describe the evolutionary ontologies and, finally, in section 4, the major differences between the two are detailed.

II. GENETIC ALGORITHMS

Genetic algorithms, intoduced by Holland and his students [9], are a family of computational models inspired by evolution, according to [27].

Genetic algorithms are used in general for solving optimization problems and in particular for combinatorial problems [18].

A. GA Individuals

In solving problem using genetic algorithms an important decision is the encoding of individuals. According to [21] there are many ways to represent individuals by type of problem to be solved. The most common encoding is a binary encoding of individuals which gives many possible chromosomes with a small number of alleles, according to [15].

In ordering or queuing problems is used mainly permutation encoding, where every chromosome is a string of integer numbers [15].

When complicated values are required, value encoding can be used. In this case every chromosome is a string of some values as form number, real number or characters [15].

B. GA Selection

In GA selection operator is designed to allow the best individual to transmit their genes to the next generation, according to [25]. Usually, selection operator works at chromosome's level by fitness function.

An important parameter in GA is selection pressure [25]. Selection pressure is the probability of selecting the best individual compared with the average probability of selection of all individuals [31]. A selection mechanism should be chosen so as to achieve convergence to the global optimum without causing a blockage in a local optimum [25].

Sivaraj classifies the methods of selection as traditional mechanisms and alternative selection mechanisms. The first category comprises Proportionate Selection methods (Roulette Wheel selection, Deterministic Sampling, Stochastic Remainder Sampling, Stochastic Remainder selection with replacement, Stochastic remainder selection without replacement, Stochastic universal selection), Ranking Selection (Linear Ranking selection, Truncate selection) and Tournament Selection (Binary Tournament Selection, Larger Tournament Selection, Boltzmann tournament selection, Correlative Tournament Selection). The second category comprises of Range selection, Gender-Specific Selection (Genetic algorithm with chromosome differentiation, Restricted mating, Correlative Family- based selection) and GR based selection (Fitness Uniform selection scheme, Reserve selection). For a detailed description of these selection methods see [25].

The purpose of all selection methods regardless of their type is to create multiple copies of individual with high value of fitness function [25]. It is impossible to say that a selection method is better than another. It is important to choose the right type of selection method for the problem to be solved so as to achieve the optimality of the solution.

C. GA Crossover

The purpose of crossover operator is to recombine two chromosomes to get a better chromosome, according to [20].

Malhotra et. al. classify crossover operator depending on the method of encoding data. If binary encoding is used, then one point, two point, uniformly or arithmetically crossover may be suitable for use. For permutation encoding one point crossover is appropriate. In case of value encoding all types of crossover used for binary encoding can be performed [15].

In the case of one point crossover, an integer number between 0 and the length of the chromosome is selected. This number is the crossover point. The first offspring is composed of the genes to the left of crossover point from the first parent and the genes to the right of crossover point from the second parent [24]. The second offspring is composed of the genes to the left of crossover point from the second parent and the genes to the right of crossover point from the first parent. If we have two chromosomes

$$x=(x_1x_2\dots x_kx_{k+1}\dots x_r) \quad (1)$$

$$y=(y_1y_2\dots y_ky_{k+1}\dots y_r) \quad (2)$$

and k is the cutting point will result two offspring

$$x'=(x_1x_2\dots x_ky_{k+1}\dots y_r) \quad (3)$$

$$y'=(y_1y_2\dots y_kx_{k+1}\dots x_r) \quad (4)$$

For two point crossover are chosen two different points between 1 and r-1 (where r este the length of the chromosome). The first offspring is composed of the genes to the left of first crossover point from the first parent, the genes to the right of first crossover point and the left of second crossover point from the second parent and the genes to the right of second crossover point from the first parent. The second offspring is composed of the genes to the left of first crossover point from the second parent, the genes to the right of first crossover point and the left of second crossover point from the first parent and the genes to the right of second crossover point from the second parent [29]. If we have two chromosomes

$$x=(x_1x_2\dots x_{k1}x_{k1+1}\dots x_{k2}x_{k2+1}\dots x_r) \quad (5)$$

$$y=(y_1y_2\dots y_{k1}y_{k1+1}\dots y_{k2}y_{k2+1}\dots y_r) \quad (6)$$

and k1, k2 are the two cutting points will result two offspring

$$x'=(x_1x_2\dots x_{k1}y_{k1+1}\dots y_{k2}x_{k2+1}\dots x_r) \quad (7)$$

$$y'=(y_1y_2\dots y_{k1}x_{k1+1}\dots x_{k2}y_{k2+1}\dots y_r) \quad (8)$$

In uniform crossover each gene of the first offspring has a probability of 0.5 of inheriting from the first parent, otherwise it inherits from the second parent. The second offspring has genes selected inversely to the corresponding gene of the first offspring [29].

In [18] is stated that for arithmetic crossover are used arithmetic operation to produce offspring. The operation depends on the representation of the individuals. Thus for binary representation, operators like AND, OR, XOR may be used, whereas operator as average may be used for float representation.

D. GA Mutation

Mutation operator has the role of causing random changes in the chromosome, generally applied in the genes. Mutation restore genetic diversity of population helping it avoid a local optimum level locking [12].

Like crossover, mutation depends on type of encoding. For binary encoding mutation transform the bit 1 into bit 0 and reverse. In permutation encoding two genes are selected randomly and their order is changed. If value encoding is used a small number will be added or subtracted from the selected genes to produce offspring [15].

III. EVOLUTIONARY ONTOLOGIES

In artificial intelligence the ontology term, borrowed from philosophy, was defined by Gruber [8] as an explicit specification of a conceptualization. Nowadays the ontologies are used in different areas from technological processes [14], [22], [23] to medical field [16], [17].

Using ontologies as individuals rather than any other data structure in a genetic algorithm an ontological evolutionary algorithm breeds. The ontological space (called "onto-space") is the definition given to the solution space, the restriction and boundaries of this evolution.

The onto-space is an ontology describing a domain specific knowledge, containing all the concepts along with their allowed and denied relationships, according to [19]. The onto-space defines the degrees of freedom as well as the boundaries of the solution space to be searched by the evolutionary process. Quite often, the solution space is infinite and special algorithms are needed for exploring it efficiently. An onto-space would be the ontology about all electronic appliances and the solution required to be found is a possible arrangement of a kitchen given some restrictions.

Formally, an onto-space

$$OS = (C, P, I) \quad (9),$$

where C is the set of classes, P is the set of properties and I is the set of instances.

Within the ontology OS, there are two disjunctive sub-ontologies

$$OS_e = (C_e, P_e, I_e) \quad (10)$$

$$OS_f = (C_f, P_f, I_f) \quad (11)$$

$$OS_e \cup OS_f = OS \quad (12)$$

OS_e is the sub-ontology which will undergo the evolutionary process and OS_f is the fixed sub-ontology, e.g. which will not change under the evolutionary process.

A. EO Individuals

An individual is a subset of the ontology, represented as

$$Ch = (C_i, P_i, I_i) \quad (13)$$

where $C_i \subset C$ is a subset of classes in OS, $P_i \subset P$ is a subset of properties in OS and $I_i \subset I$ is a subset of instances in OS. Further on, a genetic individual consists of an evolving part

$$Ch_e = (C_ie, P_ie, I_ie) \quad (14)$$

which will be changed during the evolutionary process, and a fixed part

$$Ch_f = (C_if, P_if, I_if) \quad (15)$$

The two parts hold the following relations:

$$Ch_e \cup Ch_f = Ch \quad (16)$$

$$Ch_e \cap Ch_f = \varnothing \quad (17)$$

Moreover:

$$Ch_e \subset OS_e \quad (18)$$

$$Ch_f \subset OS_f \quad (19)$$

$$Ch \subset OS \quad (20)$$

A population consists of a given (μ) such individuals and does not necessarily cover the entire onto-space, therefore

$$\cup_i Ch_i \subset OS \quad (21)$$

B. EO Selection

In the case of EO it is used deterministic selection (μ, λ) – selection or (μ + λ) – selection. For first type of selection μ parents produce λ offspring (λ > μ) and only the offspring undergo selection. For the second type of selection μ parents produce λ offspring and all solutions compete to survival.

The deterministic selection is chosen in favor of other types of selections as it is rather difficult to correlate mathematically ontologies with their relative fitness, needed for other techniques.

C. EO Crossover

For EO can distinguish three types of crossover operators: class crossover, instance crossover and relation crossover.

In an ontology classes are organized hierarchically. Two groups of related classes and subclasses are randomly selected as parents, a cutting point is chosen, it changes between the two parent classes to the point of cutting and the two resulted groups are the offspring. In doing so is likely the ontology to become inconsistent. In such cases the repair operator (see subsection 3.5) will be used to validate the ontology.

In an ontology for each class can be established as many instances as wanted. The instances are not independent, but related through object properties. For crossover, two groups of related instances are selected randomly as parents.

In an ontology, there are two types of properties: the object level and the data level. At object level are elected two object properties P1 and P2 as parents. Each property has a domain and a range from among the classes:

$$C1_1\ P1\ C1_2 \quad (22)$$

$$C2_1\ P2\ C2_2 \quad (23)$$

After crossover will get:

$$C2_1\ P1\ C1_2 \quad (24)$$

$$C1_1\ P2\ C2_2 \quad (25)$$

or

$$C1_1\ P1\ C2_2 \quad (26)$$

$$C2_1\ P2\ C1_2 \quad (27)$$

As in the class crossover case the result may be inappropriate. The repair operator (see subsection 3.5) will remove inconsistency.

In an ontology each data property (DP) has a do-main from among the classes and a range from different data types like integer, double, float etc. At data level crossover are selected as parents two classes with several data properties:

$$C1\ (DP1_1, DP1_2, DP1_3, ..., DP1_n) \quad (28)$$

$$C2\ (DP2_1, DP2_2, DP2_3, ..., DP2_m) \quad (29)$$

A cuting point is selected and the result of the crossover will be the same classes with modified properties:

$$C1\ (DP2_1, DP2_2, DP1_3, ..., DP1_n) \quad (30)$$

$$C2\ (DP1_1, DP1_2, DP2_3, ..., DP2_m) \quad (31)$$

The repair operator (see subsection 3.5) will be also applied if appropriate result is not obtained.

The pseudo code for the crossover operator is described in algorithm 1.

Algorithm 1 Crossover - pseudo code showing the crossover operator

```
procedure recombine(population)
newPopulation = ∅;
parentPopulation = select λ individuals randomly
for all ind1 and ind2 ∈ parentPopulation do
    choose a random cutting point (in the tree
      formed by the classes)
    create two offspring by preserving the ordering
      position of symbols in the corresponding
      sequences of the parents
    adjust the object properties according to the new
      class structure
    add the two offspring to the newPopulation
end for
end procedure
```

D. EO Mutation

Like crossover, the mutation operator for EO requires different treatment for classes, for instances, respectively for properties. It is applied for each individual with a probability p_m.

The pseudo code for mutation is shown in algorithm 2.

Algorithm 2 The Mutation - pseudo code describing the mutation procedure

```
procedure mutate(newPopulation)
for all ind ∈ newPopulation do
    choose a random number r ∈ [0, 1)
    if r < pm then
        choose a random integer number rm ∈ 1, 2
        if rm = 1 then
            applyInstanceMutation(ind)
        else
            applyClassMutation(ind)
        end if
    end if
end for
end procedure
```

Instance mutation means replacing a randomly selected instance *i* belonging to a class C ($i \in C$) with another individual $i' \in C$ from the onto-space OS. This operator preserves the number of ontological instances in an individual.

A class mutation means replacing all the instances in a class C by other individuals belonging to a random subclass $SC \subseteq C$ in the onto-space.

The property mutations may be approached separately for data properties, respectively for object properties.

E. Repair Operator

Applying classical genetic operators: crossover and mutation on ontologies, they can be easily corrupted on the strength of their complex structure. That is why Matei et al. [19] introduced a new operator, called repair, which is a

deterministic operator. It can be applied on the population each time another genetic operator is used or only once, after crossover and mutation. Repairing an individual means adjusting its instances and properties so that they respect all the rules defined in the onto-space.

F. Other Genetic Operator

The EO's have a strong expressivity power due to it symbolic nature. Therefore several new genetic operators may be defined at a class level as well as at individual level. Moreover, the relationships of an ontology are very suited for subject of new operators, such as union, intersection and composition.

IV. Genetic Algorithms vs. Evolutionary Ontologies

First of all, although evolutionary computation is a sub-symbolic field of artificial intelligence [11], evolutionary ontologies are symbolic simply because ontologies are symbolic intelligence [1]. However this is not the first step towards bridging the two major domains: symbolic and subsymbolic, as shown by Goertzel in [7]. For instance, Andrews et al. try in [5] to extract rules from neural networks. However, it is for the first time when Matei et al. [19] apply the genetic principles to ontologies.

The strength of EO consists in the fact that they make use of the power of mathematical algorithms and the expressivity of ontologies. We cannot say anymore that evolutionary ontologies are subsymbolic intelligence as they make use of ontologies, therefore semantic; on the other hand they are not pure symbolic because they evolve using mathematical principles, which is never the case of other symbolic fields, such as knowledge-based systems [3] and intelligent agents [28].

The individuals of EO are ontologies themselves as they contribute to the evolutionary algorithm with all their elements: classes, individuals, relations, properties. In EO classes and instances increase in number and/or receive improved properties and relations as the result of their participation in the evolutionary act. On the other hand, the individuals of GA are strings of integer or float number or characters depending on the specific problem to be solved.

The crossover operator is represented in EO in three stages: class crossover, instance crossover and relation crossover. As shown in subsection 3.3 the three types of operators behave differently depending on individuals used as operands. If it is applied class crossover operator the result would be new class structure. When applying instance crossover the result would be new instances. Finally, relation crossover determines new relations or new properties in the ontology. In GA there are several types of crossover operator depending on the data encoding. Whatever crossover operator is applied to two strings representing parents will get two strings (not very different from the default) which are the offspring.

The mutation operator in EO is also differentiated according to the ontological element to which it applies. With class mutation it is obtain a new class structure by replacing all individuals of a class with individuals of a random subclass of that class. Instance mutation signifies the replacement of an individual with another individual from the same class as the initial individual. Data property mutation means the change of initial value depending on the type of data. In GA are identified more types of mutation operator based on the kind of encoding chosen. No matter what mutation operator is applied the result would be a string representing a chromosome with random genes modified from the default.

The selection operator used in EO is based on the model used in ES, namely (μ, λ)-selection or $(\mu + \lambda)$-selection, unlike GA where are used mainly types of selection based on fitness function.

The need of repair operator is required by the results of crossover and mutation operators in EO. The resulting evolutionary ontology may present evidence of inconsistency, which will be removed by repair operator. Affenzeller et al. show in [2] that a repair operator is often used in GA in order to convert an illegal chromosome to a legal one. There are more than one repair operators in GA depending on the specificity of the problem.

From the 70 GA is an open research area. Were thus introduced several new operators to the standard ones, like a new mutation operator developed to increase GA performance to find the shortest distance in the Traveling Salesman Problem [4] or new crossover operators namely sequential and random mixed crossover applied to a deep beam problem and, a concrete mix design problem [10] and the list goes on. EO is a new field of evolutionary computation. We customized the standard genetic operators: selection, crossover, mutation to EO and we have demonstrated the need to use repair operator in EO. Further we consider as future work the implementation of new operators due to the complexity of ontology elements and the relations between them.

V. Conclusions

We have proved in this article that between genetic algorithms and evolutionary ontologies there are several similarities:

- both are instantiations of evolutionary computation;
- the general algorithms are very similar, implying individuals which evolve undergoing some genetic operators, out of which three are classical: crossover, mutation and selection;
- the aim of both is optimization.

However, the gap between them, shown in table 1 is so large that make the evolutionary ontologies a distinct domain.

Table 1. The differences between GA and EO

Aspect	GA instantiation of the aspect	EO instantiation of the aspect
Intelligence level	Subsymbolic, entirely mathematical algorithms	Symbolic concepts evolved with subsymbolic algorithms

Solution space	The set of strings with the encode depending on the problem to be solved	An ontological space containing the possible instances and their relations
The individuals	Strings	Ontologies
Crossover	Depends on the type of encoding – binary, permutation or value	Actually we have three different operators, one for classes, one for instances and one for relations
Mutation	Depends on the type of encoding – binary, permutation or value	There are three different mutations for classes, instances and relations
Repair operator	Frequently used, but not mandatory	Absolutely needed as the individuals may contain very complex internal and external relations
Selection	Based on fitness function	Deterministic
New operators	New operators were introduced and continue to occur	New specific operators may be defined because ontologies imply different concepts and princples

Acknowledgments. This paper was supported by the Post-Doctoral Programme POSDRU/159/1.5/S/137516, project co-funded from European Social Fund through the Human Resources Sectorial Operational Program 2007-2013.

REFERENCES

[1] Aerts, Diederik, and Marek, Czachor, (2003). Quantum aspects of semantic analysis and symbolic artificial intelligence. *arXiv preprint quant-ph/0309022*.

[2] Affenzeller, Michael, Wagner, Stefan, Winkler, Stephan and Beham Andreas, (2009). Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications p. 131 ISBN 978-1-58488-629-7

[3] Akerkar, Rajendra, and Priti Sajja, (2010). Knowledge-based systems p. 19 ISBN-13: 978-0-7637-7647-3.

[4] Albayrak, Murat, and Novruz, Allahverdi, (2011). Development a new mutation operator to solve the Traveling Salesman Problem by aid of Genetic Algorithms. *Expert Systems with Applications*, vol. 38, ISSN 09574174 p. 1313-1320.

[5] Andrews, Robert, Joachim Diederich, and Alan B. Tickle, (1995). Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-based systems*, vol. 8, ISSN 0950-7051 p. 373-389.

[6] Ferariu, Lavinia, (2013). Sisteme inteligente hibride. zharieBucureşti: Conspress.

[7] Goertzel, Ben, (2012). Perception processing for general intelligence: Bridging the symbolic/subsymbolic gap. *Artificial General Intelligence*, vol. 7716, ISSN 1946-0163 p. 79-88.

[8] Gruber, Thomas R., (1993). A translation approach to portable ontology specifications. *Knowledge acquisition*, vol. 5, ISSN 1042-8143 p. 199-220.

[9] Holland, John, (1975). Adaptation in natural and artificial system ISBN 0-262-58111-6

[10] Kaya, Mustafa, (2011). The effects of two new crossover operators on genetic algorithm performance. *Applied Soft Computing*, vol. 11, ISSN 1568-4946 p. 881-890.

[11] Kelley, Troy D., (2003). Symbolic and sub-symbolic representations in computational models of human cognition what can be learned from biology? *Theory & Psychology*, vol. 13, ISSN 0959-3543 p. 847-860.

[12] Konak, Abdullah, David W. Coit, and Alice E. Smith, (2006). Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering & System Safety*, vol. 91, ISSN 0951-8320 p. 992-1007.

[13] Koza, John, (1992). Genetic programming: on the programming of computers by means of natural selection ISBN 0-262-11170-5

[14] Lobonțiu, Mircea and Petrovan, Adrian, (2012). A Product development ontology(1). Information integration concepts. *Revista de Management și Inginerie Economică*, vol. 11, ISSN 1583-624X p. 43-56

[15] Malhotra, Rahul, Narinder, Singh and Yaduvir Singh, (2011). Genetic algorithms: Concepts, design for optimization of process controllers. *Computer and Information Science*, vol 4, ISSN 1913-8989 p. 39.

[16] Matei, Oliviu, (2008). Defining an Ontology for the Radiograph Images Segmentation. *9th International Conference on Development and Application Systems*, place Suceava, Romania, 22.05.2008

[17] Matei, Oliviu, (2008). Ontology-Based Knowledge Organization for the Radiograph Images Segmentation. *Advances in Electrical and Computer Engineering*, vol 8, ISSN 1582-7445 p. 56-61.

[18] Matei, Oliviu, (2008). Evolutionary Computation: Principles and Practices p. 19-29 ISBN 978-973-751-944-3

[19] Matei, Oliviu, Contraș, Diana and Pop Petrică, (2014). Applying Evolutionary Computation for Evolving Ontologies. *Proceedings of CEC 2014*, China, 06.07.2014

[20] Mathew, Tom, (2012). Genetic algorithm. Report submitted at IIT Bombay.

[21] Nedjah, Nadia and Macedo Mourelle, Luiza de, (2002). Minimal Addition-Subtraction Chains Using Genetic Algorithms. *Advances in Information Systems*, vol. 2457, ISSN 1532-0936 p. 303-313

[22] Petrovan, Adrian and Lobonțiu, Mircea, (2012). Product development ontology. A case study, *Quality - Access to Success*, vol. 13, ISSN 1582-2559 p. 393-398.

[23] Petrovan, Adrian and Lobonțiu, Mircea, (2013). Broadening the Use of Product Development Ontology for One-off Products. *Applied Mechanics and Materials*, vol. 371, ISSN 1662-7482 p. 878-882

[24] Shukla, Anupam, Ritu, Tiwari and Rahul, Kala, (2012). Real life applications of soft computing p.161 ISBN 978-1-4398-2289-0

[25] Sivaraj, R. and Ravichandran, T. (2011). A review of selection methods in genetic algorithm. *International journal of engineering science and technology*, vol. 3, ISSN 2141-2820 p. 3792-3797.

[26] Stoean, Cătălin and Stoean, Ruxandra, (2010). Evoluție și inteligență artificială. Paradigme moderne și aplicații ISBN 978-973-650-277-4

[27] Whitley, Darrell, (1994). A genetic algorithm tutorial. *Statistics and computing*, vol. 4, ISSN 0960-3174 p. 65-85.

[28] Wooldridge, Michael and Nicholas, R. Jennings, (1995). Intelligent agents: Theory and practice. *The knowledge engineering review*, vol. 10, ISSN 0269-8889 p. 115-152.

[29] Xinjie, Yu and Mitsuo Gen, (2010). Introduction to evolutionary algorithms p.43-44 ISBN 978-1-84996-129-5

[30] Zaharie, Daniela, (2013). Curs Calcul neuronal și evolutiv. Facultatea de Matematică și Informatică, Universitatea de Vest Timișoara.

[31] Zaharie, Daniela, (2006). Algoritmi genetici - Curs 3. Algoritmi evolutivi - metode de selecție