Presenting the ECO: Evolutionary Computation Ontology

Anil Yaman[1,2], Ahmed Hallawa[3], Matt Coler[2,4], and Giovanni Iacca[2(✉)]

[1] Department of Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
a.yaman@tue.nl

[2] INCAS3, P.O. Box 797, 9400 AT Assen, The Netherlands
m.coler@rug.nl, giovanni.iacca@gmail.com

[3] Chair for Integrated Signal Processing Systems, RWTH Aachen University, 52056 Aachen, Germany
hallawa@ice.rwth-aachen.de

[4] Campus Fryslân, University of Groningen, Sophialaaan 1, 8911 AE Leeuwarden, The Netherlands

AQ1

Abstract. A well-established notion in Evolutionary Computation (EC) is the importance of the balance between exploration and exploitation. Data structures (e.g. for solution encoding), evolutionary operators, selection and fitness evaluation facilitate this balance. Furthermore, the ability of an Evolutionary Algorithm (EA) to provide efficient solutions typically depends on the specific type of problem. In order to obtain the most efficient search, it is often needed to incorporate any available knowledge (both at algorithmic and domain level) into the EA. In this work, we develop an ontology to formally represent knowledge in EAs. Our approach makes use of knowledge in the EC literature, and can be used for suggesting efficient strategies for solving problems by means of EC. We call our ontology "Evolutionary Computation Ontology" (ECO). In this contribution, we show one possible use of it, i.e. to establish a link between algorithm settings and problem types. We also show that the ECO can be used as an alternative to the available parameter selection methods and as a supporting tool for algorithmic design.

Keywords: Ontology · Knowledge representation · Evolutionary computation

1 Introduction

Ontologies are a type of formal knowledge representation that make it possible to represent kinds of knowledge for different applications [1]. Ontologies structure data as a network of objects and their relations. These objects refer to entities and events (also known as concepts) in the real world, and their relations represent the semantic relations between entities. Thus, ontologies can be represented as a graph structure similar to semantic networks [2].

G. Squillero and K. Sim (Eds.): EvoApplications 2017, Part I, LNCS 10199, pp. 1–16, 2017.
DOI: 10.1007/978-3-319-55849-3_39

Ontologies have been used successfully in a number of applications, such as knowledge modeling [3] and decision support for medical diagnosis [4]. A comprehensive survey of such application was performed by [5]. The goal of building an ontology depends normally on its use: for instance, in Evolutionary Computation (EC), ontologies can be used to represent knowledge about the evolutionary operators and their parameters, as well as the problem features and their fitness landscapes. Knowledge plays a key role in various aspects of EC including encoding, population initialization, evolutionary operators [6]. For instance, individual representations facilitate implicit knowledge, while population initialization, fitness evaluation, and evolutionary operators facilitate explicit knowledge incorporation [7]. Knowledge incorporation is in turn beneficial for an efficient search, by balancing the trade-off between exploration and exploitation. In the literature there are a number of works where knowledge incorporation has been used, for instance in the design of evolutionary operators [7], in the selection of the algorithm parameters [8] and in the selection process in Interactive Evolutionary Computation (IEC) [9].

In this work, we present the "Evolutionary Computation Ontology" (ECO), an ontology that is primarily designed to include available knowledge in EC and problem domains, and to establish a link between algorithmic and domain knowledge; in this sense, it can provide the background for reasoning methods such as case-based and analogical reasoning. As we will see in Sect. 3, some motivating use cases for the ECO are operator and parameter selection, human-made selection in IEC, user guidance in EC software tools [10], and teaching and self-learning of EC-related topics [11].

As a proof-of-concept, we show how the ECO can be used for instance to define efficient strategies (i.e., algorithm settings, see Table 2 in Sect. 4 for a complete definition) based on the type of the problem. In particular, we focus on the strategies for *parameter selection*. First, we make use of the existing literature in EC to collect these strategies. However, since most of the proposed strategies are defined as vague rules that rarely are applicable off-the-shelf, we perform an empirical analysis to specify such rules in concrete terms. We then populate our ontology with concrete rules to be used for parameter selection. We describe the development process of our ontology in Sect. 4, and perform experiments to demonstrate the performance of the ECO. We then present our experimental results in Sect. 5. In these experiments, we used the ontology to extract rules established on benchmark functions, and compared the results of an evolutionary algorithm based on such rules, with the performance of a control algorithm whose parameters are chosen arbitrarily. Finally, we discuss our concluding remarks and the future work in Sect. 6.

2 Background

The term "ontology" is used in computer science to describe a type of knowledge representation that explicitly defines what exists in a domain. Ontologies represent knowledge by defining concepts, their properties and relations, and the

individuals [12]. The properties of concepts describe their features; and, the relations define how concepts semantically relate to one another. These components are essential for representing knowledge. For example, taxonomic relations are one of the fundamental relations, as they establish the concept class hierarchy using "is-a" relation. The individuals are instances of the concept classes.

A general guideline of the practical steps for developing an ontology is provided by Noy et al. [13]. An initial step for developing ontologies is to identify the domain and scope of the ontology. It is impossible to represent everything; therefore, it is required to define its scope [14]. Thereafter, the concepts in the domain are identified and hierarchically structured. Next, the properties of the concepts and the relations between them are defined. In the final step, individuals are added to the ontology.

To guarantee the interoperability across different ontologies from different domains, a number of standardized formal representation languages have been proposed for developing ontologies. These include the Resource Description Framework (RDF) [15], a framework for structuring the data as an easy-to-process graph and the Web Ontology Language (OWL) [16], an ontology development language endorsed by World Wide Web Consortium (W3C) [17]. Once ontologies are developed, the knowledge can then be retrieved by query languages such as the SPARQL Protocol or the RDF Query Language (SPARQL), a standard query language for querying RDF structured data [18].

3 Motivating Use Cases

In this section, we motivate knowledge incorporation into Evolutionary Computation by discussing some possible use cases and applications.

3.1 Operator and Parameter Selection

The knowledge involved into the evolutionary operator and parameter selection plays a key role for balancing the trade-off between the exploration and exploitation processes, a fundamental aspect for an efficient search [19]. In addition, the availability of any additional knowledge about the problem domain offers a great source of information for suggesting efficient strategies for evolutionary search. There are several approaches listed in Table 1 that can be used for selecting the evolutionary operators and adjusting their parameters.

The approaches listed in (1)–(8) are outlined by Črepinšek et al. [19]. We included an additional approach (9). Ontologies can be designed to include the knowledge involved in the approaches (2), (4), (5), (6) and (9). The ECO is designed to include the knowledge involved in (2) and (5), but it is also applicable to the approaches (4) and (9). It is applicable to (4) (also known as case-based reasoning [20]) because it establishes the links between (2) and (5), and makes it possible to reuse these links on similar problems; it is also applicable to (9) because it allows the structural mappings between different knowledge representations from different domains. Although there has been extensive research in

Table 1. Approaches to evolutionary operator/parameter selection (adapted from [19])

#	Approach
(1)	Trial and error
(2)	Following guidelines from the literature [21,22]
(3)	Parameter-less EA [23]
(4)	Using experience from previous similar problems [24]
(5)	Identifying fitness landscape's features to propose suitable parameters [25,26]
(6)	Analyzing the parameters and their effects statistically [27]
(7)	Mathematical modeling
(8)	Optimizing the parameters algorithmically [28,29]
(9)	Adapting solutions to the problems from different domains

the literature focused on evolutionary operators and parameters, to the best of our knowledge (as also demonstrated by [19]) there is no application that makes direct use of such knowledge in the approaches given in (4) and (9). Our work here aims at collecting (part of) this knowledge accumulated over many decades of EC literature, and make it available —in a structured and consistent way— for future developments in Evolutionary Computing.

We hypothesize that it is beneficial to use the existing knowledge in the ECO, i.e. that similar strategies can solve similar problems. Under this assumption, transferring the algorithm settings obtained from past problems to new problems should produce an advantage in terms of optimization. In this scenario, the ontology can be populated with example problems (representing different problem types: unimodal vs multimodal, separable vs non-separable, etc.) and their landscape properties can be described by using the concepts in the ECO. Using the links between the concepts related to problem types and strategies present in the ontology, specific strategies that apply to specific problem types can then be identified.

By leveraging the ECO, all this existent knowledge can be collected and used automatically for selecting the most efficient strategies for a problem. Using the knowledge included in the ECO, the following example questions can be answered (the definitions of the concepts mentioned in the example questions can be found in Table 2, see Sect. 4):

- What is the best strategy for unimodal functions?
- What are the operators and their parameters that cause low exploration?
- Which crossover operator is the best for mutimodal functions when the evolutionary process is in is "maturation" phase?

3.2 Human-Made Selection in Interactive Evolutionary Computation

Another example task that involves knowledge is in Interactive Evolutionary Computation, where experts and users can involve directly into the evolutionary process to select the individuals they find interesting, in order to pass their genes to the next generation [9]. However, this process is often extremely time-consuming and requires a significant effort to perform the individual selection manually. Also, it is often impossible to monitor all individuals for thousands of generations, especially if the population is large. The ECO can provide support for improving the speed of IEC processes, thus reducing the effort spent by users. For instance, the knowledge of how selection is performed by humans can be represented in the ECO, and used automatically in the selection process to find similar individuals that satisfy the user's interests or preferences.

3.3 Guidance in Software Tools Based on Evolutionary Computation

Evolutionary algorithms are now widely available in several optimization software packages [10,30,31]. In general, it is straightforward to apply evolutionary algorithms to custom problems using these tools. However, it is not always so easy to adjust their settings if the user does not have any expertise in the EC field. As we discussed in Sect. 3.1, parameter selection requires extensive expertise regarding what concerns the algorithm configuration, the problem type, and the strategies that should be applied. Therefore in many cases it can be useful that these software tools are able to provide some guidance to users from different expertise domains.

In this sense, the ECO defines the knowledge required for supporting smart human computer interaction. Explicit relations between the concepts in the evolutionary algorithms, the problem types and the strategies can make the expert knowledge available in the software packages making use of evolutionary algorithms. For instance, the ECO can be used to recommend a suitable algorithm, as well as its operator and parameter configuration, based on the problem that is introduced by a user. The ECO is a good candidate for providing this support in software.

3.4 Teaching and Self-learning

Ontologies are well suited also for introducing new topics to students. In the literature, an example use of the ontologies for education in EAs was demonstrated by Kaur and Chaudhary [11], who included useful knowledge about the background and history in evolutionary algorithms into their ontology. It should be noted, however, that the main difference between the ontology proposed in [11] and that one proposed in this work is: while the ECO proposed by Kaur and Chaudhary is only limited to historic knowledge and education purposes, whereas, our ECO extends that knowledge, and includes efficient strategies and

problem types which is used for parameter selection problem. As such, our ECO is dynamically updatable, and extensible that support a lifetime learning and knowledge-based optimization.

4 Evolutionary Computation Ontology

The development process of the ECO involves the steps elaborated in Sect. 2. In this work, we focus on representing the knowledge for problem solving making use of the literature in EC. To structure the knowledge formalized in the ECO, we manually parsed 50 of the most cited research papers in the field. We then identified the relevant concepts, as shown in Table 2, and we extracted concrete strategy instances to populate the ontology.

There are different "knowledge categories" in ECO that describe the concepts in Evolutionary Computation. These are broadly categorized under: *evolutionary algorithms*, *evolutionary processes*, *problem types*, *search properties*, and *strategies*.

Evolutionary Algorithms. One of the earliest taxonomies of evolutionary algorithms is provided by Bäck and Schwefel [34]. We limit our ontology to the algorithms provided in their overview, with the addition of genetic programming. Thus, the domain knowledge covered in this area describes four main classes of evolutionary algorithms, namely, Genetic Algorithms (GA), Genetic Programming (GP), Evolutionary Programming (EP) and Evolutionary Strategies (ES). Each class of algorithms originates from a different research line, and has different characteristics. The properties of these classes define the algorithmic characteristics (i.e. data structure, population parameters, evolutionary operators), which

Table 2. Some example concepts represented in the ECO, and their definitions

Concept	Definition
Modality	A feature of a fitness landscape representing the no. of optima [32]
Maturation phase	One of the four phases of an evolutionary run: initial, sub-maturation, maturation and convergence [33]
Strategy	Algorithm settings that can be applied to an evolutionary algorithm for solving a problem. The settings of an algorithm define the data structure, the evolutionary operators and their parameters, the selection operator, the population size and the initialization methods
Percentage of performed evaluations	The number of evaluations performed at any point of an evolutionary run, divided by the total number of evaluations allotted to the EA
Exploration	A property of a search process that aims to visit as many search space regions as possible

are in turn included in concept classes. The concepts of the evolutionary operators and parameters are represented as algorithm-independent and therefore can be manipulated and plugged into any suitable evolutionary algorithm.

Evolutionary Processes. This knowledge area includes the properties related to the runtime of the evolutionary processes. Percentage of performed evaluations, convergence rate, average, minimum and maximum fitness values are some of the properties that describe an evolutionary run. These properties are univocally determined, except the convergence rate [35] which can be used e.g. for defining the phases of an evolutionary run. For example, Zhang et al. [33] measures the convergence rate by finding the relative sizes of the clusters (subpopulations) that include the worst and the best individual. In their work, they used this rate to identify the phases of an evolutionary process which they divided into *initial*, *sub-maturation*, *maturation* and *convergence*. We find it appropriate to include these concepts into our ontology, although the methods that define the convergence rate and the evolutionary phases are not restricted to these examples. For instance, the evolutionary phases can also be split based on the percentage of performed evaluations.

Problem Types. We define here the concepts linked to the fitness landscapes properties. We include the following types of functions: unimodal, multimodal, separable, non-separable, symmetric and non-symmetric [32]. An instance of a problem can be linked to one or more of these types.

Search Properties. This category includes two main concepts, namely "Exploration" and "Exploitation". Such concepts define special properties of the search where exploration refers to the process of visiting areas of the search space that have not been visited before; exploitation defines the process of visiting neighboring solutions to those that have been visited during an evolutionary run. There is a well-established knowledge in the correlation between the problem types and the suggested amount of computational budget dedicated to each of the two search regimes. These suggestions often are coupled with an evolutionary phase that they apply to, or to the specific problem type (see the example strategies defined below).

Strategies. Finally, we define the *strategies* to establish a link between the operator/parameter settings of evolutionary algorithms and the problem types they apply to. The strategies suggested in the literature are usually conditional on some of the runtime properties. Therefore, we include the conditions that can be defined to trigger a strategy. These conditions can be based e.g. on percentage of performed evaluations, convergence rate, and/or population diversity.

To populate our ontology, we extracted strategies that are suggested in the literature. Some of these strategies are presented in Table 3. There are various levels of specificity in these statements. For example, the strategies given in (1)

Table 3. Example strategies (with references)

#	Strategy
(1)	For unimodal functions, the mutation rate is constant and there is an optimal value which is $1/l$ where l is the string length. This value is quite low because there is no need to invest more into exploration [22,36]
(2)	"For deceptive trap functions of order k, the best mutation rate is k/l" [22,36]
(3)	"If only local variation operators are used, e.g., mutation flipping only a single bit, it is easy to see that then sub-functions in a separable function are optimized independently and in parallel" [37]
(4)	"High mutation rate can help within the first phase of the evolution, but it becomes useless when we get close to the best solution" [22,36]
(5)	"[...] in the early stages a larger population is needed than in the later stages, when fine tuning of sub-optimal solutions is done."[19]
(6)	Selection can mainly be used for exploitation. Adjusting the selection pressure in the selection operator changes the level of exploitation. The selection operators can be ranked by increasing selection pressure: proportional selection, linear ranking, tournament selection, and (μ, λ)- and $(\mu + \lambda)$-selection [19,38]

and (2) are quite specific as they suggest certain mutation rates that should be used for unimodal and deceptive functions; however, the strategies given in (4) and (5) are open to different interpretations, since they use categorical terms like "high" and "large" while their definitions are not explicit. For instance the statement given in (4) suggests starting with a "high" mutation rate in the beginning and decreasing it in later stages of the evolutionary process. It is also important to note that the concept of "high" mutation rate can be dependent on the problem (e.g. number of dimensions), as well as the other parameter settings (e.g. population size, crossover, etc.).

To translate these statements to readily applicable strategies in our ontology, we then performed a systematic set of preliminary experiments, and assessed the ranges of the categorical concepts to use. For illustration purposes, we picked a unimodal and a multimodal function, namely the Sphere and the Rastrigin's functions (F_1 and F_9 in [39]), in $D = 10$ dimensions, to observe examples of performance obtained with different parameter settings on different kinds of functions. Following an experimental setting similar to that performed in [40], we consider population sizes $n \in \{10, 30, 50, 100, 200\}$, Gaussian mutation with mutation rates $p_m \in \{0.0005, 0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5\}$, elite counts $e \in \{0, 1\}$, selection operator fixed as "roulette selection", and crossover rate fixed at 0.8. With these experimental settings, we aim to establish concrete, optimal ranges for the concepts {"high", "medium", "low"}.

The Supporting Information **SI.1**[1] presents the results we obtained on the Sphere and Rastrigin's functions using different mutation rates, population sizes and elite counts. All the experiments were implemented in Matlab, using the

[1] Available online at: http://www.goo.gl/xSgVvv.

Genetic Algorithm Toolbox [31]. Tables (a), (b), (c), and (d) present results for the Sphere function with elite count 0, the Rastrigin's function with elite count 0, the Sphere function with elite count 1 and the Rastrigin's function with elite count 1, respectively. The columns and the rows of the tables are organized w.r.t. mutation rates population sizes in ascending order. We follow the evaluation criteria presented in [39]. We set the maximum number of evaluations to $D \times 10^4$, and record the fitness of the best individual for the checkpoints at 1st, 10^3th, 10^4th and 10^5th generations. We run an evolutionary algorithm 25 times for each specified parameter settings for the two selected functions, and found the average best fitness values for each checkpoint.

We defined the concept "EvolutionaryProcessPhases" based on these checkpoints. We then refer to the evolutionary process between the checkpoints 1st and 10^3th, 10^3th and 10^4th, 10^4th and 10^5th as "Initial", "Maturation", and "Convergence" respectively. Our aim is to identify how a parameter set performs within each evolutionary phase. Therefore, we find the difference between the average fitness values at the start and the end point of each phase, and divide this delta by the number of evaluations performed in each phase. In other words, this value indicates the average change of fitness per evaluation observed within a phase.

With reference to **SI.1**, the color (gray in print) intensity level in each cell indicates the magnitude of the average fitness difference observed in each phase (scaled across the values within each row). With elite count 0, some parameter settings produced negative fitness differences, which we represented as zeros for the sake of illustration.

For the Sphere function with elite count 0 and pop. sizes $\{10, 30, 50, 100, 200\}$, mutation rates $\{0.2, 0.1, 0.2, 0.2, 0.1\}$ perform better in phase 1, respectively. However, when a larger population size is used, the difference in the performance obtained with a high mutation rate gradually decreases. The comparisons between phases 1 and 2, and between phases 2 and 3, reveal that in further phases lower mutation rates generally perform better. Moreover, we observe a clear pattern within phases 2 and 3, which indicates that for larger population sizes lower mutation rates perform better. The behavioral pattern of the algorithm persists on the Sphere function using an elite count 1. On the other hand, the range of better performing mutation rates shift to the left to cover the lower values, and the performance differences among different mutation rates for each row becomes more distinct and sharp relative to table (a). If we fix the mutation rate and we compare the preferred population sizes across different phases, we observe that in the later stages smaller population sizes perform better in general. AQ2

For the Rastrigin's function, we observe similar patterns observed for the Sphere function in phase 1, for both elite counts 0 and 1. Also in this case, in the initial phase of the evolutionary process higher mutation rates are preferable. The pattern observed on the population sizes (i.e., lower population sizes are better for later stages of the evolutionary process) also persists. Moreover, we observe the trade-off between the population sizes and mutation rates specifically in table (d) where higher mutation rates work best with lower population sizes, and lower mutation rates work best with higher population sizes. These observations support

Table 4. Optimal parameters (population size and mutation rate) defined using categorical concepts for different phases and elite count 0

		Sphere function	Rastirigin's function
	Pop.Size	Mutation rate	Mutation rate
Phase 1	Small	High	–
	Medium	Medium-High	Medium
	Large	Low-Medium-High	Low-Medium
Phase 2	Small	Medium	Low-Medium
	Medium	Low-Medium	Medium
	Large	Low	Medium-High
Phase 3	Small	Medium	Low-Medium
	Medium	Low	Low-Medium
	Large	Low	Medium-High

the literature on shrinking the mutation rate [7] and population size [41,42] during an evolutionary run and comply with the suggestions presented in Table 3.

Based on these initial experiments, we then define population sizes {"Small", "Medium", "Large"} as $\{10, 50, 200\}$, and mutation rates {"Low", "Low-Medium", "Medium","Medium-High","High"} as $\{[0.001, 0.002), [0.002, 0.01), [0.01, 0.05), [0.05, 0.2), [0.2, 0.5]\}$, respectively. We map the optimal population sizes and mutation rates to their categories for each function and elite count using the experimental results given in **SI.1**. As an example, we illustrate the mapping results in Table 4 for elite count 0. On the Rastrigin's function, none of the mutation rates produced a fitness improvement for small population size; this is shown as a "–" in the table.

Finally, we populate our ontology with the rules summarized in Table 4 associated to the function types (e.g. unimodal, multimodal, etc.) represented by the two chosen benchmark functions. For example, for unimodal functions with "Small" population size our ontology would suggests starting with "High" mutation rate in phase 1, and modifying it to "Medium" in phase 2 and 3. The suggested strategies are retrieved from the ontology using the SPARQL query language. Questions such as *"What is the mutation rate for unimodal functions when the evolutionary process is in maturation phase?"* are converted into a SPARQL query (see Query 1.1). The query returns the list of mutation rates that are stored in the ontology.

```
SELECT ?MutationRate
WHERE {
 ?strategy initializes ?MutationRate.
 ?strategy suggestedFor UnimodalFunction.
 ?strategy conditionedOn MaturationPhase
}
```

Query 1.1. An example SPARQL query

For the sake of completeness, we provide a detailed scheme of our ontology and a partial illustration of the ECO as Supporting Information **SI.2** and **SI.3**[2].

5 Experimental Validation

In this section, we use the strategies in the ECO in a parameter selection example to demonstrate the advantage of incorporating knowledge into the EA. We select three test functions that were not used in the experiments performed for populating the strategy instances represented in the ECO. These functions are the Schwefel's, Rosenbrock's, and Ackley's functions (F_2, F_6 and F_8 in [39]). The Schwefel's function is unimodal while the Rosenbrock's and Ackley's functions are multimodal. Thus, we test the strategies we derived and represented for unimodal and multimodal functions correspondingly. A list of tested strategies is given in Table 5. The strategies labeled as (1), (2) and (3) were retrieved from the ECO, with strategy (1) suggested for unimodal functions and strategies (2) and (3) suggested for multimodal functions. The rest of the strategies {(4), (5), (6)} are used as control strategies, selected arbitrarily as if no prior knowledge existed on the function types.

The numerical results are summarized in Table 6. Each column in the table shows the performance of a selected strategy on a selected test function. Strategies are labeled as in Table 5. For testing, we used elite count 1 for all the selected functions. We only compared the strategies that are retrieved from the ECO with the ones that are defined arbitrarily, i.e. we did not compare the performance of all strategies across different functions. As for the mutation rates, we considered the lowest value of the categories defined above. The rows labeled as 1, 1e+03, 1e+04 and 1e+05 indicate the generations (checkpoints) when we record the best fitness value. We present the results of 1st, 7th, 13th and 25th best fitness values, and the mean and standard deviations for 25 separate evolutionary runs using the same strategy. In the table, for each function and phase the boldface indicates the strategy showing the best performance (on average, across 25 runs).

Table 5. Selected strategies for testing

Phase	Strategy (1)		Strategy (2)		Strategy (3)	
	Pop.Size	Mut.Rate	Pop.Size	Mut.Rate	Pop.Size	Mut.Rate
1	Medium	Medium-High	Medium	Medium-High	Large	Medium
2	Medium	Low	Medium	Medium	Medium	Medium
3	Medium	Low	Medium	Medium	Small	Medium
Phase	Strategy (4)		Strategy (5)		Strategy (6)	
	Pop.Size	Mut.Rate	Pop.Size	Mut.Rate	Pop.Size	Mut.Rate
1	Medium	Medium-High	Medium	Medium	Medium	Medium
2	Medium	Medium-High	Medium	Medium	Medium	Low
3	Medium	Medium-High	Medium	Medium	Medium	Low

[2] Available online at: http://www.goo.gl/xSgVvv.

Table 6. The performance of different strategies

Function		F2	F2	F2	F2	F6	F6	F6	F6	F6	F8	F8	F8	F8	F8
Strategy		1	4	5	6	2	3	4	5	6	2	3	4	5	6
1	1st	7.88E+03	8.68E+03	6.66E+03	6.81E+03	9.35E+02	7.50E+02	5.71E+02	4.83E+02	5.89E+02	7.89E+03	6.80E+03	6.02E+03	9.38E+03	9.08E+03
	7th	1.00E+04	1.18E+04	1.03E+04	1.28E+04	1.35E+03	9.79E+02	1.15E+03	1.21E+03	1.18E+03	1.24E+04	1.03E+04	1.21E+04	1.26E+04	1.32E+04
	13th	1.30E+04	1.34E+04	1.43E+04	1.42E+04	1.62E+03	1.15E+03	1.34E+03	1.39E+03	1.47E+03	1.49E+04	1.12E+04	1.39E+04	1.44E+04	1.47E+04
	25th	1.98E+04	1.77E+04	2.09E+04	1.86E+04	1.96E+03	1.52E+03	2.18E+03	2.02E+03	2.05E+03	2.07E+04	1.44E+04	1.97E+04	2.01E+04	2.04E+04
	Mean	1.32E+04	1.34E+04	1.38E+04	1.40E+04	1.55E+03	1.14E+03	1.42E+03	1.40E+03	1.43E+03	1.43E+04	1.10E+04	1.34E+04	1.50E+04	1.46E+04
	Std.	3.41E+03	2.39E+03	4.16E+03	2.82E+03	2.64E+02	2.20E+02	4.23E+02	3.41E+02	3.53E+02	3.42E+03	1.91E+03	3.26E+03	3.02E+03	3.11E+03
1E3	1st	2.08E-02	7.08E-02	7.38E-03	4.27E-03	3.18E+00	6.62E-03	8.17E-02	1.49E-02	2.13E-02	1.02E+01	2.50E+00	3.53E+01	8.91E+00	8.45E+00
	7th	7.12E-02	1.87E-01	2.06E-02	2.14E-02	4.36E+00	1.21E-02	1.30E-01	3.22E-02	3.31E-02	1.85E+01	8.15E+01	7.16E+01	8.74E+02	9.41E+00
	13th	1.08E-01	3.41E-01	2.56E-01	1.48E-01	9.14E+00	1.59E-02	1.59E-01	5.39E-01	4.21E-02	3.49E+01	1.33E+02	1.31E+02	3.16E+03	1.13E+02
	25th	2.81E-01	7.71E-01	2.18E+03	1.89E+03	2.11E+01	3.07E-02	3.93E-01	1.62E+01	1.21E+01	6.17E+03	8.17E+03	8.07E+03	9.06E+05	9.22E+03
	Mean	**1.17E-01**	3.68E-01	2.94E+02	2.26E+02	1.18E+01	**1.76E-02**	1.80E-01	4.69E+00	5.30E-01	**9.06E+02**	1.11E+03	9.39E+02	9.96E+04	2.05E+03
	Std.	6.02E-02	2.06E-01	5.91E+02	4.81E+02	7.61E+00	7.19E-03	7.17E-02	6.00E+00	2.41E+00	1.56E+03	2.17E+03	1.91E+03	2.41E+05	3.29E+03
1E4	1st	1.17E-07	1.14E-03	1.57E-05	3.04E-07	1.50E-01	4.60E-04	5.59E-03	5.04E-04	6.18E-05	1.57E-02	3.02E-02	5.26E+00	4.85E+00	7.05E-02
	7th	3.62E-07	3.34E-03	4.36E-05	5.11E-07	2.81E-01	8.74E-04	9.20E-03	1.06E-03	1.17E-04	4.61E+00	5.88E+00	7.13E+00	5.65E+00	1.76E-01
	13th	5.56E-07	5.24E-03	5.90E-05	7.63E-07	4.83E-01	1.21E-03	1.34E-02	1.39E-03	1.76E-04	5.37E+00	3.50E+01	1.06E+01	6.22E+00	8.41E-01
	25th	4.24E-06	2.01E-02	2.54E-04	6.34E-05	2.11E+01	2.24E-03	2.57E-02	2.48E-03	1.02E+01	2.60E+01	9.23E+02	1.20E+02	1.01E+03	7.03E+03
	Mean	**8.05E-07**	5.64E-03	8.25E-05	4.38E-06	6.86E+00	**1.23E-03**	1.37E-02	1.45E-03	4.08E-01	**5.70E+00**	1.55E+02	2.63E+01	1.74E+02	1.15E+03
	Std.	8.23E-07	3.74E-03	5.87E-05	1.29E-05	9.67E+00	4.58E-04	5.14E-03	5.14E-04	2.04E+00	4.78E+00	2.54E+02	3.44E+01	3.20E+02	2.43E+03
1E5	1st	3.68E-09	1.37E-05	1.53E-07	3.03E-09	1.68E-01	3.04E-04	5.80E-04	7.18E-05	6.61E-06	1.41E-02	1.26E-02	7.99E-01	1.30E+00	3.69E-03
	7th	8.47E-09	3.06E-05	4.19E-07	7.34E-09	2.91E-01	4.38E-04	1.04E-03	8.82E-05	1.16E-05	2.97E+00	3.94E+00	3.62E+00	3.74E+00	5.65E-03
	13th	9.52E-09	7.01E-05	6.53E-07	9.14E-09	3.65E-01	5.95E-04	1.31E-03	1.18E-04	1.55E-05	3.34E+00	4.48E+00	6.67E+00	4.01E+00	8.74E-03
	25th	2.56E-08	2.05E-04	2.05E-06	2.82E-08	2.13E+01	1.26E-03	2.96E-03	2.09E-04	1.02E+01	6.62E+00	2.46E+02	8.52E+01	5.25E+01	2.80E+02
	Mean	1.01E-08	7.64E-05	8.07E-07	**9.49E-09**	6.14E+00	6.53E-04	1.36E-03	1.25E-04	4.08E-01	**3.14E+00**	2.65E+01	1.39E+01	8.67E+00	1.73E+01
	Std.	4.30E-09	5.09E-05	5.01E-07	4.98E-09	9.49E+00	2.74E-04	4.87E-04	4.07E-05	2.04E+00	1.38E+00	5.03E+01	2.09E+01	1.33E+01	5.76E+01

We omit the boldface for the 1st evaluation, due to the random initialization of the evolutionary algorithm.

The first four columns show the result on the unimodal function (Schwefel's function [39]). On average, the strategy suggested by the ECO performs better than the arbitrarily selected strategies labeled as (4), (5) and (6) in the first two phases. In the last phase, the strategy (6) performs better than the strategy (1), on average. However, this fact may be ignored since two of these strategies implement the same mutation rate for the final phase. We should note that the strategy (4) is often recommended by the literature (see (1) in Table 3); however, in our experiments we observe that the strategy (1) performs better than the strategy (4) (see (4) in Table 3).

As for for multimodal functions, the ECO suggests the strategies labeled as (2) and (3). Therefore, we tested the strategies (2) and (3) on F_6 and F_8. On F_6, the strategy (3) shows the best performance. The empirical results point out that for multimodal functions, population reduction in further phases of the evolutionary process is a good strategy. However, it is interesting to see the contradiction where the strategy (5) performs better than the strategy (3). When we take a closer look, the strategy (3) performs better than (5) in phase 1 and 2. On the other hand, the results show that during the last phase a small population size caused the strategy (3) to slow down in terms of fitness improvement; whereas, a medium population size caused (5) to catch up. This suggests that in the last phase of strategy (3), a slightly higher population size might be preferable (e.g. as a last attempt to introduce diversity). Finally, from the comparison among the strategies evaluated on F_8 we can observe that the strategy (2) performs significantly better in all three phases.

6 Conclusions

In this work, we presented the ECO, an ontology designed to represent domain and algorithmic knowledge in evolutionary algorithms. As such, the ECO includes concepts relevant to evolutionary algorithms, problem domains and strategies. The strategies are defined as general guidelines in EC that were extracted from the literature. To populate the ontology, we collected the knowledge available in fifty research papers from the specialized EC-related literature. We then described some possible uses of the ontology in parameter selection, Interactive Evolutionary Computation, software design and education. We finally demonstrated numerically the performance of an evolutionary algorithm with parameters driven by the ECO on three different functions. Our results show that the incorporation of knowledge (coming from different sources, i.e. literature and/or empirical experiments) into evolutionary algorithms can improve their performance consistently. Furthermore, the use of a common ontology guarantees a systematic way of collecting, representing and sharing this knowledge among researchers, among algorithms, and across different application domains.

The main limitation of the ECO is the uncertainty that characterizes knowledge in EC. While parsing the literature, we noted that most of the strategy

statements that we found use non-specific wordings such as "high mutation rate facilitates more exploration". Clearly, the exact parameter values that should be used are not defined. Here, to define the range of these values in the ECO, we performed preliminary experiments on a unimodal (Sphere) and a multimodal (Rastrigin) function and we observed, for instance, that some mutation rates are expected to perform better depending on the population size, and vice versa. In this work, we assigned a crisp value for each category based on the experimental data. However, in future works we aim to include representation methods that can deal with gradual ranges for categories, such as fuzzy logic [33]. Another source of uncertainty is due to the fact that multiple strategies (characterized by different combinations of operators and parameters) may be equally efficient in an optimization scenario. Currently, querying over the ECO yields a list of suitable strategies. These strategies are conditional on the type of problem and other properties of the evolutionary process, e.g. its phase. However, the list of suggested strategies should be aggregated to recommend one single strategy. This aggregation is *probabilistic* in nature, because in general it summarizes conflicting and vague strategies available in the knowledge base. Moreover, different strategies may have different levels of confidence, depending on how well they perform in specific settings. Future research will focus on specific methods for providing aggregated probabilistic results into our query processing.

Another limitation of the present work obviously consists in the limited set of unimodal and multimodal functions we used for testing. However, these limited experiments were shown here as a simple proof-of-concept. In order to derive statistically rigorous conclusions and generalize the experimental results, so to transfer these generalizations to a broader set of optimization functions, we will need to perform more experiments on a much larger set of benchmark functions.

In future works, we also aim to extend our strategies to cover different kinds of evolutionary operators and different evolutionary algorithms. We believe that a similar experimental analysis would improve our current understanding of the effects of different evolutionary operators and parameter settings.

Acknowledgments. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 665347.

References

1. Gruber, T.R.: A translation approach to portable ontology specifications. Knowl. Acquis. **5**(2), 199–220 (1993)
2. Sowa, J.F.: Principles of Semantic Networks: Explorations in the Representation of Knowledge. Morgan Kaufmann, San Mateo (2014)
3. Studer, R., Benjamins, V.R., Fensel, D.: Knowledge engineering: principles and methods. Data Knowl. Eng. **25**(1), 161–197 (1998)
4. Riaño, D., Real, F., López-Vallverdú, J.A., Campana, F., Ercolani, S., Mecocci, P., Annicchiarico, R., Caltagirone, C.: An ontology-based personalization of healthcare knowledge to support clinical decisions for chronically ill patients. J. Biomed. Inf. **45**(3), 429–446 (2012)

5. Liao, S.H.: Expert system methodologies and applications-a decade review from 1995 to 2004. Expert Syst. Appl. **28**(1), 93–103 (2005)
6. Jin, Y.: Knowledge Incorporation in Evolutionary Computation, vol. 167. Springer, Heidelberg (2013)
7. Bonissone, P.P., Subbu, R., Eklund, N., Kiehl, T.R.: Evolutionary algorithms + domain knowledge = real-world evolutionary computation. IEEE Trans. Evol. Comput. **10**(3), 256–280 (2006)
8. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter control in evolutionary algorithms. IEEE Trans. Evol. Comput. **3**(2), 124–141 (1999)
9. Takagi, H.: Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation. Proc. IEEE **89**(9), 1275–1296 (2001)
10. Wagner, S., Kronberger, G., Beham, A., Kommenda, M., Scheibenpflug, A., Pitzer, E., Vonolfen, S., Kofler, M., Winkler, S., Dorfer, V., Affenzeller, M.: Architecture and design of the heuristiclab optimization environment. In: Klempous, R., Nikodem, J., Jacak, W., Chaczko, Z. (eds.) Advanced Methods and Applications in Computational Intelligence, vol. 6, pp. 197–261. Springer International Publishing, Heidelberg (2014) AQ3
11. Kaur, G., Chaudhary, D.: Evolutionary computation ontology: e-learning system. In: 2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions), pp. 1–6, September 2015
12. Roussey, C., Pinet, F., Kang, M.A., Corcho, O.: An introduction to ontologies and ontology engineering. In: Roussey, C., Pinet, F., Kang, M.A., Corcho, O. (eds.) Ontologies in Urban Development Projects, vol. 1, pp. 9–38. Springer, London (2011)
13. Noy, N.F., McGuinness, D.L., et al.: Ontology development 101: a guide to creating your first ontology (2001)
14. Davis, R., Shrobe, H., Szolovits, P.: What is a knowledge representation? AI Mag. **14**(1), 17 (1993)
15. Pan, J.Z.: Resource description framework. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies, pp. 71–90. Springer, Heidelberg (2009)
16. McGuinness, D.L., Van Harmelen, F., et al.: OWL web ontology language overview. W3C recommendation **10**(10) (2004)
17. The World Wide Web Consortium (W3C) (2016). Accessed 14 Aug 2016 AQ4
18. Quilitz, B., Leser, U.: Querying distributed RDF data sources with SPARQL. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 524–538. Springer, Heidelberg (2008). doi:10.1007/978-3-540-68234-9_39
19. Črepinšek, M., Liu, S.H., Mernik, M.: Exploration and exploitation in evolutionary algorithms: a survey. ACM Comput. Surv. **45**(3), 35:1–35:33 (2013)
20. Johnson, J., Louis, S.J.: Case-initialized genetic algorithms for knowledge extraction and incorporation. In: Jin, Y. (ed.) Knowledge Incorporation in Evolutionary Computation, pp. 57–79. Springer, Heidelberg (2005)
21. De Jong, K.A., Spears, W.M.: A formal analysis of the role of multi-point crossover in genetic algorithms. Ann. Math. Artif. Intell. **5**(1), 1–26 (1992)
22. Falco, I.D., Cioppa, A.D., Tarantino, E.: Mutation-based genetic algorithm: performance evaluation. Appl. Soft Comput. **1**(4), 285–299 (2002)
23. Bäck, T., Eiben, A.E., van der Vaart, N.A.: An empirical study on GAs "without parameters". In: International Conference on Parallel Problem Solving from Nature, pp. 315–324. Springer, London (2000) AQ5

24. Yeguas, E., Luzón, M., Pavón, R., Laza, R., Arroyo, G., Díaz, F.: Automatic parameter tuning for evolutionary algorithms using a bayesian case-based reasoning system. Appl. Soft Comput. **18**, 185–195 (2014)
25. Picek, S., Jakobovic, D.: From fitness landscape to crossover operator choice. In: Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, pp. 815–822. ACM (2014)
26. Asmus, J., Borchmann, D., Sbalzarini, I.F., Walther, D.: Towards an FCA-based recommender system for black-box optimization. In: Workshop Notes, p. 35 (2014)
27. Czarn, A., MacNish, C., Vijayan, K., Turlach, B., Gupta, R.: Statistical exploratory analysis of genetic algorithms. IEEE Trans. Evol. Comput. **8**(4), 405–421 (2004)
28. Eiben, A., Smit, S.: Parameter tuning for configuring and analyzing evolutionary algorithms. Swarm Evol. Comput. **1**(1), 19–31 (2011)
29. Neumüller, C., Wagner, S., Kronberger, G., Affenzeller, M.: Parameter meta-optimization of metaheuristic optimization algorithms. In: Moreno-Díaz, R., Pichler, F., Quesada-Arencibia, A. (eds.) EUROCAST 2011. LNCS, vol. 6927, pp. 367–374. Springer, Heidelberg (2012). doi:10.1007/978-3-642-27549-4_47
30. Inspyred: Bio-inspired Algorithms in Python (2016). Accessed 11 Nov 2016
31. Matlab Genetic Algorithm Toolbox (2016). Accessed 11 Nov 2016
32. Finck, S., Hansen, N., Ros, R., Auger, A.: Real-parameter black-box optimization benchmarking 2009: presentation of the noiseless functions. Technical report, Citeseer (2010)
33. Zhang, J., Chung, H.S., Lo, A.W., Hu, B.: Fuzzy knowledge incorporation in crossover and mutation. In: Jin, Y. (ed.) Knowledge Incorporation in Evolutionary Computation, vol. 167, pp. 123–143. Springer, Heidelberg (2005)
34. Bäck, T., Schwefel, H.P.: An overview of evolutionary algorithms for parameter optimization. Evol. Comput. **1**(1), 1–23 (1993)
35. He, J., Kang, L.: On the convergence rates of genetic algorithms. Theor. Comput. Sci. **229**(1), 23–39 (1999)
36. Mühlenbein, H.: How genetic algorithms really work: mutation and hillclimbing. PPSN **92**, 15–25 (1992)
37. Doerr, B., Sudholt, D., Witt, C.: When do evolutionary algorithms optimize separable functions in parallel?. In: Proceedings of the Twelfth Workshop on Foundations of Genetic Algorithms XII, FOGA XII 2013, pp. 51–64. ACM, New York (2013)
38. Back, T.: Selective pressure in evolutionary algorithms: a characterization of selection mechanisms. In: Proceedings of the First IEEE Conference on Evolutionary Computation, 1994, IEEE World Congress on Computational Intelligence, vol. 1, pp. 57–62, June 1994
39. Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.P., Auger, A., Tiwari, S.: Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization. Technical Report, Nanyang Technological University, Singapore, AND KanGAL Report 2005005, IIT Kanpur, India, May 2005
40. Gates, G.H., Merkle, L.D., Lamont, G.B., Pachter, R.: Simple genetic algorithm parameter selection for protein structure prediction. In: IEEE International Conference on Evolutionary Computation, vol. 2, pp. 620–624. IEEE (1995)
41. Iacca, G., Mallipeddi, R., Mininno, E., Neri, F., Suganthan, P.N.: Super-fit and population size reduction in compact differential evolution. In: 2011 IEEE Workshop on Memetic Computing (MC), pp. 1–8. IEEE (2011)
42. Tanabe, R., Fukunaga, A.S.: Improving the search performance of shade using linear population size reduction. In: IEEE Congress on Evolutionary Computation (CEC), pp. 1658–1665. IEEE (2014)