Darwin or Lamarck? Future Challenges in Evolutionary Algorithms for Knowledge Discovery and Data Mining

Katharina Holzinger[1], Vasile Palade[2,3], Raul Rabadan[4], and Andreas Holzinger[1]

[1] Medical University Graz, Institute for Medical Informatics, Statistics and Documentation
Research Unit HCI, Graz, Austria
{k.holzinger,a.holzinger}@hci4all.at
[2] Coventry University, Faculty of Engineering and Computing, Coventry, UK
vasile.palade@coventry.ac.uk
[3] Oxford University, Department of Computer Science, Oxford, UK
[4] Columbia University, Department of Systems Biology and
Department of Biomedical Informatics, New York, US
rabadan@dbmi.columbia.edu

Abstract. Evolutionary Algorithms (EAs) are a fascinating branch of computational intelligence with much potential for use in many application areas. The fundamental principle of EAs is to use ideas inspired by the biological mechanisms observed in nature, such as selection and genetic changes, to find the best solution for a given optimization problem. Generally, EAs use iterative processes, by growing a population of solutions selected in a guided random search and using parallel processing, in order to achieve a desired result. Such population based approaches, for example particle swarm and ant colony optimization (inspired from biology), are among the most popular metaheuristic methods being used in machine learning, along with others such as the simulated annealing (inspired from thermodynamics). In this paper, we provide a short survey on the state-of-the-art of EAs, beginning with some background on the theory of evolution and contrasting the original ideas of Darwin and Lamarck; we then continue with a discussion on the analogy between biological and computational sciences, and briefly describe some fundamentals of EAs, including the Genetic Algorithms, Genetic Programming, Evolution Strategies, Swarm Intelligence Algorithms (i.e., Particle Swarm Optimization, Ant Colony Optimization, Bacteria Foraging Algorithms, Bees Algorithm, Invasive Weed Optimization), Memetic Search, Differential Evolution Search, Artificial Immune Systems, Gravitational Search Algorithm, Intelligent Water Drops Algorithm. We conclude with a short description of the usefulness of EAs for Knowledge Discovery and Data Mining tasks and present some open problems and challenges to further stimulate research.

Keywords: Evolutionary Algorithms, Optimization, Nature inspired computing, Knowledge Discovery, Data Mining.

1 Introduction

The original idea behind the EAs goes back to the early days of computer science [1] and started with some initial thoughts on **adaptive systems** introduced by John H. Holland [2]. Since the 1980ies, EAs have been used to address optimization problems due to their robustness and flexibility, especially in fields where traditional greedy algorithms did not provide satisfactory results. A typical example can be found in [3] in finding near-minimal phylogenetic trees from protein sequence data; a good Web-based tool for the display, manipulation and annotation of such phylogenetic trees is described in [4].

Traditional evolutionary paradigms are usually divided into two groups according to the principle invoked to explain the biological change: While Lamarck (see section 3.3) proposed the inheritance of acquired characteristics; Darwin (see section 3.2) underlines the role of selection on *random genetic variation.* A Lamarckian Algorithm, for example, would have nothing to do with selection.

Rather than referring to Darwin's original work [5], computer scientists use terms like “natural selection theory”, “natural genetics”, “the genetic theory of natural selection”, etc., because EAs are inspired from the **selection** and **genetic principles observed in nature.** However, EAs do not prove anything with respect to the evolution in nature presumed in the original work by Darwin. So, a good question is why are we speaking then of “evolutionary algorithms”?

One aim of this paper is to shortly introduce to computer scientists the original work of Darwin, and to contrast these ideas to an earlier evolution theory of Lamarck, which might be even less familiar to the computer science community, but which has started to gain some popularity among researchers in evolutionary computing in recent years. For example, a search in the Web of Science repository, with the words “evolutionary algorithms” in the title, returns 1,886 results (as of February, 19, 2014). The oldest contribution is a paper in Lecture Notes in Economics and Mathematical Systems dating back to 1991 [6], which, interestingly, got no citation so far; the newest is a contribution in the April 2014 issue of the Journal of Industrial Management Optimization [7]; and the paper with the highest number of citations is in the Nov. 1999 issue of the IEEE Transactions on Evolutionary Computation [8].

This paper is organized as follows: First, we define the key terms to ensure mutual understanding. Then, we contrast the work of Darwin and Lamarck and focus on some computational aspects, because it will be necessary to define a new terminology for the Lamarckian version of an evolutionary algorithm. In the central part of the paper, we describe the state-of-the-art in EAs, where we shortly describe the main classes of current EA approaches. We finally stimulate a discussion on the use of EAs for Knowledge Discovery and Data Mining tasks, by presenting current challenges in the area and some new “hot ideas” that may inspire future research.

2 Glossary and Key Terms

Classification: Computational learning process to identify the class or category (from a set of possible classes) to which a new observation belongs, on basis of a training set containing observations whose category memberships are known.

Clustering: Grouping a set of objects in such a way that objects in the same group (or cluster) are more similar to each other than to those in other groups (clusters).

Epigenetics: is the study of heritable changes in genes, not caused by changes in the DNA. Whereas genetics is based on changes to the DNA sequence (the genotype), the changes in gene expression or cellular phenotype of epigenetics have other causes, therefore the prefix epi- (Greek: επί- outside) [9], [10].

Evolution: The change of inherited characteristics of biological populations over successive generations.

Evolutionary Computation (EC): Subfield of computational intelligence that involves mainly optimization with a metaheuristic or stochastic character inspired from biological processes observed in nature.

Evolutionary Algorithm (EA): An algorithm that uses mechanisms inspired by biological evolution, such as reproduction, mutation, recombination, and selection.

Genetic Algorithm (GA): Search heuristic that mimics the processes from natural genetics to generate useful solutions in optimization and search problems.

Genetic Programming (GP): Set of genetic operations and a fitness function to measure how well a computer program has performed a task, and used to optimize a population of computer programs.

Knowledge Discovery (KDD): Exploratory analysis and modeling of data and the organized process of identifying valid, novel, useful and understandable patterns from data sets.

Machine Learning: The discipline concerned with methods and systems that can built and used to learn from data; a subfield of computer science.

Multi-Objective Optimization: aka Pareto optimization, involves more objective functions to be optimized simultaneously.

Optimization: is the selection of a best solution to a given problem (with regard to some criteria) from a set of available alternatives.

Phylogenetic tree: is a branching tree diagram displaying the evolutionary relationships among biological species [11], [12].

3 Background

3.1 Basic Principles

The fundamental principle of evolutionary algorithms is to use ideas inspired by selection and genetic mechanisms observed in nature to find the best solution for a given optimization problem. Consequently, EAs include a class of optimization techniques that imitate natural selection principles and social behavior in nature, and embrace genetic algorithms, swarm optimization algorithms, ant colony algorithms, bacteria foraging algorithms, to name only a few.

Today, EAs field has grown to represent a big branch of computational intelligence and machine learning research [13]. Evolutionary methods are used in many different research fields such as medicine [14], genetics [15], or engineering [16], and there are nearly countless application areas of EAs, due to their **adaptive nature** and ability in solving difficult optimization problems [17], [18], [19].

EAs scale well into high dimensions, are robust to noise and are in general a good choice for problems where traditional methods do not provide a solid foundation. However, due to the global search process of evolutionary methods, an optimal solution within finite time cannot be guaranteed. Before we continue with recent state-of-the-art on EAs, we will shortly look back into history first.

3.2 Darwin's Theory

The theory of evolution, which Charles Darwin (1809–1882) presented in 1859 in his book "On the origin of species" [5] can be summarized with a simple algorithm: Mutation – variability – competition – selection – inheritance.

Fitness: A key concept in Darwinian evolution is the idea of fitness, or the capability of organisms to survive and reproduce. Genomic variations in the form of mutation or recombination could cause changes in fitness. Fitter organisms are positively selected and their genomic information is inherited by their descendants. The descendants inherit the selected variations and the phenotypic traits associated with them. The phenotypic variability is then caused by inherited mutations in the DNA sequence. Similar to individuals, there is also a competition among the alleles, for the presence in the DNA of the population. Alleles are the possible genetic variations of a gene that are present in the population. Depending on how successful the carriers of this specific allele are, after several generations it will either be fixed or die out – therefore, disappear from the gene pool. However, the success of an allele carrier only depends on the allele, if it occurs phenotypically in morphological, physiological or ethological terms, therefore, has an influence on appearance, body function or behavior of the organism in question. Consequently, in Darwinism, the evolution is only a secondary process. The organisms do not actively adapt to their environment, but out of a variety of different characteristics and manifestations, the ones that are selected are those that give their bearers an advantage in survival or reproduction. As has already been emphasized above, what a central role the selection plays in Darwinism, it is essential to look at the different types of selection:

Natural Selection: This is the selection by biotic or abiotic environmental factors. Abiotic factors for example include climate, biotic factors include pressure from predators. Darwin used the term as opposed to artificial selection and emphasized that natural selection must end with the death or incapacity of reproduction of the organism. *"(…) for of the many individuals of any species which are periodically born, but a small number can survive. I have called this principle, by which each slight variation, if useful, is preserved, by the term of Natural selection (...)"* [5].

Sexual Selection: In modern evolutionary biology, sexual selection is counted among natural selection. Darwin himself described sexual selection as "less rigorous" than natural selection because it does not decide over life and death, but on the number of offspring, which is only indirectly crucial for the survival or success of a species. Sexual selection is the competition within a species to reproduce, hence, the efforts of the males to impress the females and the males fighting each other for the right to mate. The structures and trades resulting from these processes do not always coincide with natural selection, but often are even contradictory to it. Well known examples of such structures are the tail feathers of a male peacock and the antlers of a male deer. As for the survival of a species, however natural selection is the stronger force.

Artificial Selection: Artificial selection occurs when humans select animals with desired characteristics and breed them. The many breeds of dogs and horses are a result of artificial selection.

Gene Selection: It is of great importance in modern evolutionary research, as individual alleles compete for the maximum frequency in the population. In modern evolutionary biology, gene selection has replaced the selection of individuals as postulated in the theory of classical Darwinism, where individuals are selected because of phenotypic characteristics.

Stabilizing selection: eliminates individuals with an extreme value of a specific characteristic, for example size. A possible scenario would be a pond with fish in different sizes, where the small fish are prayed on by birds and the large fish get caught by fishermen. Therefore medium sized fish will become a majority within the pond.

Distributive Selection: This is the exact opposite of stabilizing selection, because it eliminates individuals with mediocre value of a certain characteristic. If we return to our exemplary pond of fish, this time the medium sized fish will get prayed on by bigger birds. On the other hand the extreme fish – the small and the big – will survive.

Directional Selection: This type of selection is particularly interesting and aimed at one side of the extremes and the mediocre; e.g., in our exemplary pond directional selection, if an otter preyed on small and medium sized fish. Thus, the chances of survival increase for the fish with their size. The bigger the safer. Under such a kind of selective pressure this species of fish will gradually increase in size.

Hard selection: This refers to selective pressure at which an individual is eliminated if it does not reach a certain value, such as size or color. For example, all fish bigger than 30 cm will be caught in the nets of fishermen.

Soft selection: This does not use an absolute value, but a ratio. In our fish example soft selection would mean the biggest fish will be caught, no matter how big they are exactly.

3.3 Lamarck's Theory

However, Darwinism was not the only theory of evolution of the time. In addition to the catastrophism of Georges Cuvier (1769–1832), there is also Lamarckism, which states, unlike Darwinism, that selection is **not** the driving force of evolution, but the inheritance of acquired characteristics or inherited **"effort"** of the organisms themselves. Jean-Baptiste de Lamarck (1744–1829) assumed that appropriate characteristics arise from the desire of the organisms to achieve them (strive for perfection).

Unlike Darwinism, where evolution is only a result of competition and selection, in Lamarckism the organisms themselves control evolution. This is accomplished through practice, training, and the frequent use of specific organs. Lesser used organs, however, wither with time. The most popular example to illustrate the idea

Lamarckism is the evolution of the giraffe's neck: The giraffe is striving to reach the highest leaves, and stretched her neck. This acquired trait is inherited by her descendants, who again stretch their necks. However, this very simple explanation of a deliberate adaptation results in some questions from modern biological perspective: Why should organisms have the desire to change? Can new structures be build trough training? By what means is it decided which adaptions will be passed on? Why does not an amputated leg get inherited? In biology, Lamarckism would be possible if there was a mechanism that translates phenotypic changes into the sequence of the responsible gene. However, Lamarckism should not be entirely rejected, as it can provide some answers, especially in modern genetics and medicine. In epigenetics – which very early dealt with questions of evolution [20],[21], it was found that there are special traits which can be inherited without being part of the genetic code; That would, for example, explain a possible higher function of the thumb in the upcoming post-millennial younger generations ("Net Gen" [22]) due to frequent use of text messaging on mobile phones, which is being allegedly claimed by some people, but still to be confirmed. The possibility that acquired behavior or marks can be passed from parents to children is in serious debate and the advent of epigenetics is hailed as a profound shift in our understanding of inheritance, i.e. that genes also have a kind of "memory" [23], [24], epigenetics being an upcoming hype in medical research [25], with a very recent example in cancer research found here [26].

4 Brief Survey on Evolutionary Algorithms

4.1 Why Evolutionary Algorithms?

Due to the adaptive and robust nature of performing a global instead of a local search for solutions in the search space, which improves their handling of interactions between attributes [27], methods based on evolutionary algorithms are being used in a wide array of different research fields. They are mostly used for traditional KDD tasks, such as clustering and classification as well as for optimization. Another benefit of evolutionary methods is the possibility of using them for multi-objective optimization, making them well suited for many real-world use-cases where simultaneous optimization of several objectives is of importance [28]. There are many different algorithms in the universe of evolutionary methods, but the most prevalent are genetic algorithms and genetic programming which we describe in section 4.4.

4.2 Biological Sciences versus Computational Sciences

Darwin explains in his book *"The Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*" [5] the diversity and complexity of living organisms: Beneficial traits resulting from random variation are favored by natural selection, i.e. individuals with beneficial traits have better chances to survive, procreate and multiply, which may also be captured by the expression differential reproduction. In order to understand evolutionary algorithms, some basic notions are important, which will highlight the applicability of biological principles to computer science. Good resources for further details are: [13], which is also available in German [29], and [30], [31], [32], [33].

Evolutionary algorithms operate on a search space S, where S denotes a given set. Points are assigned via an objective function f. In the context of evolutionary algorithms, this is usually called **fitness function** $f: S \rightarrow R$, where R is the set of arbitrary possible fitness values, and the evolutionary algorithm operates on a collection of points from S, called a population P. Each member of the population (points in the search space) is called individual. A number $\mu \in \mathbb{N}$ is used to denote the size of the population, i.e. $\mu = |P|$.

A population is a multiset over S, i.e., it may contain multiple copies of individuals. Since the population changes from generation to generation, we denote the population at the t-th generation as P_t. Choosing the first population, P_0, at the beginning is called initialization.

Table 1. Biology vs Computing: basic evolutionary notions in the biological vs. computational sciences; compare with Kruse et al. (2013) [13]

NOTION	BIOLOGICAL UNIVERSE	COMPUTATIONAL UNIVERSE
Allele	"Value" of a gene	Value of a information object
Chromosome	DNA, protein, and RNA sequence in cells (describes the "construction plan" and traits of an individual)	Sequence of information objects (describes the "construction plan" and "traits of an individual")
Fitness	Aptitude/conformity of a living organism, determines chances of survival and reproduction	Aptitude/quality of a solution candidate, determines chances of survival and reproduction
Gene	Part of a Chromosome, as a fundamental unit of inheritance, which determines a (partial) characteristic of an individual	Information object, e.g. a bit, a character, number etc., fundamental unit of inheritance, which determines a (partial) characteristic of an individual
Generation	Population at a point in time	Population at a point in time
Genotype	Genetic constitution of a living organism	Encoding of a solution candidate
Individual	Living organism	Solution candidate
Locus	position of a gene, at each position in a Chromosome there is exactly one gene	position of an information object, at each position in a Chromosome there is exactly one gene
Phenotype	Physical appearance of a living organism	Implementation or application of a solution candidate
Population	Set of living organisms	Bag or multi-set of Chromosomes
Reproduction	Creating offspring of one or multiple (usually two) parent organisms	Creating (child) chromosomes from one or multiple (parent) chromosomes

For each member x of the population, its fitness $f(x)$ is computed and stored. The first step in each generation is to select some individuals from the population that will be used to create new points in the search space. These individuals are referred to as parents. This process is called selection for reproduction. Often this selection is done fitness-based, i.e., the chances of individuals to become parents increase with their fitness. Then some random variation is applied to the parents, where small changes are more likely than large changes [30].

4.3 Foundations of Evolutionary Algorithms

As already mentioned, the basic idea of an evolutionary algorithm is to apply evolutionary principles to generate increasingly better solution candidates in order to solve an optimization problem. This may be achieved by evolving a population of solution candidates by random variation and fitness-based selection of the next generation. According to [13], an EA requires the following building blocks:

- an encoding for the solution candidates,
- a method to create an initial population,
- a fitness function to evaluate the individual solutions (chromosomes),
- a selection method on the basis of the fitness function,
- a set of genetic operators to modify chromosomes,
- a termination criterion for the search, and
- values for various parameters.

The (natural) selection process of biological evolution can be simulated by a method for selecting candidate solutions according to their fitness, i.e., to select the parents of offspring that are transferred to the next generation. Such a selection method may simply transform the fitness values into a selection probability, such that better individuals have higher chances of getting chosen for the next generation. The random variation of chromosomes can be simulated by so-called genetic operators that modify and recombine chromosomes, for example, mutation, which randomly changes individual genes, and crossover, which exchanges parts of the chromosomes of parent individuals to produce offspring. While biological evolution is unbounded, we need a criterion to decide when to stop the process in order to retrieve a final solution. Such a criterion may be, for example, that the algorithm is terminated (1) after a user-specified number of generations have been created, (2) there has been no improvement (of the best solution candidate) for a user-specified number of generations, or (3) a user-specified minimum solution quality has been obtained. To complete the specification of an evolutionary algorithm, we have to choose the values of several parameters, which include, for example, the size of the population to evolve, the fraction of individuals that is chosen from each population to produce offsprings, the probability of a mutation occurring in an individual etc. [13]. The general procedure of such an evolutionary algorithm may look as presented in table 2:

Table 2. General Scheme of an Evolutionary Algorithm

```
procedure evolutionary algorithm;
begin
  t ← 0; (* initialize the generation counter *)
  initialize pop(t);(* create the initial population *)
  evaluate pop(t); (* and evaluate it (compute fitness) *)
  while not termination criterion do (* loop until termination *)
        t ← t+1;  (* count the created generation *)
                  select pop(t)from pop(t-1);(*select individuals
                  based on fitness*)
        alter pop(t);  (* apply genetic operators *)
        evaluate pop(t); (* evaluate the new population *)
  end
end
```

4.4 Types of Evolutionary Algorithms

4.4.1 Genetic Algorithms (GA)

Genetic algorithms (GAs) are a machine learning method inspired from genetic and selection mechanisms found in nature [34], which conduct a randomized and parallel search for solutions that optimize a predefined fitness function [35].

In nature, the genetic information is defined in a quaternary code, based on the four nucleotides **A**denine, **C**ytosine, **G**uanine and **T**hymine, stringed together in a DNA sequence, which forms the basis of the genetic code [36]. In transferring this structure to computer science, it seems natural to base all encodings on the ultimately binary structure of information in a computer. That is, we use chromosomes that are bit strings, to encode problem solutions, and exactly this is the distinctive feature of genetic algorithms [37]. The algorithm performs a global search in the space of solution candidates, where the space consists of data vectors. The first step is to initialize the solution candidate space with randomly generated individual solutions. At each iteration, the available candidates are mutated or crossed with other solutions in order to create new candidates. At the end of each iteration, every individual solution candidate is evaluated using a predefined fitness function. Consequently, the fitness function is the core part of every evolutionary algorithm, and designed to find out which solutions are the best fits for the problem. Once each individual has been evaluated, the least fit candidates get dismissed, leaving only the best available solutions in the population. This is Darwin's principle of survival of the fittest in solving computing problems. The loop of iterations is repeated until a predefined stopping criterion has been reached. Stopping criteria can vary in their definition from just the number of iterations to go through to a certain threshold of fitness value that has to be reached within the solution space. For more details on GAs refer to [38], [39], [40].

4.4.2 Genetic Programming (GP)

Genetic programming (GP) differs from genetic algorithms mainly in the form of the input and output values the algorithm needs and produces [41]. In the case of GP, the values are not simple data points, but parts of functions or programs. The goal is to find a procedure which solves the given problem in the most efficient way. The algorithm itself works in the same way as described in the section above, where initially there is a randomly generated space of candidate solutions (random programs in this case), which are evolved using mutation and crossover processes, generating new program trees. The end result is supposed to be a function or a program, which can be used to solve a specific type of problem. An example of linear genetic programming applied to medical classification problems from a benchmark database compared with results obtained by neural networks can be found in [42].

4.4.3 Evolution Strategies (ES)

Evolution Strategies (ES) are a stochastic approach to numerical optimization that shows good optimization performance in general and which goes attempt to imitate principles of organic evolution in the field of parameter optimization [43]. In order to improve the "self-adaptive" property of strategy parameters, Ohkura et al. (2001) [44] proposed an extended ES called **Robust Evolution Strategy (RES),** which has redundant neutral strategy parameters and which adopts new mutation mechanisms in order to utilize selectively neutral mutations to improve the adaptability of strategy parameters, a similar approach was proposed in [45] and more details can be found in [46], [47], [48].

4.4.4 Swarm Intelligence (SI)

Swarm intelligence (SI) studies the collective behavior of self-organized systems composed of many individuals interacting locally with each other and with their environment, using decentralized control to achieve their goals. Swarm-based systems have been developed in response to the observed success and efficiency of such swarms in nature [49].

Approaches that came out as a result of studying the **collective behavior** of populations of “simple agents”, i.e. individuals with limited abilities without central control, can be employed in many different areas. They have been inspired by the behavior of certain species of animals, especially social insects (ants, bees) and animals that live and search for food in swarms, flocks, herds or packs (fish, birds, deer, wolves, rabbits etc.) and also bacteria. Such swarms can find the shortest paths to food sources, they can build complex nests like bee hives, hunt for prey (for example, packs of wolves), and protect themselves against predators [13]. In joint efforts, these animals are often able to solve complex problems – demonstrating collective intelligence [50]. This is a recent and important research area in computer science [51] which can be applied for many purposes, a prominent example being the NASA crater finding [52] using human collective intelligence [53], [54].

Particle Swarm Optimization (PSO)
Particle swarm optimization (PSO) is an evolutionary computation technique developed by Eberhart & Kennedy in 1995 [55], the concept being originated from the simulation of a simplified social system, i.e. to graphically simulate the graceful but unpredictable choreography of a bird flock.

Initial simulations were modified to incorporate nearest-neighbor velocity matching, eliminate ancillary variables, and incorporate multidimensional search and acceleration by distance. At some point in the evolution of this algorithm, it was realized that the conceptual model was **an optimizer.** Through a process of trial and error, a number of parameters extraneous to optimization were eliminated from the algorithm, resulting in a very simple implementation [56], similar to a genetic algorithm, where the system is initialized with a population of random solutions.

Unlike GAs, there is no mutation operator, although each potential solution is also assigned a randomized velocity, and the potential solutions, called particles, are then "flown" through the problem space. Each particle keeps track of its personal best position in the problem space, which is associated with the best fitness value of the particle found so far. Another "best" value that is tracked by the particle swarm optimizer is the overall best value, and its location, obtained by any particle in the population [57], [58] – the so-called “global best”.

Ant Colony Optimization (ACO)
Ants do not possess a great deal of intelligence by themselves, but collectively a colony of ants performs sophisticated tasks such as finding the shortest path to food sources and sharing this information with other ants by depositing *pheromone.* Ant Colony Optimization (ACO) models the collective intelligence of ants, which are transformed into optimization techniques [59]. ACO was introduced by Dorigo et al. (1991) [60], [61] as a novel nature-inspired metaheuristic for the solution of hard combinatorial optimization (CO) problems. Such metaheuristics are approximate algorithms used to obtain satisfactory solutions to hard CO problems in a reasonable amount of computational time. Other examples of such metaheuristics are tabu search, simulated annealing, and evolutionary computation [62]. More details can be found in [63], [64] and a recent example can be found in [65]. A prominent example as a data mining algorithm is the Ant-Miner (Ant-colony-based data miner), aiming at extracting classification rules from data. This algorithm was inspired by both research on the behavior of real ant colonies and known data mining concepts [66].

Bacteria Foraging Algorithms (BFA)
Foraging theory is based on the assumption that animals search for and obtain nutrients in a way that maximizes their energy intake *E* per unit time *T* spent foraging. The Escherichia coli bacterium is probably the best understood microorganism and much what is known cytokinesis in bacteria has come from studies with E. coli, and efforts to understand fundamental processes in this organism continue to intensify [67]. When E. coli grows, it gets longer, then divides in the middle into two so-called “daughters.” Given sufficient food and held at the temperature of the human gut (one place where they live) of 37° C, this bacterium can synthesize and replicate

everything it needs to make a copy of itself in about 20 minutes. Hence, the growth of a population of bacteria is exponential, with a relatively short time to double [68]. The foraging behavior of E. coli can be used by analogy in the Bacteria Foraging Algorithms (BFA) to solve global optimization problems [69]. An example of an electrical engineering application of the BFA can be found in [70]. An approach applied in human psychology is the Information Foraging Theory by Pirolli & Card (1999) [71]; this assumes that people try to modify their information seeking strategies to maximize their rate in gaining valuable information. The adaptation analysis develops information patch models, which deal with *time allocation* and *information filtering*; *information scent* models, which address the identification of information value from proximal cues; and *information diet models*, which address decisions about the selection and pursuit of information items. The theory has been used to study e.g. the "surf behaviour" on the Web [72], but has also been used for data mining [73], and for knowledge discovery in the biomedical domain [74].

Bees Algorithm (BA)
The population-based search algorithm called the Bees Algorithm (BA) mimics the food foraging behaviour of a swarm of honey bees and was proposed by Pham et al. (2006) [75]. In its basic version, the algorithm performs a kind of neighbourhood search combined with random search, and can be used for combinatorial optimisation as well as functional optimisation [76]. BAs are also meta-heuristics, which try to model the natural behavior of bees in food foraging, such as mechanisms like waggle dance to optimally locate food sources and to search for new ones [77]. Basturk & Karaboga (2007), [78] proposed the Artificial Bee Colony (ABC) algorithm for constrained optimization problems. The idea is that the collective intelligence of bee swarms consists of three components: food sources, employed bees, and unemployed bees; the latter further segregated into onlookers and scouts. This results into three main phases of ABC: employed phase, onlooker phase, and scout phase.A recent work described the integration of Artificial Bee Colony (ABC) and Bees Algorithm (BA) to an ABC–BA algorithm which performs better than each single one [79].

Invasive Weed Optimization (IWO)
The Invasive Weed Optimization Algorithm (IWO) was proposed by Mehrabian & Lucas (2006) [80], as an ecologically inspired metaheuristic that mimics the process of weeds colonization and distribution, which is capable of solving multi-dimensional, linear and nonlinear optimization problems with appreciable efficiency. Moreover, the IWO can also be used in the validation of reached optima and in the development of regularization terms and non-conventional transfer functions that do not necessarily provide gradient information [81]. A recent example of IWO for knowledge discovery purposes can be found in [82], .

4.4.5 Memetic Algorithms (MA)

Memetic algorithms (MA) are amongst the growing areas in evolutionary computation and were inspired by Richard Dawkins' meme [83]; an implementation of an "selfish gene algorithm" can be found here [84]. The term MA is widely used as a synergy of evolutionary or any other population-based approach with separate individual learning or local improvement procedures for search problems [85]. MAs are often also referred to as Baldwinian evolutionary algorithms (see [86] about the Baldwin effect), Lamarckian EAs, cultural algorithms, or genetic local search.

A novel correlation based memetic framework (MA-C), which is a combination of a genetic algorithm (GA) and local search (LS) using correlation based filter ranking has been proposed in [87]: The local filter method fine-tunes the population of GA solutions by adding or deleting features based on Symmetrical Uncertainty (SU) measures. Such approaches have many possibilities for the use in real-world problems, particularly in bio-computing and data mining for high-dimensional problems [88]. Amongst a very recent meta-heuristic is the Grey Wolf Optimizer (GWO), which mimics the leadership hierarchy and hunting mechanism of grey wolves (canis lupus) in nature [89].

4.4.6 Differential Evolution Search

An example of Differential Evolution (DE) search is the Artificial Bee Colony (ABC) algorithm, mentioned in section 4.3.4. Its main idea is that the algorithm makes use of differential evolution operators to update the information on the food source in order to enhance the local search ability at the stage of onlooker bees, and a chaotic sequence is introduced to the differential mutation operator for this purpose. Simulation results show that this algorithm, introducing *chaotic differential evolution* search, is a promising one in terms of convergence rate and solution accuracy, compared to the ABC algorithm [90]. A memetic DE algorithm, that utilizes a chaotic local search (CLS) with a shrinking strategy, in order to improve the optimizing performance of the canonical DE by exploring a huge search space in the early run phase to avoid premature convergence can be found in [91].

4.4.7 Artificial Immune Systems (AIS)

Artificial immune systems (AIS) developed by Farmer et al. (1986) [92], can be defined as adaptive computational systems inspired from immunology, i.e. by the observed immune functions, principles and mechanisms. The idea was that the immune system as highly evolved biological system, is able to identify (and eliminate) foreign substances [93]. Consequently, it must be able to determine between gens and antigens, which requires a powerful capability of learning, memory and pattern recognition. The development and application domains of AIS follow those of soft computing paradigms [94], such as artificial neural networks (ANN) and fuzzy systems (FS). A framework which discusses the suitability of AIS as a soft computing paradigm that integrate AIS with other approaches, focusing on ANN, EA and FS has been proposed by [95].

4.4.8 Gravitational Search Algorithm (GSA)

Gravitational Search Algorithms (GSA) are based on the analogy with the law of gravity and mass interactions: the search agents are a collection of masses which interact with each other based on the Newtonian gravity and the laws of motion [96].

GSA falls also under the category of metaheuristics as general search strategies that, at the exploitation stage, exploit areas of the solution space with high quality solutions and, at the exploration stage, move to unexplored areas of the solution space [97]. GSA is a stochastic population-based metaheuristic that was originally designed for solving *continuous* optimization problems.

The Binary Gravitational Search Algorithm (BGSA) is a new variant for discrete optimization problems, and experimental results confirm its efficiency in solving various nonlinear benchmark problems [98].

A recent work on a Discrete Gravitational Search Algorithm (DGSA) to solve combinatorial optimization problems [99] can be found in [97].

4.4.9 Intelligent Water Drops Algorithm (IWD)

A natural river often finds optimal paths among a number of different possible paths in its ways from the source to destination. These near optimal or optimal (natural) paths are obtained by the actions and reactions that occur among the water drops and between the water drops and the riverbeds.

The Intelligent Water Drops (IWD) algorithm is a new population-based optimisation algorithm inspired from observing natural water drops flowing in rivers. The authors of [100] tested this algorithm to find solutions of the n-queen puzzle with a simple local heuristic, solved the travelling salesman problem (TSP) and tested it with multiple knapsack problems (MKP) in which near-optimal or optimal solutions were obtained [101].

There are various application areas for IWD thinkable, e.g. Agarwal et al. (2012) [102], propose the use of IWD as an optimised code coverage algorithm by using dynamic parameters for finding all the optimal paths using basic properties of natural water drops. A recent example application is in using IWD for solving multi-objective job-shop scheduling: Niu et al. (2013) customized it to find the best compromising solutions (Pareto non-dominance set) considering multiple criteria, namely make-span, tardiness and mean flow time of schedules, and proved that the customized IWD algorithm can identify the Pareto non-dominance schedules efficiently.

5 Evolutionary Algorithms for Knowledge Discovery and Data Mining

5.1 Classification and Clustering with EAs

In traditional data mining tasks, evolutionary algorithms can easily be used for both classification and clustering as well as for data preparation in the form of attribute generation and selection [27].

Classification is a central application for EAs, where they can be used for classification *rule mining*. These rules can be of different complexity and forms. In some cases, a whole set of rules is the goal, where interactions between the rules play an important role, whereas it is also possible to mine independent rules for classification. For details on the topic of *classification rule mining,* please refer to [27] and [103]. A very recent work, which shows that the implementation of evolutionary algorithms in machine learning can be achieved without extensive effort, meaning much experimentation can be performed quickly in order to discover novel areas where genetic algorithms can complement machine learning algorithms to achieve better classification results [104].

Clustering analysis is another application area for EAs to knowledge discovery tasks. There are different approaches to this problem, which are discussed in detail in [103]. The most important criteria when performing clustering analysis using EAs is the representation of solution candidates as well as the fitness evaluation, which can be a problematic issue considering the complexity of evaluating unsupervised knowledge discovery methods in general.

5.2 Advantages and Disadvantages

EAs have certain pros and cons as general optimization methods, including when they are used for knowledge discovery and data mining tasks, as shown in Table 3.

Table 3. Advantages and Disadvantages of EAs

Advantages	Disadvantages
Robust to noise	EA methods do not guarantee finding of an optimal solution in finite time
Deals well with attribute interaction	Domain specific knowledge has to be explicitly added using external processes
Comparatively easy to implement	Optimization runtime not constant, variance between best- and worst-case can differ greatly
Well suited for multi-objective optimization	Computational complexity can be an issue
Good scalability due to parallelization	Fitness-function needs to be specified, otherwise EAs do not work
Very flexible (widely usable)	Slower than greedy algorithms in many cases
Good option for problems without a traditional best practice method	Not the first choice if a traditional method already solves the problem in an efficient way
Good amount of programming libraries available	
Small amount of specific mathematical knowledge necessary for using EAs	
Suitable for efficiently solving NP-hard problems	

5.3 Available Software and Programming Libraries

There is a broad range of different software packages and libraries available for using EAs in KDD and DM tasks, The list below contains only the most well-known examples:

WEKA - `http://www.cs.waikato.ac.nz/ml/weka`
KEEL - `http://www.keel.es`
SolveIT - `http://www.solveitsoftware.com`
MCMLL - `http://mcmll.sourceforge.net`
Jenetics - `http://jenetics.sourceforge.net`
Jenes - `http://jenes.intelligentia.it`

jMetal - `http://jmetal.sourceforge.net`
JGAP - `http://jgap.sourceforge.net`
epochX - `http://www.epochx.org`
SAS - `https://www.sas.com`
Discipulus - `http://www.rmltech.com`
XpertRule - `http://www.attar.com`
MATLAB GA-Toolbox `http://www.mathworks.com/discovery/genetic-algorithm`

6 Open Problems and Challenges

Evolutionary algorithms, by default, are so called "blind" methods, which means that the used operators do not use or depend on domain specific knowledge. While this feature enriches their generality, it is in most cases a negative factor compared to methods making use of existing relevant knowledge within the domain [103]. This aspect can however be remedied by introducing mechanisms such as a preceding local search into the execution of evolutionary algorithms, and enriching the fitness function with domain specific data.

Another shortcoming of evolutionary approaches for knowledge discovery tasks is that they do not guarantee an optimal solution in finite time. They also do not guarantee constant optimization runtimes and the differences between the best and worst case scenarios are usually larger than for most traditional optimization methods, making EAs a suboptimal choice for real-time systems [105].

Computational complexity can, as with all other KDD methods, also be an issue. However with the increase of processing power as well as the possibility to easily parallelize evolutionary methods, especially in combination with cloud services and the island model [105], the issue should be, at most, of a temporary nature.

7 Future Work

A specific area of future research, according to [103], should be the application of genetic programming for data mining tasks. There have been attempts to create generic rule induction algorithms using GP [106], [107], but they are still comparatively under-discovered and under-used within the domain of knowledge discovery.

Biology has traditionally been a source of inspiration for evolutionary algorithms. In most organisms, evolution proceeds in small steps by random mutations and in large steps by horizontal events (recombination, reassortments, gene transfer and hybridizations). Horizontal events combine the genetic information from two or more organisms to generate a new one that incorporate alleles from parental strains. Whilst mutations allow efficient local searches in the fitness landscape, horizontal events combine information from fit individuals exploring larger regions of search space. Humans and eukaryotes in general recombine during meiosis, retroviruses during retrotranscription, each presenting different ways of combining genetic information. Segmented viruses, viruses with more than one chromosome as influenza, combine genetic information through reassortments, a process where a new individual is created by exchange of chromosomes between two or more parental strains. This is the very effective process behind **influenza pandemics** that could allow viruses to jump from one host to another and rapidly propagate in the new population. Such mechanisms, and others, are found in nature and represent different strategies to go beyond mutations with distinct advantages. Each of these evolutionary strategies can be used to address different problems – but it needs much further research, testing and experimenting.

References

1. Box, G.E.: Evolutionary operation: A method for increasing industrial productivity. Applied Statistics 6(2), 81–101 (1957)
2. Holland, J.H.: Outline for a Logical Theory of Adaptive Systems. J. ACM 9(3), 297–314 (1962)
3. Hendy, M.D., Penny, D.: Branch an Bound Algorithms to determine minimal Evolutionary Trees. Mathematical Biosciences 59(2), 277–290 (1982)
4. Letunic, I., Bork, P.: Interactive Tree Of Life (iTOL): An online tool for phylogenetic tree display and annotation. Bioinformatics 23(1), 127–128 (2007)
5. Darwin, C.: On the origin of species by means of natural selection, or the preservation of favoured races in the struggle for life. John Murray, London (1859)
6. Hoffmeister, F.: Scalable Parallelism by Evolutionary Algorithms. Lecture Notes in Economics and Mathematical Systems 367, 177–198 (1991)
7. Cheng, A., Lim, C.C.: Optimizing System-On-Chip verifications with multi-objective genetic evolutionary algorithms. Journal of Industrial and Management Optimization 10(2), 383–396 (2014)
8. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: A comparative case study and the Strength Pareto approach. IEEE Transactions on Evolutionary Computation 3(4), 257–271 (1999)
9. Waddington, C.H.: Canalization of development and the inheritance of acquired characters. Nature 150(3811), 563–565 (1942)
10. Trygve, T.: Handbook of Epigenetics. Academic Press, San Diego (2011)
11. Fitch, W.M., Margoliash, E.: Construction of phylogenetic trees. Science 155(760), 279–284 (1967)
12. Saitou, N., Nei, M.: The neighbor-joining method: A new method for reconstructing phylogenetic trees. Molecular Biology and Evolution 4(4), 406–425 (1987)
13. Kruse, R., Borgelt, C., Klawonn, F., Moewes, C., Steinbrecher, M., Held, P.: Computational Intelligence: A Methodological Introduction. Springer, Heidelberg (2013)
14. Lollini, P.-L., Motta, S., Pappalardo, F.: Discovery of cancer vaccination protocols with a genetic algorithm driving an agent based simulator. BMC Bioinformatics 7(1), 352 (2006)
15. Ritchie, M.D., Motsinger, A.A., Bush, W.S., Coffey, C.S., Moore, J.H.: Genetic programming neural networks: A powerful bioinformatics tool for human genetics. Applied Soft Computing 7(1), 471–479 (2007)
16. Winstein, K., Balakrishnan, H.: TCP ex Machina: Computer-Generated Congestion Control. In: ACM SIGCOMM, pp. 123–134. ACM (2013)
17. Gen, M., Cheng, R.: Genetic algorithms and engineering optimization. John Wiley & Sons (2000)
18. Rafael, B., Oertl, S., Affenzeller, M., Wagner, S.: Music segmentation with genetic algorithms. In: 20th International Workshop on Database and Expert Systems Application, DEXA 2009, pp. 256–260. IEEE (2009)

19. Soupios, P., Akca, I., Mpogiatzis, P., Basokur, A., Papazachos, C.: Application of Genetic Algorithms in Seismic Tomography. In: EGU General Assembly Conference Abstracts, p. 1555 (2010)
20. Waddington, C.H.: Epigenetics and Evolution. Symposia of the Society for Experimental Biology 7, 186–199 (1953)
21. Jablonka, E.: Epigenetic inheritance and evolution: The Lamarckian dimension. Oxford University Press, Oxford (1999)
22. Tapscott, D.: Grown Up Digital: How the Net Generation is Changing Your World HC. Mcgraw-Hill (2008)
23. Bird, A.: Perceptions of epigenetics. Nature 447(7143), 396–398 (2007)
24. Handel, A., Ramagopalan, S.: Is Lamarckian evolution relevant to medicine? BMC Medical Genetics 11(1), 73 (2010)
25. Kiberstis, P.A.: All Eyes on Epigenetics. Science 335(6069), 637 (2012)
26. Emmert-Streib, F., de Matos Simoes, R., Glazko, G., McDade, S., Haibe-Kains, B., Holzinger, A., Dehmer, M., Campbell, F.: Functional and genetic analysis of the colon cancer network. BMC Bioinformatics 15(suppl. 6), S6 (2014)
27. Freitas, A.A.: A survey of evolutionary algorithms for data mining and knowledge discovery. In: Advances in Evolutionary Computing, pp. 819–845. Springer (2003)
28. Coello Coello, C.A., Lechuga, M.S.: MOPSO: A proposal for multiple objective particle swarm optimization. In: Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002), pp. 1051–1056. IEEE (2002)
29. Kruse, R., Borgelt, C., Klawonn, F., Moewes, C., Ruß, G., Steinbrecher, M.: Computational Intelligence: Eine methodische Einführung in Künstliche Neuronale Netze, Evolutionäre Algorithmen, Fuzzy-Systeme und Bayes-Netze Vieweg+Teubner, Wiesbaden (2011)
30. Jansen, T.: Analyzing evolutionary algorithms: The computer science perspective. Springer Publishing Company (2013) (incorporated)
31. Yu, X., Gen, M.: Introduction to evolutionary algorithms. Springer, Heidelberg (2010)
32. Eiben, A.E., Smith, J.E.: Introduction to evolutionary computing. Springer, Berlin (2010)
33. De Jong, K.A.: Evolutionary computation: A unified approach. MIT press, Cambridge (2006)
34. Goldberg, D.E., Holland, J.H.: Genetic algorithms and machine learning. Mach. Learn. 3(2), 95–99 (1988)
35. Mitchell, T.M.: Machine learning, p. 267. McGraw-Hill, Boston (1997)
36. Forrest, S.: Genetic algorithms: Principles of natural selection applied to computation. Science 261(5123), 872–878 (1993)
37. Grefenstette, J.J.: Optimization of control parameters for genetic algorithms. IEEE Transactions on Systems Man and Cybernetics 16(1), 122–128 (1986)
38. Goldberg, D.E.: Genetic algorithms in search, optimization, and machine learning. Addison-Wesley Reading, MA (1989)
39. Mitchell, M.: An Introduction to Genetic Algorithms (Complex Adaptive Systems). MIT Press, Cambridge (1998)
40. Coley, D.A.: An introduction to Genetic Algorithms for Scientists and Engineers. World Scientific Publishing, Singapore (1999)
41. Koza, J.R.: Genetic programming as a means for programming computers by natural selection. Statistics and Computing 4(2), 87–112 (1994)
42. Brameier, M., Banzhaf, W.: A comparison of linear genetic programming and neural networks in medical data mining. IEEE Transactions on Evolutionary Computation 5(1), 17–26 (2001)

43. Bäck, T., Hoffmeister, F., Schwefel, H.P.: A survey of evolution strategies. In: Proceedings of the 4th International Conference on Genetic Algorithms, pp. 2-9 (1991)
44. Ohkura, K., Matsumura, Y., Ueda, K.: Robust evolution strategies. Applied Intelligence 15(3), 153–169 (2001)
45. Huhse, J., Zell, A.: Evolution Strategy with Neighborhood Attraction–A Robust Evolution Strategy. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2001, pp. 1026–1033 (2001)
46. Auger, A., Hansen, N.: Theory of Evolution Strategies: A new perspective. In: Auger, A., Doerr, B. (eds.) Theory of Randomized Search Heuristics: Foundations and Recent Developments, pp. 289–325. World Scientific Publishing, Singapore (2011)
47. Beyer, H.-G., Schwefel, H.-P.: Evolution strategies–A comprehensive introduction. Natural Computing 1(1), 3–52 (2002)
48. Beyer, H.-G.: The theory of evolution strategies. Springer, Heidelberg (2001)
49. Martens, D., Baesens, B., Fawcett, T.: Editorial survey: Swarm intelligence for data mining. Mach. Learn. 82(1), 1–42 (2011)
50. Franks, N.R., Pratt, S.C., Mallon, E.B., Britton, N.F., Sumpter, D.J.T.: Information flow, opinion polling and collective intelligence in house-hunting social insects. Philosophical Transactions of the Royal Society of London Series B-Biological Sciences 357(1427), 1567–1583 (2002)
51. Kennedy, J.F., Eberhart, R.C.: Swarm intelligence. Morgan Kaufmann, San Francisco (2001)
52. van't Woud, J., Sandberg, J., Wielinga, B.J.: The Mars crowdsourcing experiment: Is crowdsourcing in the form of a serious game applicable for annotation in a semantically-rich research domain? In: 2011 16th International Conference on Computer Games (CGAMES), pp. 201–208. IEEE (2011)
53. Woolley, A.W., Chabris, C.F., Pentland, A., Hashmi, N., Malone, T.W.: Evidence for a Collective Intelligence Factor in the Performance of Human Groups. Science 330(6004), 686–688 (2010)
54. Bonabeau, E.: Decisions 2.0: The power of collective intelligence. MIT Sloan Management Review 50(2), 45–52 (2009)
55. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: IEEE Proceedings of the Sixth International Symposium on Micro Machine and Human Science, MHS 1995 (1995)
56. Eberhart, R., Simpson, P., Dobbins, R.: Computational intelligence PC tools. Academic Press Professional, Inc. (1996)
57. Kennedy, J., Eberhart, R.C.: A discrete binary version of the particle swarm algorithm. In: 1997 IEEE International Conference on Systems, Man, and Cybernetics, Computational Cybernetics and Simulation, pp. 4104–4108. IEEE (1997)
58. Eberhart, R.C., Shi, Y.H.: Particle swarm optimization: Developments, applications and resources. IEEE, New York (2001)
59. Sim, K.M., Sun, W.H.: Ant colony optimization for routing and load-balancing: Survey and new directions. IEEE Trans. Syst. Man Cybern. Paart A-Syst. Hum. 33(5), 560–572 (2003)
60. Colorni, A., Dorigo, M., Maniezzo, V.: Distributed optimization by ant colonies. In: Proceedings of the First European Conference on Artificial Life, vol. 142, pp. 134–142 (1991)
61. Dorigo, M., Gambardella, L.M.: Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation 1(1), 53–66 (1997)

62. Dorigo, M., Blum, C.: Ant colony optimization theory: A survey. Theoretical Computer Science 344(2-3), 243–278 (2005)
63. Colorni, A., Dorigo, M., Maniezzo, V.: An Investigation of some Properties of an "Ant Algorithm". In: Parallel Problem Solving from Nature Conference (PPSN), pp. 509–520. Elsevier (1992)
64. Dorigo, M., Birattari, M., Stutzle, T.: Ant colony optimization - Artificial ants as a computational intelligence technique. IEEE Computational Intelligence Magazine 1(4), 28–39 (2006)
65. Chen, Y.J., Wong, M.L., Li, H.B.: Applying Ant Colony Optimization to configuring stacking ensembles for data mining. Expert Systems with Applications 41(6), 2688–2702 (2014)
66. Parpinelli, R.S., Lopes, H.S., Freitas, A.A.: Data mining with an ant colony optimization algorithm. IEEE Transactions on Evolutionary Computation 6(4), 321–332 (2002)
67. de Boer, P.A.J.: Advances in understanding E. coli cell fission. Current Opinion in Microbiology 13(6), 730–737 (2010)
68. Passino, K.M.: Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Systems 22(3), 52–67 (2002)
69. Kim, D.H., Abraham, A., Cho, J.H.: A hybrid genetic algorithm and bacterial foraging approach for global optimization. Inf. Sci. 177(18), 3918–3937 (2007)
70. Tripathy, M., Mishra, S., Lai, L.L., Zhang, Q.P.: Transmission loss reduction based on FACTS and bacteria foraging algorithm. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 222–231. Springer, Heidelberg (2006)
71. Pirolli, P., Card, S.: Information foraging. Psychological Review 106(4), 643–675 (1999)
72. Pirolli, P.: Rational analyses of information foraging on the Web. Cognitive Science 29(3), 343–373 (2005)
73. Liu, J.M., Zhang, S.W., Yang, J.: Characterizing Web usage regularities with information foraging agents. IEEE Transactions on Knowledge and Data Engineering 16(5), 566–584 (2004)
74. Goodwin, J.C., Cohen, T., Rindflesch, T.: Discovery by scent: Discovery browsing system based on the Information Foraging Theory. In: Gao, J., Dubitzky, W., Wu, C., Liebman, M., Alhaij, R., Ungar, L., Christianson, A., Hu, X. (eds.) 2012 IEEE International Conference on Bioinformatics and Biomedicine Workshops. IEEE, New York (2012)
75. Pham, D., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S., Zaidi, M.: The bees algorithm-a novel tool for complex optimisation problems. In: Proceedings of the 2nd Virtual International Conference on Intelligent Production Machines and Systems (IPROMS 2006), pp. 454–459 (2006)
76. Pham, D.T., Castellani, M.: The Bees Algorithm: Modelling foraging behaviour to solve continuous optimization problems. Proceedings of the Institution of Mechanical Engineers Part C-Journal of Mechanical Engineering Science 223(12), 2919–2938 (2009)
77. Ozbakir, L., Baykasoglu, A., Tapkan, P.: Bees algorithm for generalized assignment problem. Applied Mathematics and Computation 215(11), 3782–3795 (2010)
78. Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of global optimization 39(3), 459–471 (2007)
79. Tsai, H.C.: Integrating the artificial bee colony and bees algorithm to face constrained optimization problems. Inf. Sci. 258, 80–93 (2014)

80. Mehrabian, A.R., Lucas, C.: A novel numerical optimization algorithm inspired from weed colonization. Ecological Informatics 1(4), 355–366 (2006)
81. Giri, R., Chowdhury, A., Ghosh, A., Das, S., Abraham, A., Snasel, V.: A Modified Invasive Weed Optimization Algorithm for training of feed- forward Neural Networks. In: 2010 IEEE International Conference on Systems, Man and Cybernetics (SMC 2010), pp. 3166–3173 (2010)
82. Huang, H., Ding, S., Zhu, H., Xu, X.: Invasive Weed Optimization Algorithm for Optimizating the Parameters of Mixed Kernel Twin Support Vector Machines. Journal of Computers 8(8) (2013)
83. Dawkins, R.: The selfish gene. Oxford University Press, Oxford (1976)
84. Corno, F., Reorda, M.S., Squillero, G.: The selfish gene algorithm: a new evolutionary optimization strategy. In: Proceedings of the 1998 ACM Symposium on Applied Computing, pp. 349–355. ACM (1998)
85. Ishibuchi, H., Yoshida, T., Murata, T.: Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. IEEE Transactions on Evolutionary Computation 7(2), 204–223 (2003)
86. Simpson, G.G.: The Baldwin Effect. Evolution 7(2), 110–117 (1953)
87. Kannan, S.S., Ramaraj, N.: A novel hybrid feature selection via Symmetrical Uncertainty ranking based local memetic search algorithm. Knowledge-Based Systems 23(6), 580–585 (2010)
88. Molina, D., Lozano, M., Sanchez, A.M., Herrera, F.: Memetic algorithms based on local search chains for large scale continuous optimisation problems: MA-SSW-Chains. Soft Computing 15(11), 2201–2220 (2011)
89. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey Wolf Optimizer. Advances in Engineering Software 69, 46–61 (2014)
90. Yin, J., Meng, H.: Artificial bee colony algorithm with chaotic differential evolution search. Computer Engineering and Applications 47(29), 27–30 (2011)
91. Jia, D.L., Zheng, G.X., Khan, M.K.: An effective memetic differential evolution algorithm based on chaotic local search. Inf. Sci. 181(15), 3175–3187 (2011)
92. Farmer, J.D., Packard, N.H., Perelson, A.S.: The immune system, adaptation, and machine learning. Physica D: Nonlinear Phenomena 22(1), 187–204 (1986)
93. Parham, P.: The Immune System, 3rd edn. Garland Science, Taylor and Francis, New York (2009)
94. Dasgupta, D., Yu, S.H., Nino, F.: Recent Advances in Artificial Immune Systems: Models and Applications. Applied Soft Computing 11(2), 1574–1587 (2011)
95. de Castro, L.N., Timmis, J.I.: Artificial immune systems as a novel soft computing paradigm. Soft Computing 7(8), 526–544 (2003)
96. Rashedi, E., Nezamabadi-Pour, H., Saryazdi, S.: GSA: A Gravitational Search Algorithm. Inf. Sci. 179(13), 2232–2248 (2009)
97. Dowlatshahi, M.B., Nezamabadi-Pour, H., Mashinchi, M.: A discrete gravitational search algorithm for solving combinatorial optimization problems. Inf. Sci. 258, 94–107 (2014)
98. Rashedi, E., Nezamabadi-pour, H., Saryazdi, S.: BGSA: Binary gravitational search algorithm. Natural Computing 9(3), 727–745 (2010)
99. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Comput. Surv. 35(3), 268–308 (2003)
100. Shah-Hosseini, H.: The intelligent water drops algorithm: A nature-inspired swarm-based optimization algorithm. International Journal of Bio-Inspired Computation 1(1-2), 71–79 (2009)
101. Shah-Hosseini, H.: Problem solving by intelligent water drops. IEEE, New York (2007)

102. Agarwal, K., Goyal, M., Srivastava, P.R.: Code coverage using intelligent water drop (IWD). International Journal of Bio-Inspired Computation 4(6), 392–402 (2012)
103. Maimon, O., Rokach, L. (eds.): Data Mining and Knowledge Discovery Handbook. Springer, New York (2010)
104. Holzinger, A., Blanchard, D., Bloice, M., Holzinger, K., Palade, V., Rabadan, R.: Darwin, Lamarck, or Baldwin: Applying Evolutionary Algorithms to Machine Learning Techniques. In: World Intelligence Congress (WIC). IEEE (2014) (in print)
105. Whitley, D.: An overview of evolutionary algorithms: Practical issues and common pitfalls. Information and Software Technology 43(14), 817–831 (2001)
106. Pappa, G.L., Freitas, A.A.: Discovering new rule induction algorithms with grammar-based genetic programming. In: Soft Computing for Knowledge Discovery and Data Mining, pp. 133–152. Springer (2008)
107. Pappa, G.L., Freitas, A.A.: Evolving rule induction algorithms with multi-objective grammar-based genetic programming. Knowledge and Information Systems 19(3), 283–309 (2009)