When Evolutionary Computing meets Astro- and Geo- Informatics

Zaineb Chelly Dagdia[1,2] and Miroslav Mirchev[3]

[1]Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
[2]LARODEC, Institut Supérieur de Gestion de Tunis, Tunis, Tunisia, e-mails: chelly.zaineb@gmail.com, zaineb.chelly-dagdia@inria.fr
[3]Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University in Skopje, North Macedonia, e-mail: miroslav.mirchev@finki.ukim.mk

October 26, 2023

Abstract

Knowledge discovery from data typically include solving some type of an optimization problem that can be efficiently addressed using algorithms belonging to the class of evolutionary and bio-inspired computation. In this chapter, we give an overview of the various kinds of evolutionary algorithms such as genetic algorithms, evolutionary strategy, evolutionary and genetic programming, differential evolution and co-evolutionary algorithms, as well as several other bio-inspired approaches like swarm intelligence and artificial immune systems. After elaborating on the methodology, we provide numerous examples of applications in astronomy and geoscience and show how these algorithms can be applied within a distributed environment, by making use of parallel computing which is essential when dealing with Big Data.

Keywords— evolutionary computation, bio-inspired computing, metaheuristics, astroinformatics, geoinformatics

1 Introduction

On the origins of primitive life, biological life has been evolving every year. From the beginning, unicellular life organisms get gradually mutated to complex life form organisms (multicellular organisms). This progressive change is evolved by the process of genetic evolution. In the process of evolution, a series of natural changes cause organisms (or species) to arise, to adapt and familiarize to their environment, and turn out to be extinct. In this concern, Darwin's theory of evolution (Darwin 2004) asserts that species survive through a process named "natural selection". Those species that successfully adapt, or evolve, to meet the changing requirements of their natural habitat thrive, while those that fail to evolve and reproduce die off (Goodwin 1982).

Evolutionary computation includes a set of approaches that seek to emulate the mechanism of natural selection described in Darwin's theory, with the aim of solving complex optimization problems. In real life, there are countless applications that require optimization such as in business, in economics, in astronomy, and in geoscience. In these kinds of applications, there are many processes that can be potentially optimized. Optimization could occur in the minimization of time, cost and risk, or the maximization of profit, quality and efficiency. In astronomy and geoscience, we often need to minimize the difference from a model output to some observed data, or maximize the distance between different classes of pixels representing objects found in an image. In this chapter, we present a palette of evolutionary computation techniques that tend to solve the complex and difficult optimization problems encountered in Astro- and Geo- Informatics as these will be the main application areas studied in this chapter. More precisely, we focus on the challenges that arise in Astro- and Geo- Informatics specifically when it comes to dealing with the large amount of acquired data that became easily accessible given the emergent technologies. In this concern, we introduce parallel evolutionary computation techniques that can solve Astro- and Geo- Informatics optimization problems as they can speed up computation, solve large problems, and find better solutions.

To model and solve Astro- and Geo- Informatics optimization problems, we have to first understand the basic concepts of optimization models and related solution methods. This chapter introduces related concepts, models and solution methods of optimization including genetic algorithms, evolutionary strategy, evolutionary programming, genetic programming, ant colony optimization, particle swarm optimization, and artificial immune system optimization approaches.

This chapter is organized as follows. Section 2 introduces the basic concepts of an optimization problem including its standard formulation, the types of an optimization problem, and the multi-objective optimization model. Section 3 presents the structure of an evolutionary computation algorithm and a range of evolution operators. Section 4 addresses evolutionary computation algorithms and some bio-inspired optimization approaches by their definition, classification, models, and theories. Section 5 introduces the context of big data and discusses a set of parallel frameworks dedicated for evolutionary computing algorithms. Section 6 discusses the applications of evolutionary computing techniques in Astro- and Geo- Informatics, and Section 7 presents a summary of the main points highlighted in this chapter.

2 The optimization problem

Optimization is a key requirement for making decisions and in analyzing Astro- and Geo- Informatics systems. Formally, an optimization problem is defined in terms of a set of parameters and restrictions. The parameters chosen to describe the design of a specific system include one or many decision makers, the system's variables, single or several objectives to be achieved, and a set of structural restrictions known as constraint conditions. Several optimization studies are, indeed, formulated as problems aiming at finding the best solution(s) from among the set of all feasible solutions, i.e., solutions that satisfy all the constraints of the optimization problem.

The variables, also called decision variables, reflect the system's components for which we want to find the sought values. In model's parameters estimation or data fitting, for instance, the variables are the parameters of the given model, whether it is based on some known physical laws or it uses some generic functions.

The notation of an optimization problem also implies that there are some objective function or functions that can be improved either by performing a minimization or a maximization action, and that can also be used as a quantitative measure of effectiveness of the system. The objective function is also called fitness function, merit function or cost function. For instance, in fitting experimental data to a model, we may want to minimize the total deviation of the observed data from the data predicted with the model.

Finally, the constraints are the functions that describe the relationships between the system's variables. They define the allowable values to be taken by the variables. For example, in parameter estimation the values could be constrained within an expected range that is known in advance.

2.1 Standard formulation

Any Astro- or Geo- Informatics problem can be defined as an optimization problem. The domain of the function to be optimized is called the search space ($\mathbf{S}$). The goodness of any solution in $\mathbf{S}$ is measured with a fitness function ($\mathbf{O} : \mathbf{S} \rightarrow \mathbb{R}$). A fitness landscape that maps from a configuration space into the real numbers $\mathbb{R}$ may be considered as a triple ($\mathbf{S}$, $\mathbf{O}$, $\mathbf{D}$); where $\mathbf{D}$ is a metric defined on $\mathbf{S}$.

In general, optimization problems comprise setting a vector $\mathbf{X}$ of decision variables (x_i) of a system in order to optimize (either minimize or maximize) some fitness function $\mathbf{O}(\mathbf{X})$. In some cases, this is achieved subject to the satisfaction of a set of constraint conditions; namely the equality constraints $h_k(\mathbf{X})$, the inequality constraints $g_j(\mathbf{X})$, and with $(x_i)^L$ and $(x_i)^U$ corresponding respectively to the lower and upper bounds of the variable x_i. A solution x_i satisfying the $(P + Q)$ constraints is said "feasible" and the set of all feasible solutions defines the feasible search space denoted by Ω. Assuming the nonlinearity of an optimization problem and with respect to a set of constraints, its mathematical modeling that deals with the search for a minimum of a nonlinear function $\mathbf{O}(\mathbf{X})$ of m variables can be outlined as follows:

$$\min \mathbf{O}(\mathbf{X}) \ , \mathbf{X} = (x_1, x_2, \ldots, x_m)^T \in \mathbf{S} \tag{1}$$

$$\text{Subject to} \begin{cases} h_k(\mathbf{X}) = 0, \ k = \{1, 2, \ldots, Q\} \\ g_j(\mathbf{X}) \leq 0, \ j = \{1, 2, \ldots, P\} \\ (x_i)^L \leq (x_i) \leq (x_i)^U, \ i = \{1, 2, \ldots, m\} \end{cases} \tag{2}$$

Without any loss of generality, minimization is assumed in Equation 1 as any minimization problem can be equivalently transformed into a maximization problem by a simple modification of the fitness function. This can be achieved, for example, by $-\mathbf{O}(\mathbf{X})$, $\frac{1}{\text{constant } + \mathbf{O}(\mathbf{X})}$ or by some other means. In Equations 1 and 2, there are no specific conditions tied to the system's variables type. However, the formulization of an optimization problem is attached to the optimization problem features and variants. These will be detailed in the following section.

2.2 Types of optimization problems

Categorizing the optimization problem is an essential step in the optimization process. This is because algorithms for solving optimization problems are tailored to a specific type of problems. These can be classified in terms of the number of decision makers, the nature or type of the decision variables, the number of constraints, the number of objective functions, the linearity of the problem, and the uncertainty tied to the optimization model. These factors require careful thoughts along with mathematical details while designing the optimization models. It is also important to mention that for each specific type of any optimization problem, there is a set of dedicated optimization algorithms that explicitly deal and handle particular nature and variants of the problems' components. In what follows, we provide the key variants, features and types classifying the optimization model into distinctive various optimization problem types.

2.2.1 The number of decision makers

As previously defined, an optimization problem aims at finding the best solution(s) among the set of all feasible solutions. In such a case, when decision making is emphasized, the problem supports a human Decision Maker (DM) to find the most preferred optimal solution according to his/her subjective preferences (Branke et al. 2008). Based on the assumption that a single solution to the problem must be identified to be implemented in practice, the DM who is expected to be an expert in the problem domain, has to make his/her decision. In a more formal way, we deal with an optimization problem in case when either one decision maker is involved or when no preference methods are required, i.e., no DM is expected to be available. Otherwise, the problem that we deal with is concerned with a game that can be either cooperative or non-cooperative, depending on the decision makers' perspectives (Myerson 2013).

2.2.2 The type of the decision variables

The formulization of an optimization model is usually tied to the nature or type of the system's decision variables. In some cases, such formulization only makes sense when the system's variables take values from a discrete set; usually a subset of integers. Conversely, some other optimization models can be only formulized via variables that take any real value. Models with discrete variables are known as "discrete optimization" problems while models with continuous variables are called "continuous optimization" problems.

In discrete optimization, some or all of the variables may be binary, i.e., restricted to the values 0 and 1, integer for which only integer values may be taken, or more abstract objects drawn from sets with finitely several elements. Within this category, we may consider two divisions of optimization problems. The first branch is called "integer programming" where the discrete set of the feasible solutions is a subset of integers. This class of models is mostly common as many real-life applications are modeled with discrete variables as their handled resources are indivisible, e.g., image pixels belonging to a certain object (Bertsimas & Weismantel 2005). The second branch is called "combinatorial optimization" where the discrete set is a set of objects or combinatorial structures such as assignments, combinations, routes, schedules, or sequences (Papadimitriou & Steiglitz 1998) (Schrijver 2003).

In continuous optimization, the system's variables in the optimization model take real values. In general, continuous optimization problems tend to be simpler and easier to solve in comparison to discrete optimization problems. This can be explained by the fact that smoothing the fitness function and the constraint conditions' values at a point x can be used to infer information about some other points in the neighborhood of x. It is essential to mention that continuous optimization has an important connection to discrete optimization because many discrete optimization algorithms often require continuous optimization problems to be solved as sub-problems or relaxations.

In some other cases, it is possible that an optimization model is formulized using different variables' types. In this case, we refer to a "mixed-integer optimization" problem where the decision variables are both discrete and continuous. This class of optimization problems presents a generalization of both the discrete optimization problems and the continuous optimization problems (Pochet & Wolsey 2006).

2.2.3 The number of constraints

One further distinction in optimization problems is between problems which are defined using a set of constraints on the system's variables, called "constrained optimization", and problems in which there are no constraints on the decision variables, called "unconstrained optimization". We refer to a constrained optimization problem when there are explicit constraints on the decision variables. These constraint conditions may vary from simple bounds to systems of equalities and inequalities that model complex relationships among the variables. The formalization of a constrained optimization problem was given in Section 2.1. In unconstrained optimization problem, the model may be based on a reformulation of constrained optimization problems in which the constraints are replaced by a penalty term in the fitness function.

2.2.4 The number of objective functions

In real world applications, an optimization problem may either have a single objective function, multiple objective functions or even no objective function. "Feasibility problems" are those problems aiming at finding values for the variables that satisfy the model constraint conditions with no specific objective to optimize. We may, also, refer to "multi-objective optimization" problems when involving more than one objective function to be optimized simultaneously. These objectives are often conflicting and incommensurable. Usually, there is no single solution that is optimal with respect to all the used objective functions at the same time, but rather many different designs exist which are incomparable. Consequently, contrary to "single-objective optimization" problems where we look for the solution presenting the best performance, the resolution of a multi-objective optimization problem gives rise to a set of compromise solutions presenting the optimal trade-offs between two or more conflicting objectives. In practice, problems with multiple objectives are often reformulated as single objective problems by either forming a weighted combination of the different objectives or by replacing some of the objectives by constraints (Coello et al. 2007) (Deb 2001). Defining multi-objective optimization problems requires further mathematical definitions; these will be given in Section 2.3.

2.2.5 The linearity

An optimization problem may be categorized, indeed, as a linear problem or as a nonlinear problem. A linear optimization problem can be defined as solving an optimization problem in which the objective function(s) and all associated constraint conditions are linear. As all linear functions are convex, linear optimization problems are intrinsically simpler and easier to solve than general nonlinear problems, in which the resolution becomes more complex and the decision space is non-convex. There are several types of nonlinear optimization problems where for many of them the objective function may have many locally optimal solutions, i.e., solutions that are optimal (either maximal or minimal) within a neighboring set of candidate solutions. Finding the best of all such minima, i.e., the global solution which is the optimal solution among all possible solutions, not just those in a particular neighborhood of values, is often difficult.

2.2.6 The uncertainty tied to the optimization model

One further possible classification of optimization problems is in terms of the randomness or uncertainty tied to the data dealt with. In this concern, two main branches can be emphasized; namely the "deterministic optimization" problems and the "stochastic optimization" problems. In deterministic optimization problems, it is supposed that the problem dealt with presents data which are known accurately. Nevertheless, in practice and in many real-world problems, the given data cannot be known with such certainty for several reasons, e.g., due to measurement and model errors, or if the data are for some predicted quantities for some period in the future. In general, deterministic optimization algorithms are unidirectional, i.e., there exists at most one way to proceed, otherwise, the algorithm gets terminated, and do not use random numbers in any step of execution. On the other hand, in stochastic optimization problems or optimization under uncertainty, the uncertainty or a concept of probability is incorporated into the model that employs at least one instruction or at least one operation that makes use of random numbers. Efficient optimization methods can be applied when the system's parameters are known with certain bounds. In such a case, the aim is to find a solution that is feasible for all data and optimal in some sense. To deal with such context, stochastic optimization models replies on the fact that probability distributions governing the data are either known or they can be estimated. Hence, the aim is to find some policy that is feasible for all (or almost all) the possible data instances and optimizes the expected performance of the model.

2.3 The multi-objective optimization problem

Nowadays, most Astro- and Geo- Informatics real-world problems that are encountered in practice often involve multiple objectives to be minimized or maximized simultaneously with respect to a set of constraints. Hence, multi-objective optimization in such fields that offer highly complex search spaces has become a standard practice. Calling for this specific class of optimization problems and its related optimization algorithms became essential, as they require little domain information to operate, they are easy to use and most importantly, they are known to be flexible. The general form of a multi-objective optimization problem is based on the same definitions of the set of constraints presented in Section 2.1, with the following objective function (Deb 2001):

$$\text{minimize } \mathbf{O}(\mathbf{x}) = [O_1(x), O_2(x), \ldots, O_M(x)]^T \tag{3}$$

The resolution of a multi-objective optimization problem yields a set of trade-off solutions, called "Pareto optimal" solutions or "non-dominated" solutions, and the image of this set in the objective space is called the "Pareto front". Hence, the resolution of a multi-objective optimization problem consists in approximating the whole Pareto front. In the following, we give the key background definitions related to multi-objective optimization:

Definition-1. Pareto optimality

A solution $x^* \in \Omega$ is Pareto optimal if $\forall x \in \Omega$ and $I = \{1, 2, \ldots, M\}$ either $\forall m \in I$ we have $O_m(x) = O_m(x^*)$ or there is at least one $m \in I$ such that $O_m(x) > O_m(x^*)$. The definition of Pareto optimality states that x^*

is Pareto optimal if no feasible vector x exists which would improve some objectives without causing a simultaneous worsening in at least another one.

Definition-2. Pareto dominance
A solution $u = (u_1, u_2, \ldots, u_n)$ is said to dominate another solution $v = (v_1, v_2, \ldots, v_n)$ (denoted by $O(u) \preceq O(v)$) if and only if $O(u)$ is partially less than $O(v)$. In other words, $\forall m \in \{1, 2, \ldots, M\}$ we have $O_m(u) \leq O_m(v)$ and $\exists m \in \{1, 2, \ldots, M\}$ where $O_m(u) < O_m(v)$.

Definition-3. Pareto optimal set
For a given multi-objective optimization problem $O(x)$, the Pareto optimal set is $P^* = \{x \in \Omega | \neg \exists x^{'} \in \Omega, O(x^{'}) \preceq O(x)\}$.

Definition-4. Pareto optimal front
For a given multi-objective optimization problem $O(x)$ and its Pareto optimal set P^*, the Pareto front is $PF^* = \{O(x), x \in P^*\}$.

When solving a multi-objective optimization problem, the aim is to find not one, but the set of solutions representing the best possible trade-offs among the objectives; the so-called Pareto optimal set.

3 Evolutionary computation

In evolutionary computation, the derived evolutionary optimization algorithms use the main principles and mechanisms inferred from the Darwinian ideas of natural selection and population which was presented in Section 1. These evolutionary mathematical models operate on a population composed of a set of individuals or chromosomes. Each individual represents a potential solution to the problem being solved and is codified according to the problem's requirements; as highlighted in Section 2.2. The goodness of an individual is represented by the objective function; and the restrictions to the problem reflect how apt that individual is to survive in that environment (Eberbach 2005). Those individuals having lower fitness value are gradually eliminated by the dominant competitors. Within a population and for each individual, probabilistic operators, typically chromosomal crossover and mutation, are applied over these individuals (parents) to produce new features in the chromosome. These new features represent new individuals (offspring) that maintain some properties of their ancestors which are conserved or are eliminated via a selection. This evolution process repeats itself during a certain number of cycles or generations where species continuously strive to reach a specific genetic structure of the chromosomes that maximize their probability of survival in a given environment. This process continues until an acceptable result is achieved, i.e., the maximal fitness solution is found. These key mechanisms of evolutionary optimization algorithms are defined in what follows.

3.1 Basic structure of an evolutionary algorithm

When defining any evolutionary algorithm, the first steps are usually the most critical ones. These comprise the encoding of the candidate solution and the definition of the objective function(s). The encoding and the fitness function(s) are tied to a specific problem, and hence, adequate choices must be taken to guarantee the success of the algorithm. To make these choices, knowledge about the problem dealt with and about the expected solution should be used. Once the problem is defined, the next step is the application of the evolutionary algorithm itself. A basic representation of the different steps of a general evolutionary algorithm is shown in Figure 1.

The algorithm starts with creating randomly an initial population of chromosomes of size N. The objective function is then evaluated for each chromosome to determine the chromosome's fitness. Based on this fitness value, a termination criterion is evaluated specifying if a solution of the desired quality was found or a specific number of iterations was run. In case where the termination criterion is not satisfied, some chromosomes will be selected (some parents of size $N/2$) and then reproduced via the crossover and the mutation mechanisms, resulting in offspring. The new chromosomes will replace the old ones producing a new generation, i.e., the parents. This process continues until the termination criterion is satisfied. Finally, the fittest chromosome will be decoded, producing the best solution of the problem.

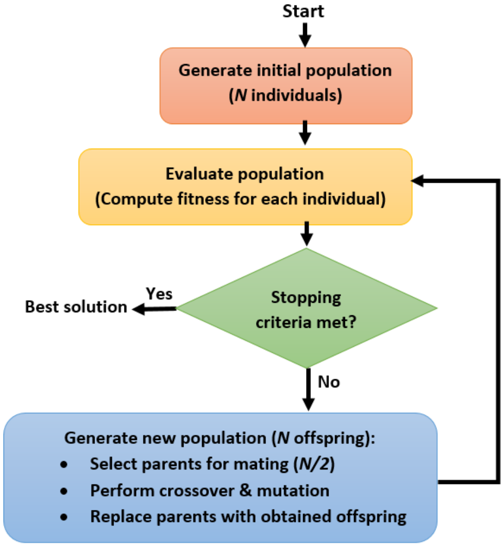
Start
Generate initial population
(N individuals)
Evaluate population
(Compute fitness for each individual)
Stopping
criteria met?
Yes
Best solution
No
Generate new population (N offspring):
• Select parents for mating (N/2)
• Perform crossover & mutation
• Replace parents with obtained offspring

Figure 1: Flowchart of a basic evolutionary algorithm.

3.2 Evolution operators

As presented in Figure 1, the key components of any evolutionary algorithm are selection, crossover and mutation. Each of these components can be realized in a number of different ways. In what follows, we will give a general overview on how to perform these basic genetic operators.

3.2.1 Selection

Selection operators are used to select a proportion of the existing population, the individuals, with a pre-defined probability to create the basis for the next new generation. It is a process designed to ensure that promising solutions of the population get a greater probability to be selected for mating, and hence forcing the population to improve over time. There are several selection operators that have been proposed in literature (Goldberg & Deb 1991) (Blickle & Thiele 1996), and it is worth mentioning that a number of studies have been conducted to compare them. In (Blickle & Thiele 1996), it has been concluded that the best selection operator is problem dependent. In the following, we will highlight the most common selection operators used in practice, mainly in Astro- and Geo-Informatics real-world problems.

Roulette wheel selection In roulette selection, each member of the population is selected according to its fitness value. The higher the fitness, the higher the probability of being selected. In such a process, and within a given population P and for each individual x_i, the related selection probability $P_s(x_i)$ is defined as:

$$P_s(x_i) = \frac{O(x_i)}{\sum_{j=1}^{N} O(x_j)}. \tag{4}$$

The population is then mapped onto a roulette wheel, where each chromosome x_i has its $P_s(x_i)$ proportional slice of area. To select an individual, a random number is generated and the individual whose slot spans the random number is then selected. The wheel is then spun N times, where N is the cardinality of the population. This is to create the next generation of parents that will undergo the simulated reproduction process of crossover and mutation.

Tournament selection In a tournament selection, a user-defined number of chromosomes or tournament pool is chosen at random from the current population. The chromosomes in this tournament pool compete with each other and the one with the best fitness value is selected to be a parent for production of the next generation.

Ranking selection The ranking selection approach is based on the idea that individuals are sorted according to their fitness values. Once they are sorted, rank-based weights are assigned to each chromosome from one generation to another in a way that the rank of each chromosome defines how likely it is to be selected to be a parent for the next new generation. In literature, there are several linear and exponential rank-based weighting schemes that were proposed (Blickle & Thiele 1996) and that can be applied with respect to the targeted context. After the application of a specific selected rank-based weighting function to the sorted population, simple roulette or any other selection operator can be used.

Truncation selection Similar to the ranking selection approach, truncation selection also uses a sorting technique to gain knowledge of the rank order of a given population. More precisely, the truncation selection operator truncates the population by only looking at a fixed number of the highest performing chromosomes. From this specific subset, a random selection technique can be applied where each highest performing chromosome has an equal chance of being selected (Mühlenbein & Schlierkamp-Voosen 1993). The truncation selection operator has the drawback of additional computational complexity due to the requirement of ranking the population based on the fitness value during each generation.

Supplement to selection: Elitism In some cases, due to the application of genetic operators, valuable genetic information may be destroyed during the search. When such case occurs, there is no guarantee whether the evolutionary algorithm will be able to rediscover these lost information or not. To prevent such lost, the concept of elitism is introduced. Elitism retains the best members, i.e., a small proportion of the fittest individuals, from the current population and ensures that they pass onto the new generation so that the valuable information remains intact within the population, and hence reduces the genetic drift. It is important to mention that the degree of elitism should be carefully adjusted. This is because a higher proportion, for instance, may cause a rise in selection pressure and hence lead to a premature convergence. On the other hand, elitism is a very useful practice that can significantly increase the performance of any evolutionary algorithm that uses any selection technique.

3.2.2 Crossover

Based on the evolutionary algorithm selection process, the most fit chromosomes should be always selected. However, it is possible that these chromosomes may be represented several times in the upcoming generations; hence leading to a population that is entirely composed of a number of copies of the same candidate solution. If the initial population is not large enough then this might be a problem as there is no guarantee that the initial population contains a global optimal solution, or even a solution that is considered good enough for the problem being solved. In this case, the evolutionary algorithm will converge to a population filled with duplicates of the best solution that is originally attained in the initial population. To overcome this limitation, crossover operators as reproduction techniques were introduced and are considered as key components of any evolutionary algorithm to efficiently evolve populations toward optimal points. A detailed study of these can be found in (De Jong & Spears 1990). In what follows, we will elucidate the most commonly used crossover operators for binary encoding.

Single-point crossover Single-point crossover is a technique where the selected parent population, the two mating chromosomes, are cut at a randomly selected location; called the pivot point or crossover point. At this cut, the genetic information to the left (or right) of the point is swapped between the two parent chromosomes to produce two offspring chromosomes (children). The single-point crossover operator is illustrated in Figure 2.

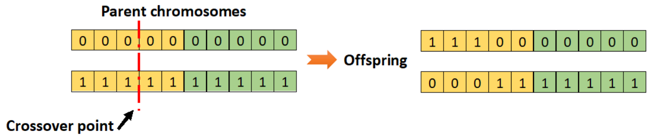

Figure 2: A single-point crossover operator.

Multi-point crossover Contrary to the single-point crossover, multi-point crossover works with more than one pivot point. This increases the extent of disruption of the original parent chromosomes. As described in Figure 3, the genetic information between two selected crossover points that is to the right of an even (or odd) number of pivot points is swapped to produce two unique offspring individuals.

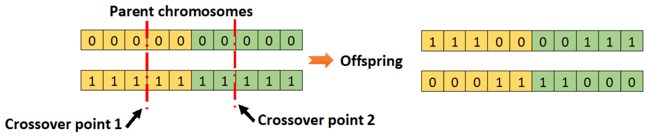

Figure 3: A multi-point crossover operator.

Uniform crossover The uniform crossover operator looks at one specific gene at a time. It produces a random crossover-vector filled with binary values where 1 indicates that a specific gene location is swapped between parents, and 0 indicates that each parent retains that specific gene. This process can be seen in Figure 4.

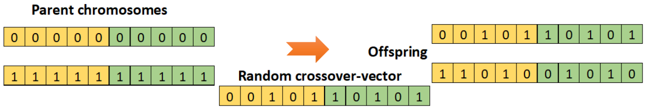

Figure 4: A uniform crossover operator.

3.2.3 Mutation

The mutation operator typically occurs after one of the crossover techniques has been applied to the chosen parent chromosomes. This genetic process perturbs one or more components (genes) of a selected chromosome and is regulated by a predefined mutation probability. Mutation is mainly applied to restore lost information or import unexplored genetic components into the population in order to distribute solutions widely across the search space and therefore avoid premature convergence.

Different mutation schemes were proposed in literature such as the bit-flipping where the operator simply flips a bit from 0 to 1 or from 1 to 0 with certain probability. Figure 5 illustrates a simple bit-flipping mutation process. The bit-flipping mutation can be generalized to mutate strings of any alphabet. Another scheme is the uniform mutation where this operator replaces the value of the chosen gene with a uniform random value selected between the user-specified upper and lower bounds for that gene. The shrink mutation operator, for instance, is another scheme which adds a random number taken from a Gaussian distribution with mean equals to the original value of each decision variable characterizing the entry parent vector. Some other mutation operators can be found in (Voigt & Anheyer 1994).

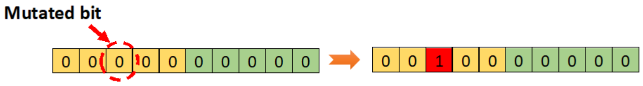

Figure 5: A bitwise mutation operator.

4 Evolutionary computing metaheuristics

Based on the discussed concepts of evolutionary computation, numerous possibilities arise to design and implement advanced evolutionary algorithms, i.e., metaheuristics which are able to efficiently explore complex search spaces in order to solve complex optimization problems. In literature, a wide range of different evolutionary optimization algorithms was proposed and a full review of these can be found in (Deb 2001) (Talbi 2009), which is out of scope of this chapter. All of these algorithms are similar in their basic approach and in making use of the evolutionary concepts, but they mainly differ in the way they represent the information. In what follows, we will give an overview of the main evolutionary optimization metaheuristics that are the most commonly used in Astro- and Geo- real-world problems. We will give a general descriptions of the algorithms and with respect to the specific problem requirements, the metaheuristic can be designed for single-objective problems, multi-objective problems or for any other type of optimization algorithm as discussed in Section 2.2. We will also emphasize some other evolutionary approaches and bio-inspired algorithms that have much to offer to Astro- and Geo- Informatics but as relatively new paradigms, they were not commonly applied to solve Astro- and Geo- optimization problems. This will be further discussed in Section 7.

4.1 Genetic algorithm

Genetic Algorithms (GAs) have been firstly proposed in (Holland 1992) to understand the adaptive processes of natural systems. After that, they have been applied to solve optimization and machine learning problems (Holland & Goldberg 1989) (De Jong 2005). The classical versions of GAs use a binary representation, i.e., the chromosome representation is based on a binary string of fixed length. However, nowadays, GAs make use of several other types of representations, e.g., nominal-valued discrete variables where each nominal value is encoded as a bit string, integer or real-valued representations, order-based representations or chromosomes of variables length and many more (Chambers 2000).

For the selection process, the algorithm uses a probabilistic selection that is originally the roulette wheel operator. A replacement selection is also performed, i.e., the selection of survivors of both the parent and the offspring populations. Specifically, a generational replacement is used where the parents are replaced systematically by the offspring. Concerning the reproduction process, it is traditionally made via the crossover and the mutation operators with a fixed probability for each of them. However, the algorithm emphasises more the importance of the crossover operator over the mutation operator. The crossover operator is based on the single/multi-point or uniform crossover while the mutation is generally bit-flipping.

4.2 Evolutionary strategy

Evolution Strategies (ESs) were originally proposed in (Rechenberg 1981). Unlike GAs which are mostly applied to discrete optimization problems, ESs are mostly applied to continuous optimization where representations are based on real-valued vectors. The first ES applications (Klockgether & Schwefel 1970) include real-valued parameter shape optimization. They usually apply the elitism concept and a Gaussian distributed mutation. Crossover is rarely applied. In ES, the representation of an individual is made by its genetic material and by a so-called strategy parameter which defines the behavior of the individual in its related environment. The genetic material is represented by floating-point variables while the strategy parameter is, generally, defined by the standard deviation of a Gaussian distribution associated with each (variable of) individual.

In many ESs, the selection operator is mainly deterministic and is based on the fitness ranking. Two types of mutation operators are commonly used namely the discrete mutation, in which the gene value of the offspring is the gene value from the parent, and intermediate mutation in which the midpoint between the gene value of the parents gives the gene value of the offspring. The mutation operator has a special implementation in ES as it mutates both the strategy parameter and the genetic material. Hence, the evolution process evolves the genetic material and the strategy parameter at the same time; and accordingly, ES is considered to be a “self-adaptive” mechanism (Meyer-Nieberg & Beyer 2007). The main ESs advantage is their efficiency in terms of time complexity.

4.3 Evolutionary programming

Evolutionary Programming (EP) was proposed in (Fogel et al. 1966). This paradigm emphasizes on mutation and does not use crossover. Classical EPs have been developed to evolve finite state automata in such a way that they were capable of solving time series prediction problems and more generally evolving learning machines (Fogel et al. 1966). Modern EPs have been later applied to solve continuous optimization problems using real-valued representations. Same as ESs, EPs use Gaussian distributed mutations and the self-adaptation paradigm. In EP, the parent selection mechanism is deterministic, and the survivor selection process (replacement) is probabilistic and is based on a stochastic tournament selection (Eiben et al. 2003).

4.4 Genetic programming

Genetic Programming (GP) (Koza 1992) is seen as an extension of the generic model, GA, in which the structures in the population are not fixed length strings that encode candidate solutions to a problem (linear representation) but programs expressed as syntax trees, i.e., nonlinear representation based on trees. In (Koza 1992), computer programs (solutions) were encoded using LISP and their representations are S-expressions where the leaves are terminals and the internal nodes are operators (functions). The definition of the leaves and the operators are strictly tied to the targeted application being solved.

In GP, generally, the parent selection is a fitness proportional and the survivor selection is a generational replacement. The crossover operator exchanges parts of two parent trees resulting in two new trees and the mutation randomly changes a function of the tree into another function, or a terminal into another terminal. One of the main problems in GP is the uncontrolled growth of trees which is a phenomenon called “bloat”. Indeed, GPs need a huge population and then they are very computationally intensive.

4.5 Other evolutionary algorithms and bio-inspired approaches

Several other Astro- and Geo- real-world problems are based on some other evolutionary approaches. Among these, we can mention differential evolution, co-evolutionary algorithms, and some other nature-inspired metaheuristics such as swarm intelligence and artificial immune systems which may also be used to solve complex optimization problems.

4.5.1 Differential evolution

Differential evolution (Storn & Price 1997, Price et al. 2006) is one of the most successful metaheuristics for continuous optimization. The algorithm uses a population of parameter vectors that encode the problem and are initially chosen uniformly random from the search space. During each generation an update is attempted for each vector in the following way, where the vector that we try to update is called a target vector. First, a mutation vector is created by randomly choosing three parameter vectors from the population, different from the target vector, and adding a scaled difference from the first two to the third vector. The difference is scaled using a mutation scaling factor. A crossover operation is then performed between the obtained mutation vector and the target vector to create a trial vector. In the selection operation, if the trial vector has a better fitness than the target vector, a replacement is made for the next generation. This idea has been integrated in a novel crossover operator of two or more solutions and a self-referential mutation operator to direct the search towards good solutions (Talbi 2009). Several differential evolution variants are possible where these algorithms mainly differ in the way parents are selected and in the form in which crossover and mutation take place. A detailed survey of various differential evolution algorithms can be found in (Das & Suganthan 2011) (Das et al. 2016).

4.5.2 Co-evolutionary algorithms

A co-evolutionary algorithm is based on the concept of natural complementary evolution of closely allied species (Durham 1991). In nature, various species represented by a collection of similar individuals coevolve based on their phenotype. A co-evolutionary algorithm, unlike traditional evolutionary algorithms where a population is composed of a single species, may be seen as a competitive-cooperative paradigm that involves different interacting populations where each represents a given species, and together optimizing coupled objectives. The competitive co-evolutionary approaches rely on the idea that different populations compete in solving the global problem. In such schema, the individual fitness defines a competition. Each population aims at minimizing a local cost specified by a local fitness function. The competition between different populations leads to an equilibrium in which the local objectives cannot be improved and hopefully the global objective is achieved (Talbi 2009). On the other hand, the cooperative models reflect the cooperative behavior of the various populations to solve the problem. In such schema, a population evolves a subcomponent of the solution. Based on the cooperative interaction between the populations, a global solution arises from the assembled species' subcomponents. The fitness of an individual of a given species will depend on its ability to cooperate with individuals from other populations (Talbi 2009).

4.5.3 Swarm intelligence

Swarm intelligence is an innovative intelligent paradigm which was successfully applied to solve optimization problems. It took its inspiration from the collective behavior of social swarms in nature such as flocks of birds, honey bees, schools of fish, and ant colonies (Bonabeau et al. 1999). Specifically, swarm intelligence is based on the common comportment of these species that compete for food. The main features of swarm intelligence based algorithms are their simplicity and their particle aspect. They are based on agents, i.e., insects or swarm individuals, which are relatively unsophisticated and which cooperate together, by doing movements in the decision space, to achieve tasks necessary for their survival. Among the most effective swarm intelligence based algorithms used in Astro- and Geo- real-world optimization problems are ant colony and particle swarm optimization. These will be detailed in what follows.

Ant colony optimization algorithms Ant Colony Optimization (ACO) algorithms are based on the idea of imitating the foraging behavior of real ants to solve complex optimization tasks such as transportation of food and finding shortest paths to the food sources (Dorigo & Di Caro 1999). In nature, ants communicate by means of chemical trails; called “pheromone”. This substance assists ants in finding the shortest paths between their nest and food. In a natural observation, ants usually wander randomly. When they find food, they return to their nest while laying down pheromone trails on the ground. This chemical, if found by other ants, will not keep them wander at random, but will help them to follow the trail and to quickly return to their nest, i.e., this trail will guide the other ants toward the target point. Meanwhile, these ants will reinforce the path if they find food. However, as ants have to travel the path back and forth and as the pheromone has to evaporate, the path becomes less prominent. In such situation, ants will look for the path having a higher density of pheromone. This means that this particular path was visited by more ants and is definitely the shortest path to take. Based on this inspiration, ant colony optimization algorithms can be seen as multi-agent systems in which each single agent is inspired by the behavior of a real ant. In literature, there are numerous successful implementations of the ACO metaheuristic. A review of their applications to a wide range of different optimization problems can be found in (Dorigo & Stützle 2003).

Particle swarm optimization Another successful swarm intelligence model is particle swarm optimization (Kennedy & Eberhart 1995, Clerc 2010). It draws inspiration from the sociological behavior of natural organisms such as bird flocking and fish schooling to find a place with sufficient food. Within these swarms’ populations, a synchronized behavior using local movements emerges without any dominant control. Each individual within its community (population) is moved to a good area based on its fitness for the environment, i.e., its flexible velocity (position change) in the search space. Indeed, based on the particle’s memory, the best position the individual has ever visited in the search space is remembered. Following this natural observation, the movements of swarms is seen as an aggregated acceleration towards their best previously visited position and towards the best particle of a topological neighborhood, i.e., the social influence between the particles. This phenomenon led to several efficient particle swarm optimization algorithms which are mainly applied to solve optimization problems (Clerc 2010).

We describe one possible PSO implementation similar to the original one proposed in (Kennedy & Eberhart 1995). The particles are randomly scattered at the beginning in an n-dimensional space. Each particle is characterized by its position C_i and its velocity v_i that is initially zero. Each particle remembers its fittest value and position P_i and also the fittest value and position G_i from the entire swarm and at each iteration every particle updates its velocity as $v_i(k+1) = \alpha v_i(k) + \beta \text{rand}()(P_i - C_i) + \gamma \text{rand}()(G_i - C_i)$, which is then added to its current position. The function rand() generates a random number from the range $(0, 1)$, while the parameters α, β, and γ can be used to balance between the inertial, cognitive and social influences.

4.5.4 Artificial immune systems

The study and design of Artificial Immune Systems (AISs) represent a relatively new area of research that tries to build computational systems that are inspired by the natural immune system (De Castro & Timmis 2002). This growing field has been mainly applied to data mining problems (DasGupta 1993), but lately, its application to optimization problems is rapidly increasing (Chandrasekaran et al. 2006) (Cutello & Nicosia 2002) (Coello & Cortés 2005).

AISs are based on the human immunological concepts. The natural immune system is a network of cells, tissues, and organs that work together to defend the body against attacks by “foreign” invaders that are trying to do it harm. This main task is achieved thanks to its capability to recognize the presence of infectious foreign cells and substances, known as “non-self” elements and to respond to them by eliminating them or neutralizing them. This distinction between the “non-self” and the body’s “self” cells is based on a process called “self-non-self discrimination” (Janeway Jr 1992). The non-self elements, also called “antigens”, are mainly microbes; tiny organisms such as bacteria, parasites, and fungi. All of these can, under the right conditions, cause damage and destruction to parts of the body and if these were left unchecked, the human body would not be able to function appropriately. Thus, it is the purpose of the immune system to act as the body’s own army. More precisely, the immune system does not rely on one single mechanism to deter invaders, but instead uses many strategies. The main division between the strategies is that between innate immunity and adaptive immunity. The innate immunity is the first line of defense against invading antigens. It is those parts of the immune system that work no matter what the damage is caused by. They are always at work and do not need to have seen the offending invader before to be able to start attacking it. The innate immune system includes anatomical barriers, secretory molecules and cellular components. In addition, the innate immune system employs a different group of cells, e.g., Antigen Presenting Cells (APCs), to eliminate threats or to interact with the rest of the immune system. The second line of defense is the adaptive immune system which affords protection against re-exposure to the same pathogen. The adaptive immune system is called into action against pathogens that are able to evade or overcome innate immune defenses. The cells of the adaptive immune system are mainly the B-cells and the T-cells, but there are also other important parts of the adaptive immune system, such as the complement cascade and the production of antibodies. These mentioned elements of the immune system do not work separately, but all work together in a co-operative fashion via specific immune proteins called the "cytokines". An inspiration from these remarkable immune properties led to the conception and the design of artificial immune systems exhibiting similar functionalities. These systems are discussed in what follows.

Clonal selection theory Clonal selection theory (Burnet et al. 1959) is used to clarify the basic response of the adaptive immune system to antigenic stimulus. Clonal selection involves two main concepts which are cloning and affinity maturation. More precisely, it establishes the idea that only those cells capable of recognizing an antigen will proliferate while other cells are selected against. Clonal selection calls both B- and T-cells. When B-cells antibodies bind with an antigen, cells become activated and differentiated either to be plasma cells or memory cells. The closer the matching between an antibody and a specific antigen is, the stronger is the bind. This property is called affinity. Plasma cells make large amounts of a specific antibody that work against a specific antigen to destroy it. Memory cells remain with the host and promote a rapid secondary response. However, before this process, clones of B-cells are produced and undergo somatic hypermutation. Consequently, diversity is introduced into the B-cell population. Moreover, a selection pressure is performed, which implies a survival of the cells with higher affinity. Let us notice that clonal selection is a kind of an evolutionary process; where an antibody represents a solution, the affinity defines the fitness function, and the antigen represents the value of the objective function to optimize. The cloning process is seen as the reproduction of solutions, the somatic hypermutation represents the mutation of a solution and the affinity maturation represents the mutation and the selection of best solutions. Based on this theory, various clonal selection algorithms have been proposed in literature and most of them are devoted to optimization problems. A detailed description and comparison of AIS clonal selection algorithms can be found in (Ulutas & Kulturel-Konak 2011).

The self-non-self theory The self-non-self theory is able to tell the difference between what is foreign and potentially harmful, and what is actually a part of its own system. The representative self-non-self theories are the negative selection and the positive selection. The purpose of the negative selection theory is to provide tolerance for self cells. During the generation of T-cells, receptors are made through a pseudo-random genetic rearrangement process.

Then, they undergo a censoring process in the thymus, called the negative selection. There, T-cells that react against self-proteins are destroyed; thus, only those that do not bind to self-proteins are allowed to leave the thymus (worse solutions are removed to get optimal solutions). These matured T-cells then circulate throughout the body to perform immunological functions and protect the body against foreign antigens. As for the positive selection theory, it works as the opposite of the negative selection process. An inspiration from the negative selection and positive selection theories gave rise to numerous AIS algorithms which are mainly used for classification to solve optimization problems (Cao et al. 2007) (Gao et al. 2008).

Immune Network Theory The immune system is a network of cells and antibodies that have a profound sense of self and the ability to remember and learn. The immune network theory states that the recognizers of the immune system, the B-cells and antibodies, not only recognize foreign particles but also recognize and interact with each other. This created network is based on interconnected B-cells for antigen recognition. The strength of the B-cells connections is directly proportional to the affinity that they share. Indeed, B-cells can both stimulate and suppress each other in order to stabilize the network. These characteristics of immune network not only maintain the diversity of the antibody population effectively but also facilitate self-organization and regulation in the biological immune system. Basic concepts of the immune network theory are implemented leading to several immune algorithms dedicated to solve optimization problems (Hajela & Yoo 1999).

Danger Theory The Danger Theory (Matzinger 2001), is a new theory which has become popular amongst immunologists. It was proposed to explain current anomalies in the understanding of how the immune system recognizes foreign invaders. The central idea in the danger theory is that the immune system does not respond to non-self but to danger. Thus, just like the self-nonself theory, it fundamentally supports the need for discrimination. However, it differs in the answer to what should be responded to. Instead of responding to foreignness, the immune system reacts to danger based on environmental context (signals) rather than the simple self-non-self principle. Specifically, the dangerous antigens stimulate the production of danger signals by stimulating cellular stress or cell death. Those signals are recognized by APCs that recognize these signals and based on this phenomena the immune system detects the danger zone and then evaluates the danger. By defining the danger zone to calculate the danger signals for each antibody, the algorithm adjusts antibodies' concentrations through its own danger signals and then triggers immune responses of self-regulation. Consequently, the population diversity can be maintained.

5 Parallel evolutionary computing metaheuristics for Big Data

In real-world applications, optimization problems are usually NP-hard and are CPU time and/or memory consuming. Hence, the application of metaheuristics is essential as the algorithms help in finding the appropriate solutions to the problem being solved while reducing the computational complexity of the search process considerably. Although the use of metaheuristics permits such gain, the used objective functions and the constraints resource requirements (e.g., CPU, memory) remain intensive, specifically when the size of the search space becomes huge. To deal with these limitations, in recent years, several parallel and distributed computer architectures have been emerged for the design of metaheuristics (Alba & Tomassini 2002).

The design of parallel metaheuristics can be realized in different ways. However, it can be categorized into three main levels, namely the algorithmic level, the iteration level and the solution level. In the algorithmic level, metaheuristics can be either independent or cooperative. If the metaheuristics are independent, i.e., the different metaheuristics are executed without any cooperation; the search will be equivalent to the sequential execution of the metaheuristics in terms of the quality of solutions. Nevertheless, in the cooperative model, i.e., the different algorithms are exchanging information related to the search with the intent to compute better and to have solutions that are more robust; the behavior of the metaheuristics is altered to enable the improvement of the quality of solutions. On the other hand, in the iteration level, the main idea is the parallelization of each iteration of metaheuristics. This design is based on the distribution of the handled solutions; where the behavior of the metaheuristic is not altered. More precisely, it is the generation and the evaluation of the neighborhood (or candidate solutions) which is done in a parallel way as this step presents the most computation intensive part of the metaheuristic. The same design can be also based on the distribution of the population. In this case, the operations commonly applied (crossover and mutation) to each of the population elements are performed in parallel. The main objective of the iteration level is to speed up the algorithm by reducing the search time. The third design is the solution level where the main focus is the parallelization of the evaluation of a single solution (objective and/or constraints) of the search space. Similar to the iteration level, in this design, the behavior of the metaheuristic is not altered.

Several parallel computer architectures have been proposed in recent years to represent an effective strategy for the design and implementation of parallel metaheuristics. Among the most popular architectures, we mention the master-slave model which is mainly used within the iterative level and the solution level, the island model and the cellular model. The two later architectures are the most widely known parallel algorithmic-level models and the mostly used for evolutionary algorithms. In the master-slave model, a single machine represents the master and it distributes the workload for executing operations to several other machines named “slaves”. As the selection and the replacement are usually sequential procedures, as they need a global management of the population, it is the master who performs these tasks. The operators like mutation, crossover, and the evaluation of the fitness function are performed by the associated slaves. This is because these operations can be performed independently and they are often among the most expensive operations. In this model, the master sends the partitions, i.e., the subpopulation, to the workers who in turn return the newly evaluated solutions to the master.

In the island model, which is also called “distributed evolutionary algorithms”, the population of each run is considered as an island. Some research papers define island models based on subpopulations that together form the population of the whole island model. In island models, the evolution of the island happens in an independent manner. However, sometimes, solutions are exchanged between the different islands in a process called migration. The main idea is to create this migration topology which is seen as a directed graph with islands as its nodes and directed edges connecting two islands. Selected individuals from each island are sent off to neighboring islands, i.e., islands that can be reached by a directed edge in the topology. These individuals, named migrants, are involved in the target island after an additional selection process. In this way, the different islands can communicate, exchange, and compete with one another. When some islands get stuck in low-fitness regions of the search space, they will be taken over by individuals from other more successful islands. This helps to coordinate search, to focus on the most promising regions of the search space, and to use the available resources efficiently.

The other well-known parallel model for evolutionary algorithms is the cellular model. Cellular models can be seen as a special case of the island models with a more fine-grained form of parallelization. In such model, the island is composed of a single individual, and is called a cell; which explains the term cellular evolutionary algorithms. Similar to the island models, the cells in cellular models are connected by a fixed topology, e.g., rings, torus graphs, etc. Each individual in a cell is only allowed to mate with its neighbors in the topology. This communication happens in every generation. The main difference to island models is that there is no evolution that happens in the cell itself. The improvements can only be obtained by cells interacting with each other. However, it is possible that an island can interact with itself. In this design, the overlapped small neighborhood helps in exploring the search space because a slow diffusion of solutions through the (sub)population provides a kind of exploration, while exploitation takes place within each neighborhood. Figure 6 shows a general scheme of a basic island model with 6 islands (on the left), and two cellular models reflecting a torus graph (on the right) and a complete graph (on the middle).

The discussed parallel and distributed computing architectures as well as many others have the ability to reduce the search time and hence helping designing real-time and interactive optimization methods, improve the quality of the obtained solutions, improve the robustness in terms of solving in an effective manner different optimization problems and different instances of a given prob-

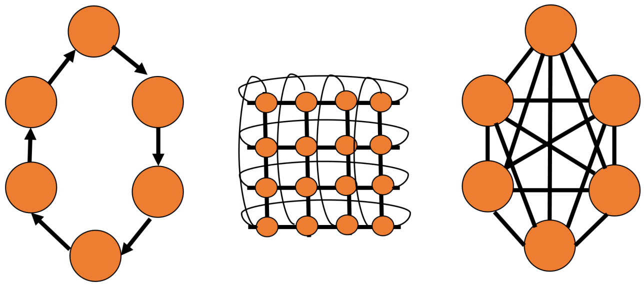

Figure 6: The traditional parallel island (on the left) and cellular models for evolutionary algorithms (on the right).

lem, and allow to solve large-scale instances of complex optimization problems. A review of the distributed computer architectures, of parallel evolutionary algorithms and their characteristics can be found in (Alba 2005, Alba et al. 1999, Gong et al. 2015).

6 Practical applications of evolutionary computing metaheuristics in the context of Astro- and Geo- Informatics

Evolutionary computing has a various range of practical applications across many different research areas and in this section we show some examples from Astro- and Geo- Informatics as well as some other related fields. For example, in (Arias-Montano et al. 2012) the reader can find many applications of evolutionary computing in Aeronautical and aerospace engineering. Our goal is to motivate the reader to indulge in application of the algorithms and to provide an insight into their characteristics when applied to real-world problems. We have grouped the various applications by the algorithm that is used, although some of the works examine or combine several different algorithms. For instance, one interesting work (Civicioglu 2012) addresses the problem of mapping geocentric Cartesian into geodetic coordinates with many different metaheuristics and proposes a new one that outperforms all the other for that particular application.

Genetic algorithm From all methaheuristics, genetic algorithms (GA) are probably the most applied. One popular article demonstrating the application of GAs to astronomy and astrophysics is (Charbonneau 1995). The article describes three different applications of GAs: fitting rotation curves in galaxies, determining pulsation periods from star's spectral lines and creating wind models for certain solar-type stars. All examples use single point crossover, uniform mutations, ranking selection, and all variables are encoded using integers. The first problem is representing rotation curves in galaxies from available data. A rotation curve is the variation of velocity around the center of the galaxy, which can be modeled as a function of the distance from the galaxy center. The velocity usually has four components: bulge, disk, interstellar gas and halo components. Brightness profiles can be determined from observations for the first two components, which can be converted into mass using mass-to-light ratios that are left as two adjustable parameters. The halo component requires two parameters, a velocity dispersion and a characteristic length scale. Two constraints strategies have been examined and resulted in different solutions, one with only positivity requirements, and another with the parameters constrained in particular positive ranges. The parameters are then learned by applying a genetic algorithm with a population of 100 individuals over 100 generations, performing crossover with a rate of 0.65 and a mutation rate of 0.003. The second problem falls within the wider class of fitting time-series data, which is often done in astronomy. The specific presented problem is fitting time-series data to a multi-periodic signal that is a sum of sinusoidal functions. Each of these functions are characterized by an amplitude, phase and frequency, which all together constitute a set of parameters that need to be determined. The authors have used a "cleaning algorithm" in which they first identify the dominant periodical function using power spectra analysis and fit it to the data. Then, the previously fitted function is subtracted from the original data and the procedure is repeated until the remaining data can be considered as noise. The method has been applied to pulsation periods in δ Scuti stars using seven sinusoidal components and a genetic algorithm with a population of 200 individuals over 1 000 generations with a variable mutation rate in conjunction with elitism. The third problem consists of finding solutions to a magnetohydrodynamic wind model, where an initial root finding problem is transformed into a minimization problem with six parameters, which can be than addressed appropriately using a population of 100 individuals over 500 generations with variable mutation rate and elitism. All three problems have been solved using a similar approach with very little code changes from one to another. Later, this approach have been also successfully applied in a parallel manner to study white dwarfs and extract their physical and structural characteristics by observing their pulsation frequencies (Metcalfe & Charbonneau 2003). The parallelization is done using the Master-slave model, where the master process performs all the calculations related to the GA, while the slave processes evaluate the fitness function.

In (Orfila et al. 2002), the authors have demonstrated how GA can be applied in developing a dynamical model of the solar cycles to monthly and yearly observations, which in turn can be used to predict its long-term future behavior. The authors try to extract the structure of the model from the observed data. Therefore, they generate a population of 120 equations with random combinations of state variables, parameters and the four basic mathematical operations, and try to find the best possible description using a genetic algorithm over 10 000 generations. A regular roulette wheel selection is employed, with a small percentage of uniform mutations and the problem is encoded using character strings.

Another work (Wahde 1998), has explored how these algorithms can be used for estimation of a large number of parameters in a pair of interacting galaxies.

The interactions can be modeled using 11 parameters, or in a simplified version with 7. A particular example for NGC 5 194 and NGC 5 195 was given in (Wahde & Donner 2001), which uses a population of 50 individuals over 70 generations (higher numbers do not bring improvements), with a single-point crossover, uniform mutations with a constant rate, a rank based selection and an elitism, in order to solve the problem encoded using integers.

In (Vachier et al. 2012), a GA was successfully applied for deriving the orbital parameters for several binary asteroid systems, such as 22 Kalliope, 3 749 Balam and 50 000 Quaoar. First, a simple Keplerian model is applied to do a wide-space search to find an initial solution. Then, the whole physical dynamical system behavior is represented as a more detailed N-body problem, which is fitted to observations with a search starting from the previous solution. The algorithm uses a population of 60 individuals, a rank-based selection that takes the fittest 40 and performs a crossover to create a new offspring population in which a mutation is induced in 2 individuals. The authors also introduce search space resizing that resembles a cataclysmic extermination in which after about 20 generations all individuals are exterminated and the search space is shrank by discarding the parts without any individuals. The algorithm terminates if a predefined acceptable error range is reached, if the fitness difference between the best and worst individual is below 10^{-2}, or a predefined maximum number of generations have passed (between 500 and 1 000).

GAs have been used for various problems in geoscience. In (Maulik & Bandyopadhyay 2003), a GA is used for land cover fuzzy clustering with a dynamic number of clusters from remote sensing imagery and the algorithm have been applied for Calcutta and Mumbai, India. The cluster membership is represented using a matrix with elements denoting if a pixel belongs to a particular cluster in a fuzzy manner, i.e., a pixel can be a member of more clusters at the same time. The centers of each cluster are then the optimization variables whose coordinates are encoded using a string of real values with variable length as the number of clusters can vary. The objective function is to minimize the Xie-Beni index that is the ratio between the total within-cluster variation and the minimal inter-cluster center separation. The selection is done using the roulette wheel with additional elitism. The crossover operator uses a single point cut, but it is slightly different from the classical version as the string length is variable. The cluster centers are considered to be indivisible, the cut position can be different in the parents, and it is required that each offspring has at least two clusters. The mutations are uniform. The GA is run using a population of 20 individuals, crossover and mutation rates of 0.8 and 0.01, over a fixed number of 100 generations, with an upper bound of 20 for the number of clusters.

GAs can be used for efficiently training or constructing Artificial Neural Networks (ANN), which could serve various tasks. For training an ANN with a GA the network weights are the optimized variables, while the objective function is the difference between the ANN outputs and some desired outputs. Unlike the classical back-propagation algorithm, GAs can search the space of weights more exhaustively and avoid being trapped in local minima. However, we should have in mind that GAs need more computational time. In (Jain & Srinivasulu

2004), a GA was used for training an ANN representing rainfall-runoff. Beside the classical black-box approach, the authors also consider an incorporation of some physical knowledge into the ANN through additional inputs, thus, making it a kind of a grey-box model. The mapping of rainfall into runoff is a very complex nonlinear phenomenon that has two principal flow components, from the surface and from the subsurface. The grey-box model incorporates a base flow component, an infiltration component, a soil moisture accounting component and an ANN component. Both black-box and grey-box approaches are examined with data from the Kentucky River basin from 26 years. The GA is coded with real values. A tournament selection is used for choosing parents in the next generation combined with a elitism that preserves the best solution from each generation. Then, a single point crossover that allows to control how near the children are from their parents allowing a better control over the range of weight values. The mutation is bounded to some predefined ranges and its distribution is controllable. A population of 290 individuals is used with crossover and mutation rates are 0.9 and 0.01. The algorithm terminates if the error falls within some acceptable range or the maximal number of generations is reached. The authors have compared several models, both black-box and grey-box, using Back-propagation and GA training, and have concluded that a GA trained grey-box ANN outperforms the rest, particularly in estimating low-magnitude flows.

In (Yan & Minsker 2006) another approach combining GA and ANN is used for groundwater remediation design. Two case studies have been considered, one simpler hypothetical case for testing and another real world case at the Umatilla Chemical Depot in Hermiston, Oregon. The goal is to find an appropriate set up of the pump and treat system in order to minimize its costs under its current capacity until the chemical contaminants are cleaned up. The authors develop a hybrid approach called adaptive neural network genetic algorithm (ANGA). This approach uses an initial set of trial designs that are first evaluated using simulation models of flow and transport, and the results are stored in a cache. The cache stores the evaluations of the design fitness, to avoid recalculation of the designs if they appear again. The existing designs are then used for creation of a new generation of designs by a GA. After a large number of simulations, the evaluation of new designs can be done using ANNs instead of a simulation model. Unlike in the previous example, here the ANNs are trained using the Back-propagation algorithm, and the GA is used to combine completely different model designs. The ANNs are initially trained and then retrained completely when required. We shortly describe the GA set up only for the real world example. A combination of tournament and roulette wheel selection is used. The population consists of 160 individuals with crossover and mutation rates of 0.5 and 0.00625. The algorithm is considered to have converged if the difference between individuals of the same generation is within some range, and in this case typically that has varied between 60 and 80. Most of the constraints are included as linear penalty terms in the cost function, while one of them is enforced directly.

In (D'Ambrosio et al. 2006) parallel GAs have been used for estimating the parameters in a cellular automata model of debris flows, which have been then used for analyzing the May 1998 Curti-Sarno case in Italy. The fitness function takes values from $[0, 1]$ and it shows how similar the simulated events are to the actual areal observations. The problem was encoded into the GA directly by representing the parameters using real values, which were constrained by previous experience from manual experimentation. The population consists of 200 individuals over 100 generations, and from each generation to the next, only the worst 15 are replaced, which is a steady state and elitist evolution. The new offspring is obtained by a binary tournament selection without replacement, where in each comparison the fitter individual has a fixed probability of 0.6 for winning. The crossover is single point with a rate of 0.8, while the mutation is uniform with a rate of 0.125. The authors have also parallelized the GA for execution over several CPUs using a Master-slave model, where the master processor performs all the operations except for the calculation of the fitness values, which is done by the other slave processors. Typically the fitness calculation is the most time-consuming so it is the operation that benefits the most from parallelization, although, the other operations could be parallelized as well. The rest of the technical details about the parallelization can be found in (D'Ambrosio et al. 2006).

Many geoscience related engineering problems have benefited from the application of GAs. One example are adaptation and mitigation strategies in water supply systems to the world's climate change, which take into account Green House Gas (GHG) emissions (Paton et al. 2014). As a case study, the Southern Adelaide System in South Australia has been analyzed, which consists of three local reservoirs and water brought from a distant river, with additional potential of including three desalination plants, reusing stormwater and introducing home rainwater containers. The studied problem involves minimizing the system's vulnerability, cost and GHG emissions, under some reliability and availability constraints. In order to evaluate many different design alternatives the system's behavior was simulated using a water resources model named WaterCress that incorporates supply and demand. The authors have generated 1 000 time-series of 30 year rainfall data, from which they have chosen 10 most representative for the optimization and the rest for evaluation of solutions. The optimization was performed using a so called Water System Multi-objective Genetic Algorithm proposed in (Wu et al. 2009). The authors have experimented with a range of parameters of the GA and as most appropriate they have found having a population of 150 over 150 generations, with crossover and mutation rates of 0.9 and 0.1, respectively. The problem was encoded using a string of integer values, which are suitable for discrete decisions, and real values where required. The next generation was produced using a special type of constrained tournament selection with elitism. As the problem is multi-objective, a set of Pareto optimal solutions was obtained, and a detailed comparative analysis of all the solution from the Pareto front was provided.

In (Wang, Veeravalli & Rana 2018) the authors have presented how GAs can be used for optimizing large-scale computations of Astronomical data, particularly for the Sloan Digital Sky Survey[1] that has more than 125 TB of data. The data consists of a huge amount of rich telescope images covering more than a third of the sky including spectra for millions of objects. The problem is finding an optimal task-scheduling strategy for the many computational procedures that are required for the processing of the astronomical data in a fog computation set up. It is a min-max combinatorial problem in which the authors minimize the maximal time for execution of each of the individual computation steps. An encoding with real values is used and a population of 100 individuals. A roulette wheel selection is performed with supplemental elitism with the 5 best fitted individuals from each generation. Two point crossover with a rate of 0.8, and a two-point mutation with a rate of 0.02 is used. For convergence speed up a local search is introduced, which tries to balance the workload among the fog computational nodes. The algorithm finishes after a fixed number of 2 500 generations have passed. The authors performed the optimization using a single day data of about 200 GB.

Another interesting application of GA is in the design of wind farms (Wang, Cholette, Zhou, Yuan, Tan & Gu 2018) where the algorithm optimizes the turbine positions and the turbine wind hub heights. In (Gonzalez et al. 2018) the authors have employed a GA to optimize the operation of a hybrid renewable energy system combining wind, photovoltaic and forest biomass energy sources.

Evolutionary strategy From the various types of evolutionary strategies (ES), Covariance Matrix Adaptation ES (CMA-ES) is probably the most applied one because of its ease of use and efficiency. The CMA-ES algorithm was proposed in (Hansen & Ostermeier 2001), while in Hansen & Kern (2004) it was shown that it can find the global optima for many types of multi-modal standard test functions. The algorithm starts with an initial parental set of individuals also called parameter vectors. At each iteration a part of the parental vectors are selected and randomly intermediately recombined to create a new set of candidate parameter vectors. The candidate vectors are then mutated by sampling an n-dimensional normal distribution of the form $N(0, \mathbf{C})$, where $\mathbf{C}$ is an adaptive covariance matrix. The parameter vectors are then ranked and the fittest ones are selected for the next generation. The algorithm uses the concept of elitism, because at each generation it could leave some of the parameter vectors non-recombined and non-mutated, if they remain fitter than the other candidate vectors. Most of the internal strategy parameters of the algorithm are self-adaptive, and do not need some special initialization efforts. The mutations tend to favor the search directions from the previous steps, thus creating an evolution path, as it is known that a random search that tends to move away from its initial location is generally faster.

In (Quast et al. 2004), the authors applied the CMA-ES method in their procedure of estimating the variability of the fine-structure constant in the cosmos. They used observations of the spectra of the quasar HE 0515-4414 taken by the ultraviolet and visual Echelle spectrograph from the ESO very large telescope,

[1] `http://www.sdss.org`

and then applied the many-multiplet technique with some parameter values obtained by CMA-ES. A similar approach was also used in (Quast et al. 2005) for decomposition of quasar spectra into individual line profiles. A CMA-ES with 100 parents and 200 offspring was used and most of the parameters of the CMA-ES were set to their default initial values. The optimization runs are terminated after 100 000 evaluations of the objective function, which is a normalized residual sum of squares. The algorithm was examined with several test cases of synthesized data with noise and its performance was compared with other classical deterministic optimization methods. The results showed that unlike the other methods, CMA-ES can find the global optimum without any special initialization.

Similarly, in (Mirchev et al. 2012) it was also shown how CMA-ES can outperform the classical optimization methods in finding a better optimum of the objective function. The problem there is in fitting the coupling coefficients of an interactive ensemble of imperfect models to a given set of "truth" data. As a case study the chaotic Lorenz 63 attractor is used, which has a behavior that is suggestive to that of the atmosphere. As a reference "truth" a Lorenz attractor with its typical values was used, while a set of three Lorenz attractors with perturbed parameter values were used to mimic imperfect models. The idea of this approach is to be able to combine multiple climate models developed at different institutions, which provide predictions with variable accuracy both in space and time. The CMA-ES was applied with its default values, and the self-adaptiveness of its internal strategy proved to work very well.

In (Chwatal & Raidl 2007), an algorithm named Exoplanet Orbit Determination by Evolutionary Strategies (ExOD-ES) has been specially developed for finding extra solar planets based on spectral observations of the central star's radial velocity. The algorithm applies the classical operations of recombination, mutation and selection, but with several problem specific modifications. The exploration of the search space is restricted to long-lasting systems. An intermediate recombination of the parameters of the internal strategy is done for all individuals, while the optimized parameters are recombined only in 10% of the individuals. The mutation operator takes into account the Hill-stability criterion. The selection operation should choose the fittest individuals from each generation, but it is modified to slightly favor newly created solutions with larger mutations, which are at the beginning of the evolution path, in order to allow for new planets to be added more easily, by reserving special places for them in the next generation. The algorithm was successfully applied to the υ-Andromedae and the 55-Cancri systems using a population of 50 individuals and 5 000 offspring candidates in each generation and it converged over about 100 generations.

Genetic and Evolutionary Programming A combination of genetic and evolutionary programming was applied in (Li et al. 2004) for characterizing radial brightness of elliptical galaxies using a dataset of 18 brightness profiles of elliptical galaxies from the Coma cluster. The authors first apply genetic programming (GP) to find a functional form of the profiles and then use evolutionary programming (EP) to fit the parameters of the obtained function to the observed data. The functional form is restricted to consist of the operations $\{+, -, *, /, \sin, \cos, \exp, \log\}$, some real constants from $[-10, 10]$ and a variable radius r along the major elliptical axis. The fitness function is a combination of the individual's fit to the observations and the length of the expression. The fit to the data is represented by the amount of hits, which is the number of points generated by the function that are within some small predefined tolerance range of 0.005, called "hits criterion". Obviously, shorter expressions should be preferred as typically they also generalize better. The GP is run with a population size of 6 000, and crossover and mutation rates of 0.9 and 0.01. A tournament selection is applied of size 6. The initial depth of the expressions is limited to 6, while later it is allowed to grow up to 17. The GP is terminated after a predefined maximal time of 6 hours has passed or a maximal number of 100 generations is reached. After finding an acceptable functional form an EP is applied for fitting the parameter values using a Cauchy mutation with a rate of 0.9, a population of 10 000, tournament selection of size 6, and a fitness function based solely on the amount of hits with a hits criterion of 0.005. The EP terminates after a fixed number of 150 generations.

Differential Evolution In (Maulik & Saha 2009), the authors explore the applicability of differential evolution (DE) to the problem of fuzzy clustering in remote sensing imagery, which was previously addressed using a genetic algorithm in (Maulik & Bandyopadhyay 2003). The authors develop a modified differential evolution (MoDE) algorithm based on the classical DE approach, and they particularly apply it to fuzzy clustering. Similarly as in (Maulik & Bandyopadhyay 2003), the problem is encoded by a vector containing the n-dimensional coordinates of all cluster centers and a special fitness function is used for evaluating how good is the clustering. The modification is introduced by allowing two types of mutation process: one where the three parent vectors are randomly chosen (as in the classical DE), and a second one in which the difference is calculated between globally and locally best vectors and the result is added to a randomly chosen third vector. The modification should bring a faster convergence toward the global optimum. At each mutation a random decision is made about which mutation process is applied, but the probability is controlled by a parameter that favors the second type less and less as the generations pass. A classical crossover is used between the mutation vector and the target vector to produce a trial vector that would replace the target vector if it has a better fitness. The proposed MoDE is shown to perform better than the classical DE, GA (Maulik & Bandyopadhyay 2003) and several other methods for fuzzy clustering, using several test functions, and generated and real data. For both DE and MoDE, a population of 20 individuals over 100 generations are used, with a crossover rate of 0.8 and a mutation scaling factor of 0.8. The GA is run with 20 individuals over 100 generations with crossover and mutation rates of 0.8 and 0.3. The MoDEFC is then successfully applied to three large datasets, two of the cities of Calcutta and Mumbai obtained with the Indian Remote Sensing Satellites, and another one of Calcutta obtained with the SPOT system.

The problem of scheduling the James Webb Space Telescope, which should be launched in 2020 after several delays, is addressed in (Giuliano & Johnston 2008) using a multi-objective DE based algorithm named Generalized Differential Evolution 3 (GDE3) (Kukkonen & Lampinen 2005). The scheduling is divided in long-term and short-term phases and the authors address the later one. The problem objectives are minimizing the schedule gaps, the momentum accumulation and the missed observation opportunities. The GDE3 algorithm uses the same mutation and crossover steps as in the classical DE that is developed for single-objective optimization, while the selection step is modified to be suitable for multi-objective optimization. The trial and target vectors are compared and if any of them dominates the other, it is selected for the next generation, otherwise both vectors are kept and the population is reduced using solutions sorting and crowding distance. The solutions sorting is done by placing the solutions in ranks, such that all solutions in the higher ranks are dominated by the solutions from the lower ranks, while the solution at the same rank are non-dominated among themselves. The crowding distance is used to differentiate between the mutually non-dominated solutions by favoring solutions from non-crowded regions. At the end, a set of solutions are obtained forming the Pareto front. Several parameter settings and strategies are explored, such as minimizing all objectives at once, one by one or in groups.

The classical DE approach was applied in (Bazi et al. 2014) in the process of classification of hyper-spectral images using a method called extreme machine learning (EML). The DE algorithm was used during the model selection step that is required by the EML method, and it is run with a population of 10 vectors over 100 function evaluations, a crossover rate and a mutation scaling factor of 0.9. The whole approach was successfully applied for land cover classification in several image datasets from Indiana, Kenedy Space Center in Florida, Washington DC and the University of Pavia in Italy.

In another interesting study (Olds et al. 2007), a classical DE was applied for designing interplanetary missions including complicated aspects such as parking orbits determination and multiple gravity assists. A thorough analysis was conducted for examining and tuning the parameters of the algorithm using a large number of trials, and it was found that using a population of 28 individuals with a crossover rate of 0.8 and a random mutation scaling factor drawn from $[-1, 1]$ results in the best outcome for this problem. Several example missions were studied like Cassini, Galileo, crewed Mars mission and a theoretical sample-return mission to the Tempel 1 comet.

Ant colony optimization In (Zhang et al. 2011), two types of ant colony optimization (ACO) algorithms were examined for extracting endmembers from hyperspectral images obtained by remote sensing. The problem consists of detecting pixels (endmembers) capturing a single ground object, and then estimating the presence proportion of the endmembers into the mixed pixels. The images were represented by a weighted directed graph that is used for selecting the endmember pixels by finding a path that goes through all of their corresponding vertices. The ants move through the graph and keep a Tabu table of vertices visited in the past. Two types of objective functions were considered, which results in two different ACOs: one where the number of endmembers and image bands are equal, and another one that allows a mismatch where several bands can be combined into a single endmember through screening. A population of 30 ants was used with a pheromone dissipation factor of 0.99. The algorithm terminates if a maximum number of 10 000 iterations have passed or if the same optimal path is reached after 3 consecutive iterations. The algorithm was successfully applied to some simulated data as well as to the AVIRIS data set obtained from Cuprite, Nevada, USA.

Particle swarm optimization One interesting application of PSO was given in (Ruiz et al. 2015), where the algorithm was used for estimating the free parameters of the SAG semi-analytic model of galaxy formation. The model uses merger trees crated by simulation of Lambda Cold Dark Mater for generating galaxy populations. Seven model parameters were left for estimation while the other were set to some given values. Observational statistics of the stellar mass and the masses of supermassive black holes in the galaxy center were used to form two data constraint relations. The PSO parameters were set to $\alpha \approx 0.72$, and $\beta = \gamma \approx 1.193$. Instead of positioning the particles randomly a Maximin Latin Hypercube technique was used, which places the particles more evenly distributed. The particles positions were constrained withing a certain range, and the velocity was inverted when the boundary is reached. The velocities were constrained to some maximal values. A population of 30 particles was used over 150 iterations. A stopping convergence criterion was also defined based on the distance of the particles from the globally fittest one. The same problem was also solved using Monte Carlo Markov Chains and the solutions are comparable, but PSO needs one order of magnitude less time to reach it.

In (Shaw & Srivastava 2007), the general problem of inverting geophysical data was addressed, which basically is fitting model parameters given some observed set of data. Both GA and PSO were considered and compared to a ridge regression (RR). Both GA an PSO were run with a population of 300 over 100 generations/iterations. The GA was run with a single point crossover at a rate of 0.6 and a mutation rate of 0.02. The PSO was set with parameters $\beta = 2.8$ and $\gamma = 3.1$, while α was varied from 0.1 to 0.05 by a drop of 99% at each iteration. The algorithms were examined using synthetic data as well as real data obtained with vertical electric sounding and magnetotellurics. The quality of the solutions of GA and PSO are comparable, as well as the required computational time. On the other hand, RR can find a similarly good solution very fast, but only if it is initialized from a position near to the global optimum. In another work (Ali Ahmadi et al. 2013), a hybrid approach combining GA and PSO was used for training an ANN that predicts a reservoir permeability.

Artificial immune system An interesting algorithm was developed in (Zhong et al. 2006), which uses unsupervised learning based on artificial immune system for classification in hyperspectral images obtained by remote sensing. Each image was represented as a feature vector consisting of all pixels and in the AIS approach it was considered as an antigen. The goal was to classify the image pixels into a given number of classes. The developed unsupervised artificial immune classifier (UAIC) starts with a random greedy initialization of antibody cells and corresponding memory cells. By an iterative process of selection, cloning and mutation of the antibodies a better affinity is developed over time. At each iteration a number of antibodies are replaced at a given displace rate. In the mean time, the population of memory cells also evolves in order to represent the classes of antibodies appropriately. As a stopping criterion a threshold is used of the percentage of pixels changing class between two iterations, or if a maximal number of iterations is reached. The algorithm was tested using images from Wuhan and Xiaqiao, with 4 and 7 classes, respectively. A set of 10 antibodies was used with a clonal rate of 15, a displace rate of 0.1, a termination threshold of 3%, and a maximal number of 20 iterations. The performance of the algorithm was shown to be better than other algorithms like k-means, fuzzy k-means, ISODATA and self-organized map.

7 Discussion and conclusions

In this chapter, we gave an overview of various methods from evolutionary computation and provided some interesting examples of their applications to real-world problems in astronomy and geoscience hoping to motivate the reader in employing them to the problems they are facing. Comparisons of the different algorithms can be made from many aspects, but typically the appropriateness of the method depends on the particular application and how the problem is formulated. Therefore, it is better to first explore the previous similar experiences for the specific problem, before deciding which algorithm is worth trying. All these methods are inspired from various evolutionary and other natural processes happening on Earth. On the other hand, in (Rashedi et al. 2009), the authors found inspiration of collection of masses and their mutual gravitational interactions to formulate an interesting metaheuristic optimization method called gravitational search algorithm (GSA). There are many different versions and modifications of GSA, such as inclusion of black hole operators in (Doraghinejad & Nezamabadi-pour 2014). Maybe this chapter could also motivate some of the readers to indulge in developing new metaheuristic algorithms by drawing inspiration from their fields of expertise.

Evolutionary computing brings plenty of benefits in facing optimization challenges. As it was outlined in (Fogel 1997), generally they are conceptually simple, have a wide range of applications, allow for an external knowledge to be incorporated, can be combined with other methods like machine learning and classical optimization, can be self-adaptive to the particular problem and to dynamical changes, provide better results than the classical optimization methods in many real applications, and can be applied to complex unsolved problems. Moreover, they can be easily parallelized, which is of great importance due to the rapid developments of computer architectures, distributed computing and graphical processing units in the last two decades. Among the different applications of parallel evolutionary algorithms, within the different distributed technologies, we can mention the work proposed in (Nebro et al. 2008) where authors proposed a parallel genetic algorithm for solving the Deoxyribonucleic Acid (DNA) fragment assembly problem in a computational grid. Authors proved that their proposed distributed genetic approach is very promising in taking advantage of a grid system to solve large DNA fragment assembly problem instances. Also, in (Alba & Luque 2006), the same real-world problem was considered but from a different perspective. In this work, authors tend to analyze the behavior of a parallel genetic algorithm over different Local Area Network (LAN) technologies. The aim of this study is to show the potential impact in the search mechanics when shifting between LANs as well as to show the actual power and utility of the proposed distributed GA to solve the DNA fragment assembly problem. Another work dealing with high dimensional data is the work proposed in (Chu & Zomaya 2006), where authors proposed a parallel ant colony optimization for 3D Protein Structure Prediction using the HP Lattice Model, and results show the performance of the distributed ACO approach in terms of accuracy and network scalability. Another optimization problem dealing with big data is the work proposed in (Luque & Alba 2011*b*). In this study, authors focused on decision making associated with workforce planning. More precisely, a parallel genetic algorithm for the workforce planning problem is proposed where two sets of decisions are considered, namely the selection and the assignment. The first step of decisions consists in selecting a small number of employees from a large number of available workers while the second decision consists in assigning this staff to the tasks to be performed. The main objective is to minimize the costs associated to the human resources needed to fulfill the work requirements. An effective workforce plan is an essential tool to identify appropriate workload staffing levels and justify budget allocations so that organizations can meet their objectives (Luque & Alba 2011*b*). Another interesting application of parallel evolutionary algorithms is the well-known natural language processing problem of part-of-speech tagging. In (Luque & Alba 2011*a*), different parallel metaheuristic approaches such as the parallel genetic algorithm and the parallel simulated annealing were considered. The study highlights the high performances achieved by the parallel algorithms for complex tagging scenarios, and states the singular advantages for every technique. For further applications of various metaheuristic for global optimization of large-scale problems can be found in (Mahdavi et al. 2015, Alba 2005, Alba et al. 1999, Gong et al. 2015). A detailed survey of the various recent distributed evolutionary algorithms and models can be found in (Gong et al. 2015), including cloud, MapReduce and GPU based implementations.

Recently, artificial immune systems as alternative metaheuristics to evolutionary algorithms have been investigated to solve Astro- and Geo- Informatics optimization problems. However, we noticed that their applications to these research areas is still limited. This might be explained by the recently emergence of this area of research. It is worth mentioning that artificial immune systems, as discussed in Section 4, is a very diverse area of research, ranging from the modelling of immune systems to the development of algorithms for specific applications, and hence it has much to offer for Astro- and Geo- Informatics problem optimization. However, when it comes to big data, there are still some theoretical issues that need to be further explored, and among these we mention the development of unified frameworks, convergence, and scalability. Therefore, more works are emerging such as in (Dagdia 2018*a*,*b*) to better fit artificial immune system techniques to real-world applications, particularly when it comes to handling big data. This might also attract the attention of researchers to either adapt or develop new artificial immune system techniques, and fit them to their Astro- and Geo- Informatics big data optimization problems.

Acknowledgment

This work is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 702527.

Biographies

Zaineb Chelly Dagdia

Zaineb Chelly Dagdia is a Research and Development Technical Project Manager at the National Institute for Research in Computer Science and Automation (Inria), in France. She received her MSc. and Ph.D. in Computer Science at the University of Tunis (ISG-Campus), Tunisia, in 2010 and 2014, respectively. After that, she hold the position of a Marie Skłodowska Curie Research Fellow (MSCA-IF-2015-EF) at Aberystwyth University, UK. Her research interests include different aspects of Artificial Intelligence. She writes on Machine Learning, Data Mining and Data Analytics, Evolutionary Algorithms, and Artificial Immune Systems, Big Data, and Uncertainty Theories.

Miroslav Mirchev

Miroslav Mirchev received his B.S. in computer engineering and M.S. in computer networks and e-technologies in 2008 and 2009 respectively from the Ss. Cyril and Methodius University in Skopje (UKIM), Macedonia. In 2010 he had a research stay at the City University, Hong Kong. In 2014 he defended his Ph.D. thesis at the Polytechnic University of Turin, Italy, and as part of his studies he spent a period at the BioCircuits Institute, UCSD, USA. His areas of interest include network science, computer networks, nonlinear systems, optimization and machine learning. Currently he is an Associate Professor at the Faculty of Computer Science and Engineering at UKIM.

References

Alba, E. (2005), *Parallel metaheuristics: a new class of algorithms*, Vol. 47, John Wiley & Sons.

Alba, E. & Luque, G. (2006), Performance of distributed gas on dna fragment assembly, *in* 'Parallel Evolutionary Computations', Springer, pp. 97–115.

Alba, E. & Tomassini, M. (2002), 'Parallelism and evolutionary algorithms', *IEEE transactions on evolutionary computation* **6**(5), 443–462.

Alba, E., Troya, J. M. et al. (1999), 'A survey of parallel distributed genetic algorithms', *Complexity* **4**(4), 31–52.

Ali Ahmadi, M., Zendehboudi, S., Lohi, A., Elkamel, A. & Chatzis, I. (2013), 'Reservoir permeability prediction by neural networks combined with hybrid genetic algorithm and particle swarm optimization', *Geophysical Prospecting* **61**(3), 582–598.

Arias-Montano, A., Coello, C. A. C. & Mezura-Montes, E. (2012), 'Multiobjective evolutionary algorithms in aeronautical and aerospace engineering', *IEEE Transactions on Evolutionary Computation* **16**(5), 662–694.

Bazi, Y., Alajlan, N., Melgani, F., AlHichri, H., Malek, S. & Yager, R. R. (2014), 'Differential evolution extreme learning machine for the classification of hyperspectral images', *IEEE Geoscience and Remote Sensing Letters* **11**(6), 1066–1070.

Bertsimas, D. & Weismantel, R. (2005), *Optimization over integers*, Vol. 13, Dynamic Ideas Belmont.

Blickle, T. & Thiele, L. (1996), 'A comparison of selection schemes used in evolutionary algorithms', *Evolutionary Computation* **4**(4), 361–394.

Bonabeau, E., Marco, D. d. R. D. F., Dorigo, M., Théraulaz, G., Theraulaz, G. et al. (1999), *Swarm intelligence: from natural to artificial systems*, number 1, Oxford university press.

Branke, J., Deb, K. & Miettinen, K. (2008), *Multiobjective optimization: Interactive and evolutionary approaches*, Vol. 5252, Springer Science & Business Media.

Burnet, S. F. M. et al. (1959), 'The clonal selection theory of acquired immunity'.

Cao, X., Qiao, H. & Xu, Y. (2007), 'Negative selection based immune optimization', *Advances in Engineering Software* **38**(10), 649–656.

Chambers, L. D. (2000), *The practical handbook of genetic algorithms: applications*, Chapman and Hall/CRC.

Chandrasekaran, M., Asokan, P., Kumanan, S., Balamurugan, T. & Nickolas, S. (2006), 'Solving job shop scheduling problems using artificial immune system', *The International Journal of Advanced Manufacturing Technology* **31**(5-6), 580–593.

Charbonneau, P. (1995), 'Genetic algorithms in astronomy and astrophysics', *The Astrophysical Journal Supplement Series* **101**, 309.

Chu, D. & Zomaya, A. (2006), Parallel ant colony optimization for 3d protein structure prediction using the hp lattice model, *in* 'Parallel Evolutionary Computations', Springer, pp. 177–198.

Chwatal, A. M. & Raidl, G. R. (2007), Determining orbital elements of extrasolar planets by evolution strategies, *in* 'International Conference on Computer Aided Systems Theory', Springer, pp. 870–877.

Civicioglu, P. (2012), 'Transforming geocentric cartesian coordinates to geodetic coordinates by using differential search algorithm', *Computers & Geosciences* **46**, 229–247.

Clerc, M. (2010), *Particle swarm optimization*, Vol. 93, John Wiley & Sons.

Coello, C. A. C. & Cortés, N. C. (2005), 'Solving multiobjective optimization problems using an artificial immune system', *Genetic Programming and Evolvable Machines* **6**(2), 163–190.

Coello, C. A. C., Lamont, G. B., Van Veldhuizen, D. A. et al. (2007), *Evolutionary algorithms for solving multi-objective problems*, Vol. 5, Springer.

Cutello, V. & Nicosia, G. (2002), An immunological approach to combinatorial optimization problems, *in* 'Ibero-American Conference on Artificial Intelligence', Springer, pp. 361–370.

Dagdia, Z. C. (2018*a*), A distributed dendritic cell algorithm for big data, *in* 'Proceedings of the Genetic and Evolutionary Computation Conference Companion', GECCO '18, ACM, New York, NY, USA, pp. 103–104.
URL: *http://doi.acm.org/10.1145/3205651.3205701*

Dagdia, Z. C. (2018*b*), 'A scalable and distributed dendritic cell algorithm for big data classification', *Swarm and Evolutionary Computation* .
URL: *https://doi.org/10.1016/j.swevo.2018.08.009*

Darwin, C. (2004), *On the origin of species, 1859*, Routledge.

Das, S., Mullick, S. S. & Suganthan, P. N. (2016), 'Recent advances in differential evolution–an updated survey', *Swarm and Evolutionary Computation* **27**, 1–30.

Das, S. & Suganthan, P. N. (2011), 'Differential evolution: A survey of the state-of-the-art', *Trans. Evol. Comp* **15**(1), 4–31.
URL: *https://doi.org/10.1109/TEVC.2010.2059031*

DasGupta, D. (1993), An overview of artificial immune systems and their applications, *in* 'Artificial immune systems and their applications', Springer, pp. 3–21.

De Castro, L. N. & Timmis, J. (2002), *Artificial immune systems: a new computational intelligence approach*, Springer Science & Business Media.

De Jong, K. (2005), 'Genetic algorithms: a 30 year perspective', *Perspectives on Adaptation in Natural and Artificial Systems* **11**.

De Jong, K. A. & Spears, W. M. (1990), An analysis of the interacting roles of population size and crossover in genetic algorithms, *in* 'International Conference on Parallel Problem Solving from Nature', Springer, pp. 38–47.

Deb, K. (2001), *Multi-objective optimization using evolutionary algorithms*, Vol. 16, John Wiley & Sons.

Doraghinejad, M. & Nezamabadi-pour, H. (2014), 'Black hole: a new operator for gravitational search algorithm', *International Journal of Computational Intelligence Systems* **7**(5), 809–826.

Dorigo, M. & Di Caro, G. (1999), Ant colony optimization: a new metaheuristic, *in* 'Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on', Vol. 2, IEEE, pp. 1470–1477.

Dorigo, M. & Stützle, T. (2003), The ant colony optimization metaheuristic: Algorithms, applications, and advances, *in* 'Handbook of metaheuristics', Springer, pp. 250–285.

Durham, W. H. (1991), *Coevolution: Genes, culture, and human diversity*, Stanford University Press.

D'Ambrosio, D., Spataro, W. & Iovine, G. (2006), 'Parallel genetic algorithms for optimising cellular automata models of natural complex phenomena: an application to debris flows', *Computers & Geosciences* **32**(7), 861–875.

Eberbach, E. (2005), 'Toward a theory of evolutionary computation', *BioSystems* **82**(1), 1–19.

Eiben, A. E., Smith, J. E. et al. (2003), *Introduction to evolutionary computing*, Vol. 53, Springer.

Fogel, D. B. (1997), The advantages of evolutionary computation., *in* 'BCEC', pp. 1–11.

Fogel, L. J., Owens, A. J. & Walsh, M. J. (1966), 'Artificial intelligence through simulated evolution'.

Gao, X.-Z., Ovaska, S. J. & Wang, X. (2008), 'A ga-based negative selection algorithm', *International Journal of Innovative Computing, Information and Control* **4**(4), 971–979.

Giuliano, M. E. & Johnston, M. D. (2008), Multi-objective evolutionary algorithms for scheduling the james webb space telescope., *in* 'ICAPS', pp. 107–115.

Goldberg, D. E. & Deb, K. (1991), A comparative analysis of selection schemes used in genetic algorithms, *in* 'Foundations of genetic algorithms', Vol. 1, Elsevier, pp. 69–93.

Gong, Y.-J., Chen, W.-N., Zhan, Z.-H., Zhang, J., Li, Y., Zhang, Q. & Li, J.-J. (2015), 'Distributed evolutionary algorithms and their models: A survey of the state-of-the-art', *Applied Soft Computing* **34**, 286–300.

Gonzalez, A., Riba, J.-R., Esteban, B. & Rius, A. (2018), 'Environmental and cost optimal design of a biomass–wind–pv electricity generation system', *Renewable Energy* **126**, 420–430.

Goodwin, B. C. (1982), 'Development and evolution', *Journal of Theoretical Biology* **97**(1), 43–55.

Hajela, P. & Yoo, J. S. (1999), Immune network modelling in design optimization, *in* 'New ideas in optimization', McGraw-Hill Ltd., UK, pp. 203–216.

Hansen, N. & Kern, S. (2004), Evaluating the cma evolution strategy on multimodal test functions, *in* 'International Conference on Parallel Problem Solving from Nature', Springer, pp. 282–291.

Hansen, N. & Ostermeier, A. (2001), 'Completely derandomized self-adaptation in evolution strategies', *Evolutionary computation* **9**(2), 159–195.

Holland, J. & Goldberg, D. (1989), 'Genetic algorithms in search, optimization and machine learning', *Massachusetts: Addison-Wesley* .

Holland, J. H. (1992), *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*, MIT press.

Jain, A. & Srinivasulu, S. (2004), 'Development of effective and efficient rainfall-runoff models using integration of deterministic, real-coded genetic algorithms and artificial neural network techniques', *Water Resources Research* **40**(4).

Janeway Jr, C. A. (1992), 'The immune system evolved to discriminate infectious nonself from noninfectious self', *Immunology today* **13**(1), 11–16.

Kennedy, J. & Eberhart, R. (1995), Particle swarm optimization, *in* 'Neural Networks, 1995. Proceedings., IEEE International Conference on', Vol. 4, pp. 1942–1948 vol.4.

Klockgether, J. & Schwefel, H.-P. (1970), Two-phase nozzle and hollow core jet experiments, *in* 'Proc. 11th Symp. Engineering Aspects of Magnetohydrodynamics', Pasadena, CA: California Institute of Technology, pp. 141–148.

Koza, J. R. (1992), *Genetic Programming II, Automatic Discovery of Reusable Subprograms*, MIT Press, Cambridge, MA.

Kukkonen, S. & Lampinen, J. (2005), Gde3: The third evolution step of generalized differential evolution, *in* 'Evolutionary Computation, 2005. The 2005 IEEE Congress on', Vol. 1, IEEE, pp. 443–450.

Li, J., Yao, X., Frayn, C., Khosroshahi, H. G. & Raychaudhury, S. (2004), An evolutionary approach to modeling radial brightness distributions in elliptical galaxies, *in* 'International Conference on Parallel Problem Solving from Nature', Springer, pp. 591–601.

Luque, G. & Alba, E. (2011*a*), Natural language tagging with parallel genetic algorithms, *in* 'Parallel Genetic Algorithms', Springer, pp. 75–89.

Luque, G. & Alba, E. (2011*b*), Parallel genetic algorithm for the workforce planning problem, *in* 'Parallel Genetic Algorithms', Springer, pp. 115–134.

Mahdavi, S., Shiri, M. E. & Rahnamayan, S. (2015), 'Metaheuristics in large-scale global continues optimization: A survey', *Information Sciences* **295**, 407–428.

Matzinger, P. (2001), 'Essay 1: the danger model in its historical context', *Scandinavian journal of immunology* **54**(1-2), 4–9.

Maulik, U. & Bandyopadhyay, S. (2003), 'Fuzzy partitioning using a real-coded variable-length genetic algorithm for pixel classification', *IEEE Transactions on geoscience and remote sensing* **41**(5), 1075–1081.

Maulik, U. & Saha, I. (2009), 'Modified differential evolution based fuzzy clustering for pixel classification in remote sensing imagery', *Pattern Recognition* **42**(9), 2135–2149.

Metcalfe, T. S. & Charbonneau, P. (2003), 'Stellar structure modeling using a parallel genetic algorithm for objective global optimization', *Journal of Computational Physics* **185**(1), 176–193.

Meyer-Nieberg, S. & Beyer, H.-G. (2007), Self-adaptation in evolutionary algorithms, *in* 'Parameter setting in evolutionary algorithms', Springer, pp. 47–75.

Mirchev, M., Duane, G. S., Tang, W. K. & Kocarev, L. (2012), 'Improved modeling by coupling imperfect models', *Communications in Nonlinear Science and Numerical Simulation* **17**(7), 2741–2751.

Mühlenbein, H. & Schlierkamp-Voosen, D. (1993), 'Predictive models for the breeder genetic algorithm i. continuous parameter optimization', *Evolutionary computation* **1**(1), 25–49.

Myerson, R. B. (2013), *Game theory*, Harvard university press.

Nebro, A. J., Luque, G., Luna, F. & Alba, E. (2008), ‘Dna fragment assembly using a grid-based genetic algorithm’, *Computers & Operations Research* **35**(9), 2776–2790.

Olds, A. D., Kluever, C. A. & Cupples, M. L. (2007), ‘Interplanetary mission design using differential evolution’, *Journal of Spacecraft and Rockets* **44**(5), 1060–1070.

Orfila, A., Ballester, J., Oliver, R., Alvarez, A. & Tintoré, J. (2002), ‘Forecasting the solar cycle with genetic algorithms’, *Astronomy & Astrophysics* **386**(1), 313–318.

Papadimitriou, C. H. & Steiglitz, K. (1998), *Combinatorial optimization: algorithms and complexity*, Courier Corporation.

Paton, F., Maier, H. & Dandy, G. (2014), ‘Including adaptation and mitigation responses to climate change in a multiobjective evolutionary algorithm framework for urban water supply systems incorporating ghg emissions’, *Water Resources Research* **50**(8), 6285–6304.

Pochet, Y. & Wolsey, L. A. (2006), *Production planning by mixed integer programming*, Springer Science & Business Media.

Price, K., Storn, R. M. & Lampinen, J. A. (2006), *Differential evolution: a practical approach to global optimization*, Springer Science & Business Media.

Quast, R., Baade, R. & Reimers, D. (2005), ‘Evolution strategies applied to the problem of line profile decomposition in qso spectra’, *Astronomy & Astrophysics* **431**(3), 1167–1175.

Quast, R., Reimers, D. & Levshakov, S. A. (2004), ‘Probing the variability of the fine-structure constant with the vlt/uves’, *Astronomy & Astrophysics* **415**(2), L7–L11.

Rashedi, E., Nezamabadi-Pour, H. & Saryazdi, S. (2009), ‘Gsa: a gravitational search algorithm’, *Information sciences* **179**(13), 2232–2248.

Rechenberg, I. (1981), ‘" evolutionsstrategie-optimierung technischer systems nach prinzipien der biologischen evolution, stuttgart: Frommannholzboog, 1973’, *New York: John Wiley* .

Ruiz, A. N., Cora, S. A., Padilla, N. D., Domínguez, M. J., Vega-Martínez, C. A., Tecce, T. E., Orsi, Á., Yaryura, Y., Lambas, D. G., Gargiulo, I. D. et al. (2015), ‘Calibration of semi-analytic models of galaxy formation using particle swarm optimization’, *The Astrophysical Journal* **801**(2), 139.

Schrijver, A. (2003), *Combinatorial optimization: polyhedra and efficiency*, Vol. 24, Springer Science & Business Media.

Shaw, R. & Srivastava, S. (2007), ‘Particle swarm optimization: A new tool to invert geophysical data’, *Geophysics* **72**(2), F75–F83.

Storn, R. & Price, K. (1997), 'Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces', *Journal of global optimization* **11**(4), 341–359.

Talbi, E.-G. (2009), *Metaheuristics: from design to implementation*, Vol. 74, John Wiley & Sons.

Ulutas, B. H. & Kulturel-Konak, S. (2011), 'A review of clonal selection algorithm and its applications', *Artificial Intelligence Review* **36**(2), 117–138.

Vachier, F., Berthier, J. & Marchis, F. (2012), 'Determination of binary asteroid orbits with a genetic-based algorithm', *Astronomy & Astrophysics* **543**, A68.

Voigt, H.-M. & Anheyer, T. (1994), Modal mutations in evolutionary algorithms, *in* 'Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on', IEEE, pp. 88–92.

Wahde, M. (1998), 'Determination of orbital parameters of interacting galaxies using a genetic algorithm-description of the method and application to artificial data', *Astronomy and Astrophysics Supplement Series* **132**(3), 417–429.

Wahde, M. & Donner, K. (2001), 'Determination of the orbital parameters of the m 51 system using a genetic algorithm', *Astronomy & Astrophysics* **379**(1), 115–124.

Wang, L., Cholette, M. E., Zhou, Y., Yuan, J., Tan, A. C. & Gu, Y. (2018), 'Effectiveness of optimized control strategy and different hub height turbines on a real wind farm optimization', *Renewable Energy* **126**, 819–829.

Wang, X., Veeravalli, B. & Rana, O. F. (2018), 'An optimal task-scheduling strategy for large-scale astronomical workloads using in-transit computation model', *INTERNATIONAL JOURNAL OF COMPUTATIONAL INTELLIGENCE SYSTEMS* **11**(1), 600–607.

Wu, W., Simpson, A. R. & Maier, H. R. (2009), 'Accounting for greenhouse gas emissions in multiobjective genetic algorithm optimization of water distribution systems', *Journal of water resources planning and management* **136**(2), 146–155.

Yan, S. & Minsker, B. (2006), 'Optimal groundwater remediation design using an adaptive neural network genetic algorithm', *Water Resources Research* **42**(5).

Zhang, B., Sun, X., Gao, L. & Yang, L. (2011), 'Endmember extraction of hyperspectral remote sensing images based on the ant colony optimization (aco) algorithm', *IEEE transactions on geoscience and remote sensing* **49**(7), 2635–2646.

Zhong, Y., Zhang, L., Huang, B. & Li, P. (2006), 'An unsupervised artificial immune classifier for multi/hyperspectral remote sensing imagery', *IEEE Transactions on Geoscience and Remote Sensing* **44**(2), 420–431.