UNIVERSITÀ DI PARMA

UNIVERSITÀ DEGLI STUDI DI PARMA

DOTTORATO DI RICERCA IN
"TECNOLOGIE DELL'INFORMAZIONE"

CICLO XXXII

Detection of relevant structures in complex systems: an information-theoretic approach with applications to machine learning and pattern recognition

Coordinatore:
Chiar.mo Prof. Marco Locatelli

Tutor:
Chiar.mo Prof. Stefano Cagnoni

Dottoranda: Laura Sani

Anni 2016/2019

Dedicated to
my family

Table of Contents

List of Figures

List of Tables

Abstract

The study of complex systems is related to the analysis of emerging properties and collective behaviors of systems whose components are usually well-known and can be defined in terms of variables.

Most natural or artificial dynamical systems are characterized by groups of variables which appear to be well coordinated among themselves and have a weaker interaction with the remainder of the system (Relevant Sets, or shortly RSs in the following). The capacity of detecting their presence can often lead to a high-level description of the dynamical organization of a complex system, and thus to its understanding.

However, the identification and monitoring of the significant or relevant portions of dynamical systems are very difficult, especially if these systems exhibit emergent or self-organizing phenomena. Moreover, in several cases, the detailed interactions among the system components are not known and, in case of systems where the components interact non-linearly, the dynamic relationships among variables might not be entirely described by a merely topological view. Therefore, it is often necessary to derive some hints about the system organization by observing the behavior of its dynamically relevant parts.

For these reasons, the work of this thesis is based on the Relevance Index (RI) metrics, a set of information-theoretical metrics that can be used to analyze complex systems by detecting the main interacting structures (RSs) within them, starting from the observation of the status of the system variables over time.

The search of relevant subsets within a dynamical system corresponds to an op-

timization problem (maximization of an RI metric). To fully describe a dynamical system based on an RI metric it would be necessary to compute such an index for all possible subsets of the system variables. Unfortunately, their number increases exponentially with the number of variables, soon reaching unrealistic requirements for computation resources, even using massively parallel hardware such as GPUs. As a consequence, to analyze large-size systems it is necessary to design efficient strategies which can limit the extension of the search by quickly identifying the most promising subsets, providing results comparable to an exhaustive search in a much shorter time. Furthermore, one does not need to know the RI metric of all possible subsets, but only to detect the ones for which the index is higher. Therefore, we propose to use population-based niching metaheuristics, derived by hybridizing genetic algorithms (GAs) and particle swarm optimization (PSO) with local searches, to deal with this multimodal scenario, analyzing many peaks in parallel.

From a methodological point of view, the work proposed in this thesis concerns also the design of some improvements to the RI metrics, in terms of both computational efficiency and effectiveness for complex systems analysis. Moreover, we propose an iterative algorithm aimed at revealing hierarchical dynamical structures.

The RI analysis can be applied to a broad range of dynamical systems, both natural and artificial. In particular, in this work we present some applications of the proposed methodology in different contexts, such as chemical systems (identification of the structure of chemical reactions, starting from the observation of the concentrations of different chemical species over time), biological networks (study of regulatory networks, e.g., the T helper network) and social networks (analysis of the dynamical behavior of the users, starting from their posts and from the expressed emotions, which have been automatically detected).

Furthermore, this work aims also at evaluating the applicability of the RI methodology to machine learning and pattern recognition problems. In particular, the properties that the RI metrics highlight in time series, when analyzing the dynamics of complex systems, can be assimilated to the properties of the multivariate distribution of the examples of a machine learning training set, e.g., the distribution of static patterns subject to noise, translations, etc. Therefore, the last part of this thesis con-

cerns the design of RI-based algorithms for machine learning and pattern recognition tasks (clustering, classification, feature extraction and preprocessing). The experimental results have been evaluated on a real-world dataset of license plates images and on some standard publicly available datasets. Finally, the proposed methodology has been applied to automatic troll detection, using data collected from Twitter.

The thesis is organized as follows:

- *Chapter 1* introduces the topic of this thesis, i.e., the analysis of complex systems, and the main motivations behind the proposed methodology;
- *Chapter 2* provides the theoretical background, describing the state of the art. In particular, the first section is dedicated to the metrics for the search of relevant structures in complex systems, with a particular focus on the RI metrics, while the second section describes some metaheuristics which can be used to deal with the curse of dimensionality when analyzing high-dimensional systems, including a description of niching techniques;
- *Chapter 3* presents a methodological analysis of the RI metrics, proposing some improvements in terms of both computational efficiency and effectiveness for complex systems analysis;
- *Chapter 4* proposes two RI-based niching metaheuristics for complex systems analysis, devised as hybridizations of GAs and PSO, respectively;
- *Chapter 5* presents an iterative algorithm, based on the RI method, aimed at revealing the hierarchical dynamical structures hidden in complex systems;
- *Chapter 6* describes some applications of the proposed methodology to different contexts, e.g., biology and social sciences;
- *Chapter 7* evaluates the applicability of the proposed methodology to machine learning and pattern recognition problems;
- Finally, *Chapter 8* summarizes the main contributions of this thesis and the open issues, which can foster future developments of this work.

Part I

Fundamentals

Chapter 1

Introduction to complex systems

The term complexity is generally used to characterize the behavior of a system (or model) whose components interact in multiple ways and follow local rules [119].
The stem of the word "complexity" (complex) combines the latin roots "cum" (meaning "with") and "plexus" (meaning "inter").

A complex system can be described using a holistic approach, which refers to the idea that this kind of system should be seen as a whole, not merely as a collection of parts (since its whole functionality is greater than the sum of its parts). Indeed, a complex system is characterized by its inter-dependencies, whose behavior is intrinsically difficult to model. The study of these complex relationships at various scales is the main goal of complex systems theory.

Generally, a complex system can be characterized by:

- **Emergence**: a "global" behavior which is not evident from the system components considered in isolation, but results from their interactions and relationships when put together in the system. The presence of emergent behaviors and properties is common in complex systems; an example is given by cellular automata [249];

- **Nonlinearity**: complex systems often exhibit nonlinear behaviors, i.e., they may behave in different ways starting from the same input, depending on their

state or on the context. Nonlinearity means that a change in the magnitude of the input does not produce a proportional change in the magnitude of the output.
Nonlinear dynamical systems, which are systems described by differential equations that have one or more nonlinear terms, are particularly interesting for complex systems analysis. Some nonlinear dynamical systems, such as the Lorenz system, can produce a mathematical phenomenon known as chaos [225], which refers to the dependence on initial conditions, or "butterfly effect", that a complex system can exhibit;

- **Adaptation**: complex adaptive systems are characterized by the ability to change and learn from experience. These dynamic systems are able to adapt to and evolve with a changing environment. Examples of complex adaptive systems are typical of the biosphere and the ecosystem, e.g., the collective behaviors of social insects and ant colonies;

- **Network structure**: the interacting components of a complex system form a network, which can be seen as a collection of discrete objects and the relationships which exist between them [2];

- **Cascading failure**: in a system of interconnected parts, the failure of one or few parts can trigger the failure of the other parts;

- **Open system**: complex systems have usually also external interactions;

- **Memory**: complex systems are dynamical systems that change over time. Thus, the history of a complex system may be important, since the prior states may influence the present states;

- **Nested systems**: the components of a complex system may themselves be complex systems;

- **Tangled hierarchies**: the levels of causality are so intertwined that we cannot identify which level is the lower and which is the upper;

- **Feedback loops**: the effects of the behavior of an element can alter the element itself. Both negative (damping) and positive (amplifying) feedbacks are often found in complex systems.

Thus, the identification and monitoring of the significant or relevant portions of complex dynamical systems are very difficult, especially if these systems exhibit emergent phenomena [64].

Indeed, most theories and models take into account only two-level systems and describe the formation of relatively simple dynamical patterns as, for example, the creation of the well-known Bénard-Marangoni hexagonal convection pattern [102]. In this case, the two levels involved are those of the water particles and of the hexagonal convection cells. Indeed, the apparatus where the phenomenon takes place (which is, of course, necessary, since it determines some major features of the phenomenon itself) is not affected by what happens at the lower levels: in other words, it just provides the fixed boundary conditions that allow the phenomenon to occur.

However, the most interesting recurrent patterns of interaction [133] take place very often at levels that can be regarded as intermediate between pre-existing layers, which are, in turn, affected by the dynamics of these patterns. There are several examples of these "sandwiched" phenomena in physics, biology and social sciences [133]. Perhaps the most evident cases are the presence of vortexes on fluids surfaces, the presence of organs and tissues in multi-cellular organisms, or the action of various groups of humans (such as companies, cooperatives, associations, factions, communities) within societies[1]. It is to be noticed that the formation of structures or patterns not explicitly designed is frequent even in artificial systems as, for example, power grids [246], e-mail networks [59], the Internet [3, 69], and so on. Thus, the detection of intermediate-level structures and patterns is a very central issue when studying complex dynamical systems.

The following chapter (in particular, section 2.1) presents some metrics for the search of relevant structures (mesolevel dynamical structures [239]) in complex systems, which will be the main subject of this thesis.

[1] The lower and upper level being constituted by the fluid particles and their global stream, by cells and the organism to which they belong, and by human beings and societies, respectively.

Chapter 2

State of the art and related work

This chapter presents the state of the art about complex systems analysis. In particular, section 2.1 describes some metrics for the search of relevant structures in complex systems, while section 2.2 describes some metaheuristics which can be used to deal with the curse of dimensionality when analyzing high-dimensional systems.

2.1 Search of relevant structures in complex systems

The study of complex systems is related to the analysis of collective behaviors and emerging properties of systems whose components are usually well-known.

Measuring the complexity of a composite system is a challenging task; dozens of measures of complexity have been proposed, several of which are based on information theory [182]. Detecting clusters of elements that interact strongly with one another is even more challenging, especially when the only information available is the evolution of their states in time.

The problem of finding groups of system elements that have a tighter dynamical interaction among themselves than with the rest of the system is a typical issue in data analysis; notable examples are the identification of functional neuronal regions in the brain and the detection of specific groups of genes ruling the dynamics of a genetic network.

Many complex systems, both natural and artificial, may be represented by networks of interacting nodes [2, 199]. Nevertheless, it is often difficult to find neat correspondences between the dynamics expressed by these systems and their network description. In addition, it is to be noticed that, in case of systems where the elements interact non-linearly, the dynamic relationships among variables might not be entirely described by the topology, which could admit, e.g., interactions without effect on target elements. In contrast, many of these systems may be described effectively in terms of coordinated dynamical behavior of groups of elements; notable examples are Boolean networks [238], chemical or biological reaction systems [240] and functional connectivity graphs in neuroscience [222, 216]. Furthermore, in several notable cases the detailed interactions among the system elements are not known; therefore, it is important to deduce some hints about system organization by observing the behavior of its dynamically relevant parts.

The emerging structures to identify may be either static entities or dynamical patterns, or some mixture of the two. In dynamical networks, static emergent structures take the form of topological features, like, e.g., motifs in genetic networks or communities in a broader context. While there is an extensive literature on community detection on graphs, this thesis focuses on a different type of mesolevel structures, namely those that are created by the dynamical interactions in the network. Nodes may exhibit coordinated behavior although they are not directly linked, since the dynamics of the system may give rise to different coordinated parts. These mesolevel structures, which may have a topological as well as a dynamical nature, can be defined as Mesolevel Dynamical Structures (MDSs) [239]. The task of identifying MDSs allows one to escape from a merely topological view, considering different subsets of the variables which describe the system under analysis, and looking for those subsets whose elements appear to be well coordinated among themselves and have a weaker interaction with the remainder of the system.

The subsets of the system variables can be analyzed using different measures of complexity, several of which are based on information theory [182], which is convenient since any dynamically changing phenomenon can be characterized in terms of the information it carries. A widely-known information-theoretic framework by

Gershenson and Fernandez [85] allows one to characterize systems in terms of emergence, self-organization, complexity and homeostasis. Such a framework has been applied, for example, to characterize adaptive peer-to-peer systems [5], communications systems [71] and agroecosystems [148].

The following subsections describe a set of information-theoretical metrics that can be used to dynamically analyze complex systems by detecting the main interacting structures within them.

2.1.1 Cluster Index

The *Cluster Index* (*CI*) was introduced by Tononi et al. [232, 231] in the study of biological neural networks close to a stationary state.

In cluster analysis, a cluster can be defined as a subset of elements that are cohesive among themselves but relatively isolated from the remaining elements. In terms of brain dynamics, a functional cluster can be defined as a subset of neural elements that interact strongly among themselves but weakly with the remainder of the system. Tononi et al. applied this notion to PET imaging data by using two multivariate measures of statistical dependence among neural elements:

- *Integration*: measure of the total statistical dependence within a subset of elements;

- *Mutual Information*: measure of the statistical dependence between a subset of elements and the rest of the system.

Given a neural system X composed of a set of N neural elements whose physiological activation is reflected by the activity value of a set of N voxels in an imaging data set, the activity of these elements can be described by a stationary multidimensional stochastic process. The joint probability density function describing such a multivariate process can be characterized in terms of entropy and mutual information [47].

According to information theory, Shannon's entropy of an element (random vari-

able) x_i is defined as:

$$H(x_i) = -\sum_{v \in V_i} p(v) \log p(v) \tag{2.1}$$

where V_i is the set of the possible values of x_i and $p(v)$ the probability of occurrence of symbol v.
The entropy of a pair of elements x_i and x_j is defined by means of their joint probabilities:

$$H(x_i, x_j) = -\sum_{v \in V_i} \sum_{w \in V_j} p(v,w) \log p(v,w) \tag{2.2}$$

Eq. 2.2 can be extended to sets of k elements considering the probability of occurrence of vectors of k values.

Let $H(S_k)$ be the entropy of a subset S_k composed of k elements, with $k < N$. The *CI* is defined as the ratio between the *Integration* of a subset S_k and the *Mutual Information* between S_k and the rest of the system:

$$CI(S_k) = \frac{I(S_k)}{MI(S_k; X \setminus S_k)} \tag{2.3}$$

where $I(S_k)$ measures the mutual dependence among the k elements in S_k, while $MI(S_k; X \setminus S_k)$ measures the mutual dependence between the subset S_k and the rest of the system $X \setminus S_k$:

$$I(S_k) = \sum_{s \in S_k} H(s) - H(S_k) \tag{2.4}$$

$$MI(S_k; X \setminus S_k) = H(S_k) + H(X \setminus S_k) - H(S_k, X \setminus S_k) \tag{2.5}$$

It is to be noticed that, being a ratio, the *CI* is undefined in all those cases where $MI(S_j; X \setminus S_j)$ vanishes. In these cases, however, the subset S_j is statistically independent from the rest of the system and, therefore, should be analyzed separately. These situations must be screened out in advance.

The value of the *CI* depends on the size of the system and on the size of the group under examination. To avoid a similar dependence, Tononi has proposed the use of the normalized Cluster Index (CI'), comparing the *CI* values with those of subsystems

having same size, but belonging to a non-clustered homogeneous system, i.e., the homogeneous system X_h, randomly generated in accordance with the probability of each single state measured in the original system X, along all its series of states. The homogeneous system is characterized by independent Gaussian variables, with pairwise correlation values all equal to zero. In order to have a reference value for the cluster index for each size, the average integration $\langle Ih_k \rangle$ and the average mutual information $\langle MIh_k \rangle$ are computed for each subsystem size k of X_h .

$$CI'(S_k) = \frac{I(S_k)}{\langle Ih_k \rangle} / \frac{MI(S_k; X \setminus S)}{\langle MIh_k \rangle} \tag{2.6}$$

A high index indicates that a subset of brain regions forms a distinct cluster having functional boundaries with the rest of the brain.

The *CI* analysis has been tested on PET data obtained from normal and schizophrenic subjects performing a set of cognitive tasks. A comparative evaluation of the functional brain regions obtained by the *CI* analysis has highlighted differences between the two groups of subjects.

From a practical point of view, several descriptive statistical techniques can be used to facilitate the identification of functional clusters in imaging data, e.g., Principal Component Analysis (PCA) and Multidimensional Scaling (MDS). However, while PCA and related approaches may be useful in identifying important relationships within the data, they are not designed to search for structures but rather to summarize a large number of dimensions. Similarly, MDS is closely related to PCA and the same considerations apply to this procedure.

2.1.2 Relevance Index

The Cluster Index, originally introduced to detect functional relationships between brain regions, has been generalized to study dynamical systems, in order to apply the method to a broad range of systems [240, 241]. The generalization of the Cluster Index has been called *Relevance Index* (*RI*)[1]. The *RI* can be used to analyze a collection

[1]The generalization of the Cluster Index was first called Dynamical Cluster Index and subsequently Dynamical Relevance Index to emphasize the possibility of applying it to actual dynamical systems.

of observations from a dynamical system, identifying the main interacting structures within it. Such structures can be viewed as subsets of the variables which describe the system status. Therefore, the *RI* analysis performs a sort of dynamical clustering of the system variables.

The purpose of the *RI* is the identification of subsets of variables that behave in a coordinated way in a dynamical system. This means that the variables belonging to the subset are integrated with each other much more tightly than with the other variables of the system. These subsets can be used to describe the whole system organization, thus they are named *Relevant Subsets* (RSs).

Given a generic system U described by N system variables (random variables) whose status changes in time (supposing the time series of their values is available) and a subset S_k composed of k variables, with $k < N$, the *RI* is defined as:

$$RI(S_k) = \frac{I(S_k)}{MI(S_k; U \setminus S_k)} \tag{2.7}$$

The *RI* of a subset S_k is computed as the ratio between the *Integration* (I, defined in Equation (2.4), which is to be maximized) of subset S_k and the *Mutual Information* (MI, defined in Equation (2.5), which is to be minimized) between S_k and the rest of the system. It is to be noticed that the formal definitions of the *RI* and of the *CI* are identical, being the *RI* a generalization of the *CI* to the study of a generic system U.

It can be observed from Equation (2.4) that $I(S_k) \geq 0$ and it is zero when the variables are independent. On the other hand, the integration increases with the decrease of the second term of Equation (2.4), i.e., with the correlation among the random variables.

Since the *RI* is defined as the ratio between Integration and Mutual Information, which are based on entropy, the *RI* computation relies on the approximation of the statistical distribution of the variables, based on the analysis of a sample of the system states observed over a given time interval. The probabilities are estimated as the relative frequencies of the values observed for each variable. I, MI and RI, and their

However, the 'dynamical' characterization was dropped since the index can be applied to more general scenarios, making it possible to discover multiple relations among system variables even when data do not belong to a time series; in fact, they may even represent the state of different systems.

normalized counterparts described below, which solve some issues with their dependency on group size, constitute a family of information-theoretical metrics that can be denoted as *RI metrics*, since they assess the relevance of a subset of variables for the description of the system organization.

It can be seen from Equation (2.7) that a high *RI* corresponds to sets in which the elements diverge as much as possible from independence among themselves and, at the same time, are as independent as possible of the rest of the elements in the system. In the context of detecting RSs in complex systems, the value of the *RI*, which can be computed using empirical entropies, is used to rank the subsets, considering those having higher index as more relevant. However, a comparison based on the expression given by Equation (2.7) would be unfair since the *RI* increases with the size k of the considered subset.

To compare subsets of different size, a normalization has been introduced using a homogeneous system as a reference. Following the normalization proposed by Tononi et al. [231], a null hypothesis can be considered: the absence of correlated subsets. The null hypothesis is provided by the homogeneous system U_h, a randomly generated reference system which preserves the priors of each single variable in U. For each subsystem size, the *RI* values are compared with those of subsystems having same size, but belonging to a non-clustered homogeneous system. The aim of the reference system is to quantify the finite-size effects affecting the information-theoretic measures of a random sampling of the system status. Therefore, in order to have a reference value for the cluster index for each size, the average integration $\langle Ih_k \rangle$ and the average mutual information $\langle MIh_k \rangle$ are computed for each subsystem size k of U_h . Equation (2.8) shows the definition of the normalized Relevance Index:

$$RI'(S_k) = \frac{I(S_k)}{\langle Ih_k \rangle} / \frac{MI(S_k; U \setminus S)}{\langle MIh_k \rangle} \tag{2.8}$$

In order to assess the statistical significance of the normalized *RI* values, Villani et al. introduced the T_c index for dynamical systems [241], which has been shown to

be numerically equal to the z-score of the simple Relevance Index.

$$T_c(S_k) = \frac{RI'(S_k) - \langle RI'h_k \rangle}{\sigma(RI'h_k)} = \frac{\nu RI(S_k) - \nu \langle RIh_k \rangle}{\nu \sigma(RIh_k)} = \frac{RI(S_k) - \langle RIh_k \rangle}{\sigma(RIh_k)} \tag{2.9}$$

where $\langle RI'h_k \rangle$ and $\sigma(RI'h_k)$ are the average and the standard deviation of the set of normalized relevance indices with the same size as S_k from the homogeneous system and $\nu = \langle MIh_k \rangle / \langle Ih_k \rangle$ is the normalization constant.

It is to be noticed that this method does not require any previous knowledge of the relationships among the system variables, but relies only on the observation of their values in time.

In this context, the possible subsets of the system variables (Candidate Relevant Sets, CRSs) can be ranked according to their T_c: the analysis returns a huge set of candidates, most of which are a subset (or superset) of other CRSs. In order to identify the most relevant information, in [78] a post-processing *sieving algorithm* has been proposed, able to reduce the list of CRSs to the most representative ones. The algorithm is based on the consideration that if CRS A is a proper subset of CRS B and ranks higher than CRS B, then CRS A should be considered more relevant than CRS B. Therefore, the algorithm keeps only those CRSs that are not included in or do not include any other CRS with higher T_c. This "sieving" action stops when no more eliminations are possible: the remaining groups of variables are the proper RSs.

To fully describe a dynamical system based on the T_c it would be necessary to compute such an index for all CRSs. Unfortunately, their number increases exponentially with the number of variables, soon reaching unrealistic requirements for computation resources. As a consequence, to extract relevant information about a system by observing its status over time, it is necessary to design efficient strategies which can limit the extension of the search by quickly identifying the most promising subsets. Section 2.2 provides an overview of some metaheuristics which can be used to counteract the complexity of an exhaustive RS search.

2.2 Metaheuristics

The term *metaehuristic* was coined by Glover [89] and combines the Greek prefix meta- (beyond in the sense of high-level) with heuristic (from the Greek heuriskein, to search). When the search space is non-continuous, high-dimensional, non-convex or multi-modal, local search methods are consistently outperformed by stochastic optimization algorithms [99]. Metaheuristics are general-purpose stochastic procedures designed to solve complex optimization problems [92, 65]. These optimization algorithms are non-deterministic and approximate, i.e., they do not always guarantee that they find the optimal solution, but they can find a good one in reasonable time. Metaheuristics require no particular knowledge about the problem structure other than the objective function itself, when defined, or a sampling of it [151]. The main objective of metaheuristics is to achieve a trade-off between diversification (exploration) and intensification (exploitation). Exploration implies generating diverse solutions to explore the search space on a global scale, while exploitation implies focusing the search onto a local region where good solutions have been found. An overview of the main proofs of convergence of metaheuristics to optimal solutions can be found in [101].

The most attractive features of metaheuristics are [253]:

- Simplicity: these algorithms are simple and easy to implement, their complexity is relatively low, they can be usually summarized in a few lines of pseudocode;
- Flexibility: these algorithms, though simple, are flexible enough to deal with a wide range of optimization problems in which more classical approaches fail. There is no need for a differentiable or continuous objective function and for information regarding the function like its gradient or Hessian;
- Ergodicity: these algorithms contain mechanisms, usually based on randomization techniques or statistical models, to search multimodal landscapes with sufficient diversity and ability to escape local optima.

Metaheuristics include:

- *Population-based methods*, in which the search process can be seen as the evolution in (discrete) time of a set of points (population of solutions) in the solution space (e.g., evolutionary algorithms [10] and particle swarm optimization [179]);
- *Trajectory methods*, in which the search process describes a trajectory in the search space and can be seen as the evolution in (discrete) time of a discrete dynamical system (e.g., simulated annealing [125] and tabu search [90]);
- *Memetic algorithms*, which are hybrid global/local search methods in which a local improvement procedure is combined with a population-based algorithm (e.g., scatter search [91]).

The following subsections describe some popular metaheuristics more in detail.

2.2.1 Evolutionary Computing

Evolutionary Algorithms (EAs) are metaheuristics based on a computational paradigm which replicates mechanisms inspired by those of biological evolution, such as reproduction, mutation, recombination, and selection, to solve optimization. Evolutionary computing has been very successful in solving hard multi-modal, multi-dimensional problems in many different tasks [61, 25].

When the dimension of the search space is large, evolutionary computing allows one to perform an efficient directed search, taking inspiration from the theories developed by Darwin ("survival of the fittest") to guide the search. In biological evolution, species are positively or negatively selected depending on their relative success in surviving and reproducing in the environment. A given environment is filled with a population of individuals that strive for survival and reproduction. In every population in nature, mutations occur from time to time. Mutations may be disruptive but may also generate individuals who are fitter with respect to the environment. The fitness of the individuals represents their chance of survival and of multiplying, producing more numerous offspring. The offspring individuals partly share their parents' genetic characters (chromosomes, made up of genes), partly define new types, obtained

by mixing such characters (crossover).

The fundamental metaphor of evolutionary computing relates natural evolution to the "trial-and-error" style of problem solving. Formally, given a problem and a quality function (fitness function), which can measure how good a solution to the problem is, processes modeled on natural evolution are used to generate solutions that have the highest possible fitness, i.e., that optimize (maximize or minimize) the fitness function. In practice, evolutionary algorithms implement an iterative process (artificial evolution) which leads to effective solutions to a given problem. Each individual of the population represents a solution to the problem; solutions improve over generations until they converge to an optimum, starting from an initial pool of very approximate, or even randomly generated, solutions. The new generations are created by applying recombination and mutation operators:

- Recombination (crossover) is applied to two selected individuals (parents), producing new individuals who derive partly from one parent and partly from the other one;

- Mutation is applied to one individual and results in one new individual.

Therefore, executing the recombination and mutation operators on the parents leads to the creation of a set of new candidate solutions (offspring). A typical evolutionary algorithm is composed of the steps described in Algorithm 1.

Algorithm 1 The general scheme (pseudocode) of an evolutionary algorithm.

```
Initialize a population (e.g., with random candidate solutions)
Evaluate population's fitness
repeat
    Select a population subset (parents), based on fitness
    Recombine pairs of parents
    Mutate the resulting offspring
    Evaluate the new population's fitness
until a termination condition is satisfied
```

Evolutionary computing includes several computational models that reproduce natural evolution processes to optimize a goal which is generally represented as a function, such as genetic algorithms [51, 117], genetic programming [180, 128] and evolution strategies [212].

Genetic algorithms (GAs) have been commonly used in two areas of research: optimization, in which GAs represent population-based optimization algorithms, and the study of adaptation in complex systems, in which the evolution of adapting entities is simulated over time [25]. The following subsection describes the canonical GA, focusing on its role as an optimizing methodology.

Genetic Algorithms

Genetic algorithms (GAs) are evolutionary algorithms inspired by the process of natural selection (survival of the fittest, crossover, mutation, etc.) [94] commonly used to solve optimization problems. A canonical GA can be described as an algorithm that operates on solution encodings, turning a population of candidate encodings into another using a number of stochastic operators.

In genetic terms, an individual's genetic code is called *genotype*, while the manifestation of the characters encoded in such a code (the individual itself) is called *phenotype*. Both genotype and phenotype play a role in evolution: the biological processes of diversity generation act on the genotype, while the *fitness* of an individual in the environment depends on the survival and reproductive success of its corresponding phenotype. Similarly, in a genetic algorithm a genotype-to-phenotype decoding mechanism needs to be defined, by which a solution is produced from its encoding, in order to evaluate its effectiveness. The genetic operators are applied to the encoding of a solution (genotype), while the phenotype associated with an encoding can have many different forms depending on the application of interest. Therefore, *chromosomes* (solutions) are represented as strings of symbols, most often bits or real numbers, and *individuals* may be anything that can be represented by a string of symbols. The structure of a canonical GA is shown in Algorithm 2.

The following paragraphs describe more in detail the main components of a GA.

Algorithm 2 The general scheme (pseudocode) of a genetic algorithm.

Determine how the solution is to be encoded as a genotype (representation) and define the fitness function
Initialize a population (e.g., with random chromosomes)
repeat
    Decode each chromosome into an individual
    Evaluate each individual's fitness
    Select some members from the current population of encodings (parents) in order to create a mating pool
    Generate a new population, partly by cloning (copying), partly by recombining, partly by mutating the chromosomes of the mating pool
until a termination condition is satisfied

Population A population is composed of individuals (chromosomes) and contains the representation of possible solutions. The population can be defined as a multiset of genotypes, i.e., a set which admits the presence of multiple copies of the same element (of solutions, points in the search space). The population *diversity* is the number of different individuals within the population, i.e., the number of distinct elements of the multiset.

A population is characterized by its *size* (number of individuals), which is most often kept constant through the generations. In some sophisticated EAs, populations are characterized also by an additional spacial structure, defined by a distance measure or a neighborhood relation, such that genetic operators can affect only members of the same neighborhood.

The efficiency of a GA can be improved by using problem-specific information to initialize the population. If good starting points are known a priori, they can be used to feed the initial population. More commonly, good starting points are not known and the population is randomly initialized.

Fitness function The fitness function corresponds to the objective function of the optimization problem to be solved by the GA. It measures the quality (effectiveness)

of a solution and must satisfy the following fundamental hypotheses (for a maximization problem):

- A measure Q exists for the quality of a solution;
- Q is positive;
- It must be maximized;
- An individual's Q is its fitness.

In summary, the fitness function is defined from the space X of the solutions to the positive real axis. GAs search a solution f such that:

$$f \in X : f = \max_{X}(Q : X \to \mathbb{R}^{+})$$

The term fitness is usually associated with maximization. However, if the optimization problem requires minimization, it can be mathematically changed into a maximization problem.

Evaluating the fitness of the individuals of the population is usually the most computationally expensive and time-consuming step in a GA. When fitness evaluation is very expensive, some efficient strategies can be adopted, in which an approximate fitness evaluation is used for most of the iterations of the GA, exposing the individuals to the real expensive fitness function only periodically during the run. Some commonly used approaches are:

- *Fitness approximation*: the original problem statement (fitness function) is replaced with a simpler one which approximates the problem of interest, assuming that a good solution for the approximated problem would be a good starting point in solving the problem of interest;
- *Fitness inheritance*: the fitness of a child is inherited from its parent(s), reducing the number of fitness function evaluations;
- *Fitness imitation*: all the individuals in a neighborhood (defined by a distance metric) are given the same fitness value.

Representation choice and generation of diversity The first step in defining a GA is to link the original problem context and the problem-solving space where evolution takes place, defining how the candidate solutions should be represented, stored and manipulated. The choice of the representation is crucial because it determines the nature of the search space and it also impacts on the appropriate design of the diversity generation operators (i.e., crossover and mutation).

Ideally, a GA works better if similar representations (genotypes) represent similar entities (phenotypes). This feature is known has *locality*: small (big) changes in the genotype should result in small (larger) changes in the phenotype and its associated fitness [25].

Three key characteristics for the quality of EAs have been presented in [186]:

- *Locality*: strong locality makes it easier to explore the neighborhood of good solutions, increasing the efficiency of the search, while weak locality makes the evolutionary search more similar to a random search;
- *Heritability*: the crossover (recombination) operator produces new individuals which combine the information contained in their parents' chromosomes in a meaningful way. Good heritability ensures that each property of a child should be inherited by one of its parents and that properties shared by both parents should be inherited by their children;
- *Heuristic bias*: in an unbiased case, if a genotype is randomly selected, each item in the phenotypic space has the same chance of occurring . With heuristic bias, some phenotypes are more likely to be created than others when sampling genotypes without any selection pressure. Inducing bias can be useful if it leads to better solutions, but at the same time it reduces genotypic diversity, which can hinder the search for the optimum.

Different kinds of representation have proposed (e.g, binary, integer, real-valued chromosomes).

Selection *Parent selection* is the strategy according to which individuals are selected for reproduction. To simulate natural selection, higher-fitness individuals must

have higher probability to be selected and become parents for the next generation. Usually, in GAs, a set of solutions is selected for mating (mating pool), then pairs of individuals are randomly extracted from the mating pool and are coupled for recombination. Offspring inherit part of their genotype from one parent and the rest from the other parent.

The design of the selection strategy determines the *selection pressure*, which is the degree of bias towards the selection of higher-fitness individuals of the population. Too low a selection pressure may lead to an inefficient search process, because the information coming from good individuals will spread slowly through the next populations, while too high a selection pressure may quickly reduce the population diversity, causing the GA to get stuck in a local optimum. Therefore, a good selection strategy should encourage the exploitation of high-fitness individuals without losing diversity in the population too quickly. To achieve this goal, different selection strategies have been proposed:

- *Fitness proportionate selection*: the probability p_i that an individual i of the current population is selected for mating is directly proportional to its fitness f_i relative to other members of the population:

  $$p_i = f_i / \sum_k f_k$$

  It is to be noticed that p_i is a proper probability, because $\sum_i p_i = 1$.
  Although this selection strategy is intuitive and easy to implement, it may produce poor results in practice because of its high selection pressure. If an individual's fitness is much higher than the average population fitness, but much lower than the optimum, it tends to be repeatedly selected, generating a mediocre uniform population (premature convergence). Conversely, if all individuals have similar fitness, they tend to have the same selection probability, causing the optimization strategy to become similar to a random search (stagnation).

- *Rank selection*: the individuals are ranked by fitness (in descending order) and their ranking is used to define a probability distribution function, decreasing

with rank, independent of fitness values. This strategy, which is not biologically plausible, is computationally heavier but allows one to avoid premature convergence, because no individual can have a selection probability much higher than any other, and stagnation, because the probability distribution does not change.

- *Tournament selection*: to select each individual, a random population subset of predefined size is picked, and the best is selected. This strategy is a commonly used, and computationally efficient (with no need for ordering), rank-selection method.
- *Elitist selection*: at least one copy of the best individual is kept in the new generation. Therefore, good solutions are not lost due to "random" selection strategies, but if the best individual's characters become dominant, this may lead to premature convergence.

Crossover Crossover (recombination) is a genetic operator that merges information from two parent genotypes, generating individuals whose genetic code derives partly from one parent and partly from the other one. Crossover implementation depends on the representation used for the chromosomes.

For binary representations, the most common crossover operators are:

- *Single-point crossover*: a position within the genome is randomly chosen and the children are created by swapping the right or the left sections;
- *Two-point crossover*: two "cuts" are made and the inner or the outer sections are swapped, considering the chromosome circular;
- *Uniform crossover*: each bit is randomly selected from one of the two parents for the first child and from the other parent for the second one.

For integer representations, the same operators can be used. If floating point vectors are used for the representation, the operators are conceptually similar (e.g., they can be modified such that elements from the chromosome of each parent are weighted averaged in order to produce the corresponding value in the child chromosome).

Mutation Mutation aims at introducing genetic diversity, exploring new regions of the search space which are not covered by the current population. Like crossover, mutation is a stochastic operator: its output depends on the outcome of a random choice.

For binary representations, a bit is chosen randomly and is inverted, while for integer representations it is possible to substitute a gene with a valid random value, or add a positive or negative quantity to it, from a probability distribution that is symmetric about zero. For floating point representations, gene values are substituted randomly, considering the values coming from a continuous rather than a discrete distribution.

Replacement Strategy Similar to parent selection, the objective of *survival selection* (*replacement*) is to distinguish individuals based on their quality. This strategy is used after the creation of the offspring from the selected parents and aims at deciding which individuals will survive in the next generation.

Replacement strategies can be grouped into two categories:

- *Fitness-based strategy*: the surviving individuals are selected from the unified multiset of parents and offspring, ranked according to their fitness;
- *Age-biased approach*: the surviving individuals are selected only by the offspring.

A lot of different replacement strategies have been proposed [25], including:

- *Direct replacement*: the parents are replaced directly by their children;
- *Random replacement*: the new population is selected randomly from the individuals of the current population and their children;
- *Replacement of the worst*: all parents and children are ranked by fitness and the lowest-fitness individuals are eliminated;
- *Tournament replacement*: the loser of the tournament is selected for replacement.

According to the replacement strategy adopted, two categories of GAs can be defined. In *generational GAs*, the number of children produced in each generation is equal to the size of the current population and during replacement the entire current population is replaced by the offspring. The ratio between the number of children produced and the size of the current population is known as the *generation gap*. It is also possible to create a offspring which is more numerous than the current population, and then select the best n (where n is the current population size). Another popular approach is used by *steady-state GAs*, in which only a small number of children (lower than the population size) are created. In this case, only some individuals of the current population are replaced.

To guarantee that the quality of the best solution candidate never worsens from one generation to the next, evolutionary algorithms often employ a technique known as *elitism*. The best individual (or the k best individuals) is transferred without modification to the next generation. This ensures that the best solutions found never get lost during the evolution.

Termination condition Ideally, if the problem has a known optimal fitness level, the stopping condition may accept a solution that approaches the optimal fitness within a given precision $\varepsilon>0$. Given the stochastic nature of GAs, there are no guarantees of reaching such an optimum, so this kind of condition might never get satisfied. Therefore, this condition must be extended with one that certainly stops the algorithm. Commonly used termination conditions are [61]:

- the maximally allowed CPU time elapses;
- the total number of fitness evaluation reaches a given limit;
- the fitness improvement remains under a threshold value for a given period of time (e.g., for a number of generations or fitness evaluations);
- the population diversity drops under a given threshold.

2.2.2 Swarm Intelligence

Swarm intelligence aims at designing intelligent systems by taking inspiration from the collective behavior of animal societies. In social behaviors, individuals can learn both from their own experience and from the experience of others. Therefore, a group can solve complex tasks which are beyond the capability of any of its members [25].

A social group is characterized by the following properties [23]:

- *Flexibility*: the group can respond to internal perturbations and external challenges;
- *Robustness*: the tasks are completed even if some individuals fail;
- *Decentralization*: there is no central control(ler);
- *Self-organization*: paths to solutions are emergent rather than predefined.

The task is not simply divided among multiple individuals, it is solved by the group using communication and a shared understanding of the problem (*social computing*). The collective behavior of a swarm of social organisms emerges in a nonlinear manner from the behaviors of the individuals of that swarm [65]. Examples of swarm intelligence in natural systems include ant colonies, bird flocking, fish schooling and microbial intelligence.

The term swarm intelligence was first used in the context of cellular robotics systems [16], where simple agents interact and organize themselves. Currently, swarm intelligence methods have become very successful in a broad range of contexts [22], especially in the area of optimization.

Commonly used swarm intelligence algorithms are Particle Swarm Optimization [60] and Ant Colony Optimization [56]. The following subsection describes Particle Swarm Optimization more in detail.

Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a bio-inspired optimization algorithm based on the simulation of the social behavior of bird flocks. The algorithm was introduced by

Eberhart and Kennedy in [60]. Since its introduction, many variants of the original algorithm have been proposed [43], applying PSO to a wide variety of problems [178], thanks to its simple implementation and ability to reach good results quickly. PSO provides an effective solution to multifaceted optimization problems and is often used for finding the optimal values for the parameters of other algorithms, such as hybrid SVM [137], deep belief neural networks [223] and artificial bee colony [97].

In PSO a swarm of s particles moves within a function domain (fitness function), searching for the optimum of the function (best fitness value). Each particle of the swarm is characterized by:

- Position (x) in the search space;
- Fitness value at this position;
- Velocity (v), used to compute the next position;
- Memory (previous best) of the best position found so far by the particle;
- Fitness value of the previous best position.

The movement of each particle is influenced by its own best known position (previous best, $\vec{p}_i$), but is also guided towards the best known positions of the other particles ($\vec{p}_g$) which are updated as better values of the fitness function are found.

PSO pseudocode is described in Algorithm 3.

Algorithm 3 The general scheme (pseudocode) of PSO.

Initialize a swarm of s particles with random positions and velocities in the D-dimensional search space
repeat
  For each particle, evaluate the optimization fitness function (in D variables)
  For each particle, update the best positions $\vec{p}_i$ and $\vec{p}_g$, comparing fitness values
  Update velocity (v) and position (x) of each particle
until a termination condition is satisfied (usually a sufficiently good fitness or a maximum number of iterations)

The particles move within the search space as a result of the following update steps [179]:

$$\begin{cases} \vec{v}_i \leftarrow \chi(\vec{v}_i + \vec{U}(0,\phi_1) \otimes (\vec{p}_i - \vec{x}_i) + \vec{U}(0,\phi_2) \otimes (\vec{p}_g - \vec{x}_i)) \\ \vec{x}_i \leftarrow \vec{x}_i + \vec{v}_i \end{cases} \tag{2.10}$$

where χ is the so-called *constriction factor* [179], U are random variables uniformly distributed in $[0,\phi_i], i = 1,2$, where ϕ_i's (acceleration coefficients) are positive constants such that

$$\phi = \phi_1 + \phi_2 > 4 \tag{2.11}$$

$$\chi = \frac{2}{\phi - 2 + \sqrt{\phi^2 - 4\phi}} \tag{2.12}$$

Such constraints have been found to guarantee system stability (finite speed) [44].

In the *global-best* PSO algorithm, $\vec{p}_g$ is the best-fitness position visited so far by any particle of the swarm, therefore it is the same for all particles. In several variants (*local-best* PSO), the swarm is subdivided into smaller neighborhoods, which can assume different topologies. Three of the most commonly used neighborhood topologies (global, ring and star) are shown in Figure 2.1. In the global topology, each particle is connected to all others. In the ring topology, the neighborhood of each particle is composed by itself, the previous K and next K particles, where K indicates the radius of the neighborhood. In the star topology, one particle is selected as "center" and acts like in a "global-best" PSO, while the neighborhood of the other particles is limited to the central one and themselves.

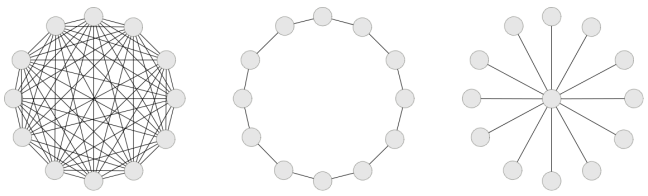


Figure 2.1: Commonly used PSO topologies: global, ring, and star.

2.2.3 Niching Algorithms

In the context of optimization, many real-world problems can be difficult to solve, since the existence of several local optima can significantly complicate the search. Optimization problems with a fitness landscape featuring many local optima are called *multimodal*. An example of multimodal function is the Rastrigin function:

$$f(x) = A \cdot n + \sum_{i=1}^{n} [x_i^2 - A \cdot cos(2 \cdot \pi \cdot x_i)] \tag{2.13}$$

where A=10 and $x_i \in [-5.12, +5.12]$. This function, represented in Figure 2.2 as a 2-dimensional function, has several local minima and a global minimum at $x = (0, ..., 0)$, where $f(x) = 0$.

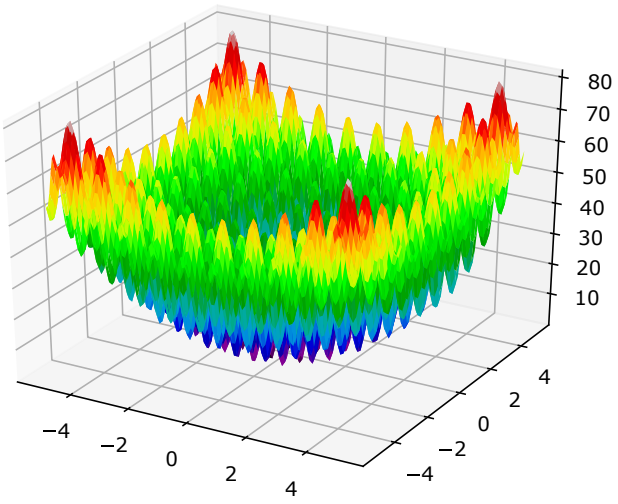


Figure 2.2: 2-dimensional Rastrigin function.

For many practical problems it can be useful to detect as many local and potentially global optima as possible. Multimodality is a typical aspect of the type of problems for which evolutionary computing and swarm intelligence are often employed, either with the aim of locating the global optimum, or identifying a set of high-fitness solutions corresponding to local optima.

Apart from the standard mechanisms related to population-based search, which combine exploration and exploitation, a lot of strategies have been proposed that explicitly aim at dealing with a multimodal solution space [181]. Among these, *niching* techniques assume great importance. Niching is a concept imported from evolutionary biology (ecology) that refers to the process by which competing species use non-overlapping resources (resource partitioning) and thus occupy different ecological niches, in a way that helps them to coexist. The concept of "sharing limited resources within an ecological niche", proposed by Holland in [117], is one of the most effective approaches for creating and maintaining stable subpopulations around the peaks of the objective function, using a population-based metaheuristic [218]. Niching metaheuristics maintain population diversity during the search process and allow the search to explore many peaks in parallel. These techniques aim at explicitly detecting basins in the solution space and focusing the search in each niche, finding as many local optima as possible.

Niching methods can be categorized in *Sequential Niching* algorithms, which develop niches sequentially over time, and *Parallel Niching* algorithms, which form and maintain several niches simultaneously.

The term *species* often denotes separate subpopulations in niching algorithms. The main speciation behaviors are [146]:

- *Sympatric speciation*: individuals are organized in species that coexist geographically in the same search space, but evolve to exploit different resources (different ecological niches);

- *Allopatric speciation*: individuals differentiation is based on spatial isolation in the search space;

- *Parapatric speciation*: new species can be generated as a result of segregated species sharing a common border.

The following subsections describe the most popular niching techniques applied to genetic algorithms and particle swarm optimization, respectively.

Niching Genetic Algorithms

Genetic Algorithms (GAs) are popular search and optimization techniques, particularly effective when little knowledge is available about the function to optimize. The population-based nature of genetic algorithms might seem suitable for identifying multiple optima, however, in practice, the finite population size, coupled with recombination between any parents (panmictic mixing) can lead to the phenomenon of *genetic drift*, where the population may suffer from a loss of variety, eventually losing some high-fitness individuals and converging around a single optimum [61].

Some studies [39, 110] show that GAs are not well-suited to fine-tuning searches in complex spaces and that their hybridization with local search methods, often referred to as memetic algorithms (MAs) [39], can greatly improve their performance and efficiency. Nevertheless, basic GAs and MAs are designed to find absolute optima and therefore are not capable of maintaining the diversity of solutions during evolution, which is essential when multimodal functions are analyzed, and the goal is to find as many local optima as possible.

To compensate for this shortcoming, Goldberg and Richardson [95] proposed an adaptation of the ecological niche concept for genetic algorithms. Niching GAs join search capabilities to good exploration properties, which allow them to explore different regions of the search space in parallel and converge onto several local/global optima.

Most niching methods, however, often require problem-specific parameters, strictly related to the features of the search space, to be set *a priori* to perform well. This is documented, for example, in [15], [37], and [174], that describe applications to mechatronics, image processing, and multimodal optimization, respectively.

Some of the most renowned niching GAs are described in the following paragraphs.

Fitness Sharing The Fitness Sharing algorithm, proposed in [95], is based on the idea of assigning the same (or similar) adjusted fitness to numerous solutions in a niche to maintain diversity in the population.

Intuitively, the individuals share the resources of a niche. To this aim, it is essen-

tial to introduce a concept of *dissimilarity* between individuals (e.g., if the individuals are represented by bit strings, the Hamming distance may be appropriate). The value of the dissimilarity is the criterion to decide whether two individuals belong to the same niche or not. Fitness sharing methods consist in assigning a shared fitness to each individual. The value of the shared fitness of each individual is obtained by dividing its raw fitness by a quantity that increases with the number of individuals resembling it:

$$f_{shared}(i) = \frac{f_{raw}(i)}{n_i}$$

where n_i (*niche count*) represents the number of points in the population within a certain distance from an individual i:

$$n_i = \sum_j sh(d(i,j))$$

where $d(i,j)$ is the distance between the individuals i and j, and $sh(d(i,j))$ (*sharing function*) defines both shape and size of the niche. The power-law sharing is commonly used as a sharing function:

$$sh(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{share}}\right)^{\alpha} & if \quad d_{ij} < \sigma_{share} \\ 0 \qquad otherwise \end{cases}$$

where σ_{share} is the *niche radius* and α is the *sharpness* parameter, which determines the shape of the sharing function.

A simple but effective variant consists in assigning all solutions in a niche to their average fitness value [129]. The selection operator cannot distinguish between the solutions anymore and is forced to treat them in the same way. Therefore, numerous worse solutions survive in the population and the evolutionary process is prevented from converging too fast to the global optimum. However, such a mechanism hinders convergence towards the most attractive regions, which represents the typical tradeoff between exploration and exploitation.

It is to be noticed that the use of fitness proportionate selection is implicit within the fitness sharing method. Studies have shown that the use of alternative selection

methods does not lead to the formation and preservation of stable subpopulations in niches [166].

The main drawbacks of fitness sharing are the appropriate choice of the niche radius σ_{share} and the algorithmic complexity ($O(N^2)$, where N is the population size), due to the calculation of distances.

Deterministic Crowding The Deterministic Crowding algorithm was proposed by Mahfoud [145] as an improvement of the the method of niching by crowding presented by De Jong [51]. Crowding is a way of preserving diversity by ensuring that offspring should always replace those individuals in the population which are most similar. Like fitness sharing, crowding needs a similarity or distance measure between individuals, but in this case it operates at the level of the replacement operator.

The main idea of Deterministic Crowding is that a pair of offspring o_1 and o_2, obtained after crossover and mutation, enters into competition only with its two parents, p_1 and p_2. This niching technique relies on the fact that offspring are likely to be similar to their parents, and the replacement of a parent by an offspring is effective only if the parent is less efficient (has a lower fitness) than the offspring, as shown in Algorithm 4.

The algorithmic complexity of Deterministic Crowding is low, since it scales as $O(N)$, where N is the population size, which is lower by an order of magnitude if compared to the Fitness Sharing algorithm. Moreover, Deterministic Crowding does not require the determination of a good setting for problem-related parameters, such as the niche radius. In fact, only the population size is significant and is chosen according to a simple criterion: the larger the number of optima to found, the larger the population.

Despite all the interesting advantages, the main drawback of Deterministic Crowding is its replacement strategy, which always favors the best individuals. This can lead the algorithm to reduce genetic drift less significantly than more powerful methods like Fitness Sharing. Otherwise, the simplicity of the method together with its low computational complexity are probably the main reasons why the usage of deterministic crowding is still often reported in the recent literature [132, 248, 255].

Algorithm 4 Deterministic Crowding pseudocode.

The parent population is randomly paired
Each pair produces two offspring via recombination
These offspring are mutated and then evaluated
The four pairwise distances between parents (p_1 and p_2) and offspring (o_1 and o_2) are computed
Each offspring competes for survival in a tournament with one parent so that the intercompetition distances are minimized:

```
if distance(p1,o1) + distance(p2,o2) ≤ distance(p1,o2) + distance(p2,o1) then
    if fitness(o1) > fitness(p1) then
        p1 ← o1
    end if
    if fitness(o2) > fitness(p2) then
        p2 ← o2
    end if
else
    if fitness(o2) > fitness(p1) then
        p1 ← o2
    end if
    if fitness(o1) > fitness(p2) then
        p2 ← o1
    end if
end if
```

Clearing The Clearing algorithm, presented by Petrowski in [176], is based on limited resource sharing within ecological niches, according to the principle proposed by Holland [117]. The resources are not equally distributed among the individuals. Indeed, the clearing procedure typically assigns all the resources of a niche to the best individual (dominant individual), while the other individuals in the same niche (dominated individuals) will not have anything (fitness=0).

The algorithm determines a set of subpopulations in which the dominant individuals are identified. To do so, a distance which is significant for the problem is chosen and the niches are represented as "balls" of radius σ_{clear} (*niche radius*) centered on the dominant individuals. The parameter σ_{clear} is analogous to the σ_{share} parameter of the Fitness Sharing algorithm. Its value should be lower than the distance between two optima of the fitness function, so they can be distinguished.

To discover all the dominant individuals in the population, a Clearing procedure is iterated. The population is initially sorted by decreasing fitness value. Then, the Clearing procedure (described in Algorithm 5) is applied.

Algorithm 5 Clearing procedure pseudocode.

The first individual of the population is the best individual (dominant individual)
The distances of all the individuals from the dominant one are computed
The individuals located at a distance $d < \sigma_{clear}$ belong to a niche centered on the dominant individual. Thus, they are dominated and are assigned a fitness value of 0.
The dominant and dominated individuals are virtually withdrawn from the population
The procedure is then reapplied to the new reduced population

The Clearing procedure is applied just before the application of proportional selection and is iterated while the algorithm finds dominant individuals. This strategy is elitist, since it preserve the dominant individuals from the subpopulations to transfer them into the next generation.

The principal quality of the method is given by its high resistance to the loss of diversity by genetic drift. Conversely, the main difficulty lies in the choice of the

niche radius (σ_{clear}) parameter.

The Clearing procedure requires about $O(c \cdot N)$ distance computations, where c is the number of niches and N is the population size.

Restricted Tournament Selection Restricted Tournament Selection, presented by Harik in [103], is based on the concept of direct local competition. The algorithm adapts standard tournament selection for multimodal optimization. The selection process is adapted as described in Algorithm 6.

Algorithm 6 Restricted Tournament Selection pseudocode.

Two individuals A and B are randomly selected from the population
The selected individuals are crossed and mutated, generating two new individuals (A' and B')
For each of A' and B', a distinct group of w members of the current population is evaluated. Among the groups, the closest individuals to A' and B' are called A'' and B'', respectively
A' then competes against A'' for a spot in the next generation, the same for B' and B''

This kind of tournament should prevent an entering individual from competing with others that are too different from it. The procedure described in Algorithm 6 is repeated $N/2$ times, where N is the population size.

The algorithmic complexity of Restricted Tournament Selection is $O(w \cdot N)$.

Niching Particle Swarm Optimization

The aim of standard PSO is to find a single point representing the global optimum of an N-dimensional function. Thus, canonical PSO is not suitable for application to multimodal optimization problems due to its convergent nature. The standard PSO must be modified to favor an efficient location of multiple solutions [66].

Various approaches have been developed to maintain diversity in PSO, increasing the particles' diversity degree in order to separately explore different regions of the search space [25, 136].

The particles in the swarm interact in a straightforward way, by sharing knowledge of the search space with the neighbors. Thus, the neighborhood structure of the swarm is a relevant aspect in the design of Niching versions of PSO.

The following paragraphs briefly summarize various Niching PSO algorithms.

Objective Function Stretching Introduced by Parsopoulos et al. [171], Objective Function Stretching was one of the first strategies developed for analyzing multimodal functions. Its main purpose is to overcome the limitations of PSO due to untimely convergence to local solutions. The stretching approach modifies the fitness function to remove previously identified local optima. In this way, successive iterations of PSO can explore different regions of the research space and identify new solutions.

Considering a minimization problem, when the swarm converges onto a (possibly local) minimum, the fitness function landscape is transformed by applying a stretching function to the original function. After storing its original value, the fitness of that position is stretched so that it becomes a local maximum, to prevent the swarm from returning to the previously discovered minimum. This algorithm has been further improved by Parsopoulos and Vrahatis [172] to use it as a sequential niching approach.

The main advantage of the stretching technique is that it requires no modifications of the underlying PSO algorithm. The performance of this method is good on many multimodal problems but in some cases the stretching technique can introduce false minima, since its effectiveness is not uniform on every function.

NichePSO In 2002, Brits et al. introduced the nbest PSO algorithm, the first technique using parallel niching in particle swarm [28], which is particularly suitable for finding multiple solutions in a system of equations. The same authors have also proposed another approach, which employs sub-swarms to locate multiple solutions in multimodal optimization problems, called NichePSO [27].

In NichePSO, a main swarm can generate sub-swarms each time a possible niche is identified. The main swarm updates particle velocities using the cognition-only

model, which considers only their personal best position. Therefore, particles perform a local search and move towards the most promising regions of the search space, facilitating sub-swarm formation. The algorithm performs alternatively a step of the cognition-only model on the main swarm and a step of the Guaranteed Convergence PSO algorithm (which ensures convergence to a local optimum) on the sub-swarms.

The standard deviation σ_i of the fitness of particle i is tracked over e_σ iterations and a new sub-swarm is created when $\sigma_i < \delta$, where δ is a problem-dependent small value. The closest neighbor of particle i is then added to the sub-swarm.

A sub-swarm S_j can absorb new particles or can be merged with other swarms according to some rules which depend on the measure of the sub-swarm radius R_j:

$$R_j = \max_{x \in S_j} ||x^*_{S_j} - x||$$

where $x^*_{S_j}$ is the best particle in the sub-swarm. A particle is absorbed into a sub-swarm if it moves into it, while two sub-swarms are merged if they intersect.

NichePSO is able to successfully locate and maintain multiple optimal solutions, but its performance strongly depends on tunable parameters (e.g., δ).

Species-based PSO The Species-based PSO, proposed by Li in [135], encompasses the idea of classifying the population in groups of species.

The procedure for the species identification is analogous to the Clearing procedure described in section 2.2.3. The definition of a species includes the seed, i.e., the particle with best fitness function, and the radius of the species, representing the Euclidean distance from the seed to the borders of the species. All particles within this radius belong to the same species. This approach implicitly assumes that all the niches have approximately the same extension.

Species-based PSO can dynamically identify the number of niches in the fitness landscape. However, the problem with this approach is the need to set the value of the radius, especially when the function presents niches of different dimensions.

Adaptive Niching PSO Bird and Li proposed the Adaptive Niching PSO (ANPSO) [20], which removes the need to set the niche radius according to the specific problem

and allows one to compute the parameters adaptively during execution.

ANPSO uses an indirect graph to track the minimum distance between the particles, during the execution of the algorithm, and the niches are formed through subgraphs that exclude all disconnected particles.

This algorithm is more flexible than the previously presented techniques, since it does not require setting problem-dependent parameters. However, the computational cost of the niche formation procedure is higher with respect to other techniques.

Parallel Vector-based PSO The Vector-Based PSO has been proposed by Schoeman and Engelbrecht [209, 210]. This method identifies the niches by exploiting some properties of the velocity vectors and some operations on them already present in the canonical PSO.

The original algorithm identified and exploited niches in a sequential way, starting from the global best and then iterating the procedure for the particles outside its niche. The same authors have developed a parallel version of the algorithm, based on the same principles. The niches are identified and maintained in parallel, with the introduction of a procedure to merge niches when they become closer than a specified threshold.

The algorithm is efficient but it is outperformed by other approaches, such as NichePSO.

K-means PSO K-means PSO, proposed in [173], combines the canonical PSO algorithm with a clustering technique, namely K-means [144], which is used to group the particles in sub-swarms.

A local search is performed in each sub-swarm using the PSO *gbest* topology, which considers the best individual's position as the only global attractor. Clustering is repeated at regular intervals.

The algorithm is simple and its performance is competitive with the previously presented techniques. However, it has two problem-dependent parameters: the number of clusters for k-means clustering and the number of PSO iterations between two clustering applications.

Part II

Proposed Methods

Chapter 3

Advanced RI metrics

The Relevance Index (RI) metrics are a set of information-theoretical metrics derived from the Relevance Index method, described in section 2.1.2, which can be used to analyze complex systems by detecting the main interacting structures within them. Starting from a collection of observations of the state of the system variables, the purpose of the RI metrics is the identification of subsets of variables that behave in a coherent and coordinated way, while loosely interacting with the remainder of the system. These subsets can be used to describe the whole system organization, thus they are named Relevant Subsets (RSs).

This chapter presents a methodological analysis of the RI metrics, proposing some improvements in terms of both computational efficiency and effectiveness for complex systems analysis. In particular, sections 3.2 and 3.3 propose two advanced metrics based on the RI methodology, while section 3.4 presents a GPU-based massively parallel implementation of the algorithm used to compute them.

The advanced metrics have been evaluated on different case studies, representing both natural and artificial systems. A detailed description of such systems is reported in the following section.

3.1 Case studies

The RI methodology can be applied to a broad range of dynamical systems, since it relies only on the observations of the states of the system variables over time, as shown in section 2.1.2.

The present section describes some case studies which will be used in this thesis to assess the performance of the proposed methodologies. In particular, subsections 3.1.1 and 3.1.2 describe artificial systems obtained by generating synthetic data, while subsections 3.1.3 and 3.1.4 describe real-world systems.

3.1.1 Boolean Networks

A Boolean Network [122] (BN) is a discrete dynamical system composed of a set of N boolean variables. Each variable represents a node in a directed graph with links from K neighboring nodes, and associated with a boolean function assuming binary values 0 or 1 (inactive/active). At each time step, the value of each node's boolean variable is obtained by evaluating its function using the current values of its neighbor nodes as arguments. The state of each node at time $t+1$ depends on the state of the neighbors at time t.

In a Random Boolean Network (RBN), the functions, links, and initial conditions are all randomly generated. Extensive studies have shown that these kinds of networks can lead to regular behavior patterns.

The path (trajectory) described by the transitions will sooner or later reach a previously visited state, and thus, since the dynamics are deterministic (both the topology and the Boolean function associated to each node do not change in time), the trajectory will fall into a steady state or cycle, which is called attractor. The network dynamics is discrete and synchronous, so fixed points and cycles are the only possible asymptotic states in finite networks.

The Boolean framework, despite its apparent simplicity, has obtained remarkable results in simulating several aspects of real gene regulatory networks [124, 215] which spontaneously display some rather stable, ordered behavior.

The aim of this case study is to check whether the RI methodology is able to rec-

ognize special topological cases, such as causally independent subnetworks, starting from the observation of the variable states. Figure 3.1 shows an example of a system composed of independent RBNs.

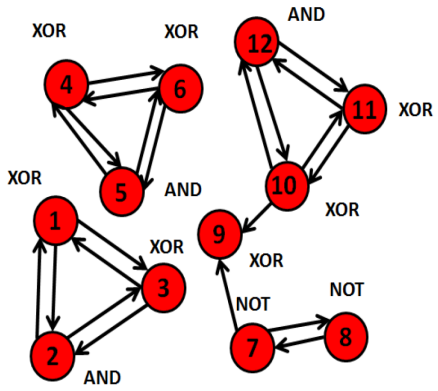


Figure 3.1: A system composed of independent RBNs.

The analyzed data can be obtained from the juxtaposition of different states belonging to the different attractors of each network [240] or of different trajectories coming from different perturbations (e.g., temporarily changing a randomly chosen value), as will be shown in section 3.2. It is to be noticed that the RI approach can be applied to this kind of data because the results it provides do not depend on the order in which observations are taken into consideration.

3.1.2 Leader-Follower model

This case study represents a simplified model of the dynamics of opinion diffusion. The model abstracts situations where agents (followers) modify their opinion agreeing with (or contrasting) the opinion of other specific agents, called leaders.

The system is composed of a vector of n binary variables $\{X_1, X_2, ..., X_n\}$ representing, for example, the positive or negative opinion of n agents about a given proposal. The model generates independent observations of the system state, i.e., each observation is a binary n-vector generated independently of the others, according to

some predefined rules.
At each step, the leaders' variables randomly assume value 0 or 1 according to preset probability values. The other variables (i.e., the followers) generally copy or negate the values of their leaders. The values of the followers' variables can also be given by computing boolean functions (e.g., OR, AND, ...) of two or more leaders' variables. Given these rules, the system comprises only well-defined and non-interacting groups of agents, dynamically separate with respect to the other independent random variables. However, its stochasticity could occasionally support the emergence of spurious relationships, which make the automatic detection of groups non-trivial.

This case study aims at evaluating the ability of the RI methodology to recognize the groups of agents characterized by different Leader-Follower dynamics.

3.1.3 Catalytic Reaction System

This case study simulates a collection of molecules able to collectively self-replicate [124], a situation frequently studied in researches about the origin of life [147]; very similar assemblies could play an important role also in future bio-technological applications [221]. Indeed, currently, living beings are based on self-replicating chemical structures, where the presence of enzymes (biological catalyzers) plays an essential role.

The data come from dynamical simulations of autocatalytic reaction systems occurring within a Continuous-flow Stirred-Tank Reactor (CSTR for short), featuring a constant influx of feed molecules and a continuous outgoing flux of all the molecular species proportional to their concentration (see [240] for a more detailed description of the model). The reactions occur only in the presence of a specific catalyst, since spontaneous reactions are assumed to occur too slowly to affect the system behavior.

The dynamics of this kind of system have been described adopting a deterministic approach whereby the reaction scheme is translated into a set of Ordinary Differential Equations ruled by the mass action law (see [214] for further details) and integrated by means of a Euler method with step-size control.

We simulated different kinds of chemical systems featuring distinct reaction pathways. Figure 3.2 shows an example of the reaction scheme.

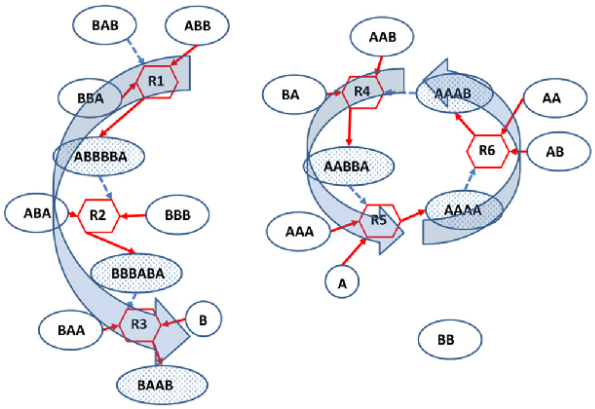


Figure 3.2: An example of the reaction scheme of the catalytic reaction system: white ellipses represent the chemicals injected in the incoming flux, meshed ellipses represent the chemicals produced inside the CSTR vessel, hexagons represent the reactions; continuous arrows represent the consumptions/productions and dashed arrows represent the catalytic activities. Chemical BB does not participate in any reaction, and is used as reference. The six reactions are arranged into two independent groups: a linear chain and an autocatalytic circle.

The objective of this case study is the detection of the groups of chemicals that participate in distinct dynamical organizations, by simply observing their concentration in time.

The asymptotic state of this system consists of constant concentrations. In this case, the typical attractors are fixed points, which do not provide any useful information for computing the RI. In order to apply the RI analysis, however, we need to observe the feedbacks dynamically: thus, the concentration of some molecules has been perturbed in order to trigger a response (i.e., a series of changes) in the concentration of (some) other species. The perturbations consisted of temporarily setting to zero the concentration of some species after the system reached its stationary state.

The analyzed variables can represent:

- the final products of the reactions;

- the reagents and the products;
- the reagents, the products and the intermediate complexes (combining the substrates and the catalysts).

In order to analyze the system response to perturbations, the system has been observed within equally-sized, non-overlapping time windows, e.g., the behavior of the chemical concentrations within this intervals can be discretized as follows:

- two-level encoding: “chemical concentration changing” (“1”) and “no change in chemical concentration” (“0”);
- three-level encoding: for each species, the digits “0”, “1” and “2” stand respectively for “decreasing concentration", “no change" and “increasing concentration”.

3.1.4 Green Community Network

This case study corresponds to a real-world environment (participation of partners in project meetings) and consists of a set of data extracted from a very large and complex corpus collected during the monitoring of the Green Community (GC) project. The project started in Italy in 2012, within a call supported by the EU Interregional Operational Program; it initially involved only four mountain communities, dealing with a core topic about energy efficiency and renewable energy production-related issues. Later, it was extended to other social and organizational themes, gradually involving many heterogeneous stakeholders, including specialists, engineers, researchers, local administrations and representatives. In order to manage this increasing complexity, the project decided to take advantage of an evaluation approach, the Dynamic Evaluation, developed within the “Emergence by Design” European project[1], aimed at supporting the monitoring, management, and development of projects and programs [7].

The data are extensive and rather complex; on the one hand, the overall data structure was not designed with the RI application in mind; on the other hand, the

[1] http://cordis.europa.eu/project/rcn/102441_en.html

data do contain information that social scientists are not used to analyze and that could be analyzed successfully through our techniques.

Therefore, in order to observe the presence of (formal or informal) coalitions within the four mountain communities, we decided to extract from this database only the data about the involved stakeholders' attendance (or absence) at some significant points-of-control[2] of the GC project. Our idea was that the simple presence or absence at important meetings (or even the mere permission to participate in them) could carry significant information about the project. We obtained a very sparse matrix, composed of 137 variables and 124 points-of-control (observations, corresponding to the meetings).
The objective is to infer some insightful indications about the formation of coalitions during the GC project, analyzing this matrix using the RI methodology, in order to verify whether the RI analysis can evidence the formation of specific dynamics among subsets of participants, despite the apparently simplicity of the information carried by the observations.

Given the large size of this system, the RI analysis requires the use of some metaheuristics, which will be described in chapter 4. The experimental results which will be shown in the present chapter are related to the case studies described in the previous subsections (3.1.1, 3.1.2 and 3.1.3).

3.2 Homogeneous system with correlation

As shown in section 2.1.2, the value of the RI depends on the size of the group under examination. Therefore, it should be normalized to deal with subsets of different size. Following the normalization proposed by Tononi et al. [231], a null hypothesis can be considered: the absence of correlated subsets. The null hypothesis is provided by a reference (homogeneous) system, in which the variables have, individually, the same marginal distributions as in the dataset, and all have the same pairwise correlation. In

[2]These points-of-control are very heterogeneous and include official meetings, global conferences, other relevant panels, e-mail discussions. Moreover, they are distributed over time without a predefined frequency.

particular, a system characterized by independent Gaussian variables, i.e., with zero pairwise correlation, was originally taken as a reference [240].

This section presents a further improvement to the way the homogeneous system is conceived from a theoretical viewpoint. Firstly, we consider the system components as dependent and with equal pairwise correlations, which implies a non-diagonal correlation matrix of the homogeneous system. Then, we generate the components of the homogeneous system according to a multivariate Bernoulli distribution, by exploiting the NORTA method [36], which is able to create samples of a desired random vector, given its marginal distributions and its correlation matrix. A more detailed description of the proposed methodology is provided in section 3.2.1.

The introduction of the pairwise correlations has allowed us to identify particularly interesting groups of variables, undetected in previous experiments. The improved method has been applied to three different case studies, obtaining better results compared with the traditional method based on the homogeneous system with independent Gaussian variables. The experimental evaluation is presented in section 3.2.2.

3.2.1 Theoretical approach

The generation of the homogeneous system is critical, as stated also in [240], and often, in the past, a simple but general and easy to compute solution was preferred. This solution encompassed the computation of the frequency of occurrence of each variable, given the available observations, and the generation of a new random series of samples, where each variable had a prior probability equal to the frequency of the original observations. The homogeneity required by Tononi [231] was achieved by considering the components of the random vector U_h to be Gaussian and independent. This caused:

1. the correlation matrix of the homogeneous system to be a diagonal matrix, i.e., with pairwise correlations set to zero;

2. the integration $I(S_k)$ to be zero for all subsets of the homogeneous system.

The proposed improvement consists in taking the pairwise correlation between variables describing the homogeneous system into account, requiring that it be not null, which seems a more realistic assumption. In this way, we remove a hypothesis (the independence of the variables) that is not true in general. Moreover, we maintain the homogeneity required by Tononi, by forcing all off-diagonal elements of the correlation matrix to have the same constant value ρ:

$$CORR(U_h) = \begin{bmatrix} 1 & \rho & \dots & \rho \\ \rho & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \rho \\ \rho & \dots & \rho & 1 \end{bmatrix}$$

while we normalize all variances to 1. The value of ρ is computed, in a first approximation, as the average value of all pairwise correlations of the observed variables.

In order to generate a homogeneous system with the aforementioned features, we take advantage of the NORTA method [36], whose theoretical background is presented in the following subsection, together with the description of its application to the RI method.

The NORTA Method

NORTA ("NORmal To Anything") is a method devised to generate specifically correlated random vectors [36]. This is a mathematical procedure that solves the issue of creating random vectors of correlated samples, given the set of their marginal distributions (marginals) and a measure of the dependence among them.

This is a good choice in our scenario, since, usually, the majority of complex systems components experience a certain degree of mutual dependence [250]. Some recent examples, where NORTA has been successfully employed in different fields, include wind power generation in renewable power supply systems [165] and the modeling of probabilistic load flows, based on Latin Hypercube Sampling [251]. Indeed, NORTA presents some degrees of uncertainty in the estimation of the marginal distributions and of the correlation matrix [250], since it is not always guaranteed that

its samples have exactly the desired correlation matrix. However, we found it useful to overcome some issues encountered in applying the original RI method to some simple systems described by a moderate number of variables.

The measure of dependence we used in NORTA is the usual *product-moment* correlation matrix, based on the linear Pearson correlation coefficient with entries defined according to the following formula:

$$\rho(X_i, X_j) = \frac{COV(X_i, X_j)}{\sigma_{X_i}\sigma_{X_j}}. \tag{3.1}$$

The NORTA method creates independent and identically distributed replicas of a random vector $\mathbf{X} = (X_1, X_2, ..., X_n)$, based on its (known) marginal distributions $F_i(x) = P(X_i \leq x), \quad i = 1, \ldots, n$ and the correlation matrix $CORR(\mathbf{X})$.
In summary, the NORTA procedure performs the following steps:

1. it generates a normal random vector $\mathbf{Z} = (Z_1, Z_2, ...Z_n)$ with zero mean and covariance matrix $COV(\mathbf{Z})$, with 1s on the main diagonal;

2. it obtains the prescribed marginal distributions by computing the replica $\mathbf{X}' = (X_1', X_2', ..., X_n')$ according to the following equation:

$$X_i' = F_i^{-1}(\Phi(Z_i)) \; i = 1...n, \tag{3.2}$$

   where Φ is the distribution function of a standard Gaussian random variable and F_i^{-1} is the inverse of F_i,defined as:

$$F_i^{-1}(u) = \inf\{x : F_i(x) \geq u\}. \tag{3.3}$$

3. it chooses $COV(\mathbf{Z})$ in order to induce the requested correlation matrix $CORR(\mathbf{X})$. In this case there is no closed-form solution and the method often relies on an efficient numerical search, by solving a number of one-dimensional root-finding problems. In some cases the procedure does not lead to the exact desired correlation matrix, failing to produce a positive semidefinite matrix, which is a requirement for a valid correlation matrix. However, NORTA can often get very close to the desired correlation matrix.

In the approach we propose, NORTA is used to generate the homogeneous system based on the R implementation known as NORTARA[3]. This package generates n-dimensional random vectors with given marginal distributions and correlation matrix. The NORTA algorithm, which generates a standard normal random vector and then transforms it into a random vector with specified marginal distributions, is combined with the RA (Retrospective Approximation) algorithm, which is a generic stochastic root-finding algorithm.

3.2.2 Experimental results

The effectiveness of the proposed methodology has been evaluated on three different systems whose dynamics are precisely known. In particular, we studied the consequences of using different homogeneous systems: the one produced with the method proposed in this work (where we "inject" the average correlation which characterizes the system under study - H_{wiC}, where *wiC* means "with correlation") and the one produced with the original method (H_{noC}). In the following, we focus on the application of the RI analysis, possibly applying the sieving algorithm in order to simplify the results. The binary nature of the variables of the test systems we used allows one to apply the H_{wiC} approach with simple Bernoulli distributions to all situations.

The three case studies taken into consideration, which have been presented in section 3.1, are representative of valuable research fields, that is, (i) the dynamics of Random Boolean Networks, (ii) dynamical simulations of autocatalytic reaction systems happening within a Continuous-flow Stirred-Tank Reactor (CSTR) and (iii) simplified models of the dynamics of opinion diffusion.

The first case study consists in the analysis of **Random Boolean Networks**, which have been introduced in section 3.1.1. In particular, here we present a collection of five different Boolean systems (denoted as RBN1, ..., RBN5) composed of 12 nodes, synchronously updated on the basis of either a Boolean function or a random Boolean value generator.
In each analysis, instead of juxtaposing different states belonging to the different at-

[3] `http://cran.r-project.org/web/packages/NORTARA/`

tractors of each system [240], we follow single trajectories, perturbed every 20 steps by temporarily changing a randomly chosen variable from 0 to 1 (or vice versa).

In the **CSTR** case study, which has been described in section 3.1.3, we tested a simple system featuring two distinct reaction pathways, a Linear reactions CHain (LCH) and an AutoCatalytic set of molecular Species (ACS). Both LCH and ACS pathways occur in an open CSTR with a constant influx of feed molecules and a continuous outgoing flux of all the molecular species proportional to their concentration. The asymptotic behavior of this kind of systems is a single fixed point [240], due to the system feedback structure. In order to apply our analysis, we need to observe the feedbacks dynamically; so, we perturb the concentration of some molecules in order to trigger a response in the concentration of other species. The behavior of the chemical concentrations within equally-sized time intervals is discretized using a two-level encoding: "chemical concentration changing" (corresponding to tag "1") and "no change in chemical concentration" (corresponding to tag "0").

Finally, in the third case study, we compare the results of the application of the RI to different homogeneous systems on a simple model, commonly used in opinion dynamics studies. Using this model, the integration among variables in a subsystem under observation and its mutual information with the remaining part of the system can be tuned by acting on few parameters. The model abstracts from specific functional relationships among elements of the system and could resemble a basic **Leader-Follower model** (LF), which has been described in section 3.1.2.

The model generates independent observations of the system state, based on the following rules:

- Variables are divided into three groups, G1 = $\{L_a, F1_a, F2_a, F3_a\}$, G2 = $\{L_b, F1_b, F2_b\}$, and G3 = $\{L_c, F1_c, \dots, F8_c\}$.
- L_a, L_b, L_c are the leaders of their groups[4], and they have a probability p_{lcpy} to copy the value of another leader, and a probability of 1-p_{lcpy} to independently assume a random value in {0,1} (with probability of obtaining a "1" equal to 0.4, 0.3, and 0.3, respectively).

[4] In details, $L_b(t) = L_a(t-1)$ and $L_c(t) = L_b(t-1)$.

- The values of the followers of the three groups are set as a copy (or negation) of their leaders with probability p_{copy} and randomly (according to a Bernoulli distribution with probability 0.5) otherwise.

- The three groups are submerged into a "sea" of random variables following a Bernoulli distribution with $P(x=0) = P(x=1) = 0.5$.

It is possible to tune the integration among elements within groups and the mutual information between groups by changing p_{lcpy} or p_{copy}. In our examples, we set for simplicity p_{lcpy}=0.0 (non-interacting groups) with p_{copy}=1.00 (perfect followers) and p_{copy}=0.98 (imperfect followers).

Results

In Figure 3.3, we report the relevant subsets identified by the RI analysis performed on RBNs using the H_{noC} or H_{wiC} homogeneous systems as a reference for the T_c computation. The RBN systems are relatively simple, and the most interesting relevant subsets are evident also without applying the sieving algorithm (see Table 3.1).

Figure 3.3 reports the two groups that rank highest according to the T_c value (the first four ranks for case RBN5).
Both approaches find the same solutions in cases RBN1, RBN2 and RBN3. In particular, the two methods directly identify the two correct solutions of RBN1, the two fundamental groups composing the correct solution of RBN2, and the correct solution of case RBN3. In RBN2, the simple iteration of the RI method after the application of the sieving algorithm is able to identify the correct big group (including variables C, D, E, H, I and L)[5]. In RBN3, the slightly preeminent position of the first triplet in Fig. 3.3 is due to the particular set of samples that has been chosen; indeed, by analyzing several sets of samples both triplets are equally represented.

The structure in case RBN4 is highly heterogeneous and comprises loosely integrated parts: the H_{noC} approach (though identifying correct nodes) is not able to spot out most variables composing the groups acting within the system, whereas the

[5] data not shown

Table 3.1: Table showing the relationships among nodes of the considered RBNs.

Node	**Node rule**				
	RBN 1	**RBN 2**	**RBN 3**	**RBN 4**	**RBN 5**
A	RND(0.5)	RND(0.5)	RND(0.5)	RND(0.5)	RND(0.5)
B	RND(0.5)	RND(0.5)	RND(0.5)	RND(0.5)	RND(0.5)
C	(D⊕E)	(D⊕E)	L^(D⊕E)	(D⊕E)	(D⊕E)
D	(C⊕E)	(C⊕E)	(C⊕E)	(C⊕E)	(C⊕E)
E	(C⊕D)	(C⊕D)	(C⊕D)	(C⊕D)	(C⊕D)
F	RND(0.5)	RND(0.5)	RND(0.5)	(E⊕H)	(E⊕H)
G	RND(0.5)	RND(0.5)	RND(0.5)	(G+H+I+L)≥2	RND(0.5)
H	(I⊕L)	E^(I⊕L)	E^(I⊕L)	(C⊕L)	(I⊕L)
I	(H⊕L)	(H⊕L)	(H⊕L)	(D+E+G+H)≥2	(H⊕L)
L	(H⊕I)	(H⊕I)	(H⊕I)	F⊕(E⊕I)	(E⊕I)
M	RND(0.5)	RND(0.5)	RND(0.5)	RND(0.5)	RND(0.5)
N	RND(0.5)	RND(0.5)	RND(0.5)	RND(0.5)	RND(0.5)

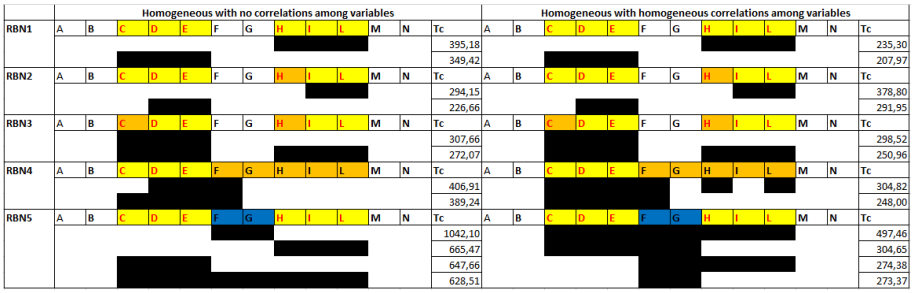


Figure 3.3: The first two candidate relevant sets for each RBN case (four candidate relevant sets in the RBN5 case) and their T_c values, computed by using the "classical" homogeneous system (H_{noC}, left) and by using the homogeneous system built using NORTA (H_{wiC}, right). In each row, a black cell indicates that the corresponding variable is selected in the candidate relevant set, whereas white cells denote the variables not belonging to the candidate relevant set. For each RBN case, we also report the correct solution, in which the different colors denote particular nodes or subdivisions of the dynamical structure of the systems. In particular: (i) case RBN1 hosts two dynamically-independent structures, (ii-iii) which in cases RBN2 and RBN3 are linked through the orange nodes; (iv) in case RBN4, a structure, observable also in case RBN1, is providing signals to other 5 nodes (highlighted in orange, and in turn exchanging messages among each other in various ways); (v) in case RBN5, the two structures, present also in case RBN1, are both sending signals to the blue nodes F and G. For a more detailed description see Table 3.1.

H_{wiC} approach identifies almost all the correct nodes. The variables not detected are just nodes G and I, which indeed have a very low coupling with the other variables (see Table 3.1 for details): so, the H_{wiC} approach seems to be more accurate than the H_{noC} approach. In other words, the T_c rankings obtained by the two approaches are different, but often the H_{wiC} approach identifies larger groups, which are also the correct ones. In case RBN5, for example, the most relevant group is composed of eight nodes and it is immediately identified by the H_{wiC} approach, whereas the H_{noC} approach identifies in the first positions only the small subsets composing the largest

group of variables.

This hypothesis is supported by the analysis of the CSTR case: the H_{noC} approach is able to identify merely small subsets of the ACS system, whereas the H_{wiC} approach directly identifies in its first iteration almost all the members of the ACS. At the same time, this approach identifies also the largest part of the LCH structure. In this case, we repeated the RI analysis several times, by using different H_{noC} and H_{wiC} homogeneous systems: all these analyses consistently confirmed these results. Figure 3.4, which, for simplicity, shows only the relevant sets selected by the application of the sieving algorithm (two sets using H_{noC} and two sets using H_{wiC}) which have a much higher value than the other possible sets, strongly supports these observations.

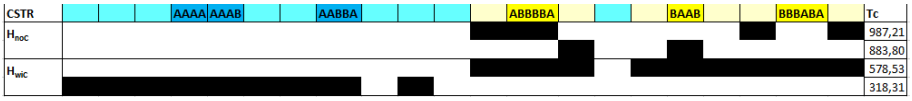


Figure 3.4: The two candidate relevant sets - obtained by applying the sieving algorithm to the results of the RI analysis - of the CSTR case and their T_c values, computed using the "classical" homogeneous system (H_{noC}, first two rows) and using the homogeneous system built with NORTA (H_{wiC}, last two rows). The groups of variables remaining after the application of the sieving algorithm have T_c values by far lower than those shown here. The blue and yellow colors indicate the chemical species belonging to LCH and ACS structures, respectively; darker colors indicate the chemical species produced by the reactions happening within the CSTR reactor (for these species the names are also reported). The constant species are not included in the table reported in this figure; the colored nodes without name indicate substrates or intermediate complexes.

The LF scenario described in the previous part of the section is a particularly difficult case for the RI analysis. Indeed, the addition to a group of size N_v of a variable, which is an almost perfect function of a variable already present within the group itself, leads to a new group of size $N_v + 1$ with a normalized integration very similar to the normalized integration of the initial group. Indeed, the integration of the

group of size N_v subtracted from the integration of the group of size $N_v + 1$ is equal to the entropy of the added variable: the same holds for the homogeneous system if the difference between the average integrations of groups of size $N_v + 1$ and of size N_v is taken into consideration.

In the case of p_{copy}=1 (perfect followers) the H_{noC} approach ranks in the top 130 positions almost all subsets of group G3 (of sizes 7, 6 and 8, in frequency order)[6], before identifying the correct G3 group, whereas the H_{wiC} approach identifies the correct G3 group (with T_c= 522.803) immediately after its 8 subsets composed of 8 variables (with T_c values slightly lower than 524). Figure 3.5 shows the superposition of these subsets, which highlights the presence of the G3 group, and the corresponding T_c values range: indeed, the H_{wiC} approach is able to discriminate among all the possibilities in a sophisticated way, thereby effectively identifying the correct G3 position.

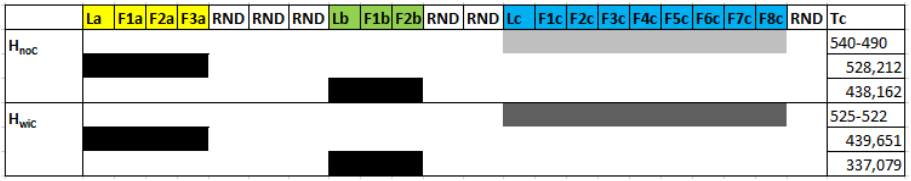


Figure 3.5: The candidate relevant sets identified by using the "classical" homogeneous system (H_{noC}) and by using the homogeneous system built with NORTA (H_{wiC}), related with the Leader-Followers case (different colors highlight different LF groups). The variable sets reported in the top line (in light grey for H_{noC} and in dark grey for H_{wiC}) actually represent the top-ranked 130 (H_{noC}) and 8 (H_{wiC}) sets, all subsets of the same variables, that were detected; in this case the T_c column shows the range of the subsets' T_c values. Notice that the T_c range identified by the H_wiC approach is significantly smaller than the range identified by the H_{noC} approach.

Eventually, both systems correctly identify the G1 and G2 groups. Similar results

[6]Notice that, in case of a perfect copy, the action of excluding a particular variable and including another one leads to groups having the same T_c value.

hold for p_{copy}=0.98 (data not shown).

3.2.3 Final remarks

In this section, we have proposed an improvement to the RI method for identifying relevant subsets in complex systems. In particular, we have introduced a constant nonzero degree of statistical dependence in the variables composing the homogeneous reference system, by imposing that all variable pairs share the same pairwise correlation. The results coming from three relevant case studies demonstrate that this improvement allows one to identify sets of interacting variables of larger size compared with the previous way of generating the homogeneous system.

The main drawback of the proposed methodology is that there exist marginal distributions and correlation matrices that the NORTA method cannot match, even though a random vector with the prescribed qualities exists. In particular, in [86] it is stated that the NORTA method becomes more and more ineffective as the dimension of the random vector increases. Therefore, Nortara becomes increasingly resource-demanding, in terms of CPU time and occupied memory, as the dimension of the random vector increases.

In conclusion, the proposed approach has shown improved results when applied to systems with small dimensions, but it cannot be applied to large-size systems in a reasonable time. The following section presents another RI-based metric that overcomes the inefficiency due to the homogeneous system computation.

3.3 The *zI* metric

The *RI* and T_c indices can be used to analyze complex systems by detecting the main interacting structures within them. However, they suffer from some limitations:

- The division by mutual information (MI) is a problematic aspect, since the mutual information is zero when the two groups of RVs are independent. Problems can derive also from values of MI close to zero. Low MI values can derive from weak interactions between the subgroup and every element of the rest of the

system, or from strong interactions between the subgroup and a small part of the rest of the system, while the other parts are not involved in the interactions. RI and T_c do not distinguish between the two situations.

- It is not easy to compute the averages and standard deviations of integration and mutual information for a suitable homogeneous system. Typically, one must refer to the use of very onerous computational techniques to "simulate" the homogeneous system distribution and to compute the statistics for each subset dimension.

These limitations can be addressed by using the integration alone through its z-score, zI in the following. Moreover, an interesting observation allows us to foresee a new approach that does not require any simulation of homogeneous systems: the integration, multiplied by twice the number of observations, approximately follows a chi-squared distribution with degrees of freedom depending on the number of variables belonging to the analyzed subgroup and on the cardinality of their alphabets. Such an approximation can be used to obtain approximate z-scores with negligible computational effort.

In this section we provide an overview of the theoretical aspects of the proposed method, describing the following points:

- the Integration in the context of the Relevance Index metrics;
- the derivation of the zI metric;
- the experimental results we obtained.

3.3.1 Integration as a Relevance Index metric

Let us consider a system U, composed of N random variables (RVs), that at each epoch takes values from a finite alphabet. The system can be described by its state $\bar{X}_U = [X_1, ..., X_N]$, i.e., a random vector whose elements are the N random variables. After a relaxation time, when the system is at quasi-equilibrium, the state of the system $\bar{X}_U$ takes values from a finite set, the set of attractors. Given this scenario,

we consider n independent observations of the state $\{\bar{X}_U(i)\}_{i=1}^{n}$. Therefore, for each RV X_j composing the state vector we have a sequence $\{X_j(1),..,X_j(n)\}$ of independent random variables, identically distributed (*iid*). If we consider n to be sufficiently large, ideally tending to infinite, we have, from the Asymptotic Equipartition Property (AEP) [47], that the empirical entropy of a sequence of *iid* RVs converges in probability (ip) to the real individual entropy:

$$-\frac{1}{n}\sum_{i=1}^{n}\log p(x_j(i)) \xrightarrow{ip} -\sum_{x_j\in T_j} p(x_j)\log p(x_j) \tag{3.4}$$

being T_j the alphabet in which X_j takes values and $p(x_j) = P\{X_j = x_j\}$.

We have dealt with entropies because they are involved in the definition of the integration (I), which has been presented in section 2.1.1 (equation 2.4). Integration measures the mutual dependence among the k elements in S_k. It is defined as the difference between the summation of the entropies of the single variables composing subset S_k and the total entropy of subset S_k itself:

$$I(S_k) = \sum_{j=1}^{k} H(X_j) - H(X^k) \tag{3.5}$$

where $H(X^k)$ is the entropy of the sequence $\{X_1,....,X_k\}$. It can be seen that $I(S_k) \geq 0$ since $H(X^k) = \sum_{j=1}^{k} H(X_j|X^{j-1}) \leq \sum_{j=1}^{k} H(X_j)$ because of the chain rule and the property that conditioning reduces entropy [47]. The integration is zero when the RVs are independent, such that $H(X^k) = \sum_{j=1}^{k} H(X_j|X^{j-1}) = \sum_{j=1}^{k} H(X_j)$. On the other hand, the integration increases with the decrease of the second term, i.e. with the correlations among the random variables. One can notice that the definition of integration in Eq. 3.5 can be rewritten in terms of Kullback-Leibler distance [47] as follows:

$$\begin{aligned} I(X_1,..,X_k) &= D(p(X_1,..,X_k)||p(X_1)...p(X_k)) \\ &= \sum_{x_1\in T_1} .. \sum_{x_k\in T_k} p(x_1,..,x_k)\log\frac{p(x_1,..,x_k)}{p(x_1)...p(x_k)} \end{aligned} \tag{3.6}$$

where $p(X_1,...,X_k)$ is the joint PMF of the sequence of RVs and $p(X_1)..p(X_k)$ is the product of the marginal PMFs. It is important to recall that the relationship

$p(X_1,...,X_k) = p(X_1)...p(X_k)$ holds only if the RVs are independent. The equivalence between Eq. 3.5 and Eq. 3.6 is proven as follows:

$$\begin{aligned}
&\sum_{x_1\in T_1} .. \sum_{x_k\in T_k} p(x_1,..,x_k)\log\frac{p(x_1,..,x_k)}{p(x_1)...p(x_k)} = \\
&\sum_{x_1\in T_1} .. \sum_{x_k\in T_k} p(x_1,..,x_k)\log p(x_1,..,x_k) + \\
&-\sum_{x_1\in T_1} .. \sum_{x_k\in T_k} p(x_1,..,x_k)\log p(x_1)...p(x_k) = \\
&-H(X_1,...,X_k) - \sum_{x_1\in T_1} .. \sum_{x_k\in T_k} p(x_1,..,x_k)\sum_{j=1}^{k}\log p(x_j) = \\
&-H(X_1,...,X_k) + \sum_{j=1}^{k} H(X_j)
\end{aligned}$$

in which the last step holds by considering, for instance, the $j = 1$ term and noticing that the second part of the expression can be written as

$$-\sum_{x_1\in T_1}\log p(x_1)\sum_{x_2\in T_2} .. \sum_{x_k\in T_k} p(x_1,..,x_k) = \tag{3.7}$$

$$-\sum_{x_1\in T_1} p(x_1)\log p(x_1) = \tag{3.8}$$

$$H(X_1) \tag{3.9}$$

where the equivalence between Eq. 3.7 and Eq. 3.8 holds for marginalization [169] of $p(x_1,..,x_k)$. Thus, the integration can be seen as a measure of deviation from statistical independence, being expressed as the divergence between a joint PMF and the product of its marginals.

It is also important to highlight the relationship between the empirical entropy and its theoretical counterpart. It has been known since Miller & Madow [156] that, on average, the empirical entropy $\hat{H}_n(S_k)$ underestimates its theoretical counterpart $H(S_k)$:

$$\mathbb{E}[\hat{H}_n(S_k)] \quad = \quad H(S_k) - \frac{c(S_k)-1}{2n} + O\left(\frac{1}{n^2}\right), \tag{3.10}$$

where $c(S) = \prod_{j=1}^{k} |T_j| > 1$, $|T_j|$ denotes the cardinality of the alphabet T_j within which X_j takes values and it is understood that $p_{S_k}(x_1, \ldots, x_k) > 0$ for all $x_1 \in T_1, \ldots x_k \in T_k$. It follows that the empirical integration overestimates its theoretical counterpart:

$$\mathbb{E}[\hat{I}_n(S_k)] \quad = \quad I(S_k) + \frac{d_k}{2n} + O\left(\frac{1}{n^2}\right), \tag{3.11}$$

where

$$d_k \quad = \quad c(S_k) - 1 + k - \sum_{j=1}^{k} |T_j| \tag{3.12}$$

is a strictly positive integer. In the special case $k = 2$, first studied by Miller [155], the quantity in (3.12) can be written as $d_2 = (|T_1| - 1)(|T_2| - 1)$; see Luce [143].

As shown in section 2.1.2, in the context of the RI metrics, the value of the integration of a subset S_k is usually divided by the mutual information, which is a well known metric that measures the mutual dependence between S_k and the remaining part of the system $U \setminus S_k$. However, in many cases, the sole integration seems to provide by itself the majority of the relevant information regarding the tightness of subset variables. Moreover, the approach based on the division by mutual information has some limitations. First of all, the ratio itself could be sometimes problematic, since it could tend theoretically to infinite. Secondly, the z-score form of the T_c index, introduced to allow a fair comparison between the *RI* of subsets of different size (section 2.1.2, Eq. 2.9), involves the creation of the homogeneous system and the computation of its statistics. This direct computation can be avoided if we consider a metric for which the distribution under the null hypothesis is known, such that the z-score can be built without the explicit generation of the null system. In the following subsection, the derivation of this new integration-based index, called *zI* since it is a sort of z-score, is presented, starting from theoretical reasoning on the distributions.

3.3.2 Derivation of the *zI* metric

The *zI* metric is defined as follows, where n is the number of observations or instances of the whole system, S_k a subset of k out of N variables and S_{NULL} is a subset of

dimension k extracted from a homogeneous system U_h (i.e., a system with the same number of variables of the analyzed system and no dynamical organization, with all N variables being independent):

$$zI(S_k) = \frac{2nI(S_k) - \langle 2nI(S_{NULL})\rangle}{\sigma(2nI(S_{NULL}))} \tag{3.13}$$

In other words, we propose:

- to use only the integration, which, by itself, describes the tight interaction between variables belonging to the same subset;
- to adopt a normalization procedure by considering the average value and the standard deviation of the metric, whose computation can be accelerated thanks to the theoretical hints described in the following.

The derivation of the zI can be described as follows.

Considering the state vector $\bar{X}_U$ as described previously, we take n independent observations of the state vector $\{\bar{X}_U(i)\}_{i=1}^n$ with joint distribution

$$P\{\bar{X}_U(1) = \bar{x}(1), ..., \bar{X}_U(n) = \bar{x}(n)\} = \prod_{i=1}^{n} \pi_U(\bar{x}(i)) \tag{3.14}$$

where $\pi_U(\bar{x}(i)) = P\{\bar{X_U}(i) = \bar{x}(i)\}$.

Considering now the null hypothesis of independence of the N RVs, the PMF of $\bar{X}_U$ can be written as $P\{\bar{X}_U = \bar{x}\} = P\{X_1 = x_1\}P\{X_2 = x_2\}...P\{X_N = x_N\} = \prod_{j=1}^{N} \pi_j(x_j)$ Therefore, if we consider n independent observations of $\bar{X}_U$ we have that Eq. 3.14 ca be re-written as:

$$P\{\bar{X}_U(1) = \bar{x}(1)\}..P\{\bar{X}_U(n) = \bar{x}(n)\} = \prod_{i=1}^{n}\prod_{j=1}^{N} \pi_j(x_j(i)) \tag{3.15}$$

where $\pi_j(x_j(i)) = P\{X_j(i) = x_j(i)\}$.

An estimator of the unknown distribution (non-parametric likelihood estimator) can be found by choosing the distribution that maximizes the likelihood function given $\bar{x}(1), .., \bar{x}(n)$, formally

$$\tilde{\pi}_U = argmax \prod_{i=1}^{n}\prod_{j=1}^{N} \pi_j(x_j(i)) \qquad \pi_U \in SI_U \tag{3.16}$$

in which SI_U is the simplex. Equivalently, we can maximize the average log-likelihood function

$$L_n(\pi_1,....,\pi_N) = \sum_{j=1}^{N} \frac{1}{n} \sum_{i=1}^{n} \log \pi_j(x_j(i)) = \sum_{j=1}^{N} L_n(\pi_j) \tag{3.17}$$

that is the sum of the marginal log-likelihoods. Each marginal average log-likelihood can be rewritten as

$$L_n(\pi_j) = \sum_{x_m \in T_j} f_j^n(x_m) \log \pi_j(x_m) \tag{3.18}$$

in which T_j is the alphabet within which X_j takes values and $f_j^n(x_m) = \frac{1}{n}\sum_{i=1}^{n} \delta(x_j(i) - x_m)$ is the marginal relative frequency of x_m, a value in the alphabet of X_j, over the n observations. Since we are under the assumption of independence, we can split this optimization problem in N independent problems. The optimal solution of each problem can be found with the Lagrange multipliers, considering as the Lagrangian function $F(\pi_j(x_m), \lambda) = \sum_{x_m \in T_j} f_j^n(x_m) \log \pi_j(x_m) - \lambda \sum_{x_m \in T_j} \pi_j(x_m)$, being $\sum_{x_m \in T_j} \pi_j(x_m) = 1$ the constraint. Omitting some steps, we find that the marginal distribution that maximizes the marginal average log-likelihood is

$$\tilde{\pi}_j(x) = f_j^n(x) \qquad x \in T_j \tag{3.19}$$

for each RV X_j in the state random vector $\bar{X}_U$. Thanks to the independence assumption, we can conclude that the optimal joint distribution is

$$\tilde{\pi}_U(\bar{x}) = \prod_{j=1}^{N} \tilde{\pi}_j(x) \qquad x \in T_j \tag{3.20}$$

Thus, the average log-likelihood function at the optimal solution is

$$\begin{aligned} L_n(\tilde{\pi}_U) &= \sum_{j=1}^{N} L_n(\tilde{\pi}_j) \\ &= \sum_{j=1}^{N} \sum_{x_m \in T_j} f_j^n(x_m) \log f_j^n(x_m) \\ &= -\sum_{j=1}^{N} H(X_j) \end{aligned} \tag{3.21}$$

If we had no information about the relationships among the N RVs the general expression of Eq. 3.21 would be

$$L_n(\hat{\pi}_U) = \sum_{j=1}^{|T_U|} f_U^n(\bar{x}_j) \log f_U^n(\bar{x}_j) = -H(X_U) \tag{3.22}$$

Considering now the application of the definition of Integration in Eq. 3.5 to the set U composed of all the N RVs in the system, we get that

$$I(U) = \sum_{i=1}^{N} H(X_i) - H(X^N) \tag{3.23}$$

and comparing this expression with equations 3.22 and 3.21 it can be seen that

$$I(U) = L_n(\hat{\pi}_U) - L_n(\tilde{\pi}_U) \tag{3.24}$$

in which $\tilde{\pi}_U$ is the distribution of the state vector that maximizes the average log-likelihood under the null hypothesis and $\hat{\pi}_U$ the optimal one without independence assumption.

Considering the log-likelihoods instead of average log-likelihoods and making some manipulations on Eq. 3.24 we have that

$$nI(U) = -n(L_n(\tilde{\pi}_U) - L_n(\hat{\pi}_U)) \tag{3.25}$$

where the term in parentheses represents the difference between the log-likelihood under the null hypothesis and the log-likelihood under the so-called alternative hypothesis.

According to Wilks' theorem [247], it is known that the quantity defined as

$$D = -2log\left(\frac{likelihood\,under\,null\,hypothesis}{likelihood\,under\,alternative\,hypothesis}\right) \tag{3.26}$$

for $n \to +\infty$ is chi-squared [169] distributed with degrees of freedom $d = DoF_{alternative} - DoF_{null}$, when the null hypothesis holds. Namely

$$D \approx \chi_d^2 \qquad n \to +\infty, H = H_o \tag{3.27}$$

where H_o represents the null hypothesis. According to [168], this result can be extended to nonparametric likelihood functions, which is our case of interest.

The comparison between Eq. 3.24 and Eq. 3.26 leads to the conclusion that $2nI = D$, with the following implication

$$2nI \approx \chi^2_d \quad n \to +\infty \tag{3.28}$$

under the null hypothesis of independent RVs.
Under the null hypothesis it is true that $P\{\bar{X}_U = \bar{x}\} = \prod_{j=1}^{N} P\{X_j = x_j\}$ and each X_j is subject to the constraint $\sum_{x_m \in T_j} P\{X_j = x_m\} = 1$. Therefore the PMF of each RV has $|T_j|$-1 degrees of freedom, where $|T_j|$ is the cardinality of the alphabet of X_j. The degrees of freedom under the null hypothesis are therefore the sum of the degrees of freedom for each independent RV. Namely $DoF_{null} = \sum_{j=1}^{N}(|T_j| - 1)$.
In the same way, under the alternative hypothesis (without independence) the PMF of the state vector is $P\{\bar{X}_U = \bar{x}\} = P\{X_1 = x_1, ..., X_N = x_N\}$ subject to the constraint $\sum_{\bar{x} \in T_U} P\{X_U = \bar{x}\} = 1$. Therefore the degrees of freedom are $|T_U| - 1$ with $|T_U| = \prod_{j=1}^{N} |T_j|$. Thus $DoF_{alt} = \prod_{j=1}^{N} |T_j| - 1$. Combining these considerations we have an expression for the degrees of freedom for equation 3.28 as follows

$$d = \prod_{j=1}^{N} |T_j| - 1 - \sum_{j=1}^{N} (|T_j| - 1) \tag{3.29}$$

It is known [169] that the mean and the standard deviation of the chi-squared distribution with g degrees of freedom are

$$\mu = d \tag{3.30}$$

$$\sigma = \sqrt{2d} \tag{3.31}$$

In summary, we have found a quantity $2nI$, for which we know the distribution and the statistics when the null hypothesis of independence holds. As a consequence, only the number of degrees of freedom $d(S_k) = d_k$ needs to be computed, which is straightforward knowing the following formulas:

$$d_k = dim(T_S) - \sum_{j=1}^{k} dim(T_j) \tag{3.32}$$

$$dim(T_S) = \prod_{j=1}^{k} L_j \quad - 1 \tag{3.33}$$

$$dim(T_j) = L_j - 1 \tag{3.34}$$

and where L_j is the cardinality of the alphabet of the j^{th} variable among the k.

Exploiting these results, we can build a new index in the same z-score equation according to the following formula:

$$zI(S_k) = \frac{2nI(S_k) - d_k}{\sqrt{2gd_k}} \tag{3.35}$$

where S_k is a subset of k elements out of N.

Eq. 3.35 is the same as Eq. 3.13 but with all elements known and with no need to compute the statistics of the null system.

The zI metric allows one to overcome the inefficiency caused by the generation of the homogeneous system, since its statistics can be obtained directly from the degrees of freedom, which depend only on the size of the analyzed subset and on the cardinality of the alphabet of its variables.

3.3.3 Experimental evaluation

The effectiveness of the zI metric for complex systems analysis has been assessed from both a simulation and an empirical perspective. The following subsections analyze the performance of the zI for a dynamically homogeneous system and a dynamically organized system, respectively.

Dynamically homogeneous systems

In order to validate our proposed approach, we carried out the analysis of several homogeneous systems. In the following, we show the results of the analysis of a set of observations, having different lengths, extracted from a homogeneous system composed of 21 binary variables whose two symbols have equal probability of appearing.

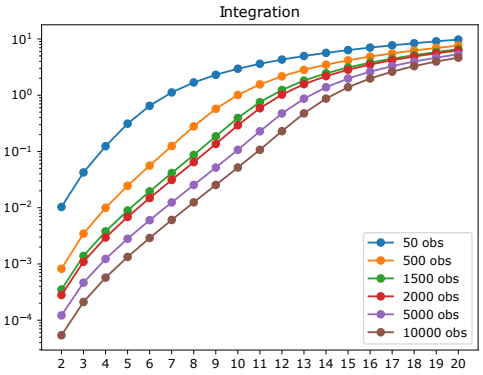
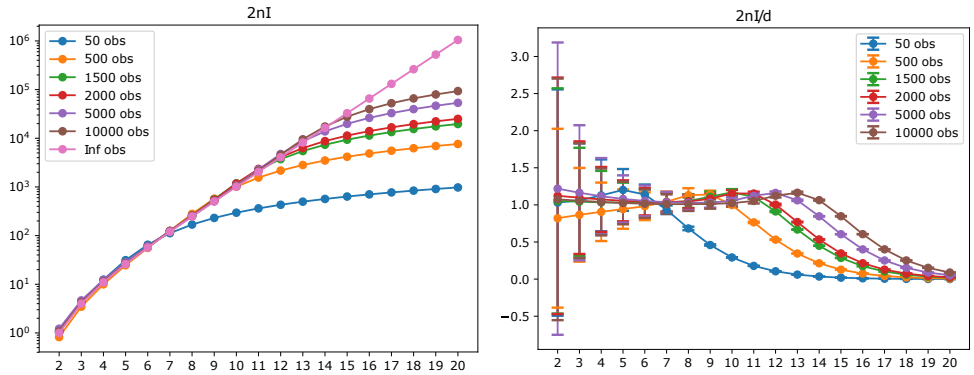


Figure 3.6: Homogeneous system. Average integration by analyzed group size, with varying number of observations ($n = 50$, 500, 1500, 2000, 5000 and 10000).

Figure 3.6 shows the average integration by group size for sets of observations having different lengths. To consider all possible groups, the integration values have been computed in parallel using a CUDA GPU implementation, which will be described in details in section 3.4. We can observe that longer sets of observations, as well as larger groups, lead to higher average integration, which reflects the bias term $d_k/2n$ in equation (3.11). Indeed, as illustrated in Fig. 3.7a, the average values of $2nI$ do not depend on n for small group sizes, which shows that, in such cases, the chosen number of observations is sufficient to provide a reliable estimate of integration. Furthermore, it can be seen in Fig. 3.7b that the average values of $2nI/d$ (where with d we briefly indicate the number of degrees of freedom d_k) are close to 1.0 for small groups sizes, which confirms the reliability of such estimates. Note that the maximum group size such that the estimated integration is reliable grows with the number of observations, from $k = 7$ with $n = 50$ observations to $k = 14$ with $n = 10000$ observations, which is consistent with the fact d can be seen as the average of $2nI$ with infinite observations. Note, as well, that the width of the error bars in Fig. 3.7b is larger for small group sizes, where there is little overlap between distinct groups, while it drops monotonically as the number of observations increases.

Figure 3.7b shows that the average of $2nI/d$ quickly drops below the value 1.0 for



(a) (b)

Figure 3.7: Homogeneous system. (a) Average of $2nI$, by group size k, for six different trajectory lengths, compared with the mean d of the chi-squared distribution (average with infinite observations). (b) Same as (a) for $2nI/d$, with error bars showing the standard deviation of measurements.

large groups. This fact indicates that, in the examined sets of observations, the values of $2nI/d$ for these groups are smaller than they should be: large size groups are not very evident, or, in other words, they are difficult to detect. Indeed, the same limit would hold for groups formed by few variables having a large number of symbols. Actually, systems with a number of degrees of freedom greater than the number of observations necessarily cannot be fully observed, which results in a fundamental limit regarding the reliability of their identification. This problem is mainly related with the number of performed observations rather than with the method used to detect the groups.

In the following, we study the performance of four candidate relevance indices: the T_c index, the zI metric, one minus the chi-squared p-value of $2nI$, denoted by $ChSq$, and the simple "normalized" index $2nI/d$.

Figure 3.8a shows the maximum values of the four indices for each group size, sorted by group size. Basically, in a homogeneous system, there should not be any relevant subset, although some will spuriously show up due to the finiteness of the

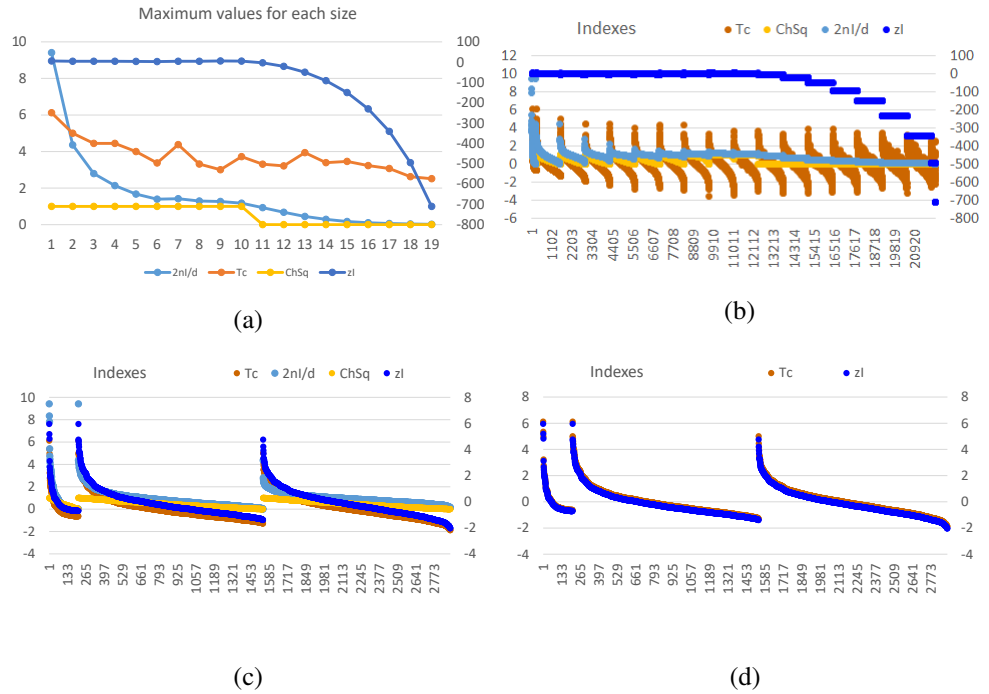


Figure 3.8: Homogeneous system: values of the four candidate relevance indices (T_c, zI, $ChSq$, $2nI/d$); the vertical axis for zI is the one on the right. (a) Maximum values for each size, sorted by group size. (b) Values for each single group, sorted by group size and then by zI. (c) An enlargement of the plot in (b), limited to groups of up to four variables. (d) The same as in (c), limited to zI and T_c; the two vertical axes here have the same scale.

number of observations. Indeed, the values of the indices are all low (with the notable exception of $ChSq$) and they tend to diminish further as the size of the groups taken into consideration increases. It can be seen from figure 3.8b that each group size presents many extreme values, a fact that indicates the presence of groups that show (because of random factors) some coordination activity. Note however that the zI metric dramatically decreases in correspondence of large groups sizes, correctly indicating an increasingly low evidence associated to these groups.

Fig. 3.8 suggests other remarkable considerations:

1. within each group size, all indices provide the same ranking, as shown for instance in Fig 3.8c (magnification on the smallest size groups), where the values are sorted by *z*I, but the other indices would give the same result;

2. *zI* and T_c (at least on small size groups) seem to provide exactly the same kind of information (Fig. 3.8d);

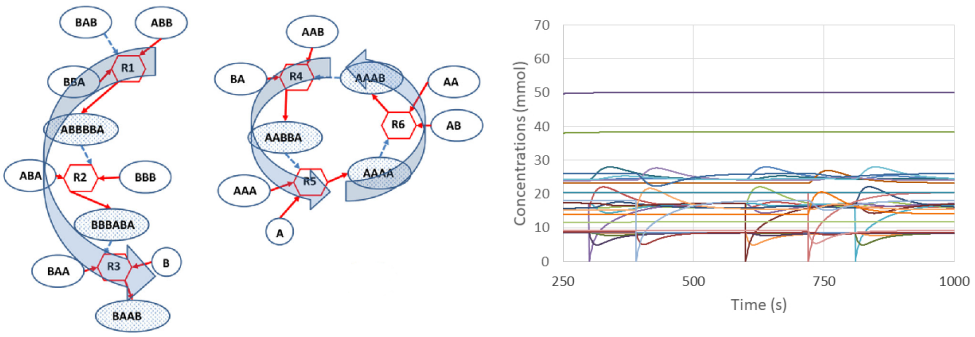
3. within the same group size, the *ChSq* alone seems to have a fairly linear trend (Fig. 3.8c).

As we shall see in the next section, point (2) turns out to be (almost always) correct only in the case of homogeneous systems, while point (3) is counterproductive in case we are interested in searching for the best groups (indeed, our main aim).

A dynamically organized system

In the previous case studies, we used the homogeneous system in order to verify the effectiveness of the chi-squared distribution and to make some general remarks on the application of different indices for the evaluation of the degree of dynamic organization of subgroups of variables. Our purpose, however, is to look for groups that are dynamically relevant (the RSs), whose identification can facilitate the understanding of the dynamic organization of the analyzed system. To this aim, the homogeneous system is used as a reference system, while we are interested in analyzing dynamically non-homogeneous systems.

In this section, we analyze a dynamically organized system, regarding the identification of autocatalytic sets of reactions (CSTR), presented in section 3.1.3. The system, described in figure 3.9, is composed of 19 variables representing different chemical species (reaction products and reagents), in which there are two distinct reaction pathways: a linear chain and an autocatalytic circle. The state of the system variables is boolean, representing the dynamical behavior of the chemical concentrations within equally-sized time intervals after a perturbation of the system (“1”: “chemical concentration changing”, “0”:“no change in chemical concentration”).



(a) (b)

Figure 3.9: CSTR system: (a) The chemical system under analysis. Circular nodes depict chemical species, the blue ones stand for those injected on the CSTR (food species) and the green ones represent the more complex species built by specific concatenations of the food species. Diamond shapes represent reactions where incoming arrows go from substrates to reactions and outgoing arrows go from reactions to products. Dashed lines indicate the catalytic role of a particular molecular species within the specific reaction context. (b) The system behavior within the time interval under analysis. The kinetic constants of all present reactions have the same value $k = 0.0025 s^{-1} mol^{-1}$; the incoming concentration of each food species is $0.001M$, whereas each second the 2% of the CSTR volume is renewed.

Let us analyze the experimental results. The schema of figure 3.10 is very similar to that of figure 3.8, but the results are quite different. Indeed, figure 3.10a confirms the tendency of the maximum index values to decrease as the size of the groups taken into consideration increases, with the remarkable exception of index *ChSq*, which is constant with respect to the group size. Figure 3.10b, however, clearly indicates the presence of a significant dynamical organization in relatively small groups, while all indices in large groups have very low values. An enlargement focused on small groups (figure 3.10c) reveals that, if *ChSq*, zI and $2nI/d$ again show similar rankings, the T_c index presents strong oscillations.

This problematic aspect of T_c index derives from the structure of the Relevance Index, which consists in the ratio between the integration inside the subgroup under examination and the mutual information between the subgroup and the rest of the system. Analyzing systems characterized by a strong dynamical organization, low values of mutual information can derive both from a low information exchange between the subgroup and every element of the rest of the system and from a high exchange of information between the subgroup and a small part of the rest of the system, while the other parts are not involved in the exchange. The latter situation occurs when a subgroup of size k is actually part of slightly larger subgroups, a condition not present in homogeneous systems. The simple quotient between integration and mutual information does not allow for an easy discrimination between the two cases. For example, in Fig. 3.9, the pair of variables BBB and ABA is in strong association (but with different intensity) with the variation of concentrations of ABBBA and BBBABA, but even of BAA or of BAAB.

For these reasons, we focused our analysis on zI, $2nI/d$ and *ChSq*. Figure 3.10d emphasizes (on groups of size 2) a situation that occurs within any group size: *ChSq* cannot discriminate among the "best" groups (a situation indeed already present in figure 3.8a, at least for the smaller groups). Indeed, the p-values associated to values that are hundreds of standard deviations away from the average are all very low, or, in other words, the presence of these groups is not possible under the null hypothesis of no dynamical organization within the analyzed system. On the other hand, the $2nI/d$ index does not give us clues about the significance of the measures. Therefore, we

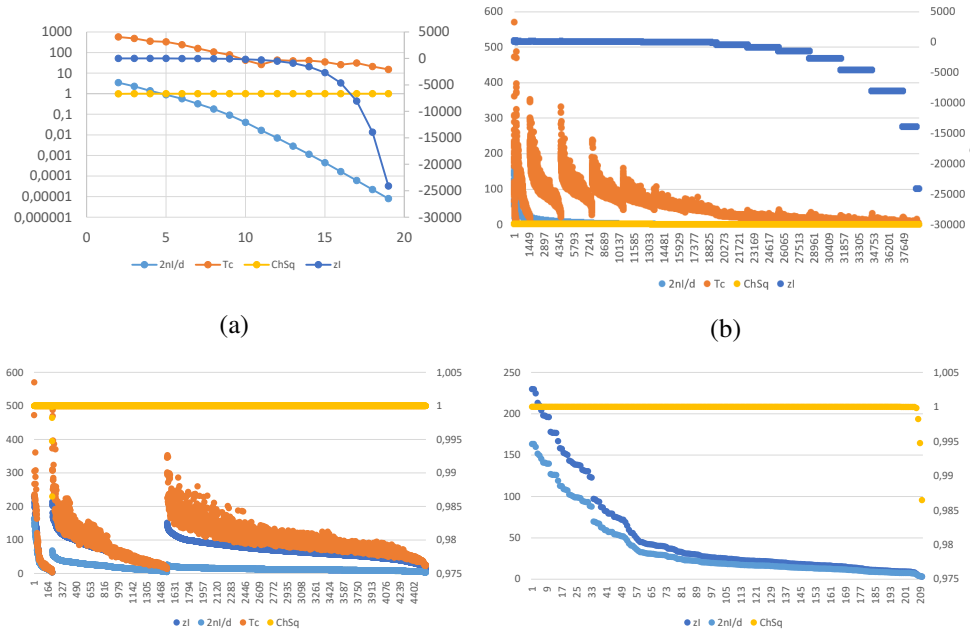


Figure 3.10: CSTR system: values of the four candidate relevance indices (T_c, zI, $ChSq$, $2nI/d$). (a) Maximum values for each size, sorted by group size; the vertical axis for zI is the one on the right. (b) Values for each single group, sorted by group size and then by zI; the vertical axis for zI is the one on the right. (c) An enlargement of the plot in (b), limited to groups of up to four variables; the vertical axis for $ChSq$ is the one on the right. (d) An enlargement of the plot in (c), limited to zI, $2nI/d$ and $ChSq$; the vertical axis for $ChSq$ is the one on the right.

propose the use of zI, which has a clear statistical meaning and, at the same time, is able to enhance the differences between the best groups (figure 3.10d).

3.3.4 Final remarks

In this work we have evaluated the effectiveness of a set of indices useful for identifying dynamically relevant variable subsets in complex systems. Within this set we propose the zI metric as a new RI metric. Its theoretical distribution allows one to avoid the excessively onerous bootstrap calculation of the homogeneous system, necessary to compare groups of different size. Moreover, we have shown that an increase in the number of observations leads to the identification of larger groups, up to the asymptotic complete observability in the case of infinite observations.

Some useful applications of the zI metric to real-world systems will be presented in chapters 6 and 7.

The analyzed metrics provide a functional description of the subsets of system variables, but they cannot describe hierarchical structures. Chapter 5 will present a new algorithm based on the RI metrics to derive a hierarchy of RSs by iteratively grouping one or more RSs into a single entity. The proposed methodology is composed of iterative runs of the RI analysis on the same data, each time using a new representation. This thesis proposes also some efficient strategies to perform this onerous iterative analysis. In particular, the following section is dedicated to the optimization of the RI metrics computation, while chapter 4 will present some niching metaheuristics for the efficient detection of RSs in large size systems.

3.4 GPU-based parallel search

To fully describe a complex system by means of an RI metric it would be necessary to compute such an index for all possible subsets of the system variables. Unfortunately, the number of subsets increases exponentially with the number of variables, soon reaching unrealistic requirements for computation resources. In practice, this procedure is feasible only for small-size systems in a reasonable amount of time. Therefore, an efficient implementation of the RI metrics is definitely necessary.

Considering that the computation of the RI metrics for each Candidate Relevant Set (CRS) can be performed independently of the others, using GPU-based parallel code seems to be the most efficient way of computing this kind of metrics.

Since the T_c index is the most computationally expensive RI metric, we propose a parallel implementation for the computation of such an index. The other RI metrics described in sections 2.1.2 and 3.3 can be easily obtained from the main building blocks of the parallel implementation of the T_c index computation.

This section presents a set of CUDA C[7] kernels [1] that provide a fine-grained parallel implementation of the main building blocks needed to compute the T_c index, upon which smart and efficient search algorithms can be designed.

The parallel functions have been developed to accomplish three different goals:

1. Speeding up an exhaustive sequential search by computing the T_c values of several candidate RSs in parallel;

2. Providing a computationally-efficient objective function for a metaheuristic that searches for the RSs of large dynamical systems for which an exhaustive search is impractical;

3. Making it possible to explore more complex systems and detect possible hierarchical dependencies between RSs.

This section presents the results related to the first goal, while goals 2 and 3 are addressed in Chapters 4 and 5, respectively.

In the next subsections we analyze the computational problem, identifying the algorithm blocks that are most suitable for parallelization, and describing their GPU-based implementation. We then report the results of the tests in which we compare the performance of our parallel code with respect to a standard single-CPU sequential implementation.

[7] `https://developer.nvidia.com`

3.4.1 Parallel algorithm

This section describes the parallel implementation of the main building blocks of the code needed to compute the T_c index (described in Section 2.1.2).

The GPU is specialized for compute-intensive and highly parallel computation and is capable of addressing highly arithmetically-intense problems that can be expressed as data-parallel computations. The computation of T_c for each CRS is independent of the others, thus a GPU-based parallel code is suitable for efficiently computing such an index. That is why we have developed CUDA C code for searching RSs in complex systems.

In order to understand how our code is organized we should consider that the exhaustive computation of the T_c index for all the CRSs of a dynamical system can be divided into the following steps:

1. Computation of the probability distribution function (estimated as a frequency histogram) for each system variable, from the observations of the state of the system variables over time;

2. Generation of the homogeneous system;

3. RI computation for each subset of variables of the homogeneous system;

4. T_c computation for each CRS of the system variables.

From the point of view of implementation:

- Each sample is stored in a memory area including S adjacent unsigned ints large enough to contain the N_{bit} bits needed to represent the N variables of the system. For example, if we consider a system consisting of N binary variables, then $N_{bit} = N$ and $S = \lceil N_{bit} / \mathsf{sizeof(unsigned\ int)} \rceil$. If M is the number of samples, then the system data can be stored in an array of $M \cdot S$ unsigned integers.

- Each CRS is represented as a bitmask of N_{bit} bits, where the i^{th} bit is set to 1 if the i^{th} variable is contained in the CRS.

Computation of the probability distribution function

Each variable of the system is examined individually in order to compute its probability distribution function. In case of binary variables, for example, the distribution of the i^{th} variable is approximated by the frequency of the values 0 and 1 (f_{i0} and f_{i1}). The frequency information thus obtained will be used for the generation of the homogeneous system.

The frequencies of occurrence of the variables are also used to compute the entropy of each variable, necessary for the computation of the RI. If we consider a binary variable, then the entropy is defined by:

$$H_i = -f_{i0} \cdot log_2 f_{i0} - f_{i1} \cdot log_2 f_{i1}$$

Homogeneous system generation

The homogeneous system (HS) is generated from N random variables, homogeneously correlated with one another, having the same probability distribution as the corresponding variables of the dynamical system to be studied.

We obtain M samples by assigning to the i^{th} variable, for each sample, a randomly generated value from the previously estimated distribution.

In case of a system described by binary variables, the i^{th} variable of the homogeneous system, for each sample, will be 0 with probability f_{i0} and 1 with probability f_{i1}. In this way, the HS meets the homogeneity requirement while, at the same time, it maintains a relationship with the dynamical system under consideration.

RI computation on the homogeneous system

All possible CRS sizes (or classes) from 2 to $N-1$ are analyzed in order to compute, for each of them, the mean value and the standard deviation of the RI. If the considered size is r, then the CRSs to be examined are selected by scanning all possible permutations of an N-bit string containing r bits set to 1 and $N-r$ bits set to 0.

The selected CRSs are grouped into grids of T threads each, where each thread is responsible for computing the RI of one CRS. We have $T = N_B N_T$, where N_B is the

number of blocks per grid and N_T is the number of threads per block. Each CRS of a certain size is coupled with its complementary cluster, whose entropy is necessary for computing the mutual information. In other words, each grid is composed of $T/2$ complementary CRS pairs. By synchronizing the execution of parallel threads in order for the entropy of one CRS to be available at the right time, it is possible to compute the statistics of classes r and $N-r$ at the same time, halving the computation time with respect to the original algorithm.

The outputs of this processing step are the mean value and the standard deviation of the RI for each CRS class of the homogeneous system, which are necessary for computing the T_c index.

T_c computation

In the following, we describe the main modules involved in the computation of the T_c index, as shown in Figure 3.11.

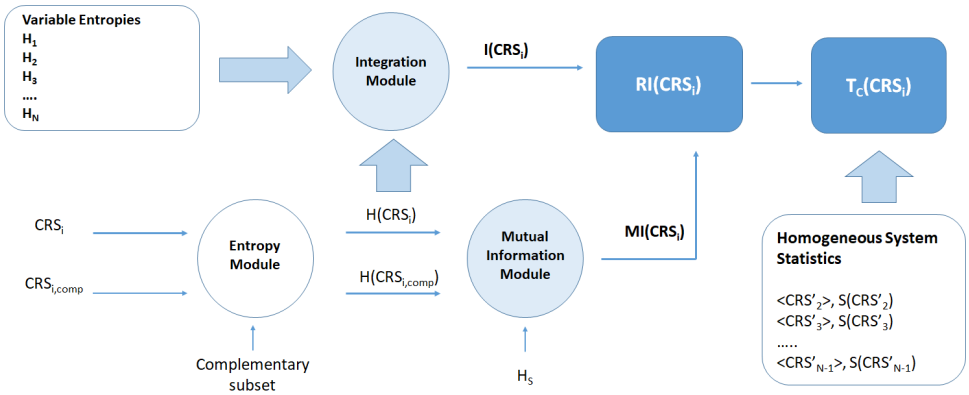


Figure 3.11: T_c computation.

RI module: The computation of the *RI* of a CRS consists of three phases:

1. *Creation of the frequency histogram*: the number of occurrences of each value of the CRS is counted; the result is a list of value/number of occurrence pairs;

2. *Entropy computation*: based on the list obtained in the previous phase, the entropy is computed according to Eq. 2.1;

3. *Computation of the final output*: the threads of the block are synchronized to make the complementary entropy available to each CRS. This enables the computation of the mutual information, which, along with the integration, is used to compute the RI.

Calculating the frequency histogram is, computationally, the heaviest step. In particular we need:

- Processing resources to extract the value of the variables in the CRS from each sample of the system;

- Memory to store the frequency histogram of the CRS.

To obtain a good trade-off between performance and memory usage, we generate a hash map, pre-allocated for each thread to be managed by the GPU kernel that computes the histogram.

T_c module: The module that computes the T_c statistical index is a simple extension of the one which computes the RI, that takes advantage of the above-mentioned organization into coupled threads. Particularly, in this case, the CRSs of each class, ranging from 2 to N/2, are placed aside their complementary CRSs and inserted, as for the RI computation for the homogeneous system, in parallel computation batches, each composed of T threads. Once the RI has been computed, it is sufficient to normalize it according to the statistics (expected value and standard deviation) of the homogeneous system that were obtained earlier by the RI homogeneous system module. As the T_c module simply extends the RI module, both call the same CUDA kernel to perform their computations; the calls differ only in the input parameters.

Once the T_c indices of all the CRSs of the system are obtained, they are compared to select the CRSs having the highest index values.

Resource occupation and scalability

If N is the number of variables that compose the system, the total number of possible CRSs is 2^{N-1}. Thus, the computational complexity of the problem is $O(2^N)$. Parallelizing the computation allows one to obtain a relevant reduction of the execution time. However, this is still not enough to perform an exhaustive search on systems characterized by a large number of variables.

Different considerations can be made regarding memory occupation. Our implementation is based on a simple fact: it is not possible for a CRS to assume a number of configurations that is higher than the number of available samples M, which is usually much lower than the total number of possible CRS configurations (i.e., $M \ll 2^N$). Thus, for each CRS it is possible to pre-allocate a hash table with maximum size M. For this reason, the device memory that is necessary to contain the hash tables of a grid of threads is directly proportional to three independent variables, namely:

- T: number of threads per grid;
- N_{bit}: number of bits needed to store a sample;
- M: number of samples.

Accordingly, the memory occupation increases linearly with the problem size. A good estimation of the device memory needed is:

$$MEM_{TOT} = M \cdot T \cdot (S+2) \cdot \text{sizeof(unsigned int)} \tag{3.36}$$

where S is the number of unsigned int that are necessary to store N_{bit} bits. On a device provided with 2 GB of memory, it is possible, for example, to launch 1024 parallel threads and compute the T_c of the same number of CRSs from a system characterized by 1000 binary variables, with $M = 10000$ available samples (in this case, $MEM_{TOT} \simeq 1.4$ GB).

These considerations show that, to analyze large systems, the exponential dependence on the problem size makes an exhaustive search computationally unfeasible. However, an approach based on a metaheuristic would definitely be feasible, as the device memory occupation scales linearly with the problem size.

3.4.2 Experimental results

In this section we illustrate the experimental results we have obtained on four different dynamical systems. The algorithm has been evaluated on both artificial and biological systems.

The first case study is described by 10 variables and consists of three independent groups, each of which replicates a simple **Leader-Follower** (LF) dynamic, which has been described in section 3.1.2. The system is simply composed of a vector of 10 binary variables $x_1, x_2, ..., x_{10}$ that represent, for example, the positive or negative opinion of 10 agents about a given proposal. The model generates a series of binary vectors (each vector representing an observation of the system) according to the following rules:

- variables are divided into three groups, $G1 = [x_1, ..., x_3]$, $G2 = [x_4, ..., x_6]$ and $G3 = [x_7, ..., x_{10}]$;

- x_1 is a leader; at each step its value is a random value in $\{0, 1\}$;

- the values of the followers x_2 and x_3 are set as a copy of x_1 with probability $1 - p_{noise}$ and randomly with probability p_{noise};

- x_4 and x_7 are "second order" leaders; in each step their values are randomly assigned in $\{0, 1\}$ with probability $1 - p_{copy}$; otherwise x_4 is a copy of x_1 and x_7 is a copy of x_4;

- the values of the followers x_5 and x_6 are set as a copy of x_4 with probability $1 - p_{noise}$ and randomly with probability p_{noise};

- the values of the followers x_8, x_9 and x_{10} are set as a copy of x_7 with probability $1 - p_{noise}$ and randomly with probability p_{noise}.

It is therefore possible to tune the integration among elements in $G1$, $G2$ and $G3$ and the mutual information between $G1$ and $G2$, and between $G2$ and $G3$ by changing p_{noise} and p_{copy} [77, 241].

The second and third cases model simplified gene regulatory networks. In particular, the second case study (referred to as AT) models the gene regulatory network shaping the developmental process of **Arabidopsis Thaliana**; although the whole network is largely unknown, a certain subsystem has been identified as responsible for the floral organ specification. The network is modeled by means of a Boolean Network described in [38], having 15 nodes and 10 different attractors (all fixed points): in order to perform an analysis we therefore built a data series containing a number of repetitions of these attractors proportional to the size of their basins of attraction.

The third case (referred to as TH) features 23 Boolean variables, used in [149] to model the regulatory network controlling the **T-helper cell differentiation**; also in this case we built a data series containing a number of repetitions of the Boolean system attractors proportional to the size of their basins of attraction. We will not discuss about the adequacy of these simplified models, but we will take them for granted and apply our method to test whether it can discover significant MDSs (Mesolevel Dynamical Structures).

The fourth case study is a deterministic simulation of a catalytic chemical system (**catalytic reaction system**), which follows two distinct reaction pathways: a linear chain and an autocatalytic circle. The system is composed of 26 variables (reagents, products and intermediate complexes, combining the substrates and the catalysts). The reactions happen in an open well-stirred chemostat (CSTR), which has been described in section 3.1.3. The asymptotic state of this system consists of constant concentrations. In order to apply our analysis, however, we need to observe the feedbacks dynamically: thus, the concentration of some molecules has been perturbed in order to trigger a response (i.e., a series of changes) in the concentration of (some) other species. To analyze the system response we used a three-level coding where, for each species, the digit “0”, “1” and “2” stand respectively for “decreasing concentration”, “no change” and “increasing concentration” (Figure 3.12) [240, 241].

The four cases present different dynamics and representations: in particular, the first test case consists of a binary time series, whereas the second and third cases are the juxtaposition of the binary states of several different attractors, and the fourth case is the encoding of a continuously perturbed situation into a three-level representation.

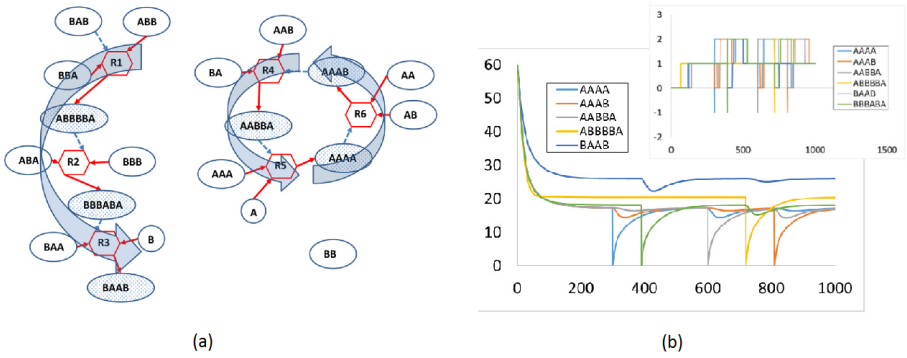


Figure 3.12: (a) The reaction scheme of the catalytic reaction system: white ellipses represent the chemicals injected in the incoming flux, meshed ellipses represent the chemicals produced inside the CSTR vessel, hexagons represent the reactions; continuous arrows represent the consumptions/productions, and dashed arrows represent the catalytic activities. Chemical BB does not participate in any reaction, and it is used as reference. The six reactions are arranged into two independent groups: a linear chain and an autocatalytic circle. (b) A time series of the six produced chemicals and the corresponding three-level encoding.

The method we implemented on GPU is able to find the correct relevant sets in all situations (some of them being discussed in details in [240, 241]): in this section, however, we focus our interest on the performance of the sequential and parallel algorithms.

The parallel algorithm (PA) has been evaluated in terms of correctness and efficiency (speedup), compared to the sequential algorithm (SA). To this purpose, we have used a Linux server provided with CPU Intel(R) Xeon(R) 2.10 GHz, 64 GB of RAM and a GPU NVIDIA GeForce GTX 1070. We have executed 10 independent runs for each example, using different random seeds when generating the homogeneous system.

Table 3.2 summarizes the performance of the algorithms in relation to the system size (expressed as number of variables) and to the number of samples.

Table 3.2: Performance summary of the sequential (SA) and parallel (PA) algorithms

System	#Variables	#Samples	Time (SA)	Time (PA)	Speedup
LF	10	500	2.15 s	11 ms	195.5
AT	15	5000	861 s	0.23 s	3743.5
TH	23	5000	60 h *	27.5 s	7854.5
CSTR	26	751	20 d *	245 s	7053.1

* estimated

In all these case studies the results are correct: they are equal to the ones obtained by the sequential implementation, but they have been computed in a significantly shorter time (e.g., with a speedup of 7854.5 for the system described by 23 variables). Using our parallel implementation we can now exhaustively analyze systems of up to 35 variables in less than 24 hours on the same hardware configuration described in the present section.

The speedup with respect to the sequential CPU implementation is very relevant.

3.4.3 Final remarks

In this section we have presented a fine-grained parallel implementation of the main building blocks needed to compute the T_c index. In summary, the most relevant choices, aiming at algorithm efficiency, are:

- Subgroup-wise parallelization (as opposed to a possible system data-wise parallelization);
- "Smart" allocation of threads/data (like using a hashmap for each thread, implemented on the graphics device).

These choices produce an algorithm which obtains a large speedup, but they are a little more critical as concerns memory allocation.

Considering multi-GPU implementations, the structure of the parallel algorithm is such that the computation of each T_c index is totally independent of the others,

which suggests that the number of T_c computations scale almost perfectly linearly with the number of GPUs.

In the benchmarks we took into consideration, the algorithm obtained a dramatic speedup with respect to the sequential implementation, allowing us to detect RSs in dynamical systems of much larger size than previously possible.

When large systems are analyzed, the increasing number of CRSs makes it impossible to compute the T_c index for every possible subset, even using massively parallel hardware such as GPUs, so we need to design efficient strategies to quickly identify the most promising subsets, limiting the extension of the search.
Smart and efficient search algorithms can be easily designed upon our parallel implementation, which provides a computationally-efficient objective function for a metaheuristic that searches for the RSs of large systems. In particular, two niching metaheuristics designed for complex systems analysis are presented in Chapter 4.

An extended description of the work described in this chapter can be found in [201, 237].

Chapter 4

Metaheuristics for relevant set detection

A particularly critical issue in the RI analysis is the efficient detection of the highest-ranked CRSs according to the corresponding RI metric (e.g., the T_c index). Indeed, the number of CRSs increases exponentially with the system dimension, the number of CRSs of size k in a system of size N being $\binom{N}{k}$. Because of this, smart search methods are needed to analyze large-size systems using such an approach.

To characterize a dynamical system of interest, one does not need to know the T_c index of all possible CRSs, but only to detect the CRSs for which the T_c is highest. The search of relevant subsets within a dynamical system corresponds to an optimization problem (T_c index maximization). Population-based niching metaheuristics, which have been described in section 2.2.3, fit well this multimodal optimization scenario.

To this aim, we developed two niching metaheuristics, based on hybridizations of GAs and PSO, respectively. The proposed metaheuristics, described in sections 4.1 and 4.2, are presented herein for T_c index maximization, but they can be applied to complex systems analysis also using other RI metrics as fitness functions.

4.1 HyReSS: Hybrid Relevant Set Search

This section presents HyReSS (Hybrid Relevant Set Search), a hybrid metaheuristic for searching relevant sets within dynamical systems, based on the hybridization of an evolutionary algorithm with local search strategies.

In the tests we have made on data describing both real and synthetic systems, HyReSS has been shown to be very efficient and to produce results comparable to an exhaustive search in a much shorter time.

4.1.1 HyReSS algorithm

HyReSS hybridizes a basic genetic algorithm with local search strategies that are driven by statistics, computed at runtime, on the results that the algorithm is obtaining.

A niching genetic algorithm is first run to draw the search towards the basins of attraction of the main local maxima in the search space. Then, the results are improved by performing a series of local searches to explore those regions more finely and extensively.

The method can be subdivided into five main cascaded steps:

1. Genetic algorithm;
2. CRS relevance-based local search;
3. CRS frequency-based local search;
4. Group cardinality-based local search;
5. Merging.

Genetic Algorithm

As specified above, HyReSS does not search a single CRS, but the set $\mathscr{B}$ of the N_{best} highest-T_c CRSs. The first evolutionary phase is a genetic algorithm based on the Deterministic Crowding (DC) algorithm, one of the most efficient and commonly used

niching techniques. We have used the Deterministic Crowding algorithm because it does not require *a priori* setting of problem-related parameters, such as the similarity radius, and its complexity is low, since it scales as $O(n)$ with the number of dimensions of the search space. The Deterministic Crowding algorithm is described in detail in section 2.2.3.

Each individual corresponds to one CRS and is a binary string of size N, where each bit set to 1 denotes the inclusion in the CRS of the corresponding variable, out of the N that describe the system. A list (termed "best-CRS memory" in the following) is created to store the best individuals that have been found along with their fitness values. At the end of the run, it should contain all CRSs in $\mathscr{B}$.

The initial population, of size p, is obtained by generating random individuals according to a pre-set distribution of cardinality (pairs, triplets, etc.). This kind of generation aims to create a sample that is as diversified as possible (avoiding repetitions) as well as representative of the whole search space.

The fitness function to be maximized corresponds directly to the T_c and is implemented through the CUDA kernel described in section 3.4, which can compute in parallel the fitness values of large blocks of individuals.

Evolution proceeds by selecting $p/2$ random pairs of individuals and creating p children by means of a single-point crossover. After crossover, each child possibly replaces the most similar parent of lower fitness. To safeguard genetic diversity, a parent is only replaced if the child is not already present in the population.

This evolutionary process is iterated until the population is no more able to evolve, i.e., the new generation remains equal to the previous one. When that happens, new random parents are generated.

Mutation (implemented as bit flips) is applied with a low probability (P_{mut}) after each mating.

The termination condition for this evolutionary phase is reached when the number of evaluations of the fitness function exceeds a threshold α_f or new parents have been generated for α_p times.

The implemented algorithm is elitist, since a child is inserted in the new population only if its fitness is better than the fitness of the parent it substitutes. Therefore,

the overall fitness of the population increases monotonically with the algorithm iterations. After the end of the evolutionary algorithm, the N_{gBest} fittest individuals are selected to seed the subsequent phases.

Variable relevance-based search

While running the genetic algorithm, a relevance coefficient RC_i is computed for each variable i of the system under examination. RC_i is higher if variable i is frequently included in high-fitness CRSs.

At the end of each generation t of the GA, a fitness threshold τ is set, corresponding to a certain percentile β of the whole fitness range, to separate high-fitness CRSs from low-fitness ones.

$$\tau(t) = minFitness + (maxFitness - minFitness) * \beta \tag{4.1}$$

A presence coefficient (PC_i) and an absence coefficient (AC_i) are defined, for variable i, as the sum of the fitness values of the CRSs having fitness greater than τ, in which the variable has been present or absent, respectively, cumulated over the generations and normalized with respect to the number of generations in which the corresponding CRSs have been included.

Based on these two coefficients, the ratio $R_{ap,i} = AC_i/PC_i$ is computed. The variable is classified as relevant if PC_i is greater than a threshold (the γth percentile of the full range of PC_i values) and $R_{ap,i}$ is lower than a certain threshold δ.

The corresponding local search procedure performs a recombination of the most relevant variables with other, randomly chosen, ones. As a first step, all possible subsets (simple combinations) of the most relevant variables are computed, excluding the subsets of cardinality 0 and 1. Then, for each subset dimension, the individual with the highest fitness is selected. Such individuals are the basis for generating new CRSs, by forcing the presence or absence of relevant/irrelevant variables and by randomly adding other variables into the RCSs. Every newly generated individual is evaluated and, should its fitness be higher, replaces the lowest-fitness individual in the best-CRS memory.

At the end of this phase, a local search is performed in the neighborhood of the best individual of the best-CRS memory, which is updated in case new individuals with appropriate fitness are obtained.

Variable frequency-based search

In this phase, the same procedure used to generate new individuals and to explore the neighborhood of the best one is repeated, based on a different criterion.

We consider the frequency with which each variable has been included in the CRSs evaluated in the previous phases and use this value to identify two classes of variables, which are assigned higher probability of being included in the newly generated CRSs:

- variables with frequency much lower than the average;
- variables with frequency much greater than the average.

In fact, variables of the former kind may have been previously "neglected", thus it may be worth verifying whether they are able to generate good individuals, while variables of the latter kind are likely to have been selected very frequently in the evolutionary process because they actually have a significant relevance.

Group cardinality-based search

During the previous phases, HyReSS records the frequency with which groups of each possible cardinality (2, ..., N-1) have occurred. These indices are then normalized according to the a priori probability of occurrence of such groups, given by the corresponding binomial coefficient $\binom{N}{c}$ where N is the total number of variables and c the cardinality of the group.

New CRSs are then generated using a procedure driven by such indices, such that cardinalities having lower values have higher probability of occurring and are possibly stored into the best-CRS memory according to their fitness.

Merging

In this phase a limited pool of variables is selected by considering all variables that are included in the highest-fitness CRSs in the best-CRS memory. In practice, a size θ for the pool is set; then, the best individuals are progressively OR-ed bitwise, in decreasing order of fitness starting from the best two CRSs, until the result of the bitwise OR contains θ bits set to 1 or all the CRSs have been processed. A final exhaustive search over all the possible CRSs that comprise the selected variables, is made, and the best-CRS memory is updated accordingly.

4.1.2 Experimental results

In this section we illustrate three examples of dynamical systems we have used as benchmarks for HyReSS. The first one is a deterministic simulation of a *Catalytic Reaction System (CSTR)*, described by 26 variables. The second one is a stochastic artificial system reproducing a *Leaders & Followers (LF)* behavior, featuring 28 variables. These examples have been analyzed using both an exhaustive search and HyReSS. The third example, denoted as *Green Community Network (GCN)*, features 137 variables, a size for which an exhaustive search is not feasible on a standard computer. Thus, it was analyzed only by HyReSS. However, we could compare its results with those provided by field experts.

In all our test, we have performed 10 independent runs of HyReSS, to take the stochastic nature of the tool properly into account. We evaluated the results of HyReSS, when possible, by comparing the list of highest-T_c subsets it produced with the results of an exhaustive search. To let results be comparable, we relied on the same homogeneous system to compute normalized DCI values in both approaches. Tests were run on a Linux server equipped with a 1.6 GHz Intel I7 CPU, 6 GB of RAM and a GeForce GTX 680 GPU by NVIDIA. The parameters regulating the behavior of HyReSS were set as reported in Table 4.1.

Results are summarized in Table 4.2 and are discussed in the following subsections.

Table 4.1: HyReSS parameter settings. The parameters are defined in section 4.1.1.

System	P_{mut}	p	α_f	α_p	β	γ	δ	θ
CSTR	.1	16384	163840	3	.75	.75	.3	15
LF	.1	16384	163840	3	.75	.75	.3	15
GCN (56 vars.)	.1	25600	256000	3	.75	.75	.3	15
GCN (137 vars.)	.1	50176	501760	3	.75	.75	.3	15

Table 4.2: Summary of HyReSS performances and comparison with an exhaustive search (ES), when possible.

System	N. Variables	N. Samples	Time (ES) [s]	Time (HyReSS) [s]	Speedup
CSTR	26	751	180	24	7.5
LF	28	150	300	19	15.8
GCN	56	124	n.a.	71	n.a.
GCN	137	124	n.a.	258	n.a.

Catalytic Reaction System

The set of observations comes from the simulation of an open well-stirred chemostat (CSTR) with a constant incoming flux of feed molecules (empty ellipses in Figure 4.1) and a continuous outgoing flux of all molecular species proportionally to their concentration. This case study has been presented in section 3.1.3.
Six catalyzed reactions produce six new chemical species (pattern-filled ellipses in Figure 4.1) and are divided in two dynamical arrangements, a linear chain and a circle. The system asymptotic behavior is a fixed point: we perturbed each single produced chemical species, in order to allow the variables to change their concentrations over time and thus highlight their dynamic relationships. In this work, we encoded each species' trajectory as a binary variable, the 0 and 1 symbols meaning

"concentration is changing" and "concentration is not changing" respectively.

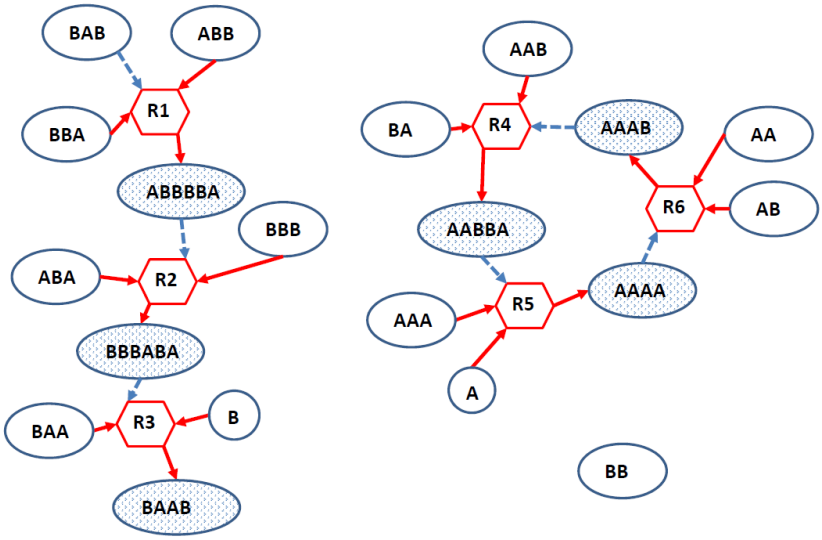


Figure 4.1: The simulated CSTR. Circular nodes represent chemical species, while the white ones represent the species injected in the CSTR and the pattern-filled ones those produced by the reactions. The diamond shapes represent reactions, where incoming arrows go from substrates to reactions and outgoing arrows go from reactions to products. Dashed lines indicate the catalytic activities.

Since the system has "only" 26 variables (corresponding to reagents, products and intermediate complexes), it has been possible to perform an exhaustive search to be used as reference, which took about 180 s. Producing almost identical results (the resulting error rate is less than 0.02[1]), the average running time of HyReSS was 24 s.

[1] In one of the 10 runs, HyReSS failed to detect 1 of the first 50 RSs detected by the exhaustive search.

Leaders & Followers

The model is an abstract representation of the basic Leader-Follower (LF) model, which has been presented in the previous chapter (section 3.1.2). The model generates independent observations on the basis of the following rules:

- the variables are divided into four groups:
  - $G1 = \{A0, A1, A2, A3\}$
  - $G2 = \{A7, A8, A9\}$
  - $G3 = \{A12, A13, A14, A15, A16, A17, A18, A19, A20\}$
  - $G4 = \{A22, A23, A24, A25, A26, A27\}$
- the remaining variables $A4, A5, A6, A10, A11$ and $A21$ assume the value 0 or 1 with identical probability;
- variables $A0, A7, A12, A22$ and $A23$ are the leaders of their respective groups; at each step they randomly assume value 0 respectively with probability 0.4, 0.3, 0.3, 0.3 and 0.6, 1 otherwise;
- the other variables (i.e., the followers) copy or negate the values of their leaders, with the exception of the followers belonging to group $G4$, which compute the OR or AND function of their two leaders.

Thus, the LF case we have considered features 28 variables. An exhaustive search takes about 300 s. HyReSS completes its run in 19 s, on average, always providing the same results as the exhaustive search, considering the 50 highest-T_c subsets as a reference.

Green Community Network

In this case, the data come from a real situation and show the participation (*i.e.*, the presence or absence) of 137 people in a series of 124 meetings, held during a project

(the so-called Green Community project [7]) which involved four mountain communities and focused on studies addressing energy efficiency and renewable energy production. This case study has been described more in detail in section 3.1.4.

The objective of the analysis is to evidence the formation of specific dynamics among subsets of participants, despite the apparently simplicity of the information carried by the observations.

We have considered two versions of the GCN. The first one includes all 137 variables, whereas the second one has only 56 variables, representing people who attended more than one meeting. Both cases are described by too many variables to perform an exhaustive search.

The real situation is complex, with a part of the sociologically significant groups composed by smaller but very active (and sometimes partially overlapping) subgroups, having very often T_c values higher than the values computed for larger groups which include them: as a consequence, the complete analysis would require the application of the sieving algorithm, which has been described in section 2.1.2 and will be analyzed more in details in chapter 5. In these tests, however, we are focusing on the search of the highest-ranked CRSs, a step fundamental for the correct detection of the sociologically significant groups. In this regard, HyReSS is quite effective, (i) finding almost all the expected highest-ranked CRSs, (ii) identifying unknown groups certified *a posteriori* by the human experts and (iii) highlighting the presence of groups judged "interesting and requiring further investigations" by the human experts. HyReSS achieved these results in a very efficient way, with average running time of 71 s for 56 variables and 258 s for 137 variables.

4.1.3 Final remarks

In this section we have presented HyReSS, an ad-hoc hybrid metaheuristic, tailored to the problem of finding the candidate Relevant Subsets of variables that describe a dynamical system. In developing our search algorithm, we have combined GAs' capacity of providing a good tradeoff between convergence speed and exploration, with local searches which refine and extend results, when correctly seeded. Using the deterministic crowding algorithm as a basis for the GA we guarantee that a large

number of local maxima are taken into consideration in the early stages of HyReSS. The subsequent local search stages extend the results of the GA (stochastic) search more systematically to those CRSs that are most likely to have high fitness values, according to a few rules essentially derived from common sense.

In the benchmarks we took into consideration, HyReSS was very fast on an absolute scale, thanks to the GPU implementation of the fitness function. Even more importantly, at least for the smaller-size systems for which the comparison was possible, it could provide the same results as an exhaustive search based on the same parallel code, performing much fewer fitness evaluations and, consequently, in a significantly shorter time. The results obtained on the larger problems, on which the speedup with respect to an exhaustive search is virtually incommensurable and for which a ground truth is therefore not available, were qualitatively aligned with the expectations of a domain expert who analyzed the data.

The availability of an efficient algorithm has allowed us to extend our research on the detection of candidate RSs to dynamical systems of much larger sizes than previously possible (see chapters 6 and 7). At the same time, it has made it possible to devise more complex analyses, by which we aim to detect also hierarchical relationships among RSs (see chapter 5).

4.2 K-means PSO for relevant set detection

In this section, we use K-means PSO [173] for searching relevant dynamical structures of complex systems and for extracting the RSs when the dimension of the variable space makes an exhaustive search unfeasible. K-means [144] is used as a niching technique to preserve diversity within the swarm, exploring many peaks in parallel. PSO, even if mainly utilized for continuous optimization, is employed in this case also on a discrete domain problem. The representation used in this context (see Section 4.2.2) has already been successfully employed for feature selection [162].

The following subsections describe in detail the metaheuristic, the algorithms on which it is based, and their application to the analysis of complex systems.

We applied K-means PSO to a set of complex systems of different size, com-

paring, when possible, its results with the ground truth represented by the results of an exhaustive search, while we rely on the analysis of a domain expert to assess the results of larger systems. In all cases, we also compare the results of K-means PSO to HyReSS, which has been presented in section 4.1.

4.2.1 K-means PSO algorithm

The canonical PSO algorithm, described in section 2.2.2, aims at finding a single point representing the global optimum of an N-dimensional function. The convergence of PSO on a Rastrigin function of two variables is shown in Figure 4.2, which shows that the global optimum of the fitness function has been finally found and virtually the whole swarm has converged onto it. However, because of this, the local optima have not been identified. The enhancement of this process using a niching technique as K-means PSO allows the swarm to converge onto many peaks in parallel (see also Figure 4.3).

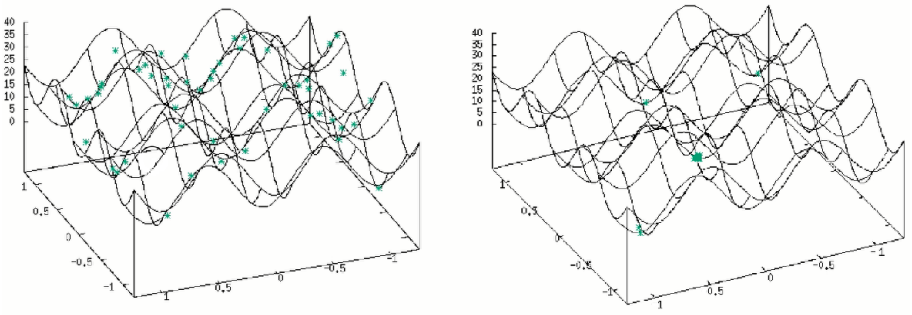


Figure 4.2: Particle Swarm Optimization: the random initialization of the swarm positions, using a Rastrigin function of two variables as fitness function (left) and the results at convergence (right).

K-means PSO [173] combines the canonical PSO algorithm with K-means clustering [144], which is used to group the particles in sub-swarms.

K-means clustering aims at partitioning n observations into k clusters in which

each observation belongs to the cluster whose mean (centroid) is the nearest to it. K-means clustering uses the Euclidean distance as similarity criterion and

$$J = \sum_{k=1,Nc} \sum_{y^{(i)} \sim \omega_k} |y^{(i)} - \mu_k| \tag{4.2}$$

as the function to be optimized, where $y^{(i)}$ is the i^{th} sample, μ_k is the centroid of the k^{th} cluster, N_c is the number of clusters, and $y^{(i)} \sim \omega_k$ refers to all samples $y^{(i)}$ assigned to cluster k.
$M = \{\mu_1, \mu_2, ...\mu_{Nc}\}$ is the set of reference vectors, each of which represents the prototype for a class. J is minimized by computing μ_k as the sample mean of the data belonging to cluster k.
In practice, the algorithm partitions the input space S into k (the number of clusters, that must be set by the user) subspaces induced by the Euclidian distance. Each subspace s_i of S is defined as:

$$s_i = \{x_j \in S \mid d(x_j, \mu_i) = min_t \; d(x_j, \mu_i)\} \tag{4.3}$$

This results in a partitioning of the data space into Voronoi cells (Voronoi tessellation).

K-means clustering is combined with PSO algorithm to deal with multimodal optimization problems. With respect to the basic PSO algorithm, in the K-means PSO the search process is enhanced by a niching technique that maintains the diversity among the particles of the swarm and allows the swarm to explore and converge onto many peaks in parallel. In particular, in K-means PSO, at regular intervals, the K-means clustering algorithm [144] is applied to the swarm to reorganize it into subswarms characterized by the proximity of their elements in the search space. The standard PSO algorithm is then independently applied to each sub-swarm thus identified.

The K-means PSO pseudo-code is reported in Algorithm 7. Lines 5, 6 and 7 represent the application of K-means clustering.

With respect to standard PSO, K-means PSO requires some additional parameters: C is the number of PSO cycles to be performed between two clustering opera-

Algorithm 7 K-means PSO pseudo-code.

```
1: procedure KPSO
2:     randomly initialize the particles' positions and velocities
3:     compute each particle's fitness
4:     for t = 1 to T do                          ▷ T = number of iterations
5:         if t mod C = 0 then                                ▷ every C steps
6:             run K-means algorithm to identify niches
7:         end if
8:         update each particle's velocity                  ▷ as in standard PSO
9:         update each particle's position
10:        compute each particle's fitness
11:        update each particle's and each niche's best position
12:    end for
13: end procedure
```

tions, k represents the number of clusters (it should be slightly higher than the number of local optima of the fitness function).

The convergence of K-means PSO using a Rastrigin function of two variables as fitness function is shown in Figure 4.3 (right). The algorithm is able to explore many peaks in parallel, finding as many local optima as possible.
Figure 4.3 also compares the results of PSO and K-means PSO using a two-dimensional Rastrigin function as fitness function.

4.2.2 Relevant set search using K-means PSO

In our work, K-means PSO has been applied to the analysis of complex systems for detecting the highest-T_c CRSs in large-sized system.

In our method, each particle i of the swarm represents CRSs as a binary string P_i of size N, where N is the number of variables that describe the system. A bit of this

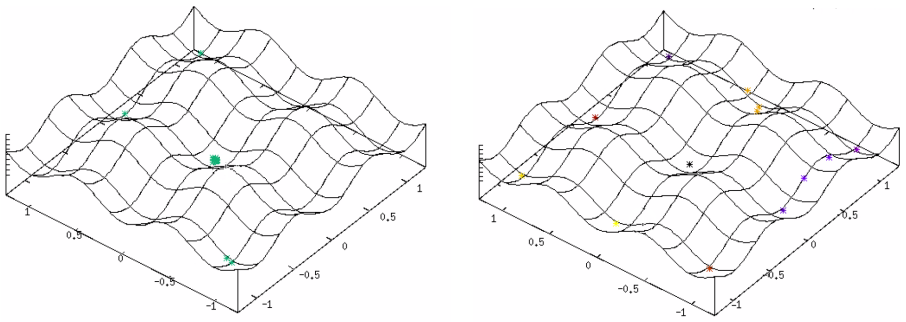


Figure 4.3: Results of PSO (left) and K-means PSO (right) using a Rastrigin function of two variables as fitness function.

binary string is set to 1 if the corresponding variable is included in the CRS.

$$P_i(j) = \begin{cases} 1 & \text{if variable } j \text{ is included in the CRS} \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

where $j \in [1,N]$.

Since PSO operates in $\mathscr{R}^N$ and each particle i is therefore represented by its coordinates $p_r^i \in \mathscr{R}^N$, the corresponding binary vector P_i is obtained from each particle by setting the bits corresponding to positive coordinates in the search space to 1, and setting the others to 0 .

$$P_i(j) = \begin{cases} 1 & \text{if } p_r^i(j) \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

The other steps of the algorithm, for example position and velocity update, are applied to the floating-point vectors p_r^i.

The fitness function to be maximized corresponds to the T_c index of the CRS associated to the particle and is implemented through a CUDA C kernel, described in section 3.4, that can compute in parallel the fitness values of large blocks of particles. Position and velocity updates have been parallelized as well.

A buffer has been introduced to store the best subsets (those having the highest T_c index) found during the run, and their corresponding fitness (T_c) values. Thus, at the end of the run, the best CRSs are not only the ones represented by the last swarm, but also the best ones found during the whole search process, which are stored in the buffer.

4.2.3 Experimental results

The K-means PSO has been evaluated on a set of meaningful systems described by Boolean variables. The results have been compared with those achieved by an exhaustive search, when computationally feasible, and by HyReSS (described in section 4.1). All the algorithms rely on the same GPU implementation of the fitness function.

Given the stochastic nature of the two meta-heuristics, 30 independent runs of the algorithm were executed to assess its performance.

The results are summarized in Table 4.3. The case studies are the same which have been used for the experimental evaluation of HyReSS (see sections 3.1 and 4.1.2). The first case study is a simulation of a **Catalytic Reaction System** (CSTR), featuring 26 variables. The second one is a stochastic artificial system reproducing a **Leaders & Followers** (LF) behavior, described by 28 variables. In the third example, the data come from the **Green Community Network** (GCN) and are described by 56 variables, a size for which an exhaustive search is unfeasible on a standard computer, even using GPU parallelization. Therefore, this case study has been analyzed only by the two meta-heuristics.

Tests were run on a Linux server equipped with a 1.6 GHz Intel I7 CPU, 6 GB of RAM and a GeForce GTX 680 GPU by NVIDIA.

The parameters regulating the behavior of K-means PSO have been set as reported in Table 4.4. They are the ones that led to the best results. Moreover, ϕ_1, ϕ_2 and χ were obtained considering also the constraints presented in Equations 2.11 and 2.12 (see Section 2.2.2).

The results of the two metaheuristics have been evaluated in terms of both quality and speed-up with respect to an exhaustive search. Quality has been evaluated, when

Table 4.3: Results obtained by K-means PSO (K) on three different systems and comparison with HyReSS (H) in terms of time and speed-up with respect to an exhaustive search (E).

System	Size	N. data	Time (E) [s]	Time (H) [s]	Time (K) [s]	Speedup (H)	Speedup (K)
CSTR	26	751	53	24	6	2.2	8.8
LF	28	150	196	19	2	10.3	98
GCN	56	124	n.a.	71	18	n.a.	n.a.

Table 4.4: K-means PSO parameter settings. The parameters are defined in sections 2.2.2 and 4.2.1.

System	s	T	k	C	ϕ_1	ϕ_2	χ
CSTR	2000	501	10	20	2.05	2.05	0.73
LF	1000	501	10	20	2.05	2.05	0.73
GCN	2000	2001	10	20	2.05	2.05	0.73

feasible, counting the number of highest-T_c CRSs detected by the exhaustive search, but not by K-means PSO. Regarding the large-sized system, for which the results of the exhaustive search were not available, we have relied on the opinion of an expert to assess their quality.

The results obtained with the smaller-size systems, for which the comparison is possible, are almost always the same as those provided by an exhaustive search. Only at most one out of the top 50 sets was not detected by both metaheuristics, and we considered this acceptable and more than enough to understand the main dynamics of the systems. For this reason, we were able to compare the results of the two metaheuristics with those of the exhaustive search.
The results obtained on the GCN were judged as reasonable by an expert. In this case, too, the same results were obtained in different runs, with marginal occasional differences only within the least significant sets.
We analyzed also an extended version of the GCN system, described by 137 variables. In this case, some of the most significant sets, identified by HyReSS with an average running time of 258 s, were not identified by K-means PSO (with a running time from 100 to 300 s). This is probably due to the absence of a local search step.

In general, K-means PSO exhibits both good exploration capabilities, thanks to its niching behavior, and fast convergence. The latter is probably justified by the nature of the problem that is such that *dominant* variables exist, which are repeatedly present in the relevant sets. In other words, K-means PSO, which converges extremely fast when solving optimization problems with separable functions, but struggles when dealing with strongly-dependent variables, can easily find those variables that are dominant across most groups and rapidly converges onto the most relevant groups that include them.

This makes it possible for K-means PSO to achieve a significant speed-up with respect to both the exhaustive search and HyReSS. Basically, the lower speed-up of the latter is partially due to the presence of the local search, which ensures a better exploration of the neighborhoods of the local minima. In the tests we have made, however, K-means PSO has not suffered from the lack of such a feature, except for the larger system analyzed (137 variables).

It is worth highlighting again that all methods rely on the same GPU implementation of the fitness function, which means that the observed differences depend only on the efficiency and complexity of the algorithms and not on their implementation.

4.2.4 Final remarks

Evolutionary and swarm intelligence algorithms are very good candidate solutions for extracting hidden relationships using the Relevance Index method, when the size of the system under consideration is large.

The work described in this section is mainly related with the study of custom versions of swarm intelligence algorithms to search for relevant sets as precisely and quickly as possible. In particular we employed K-means PSO for searching relevant sets in a number of complex systems, comparing the results obtained with such an algorithm with those obtained by HyReSS, a genetic algorithm-based metaheuristic, which has been described in section 4.1.

K-means PSO achieves a very good compromise between efficiency and precision in detecting the relevant sets. Compared to HyReSS, it appears to be generally quicker but slightly less precise, because of the presence, in the latter, of a local search step.

An extended description of the work described in this chapter can be found in [200, 219].

Chapter 5

The iterative sieving algorithm

In this chapter we present a methodology, based on the Relevance Index method, aimed at revealing the hierarchical dynamical structures hidden in complex systems.

The method iterates two basic steps: detection of relevant variable sets based on the computation of an RI metric, and application of a sieving algorithm, which refines the results. This approach is able to highlight the organization of a complex system into sets of variables, which interact with one another at different hierarchical levels, detected, in turn, in the different iterations of the sieve.

The method can be applied directly to systems composed of a small number of variables, whereas it requires the help of a custom metaheuristic in case of systems with larger dimensions. We have evaluated the potential of the method by applying it to three case studies: synthetic data generated by a non-linear stochastic dynamical system, a small-sized and well known system modelling a catalytic reaction, and a larger one, which describes the interactions within a social community, that requires the use of the metaheuristic. The experiments we made to validate the method produced interesting results, effectively uncovering hidden details of the systems to which it was applied.

5.1 Hierarchical structures in complex systems

Systems that exhibit complex behaviors are generally made up of elementary constituents interacting in a non-linear way. These constituents could be organized in a hierarchical structure [11, 220] as, for example, in biological systems (composed of cells that form tissues, which compose organs that compose organisms). However, very often complex systems are characterised by relations among components at different levels, so that the structure of their interactions cannot be represented by a clear tree-like topology, but rather by entangled hierarchies [13].

The identification of such structures involves the detection of the parts composing the system and their sub-components (up to a predefined level of granularity) and, possibly, the interactions among these components. Frequently, these structures need to be inferred by observing a system's dynamics, either in its unperturbed condition or after perturbations. The RI metrics seem suitable for exploring the organization of complex systems. The RI makes it possible to identify, as components of a system, relevant sets of variables that show an integrated behavior and interact more weakly with the rest of the system. Moreover, the zI metric, presented in section 3.3, shows that in several cases the sole integration can provide sufficient information to describe the organization of dynamical systems.

The RI method has been applied with interesting results to several systems: some of them had been artificially designed in order to test the effectiveness of the technique, while others referred to interesting physical, chemical, biological or socio-economic systems [191, 194]. In addition, the efficiency of the method can also be improved by using a parallel implementation of the RI computation (described in section 3.4) and some metaheuristics to deal with the "curse of dimensionality" when analysing high-dimensional systems (described in chapter 4).

In general, the method can be applied every time a collection of observations of the values of the system variables at different instants or conditions is available. This may be the case in off-line system analysis, but also in on-line analysis, when observations come from a data stream. In general, we suppose that information on the relations among the system's variables is not available; however, this method

can be successfully combined with community detection algorithms in complex networks [159].

In many cases the components identified by means of the RI have an intricate nested structure, which makes it hard to understand which groups of variables are really important. To overcome this problem, a filtering or sieving algorithm has been introduced [78], whose aim is to filter out any proper subset or superset of a reference variable set which has a larger RI value. By iteratively considering variable sets in descending order of index values and running the sieve, one can obtain an ensemble of disjoint or only partially overlapping variable sets, which can be considered the most relevant building blocks of the system. Since this algorithm is based on a well-defined criterion, user interpretation is not required for detecting the most relevant groups of variables after computing the RI for each possible subset. Nevertheless, at this stage, only the lowest-level components of a possible hierarchy can be uncovered. The proposed iterative application of the method, which will be described in details in section 5.3, is able to construct a hierarchical view of the system without any prior information about its structure, just by analyzing data that describe its dynamics.

Somehow, we could say that our method represents a way of giving a purely "objective" description of the structure of a complex system, based only on the observation of its states. In fact, our analysis anticipates any possible interpretation of its results that can be given *a posteriori*, as it is able to detect intrinsic dependencies among system variables or variable sets. Moreover, whenever the system can be observed in different operational conditions (or from different initial conditions), the bias depending on specific samples may be drastically reduced.

The following section describes in details the basic step of the iterative algorithm we propose: the sieving step. The methodology is presented using the T_c index, but can be also applied using the other RI metrics (e.g., the zI metric) for system analysis.

5.2 The sieving method

Whatever the method used to compute the index, the collection of RSs returned is likely to contain RSs included in others or partially overlapping, which requires fur-

ther analyses to assess their actual relevance. Indeed, having a high T_c value is not sufficient to characterise a RS, because such a value might result from the inclusion of a smaller set characterized by a higher T_c (i.e., the set under consideration is a superset of a more relevant one), in which case the only relevant set would be the latter. Conversely, a set having a high T_c value might reach an even higher value, if some other relevant variables are added to it (i.e., the set under consideration is a subset of a more relevant one), in which case we would consider only the larger set as relevant.

To tackle this problem, in [78] a post-processing sieving algorithm has been proposed to reduce the overall number of subsets. The main assumption of the procedure is that if a set A is a proper subset of B, i.e., $A \subset B$, then only the higher-T_c subset is taken into consideration. Therefore, only disjoint or partially overlapping subsets are kept. After this post-processing procedure, the remaining subsets cannot be decomposed any further, and thus they represent the building blocks of the dynamical organization of the system. The pseudo-code of the sieve is presented in Algorithm 8.

5.3 The iterative sieving method

The method previously described allows one to identify a plausible organization of the system in terms of its lowest-level, possibly overlapping, subsets of variables. Nevertheless, since complex systems have often a hierarchical structure, one may want to be able to make hypotheses on aggregated relations among the RSs thus identified, so as to derive a hierarchy of RSs. To this aim, we devised an iterative version of the sieving method, which acts on the data by iteratively grouping one or more RSs into a single entity. In fact, there are several ways to do so; the simplest one, yet quite effective, consists in iteratively running the sieving algorithm on the same data, each time using a new representation, in which the top-ranked RS of the previous iteration, in terms of T_c values, is considered as atomic and substituted by a single variable (henceforth called a *group variable*). In this way, each run produces a new atomic group of variables composed of both single variables and group variables introduced in previous iterations.

Suppose we have four variables denoted as A, B, C, D in the system and that

Algorithm 8 Sieving algorithm

Input: The array C of all the CRSs, ranked by their T_c value in descending order.
Output: RS, a subset of C
$RS \leftarrow \emptyset$
$n \leftarrow |C|$
Initialize auxiliary array $Del[k] \leftarrow \{0 \text{ for } k \text{ in } 1 \ldots n\}$
for $i = 1$ to $n-1$ **do**
  for $j = i+1$ to n **do**
    if $Del[i] \neq 1 \ \wedge \ Del[j] \neq 1$ **then**
      if $C[i] \subset C[j] \ \vee \ C[j] \subset C[i]$ **then**
        $Del[j] \leftarrow 1$
      end if
    end if
  end for
end for
for $i = 1$ to n **do**
  if $Del[i] = 0$ **then**
    $RS \leftarrow RS \cup \{C[i]\}$
  end if
end for

the group (AB) is the most relevant set detected by the first iteration of the algorithm. Then, the second iteration will analyse the dynamics of a system comprising the three variables (AB), C, and D, and so on, until the T_c value of the most relevant set detected falls below a pre-set threshold, which we usually set equal to 3.0.

As will be shown in section 5.3.2, this version of the iterative sieve is quite effective. However, other variants may be implemented, which may produce more than one group variable per iteration. In this latter case, instead of considering just the top-ranked RS as a new group variable, one may possibly transform the first q sets into group variables, with q chosen according to some empirical criterion.

The iterations of the sieving algorithm come to an end when the T_c of the top-ranked RS falls below a pre-set threshold (usually equal to 3.0), which means it is no longer possible to find new RSs that deviate significantly from the behavior of the reference system.

5.3.1 Implementation

The problem of finding the RSs of a complex system is a combinatorial optimization problem, whose size increases exponentially with the system dimension, soon making it infeasible to follow an exhaustive approach, in which T_c is computed for each possible subset of system variables. The computation of T_c itself is a rather lengthy procedure. Therefore, we tried to limit the computation time needed to run our method, on the one hand, by implementing the T_c computation algorithm as massively-parallel GPU code (described in section 3.4), while, on the other hand, by designing some Niching metaheuristics (described in chapter 4) to address the search toward the basins of attraction of the main local maxima.

The implementation of the main building blocks of the algorithm is modular and can be used to detect relevant hierarchical structures in complex systems using different RI metrics (e.g., the CUDA module for T_c computation can be replaced by an equivalent parallel module for zI computation).

5.3.2 Experimental results

In this section, we present three case studies analyzed by means of the proposed iterative RI method using the T_c index. In all the analyses performed on our test cases, we retain the highest-ranked RS of each sieve iteration, merge its variables into a new entity, treated from then on as a single variable replacing its component variables, and iterate the procedure until the algorithm reaches a stopping condition based on the T_c of the group detected in the last iteration.

The first two case studies have been chosen to validate the intrinsic properties of the method on systems for which a "ground truth" is available. In fact, the relationships between the variables of the first system have been hand-coded and the dynamical behavior of the second system is also very well known.

The first test case consists of two sets of data generated synthetically to assess the effectiveness of our method on a system which has clear and well-established dynamics, and compare its performance with classical pairwise correlation techniques.

The second example is a deterministic simulation of a chemical system (a Catalytic Reaction Network) described by 22 variables. Given its limited dimension, this system has been analyzed using an exhaustive search over all possible variable subsets.

The third case study regards the Green Community (GC) (see section 3.1.4), described by 136 variables. Given the size of the system, for which an exhaustive search is obviously infeasible independently of the available computational power, this case study has been analyzed using HyReSS metaheuristic, which has been described in section 4.1. For each iteration of the sieving algorithm, ten independent runs of the metaheuristic were performed, to take its stochastic nature properly into account and assess the repeatability of its results. For each iteration, the results provided by this algorithm were almost identical in all ten runs (only occasionally, in one of the runs, one of the 50 highest-T_c RSs was different from the ones detected in the other nine runs). For the smaller-size systems for which the comparison was possible (iterations of the sieving algorithm on systems described by fewer than 30 variables), the results were the same as those provided by an exhaustive search based on the same parallel code.

The parameters regulating the behavior of the metaheuristic were set as reported in Table 5.1.

Table 5.1: Settings of the parameters of the metaheuristic. The parameters are defined in section 4.1.

P_{mut}	p	α_f	α_p	β	γ	δ	θ
0.1	50176	501760	3	0.75	0.75	0.3	15

For all case studies taken into consideration, tests have been performed on a Linux server equipped with two Intel Xeon E5-2620v4 (2×8 cores, 2.1 GHz) CPUs, 64 GB RAM, two NVIDIA GeForce GTX 1070 GPUs.

Moreover, given the stochastic nature of the generation of the reference system, 100 independent runs with different random seeds were executed to assess the performance of the algorithm.

Execution times are summarized in Table 5.2 and results are discussed in the following subsections. The execution time on the Green Community case study has been computed considering only one run of the metaheuristic for each iteration of the sieving algorithm. For the synthetic case, we only report the execution times of the tests with $p = 0.5$, since different settings produce only slightly different results.

Table 5.2: Summary of the parameters of the analyzed systems and execution times of the sieving algorithm.

System	Variables	Samples	Time [s]
Artificial data 1 (p=0.5)	12	1000	7.81 ± 1.39
Artificial data 2 (p=0.5)	18	1000	18.37 ± 0.14
Catalytic Reactions Network	22	312	32.1 ± 0.4
Green Community	136	101	3904 ± 463

In the following subsections, we describe the three systems and provide an inter-

pretation of the results we obtained in our experiments.

Synthetic data

The systems we consider are composed of networks whose nodes are updated in terms of non-linear, possibly probabilistic, functions at discrete time steps. These systems can be considered as archetypal examples of more complex systems, because their interactions are examples of typical interactions occurring in real systems. At the same time, as we will see in the following, the more-than-binary nature of the relationships makes it impossible to detect the dynamically relevant parts of the systems by using classical pairwise correlation techniques: on the contrary, given also its ability to directly deal with large-size groups, our methodology is able to correctly identify the hidden structures.

Synthetic data 1: test case

The first system has been designed as a test case and is composed of three subsystems: $S = S_1 \cup S_2 \cup S_3$. Each of the first two subsystems is composed of a clique of three nodes, in which the state of each node at time $t+1$ depends on the state of the other two nodes at time t. The update function is an *exclusive OR* (XOR) of the two input nodes. This function is a prototypical example of a function that depends on both inputs. The third system, S_3, is composed of six independent nodes, each assuming value 0 or 1 according to a Bernoulli distribution with probability 0.5. In summary, the system is composed of two Boolean networks (BNs) immersed in a noisy environment. The Boolean Network framework has been described in section 3.1.1.
The dynamics of these BNs is synchronous and deterministic; therefore, after a short transient, the BNs settle into an attractor. In this case, each clique has four fixed points ($\{(000),(011),(101),(110)\}$), each with a basin of attraction of size two. As previously done in the case of BNs, we consider the attractors as representative of the dynamics of the whole system; the frequency of occurrence of each attractor in the data to be analyzed is proportional to its basin of attraction [241]. If S_1 and S_2 are independent, then we expect the method to be able to identify the two cliques,

distinguishing them from the random nodes. Conversely, if a dependence between S_1 and S_2 exists, then the system should still detect the two cliques, but it should also identify, in a further step, a superset including both BNs. This happens because their state space is strongly constrained with respect to the possible states assumed by the six random nodes, which, as such, are expected to have a negligible T_c. To perform our test, we produced data representing a collection of state values of S according to the following procedure:

1. the first three values ($S_1 = \{x_1, x_2, x_3\}$) are set by choosing at random one among the fixed points of the BN;
2. the fourth to the sixth values ($S_2 = \{x_4, x_5, x_6\}$)) are either an ordered copy of the previous values ($\{x_1, x_2, x_3\}$) with probability p, or they are set independently by choosing at random one fixed point with probability $1 - p$;
3. the remaining nodes ($S_3 = \{x_7, x_8, \ldots, x_{12}\}$) are independently set to 0/1 with probability 0.5.

Note that if $p = 0$, then S_1 and S_2 are independent, otherwise S_2 depends on S_1. The system is depicted in Figure 5.1.

A selection of representative results of the analysis of this first system is summarized in Table 5.3, for the case of independent cliques, and Table 5.4 for the case with probabilistic dependence between S_1 and S_2 with $p = 0.75$. We can observe that, in the case with $p = 0$, the method first detects S_1 and then S_2; in the third iteration, it groups S_1 and S_2 and adds an extra node, but the T_c value of this subset is very low. Indeed, in this case, after the second iteration we cannot expect any further meaningful clustering of relevant variables. It is worth observing that, since S_1 and S_2 are independent, in the first iterations the two sets of variables have a very similar T_c value. The case where S_1 and S_2 are dependent is quite trivial: the method already groups the variables composing the two systems in the very first iteration. We also generated data with $p = 0.25$, $p = 0.5$, and $p = 1$. In the cases with $p > 0.25$ results were qualitatively the same as in the case with $p = 0.75$, while, for $p = 0.25$, which represents a mild dependence, results were comparable to independence.

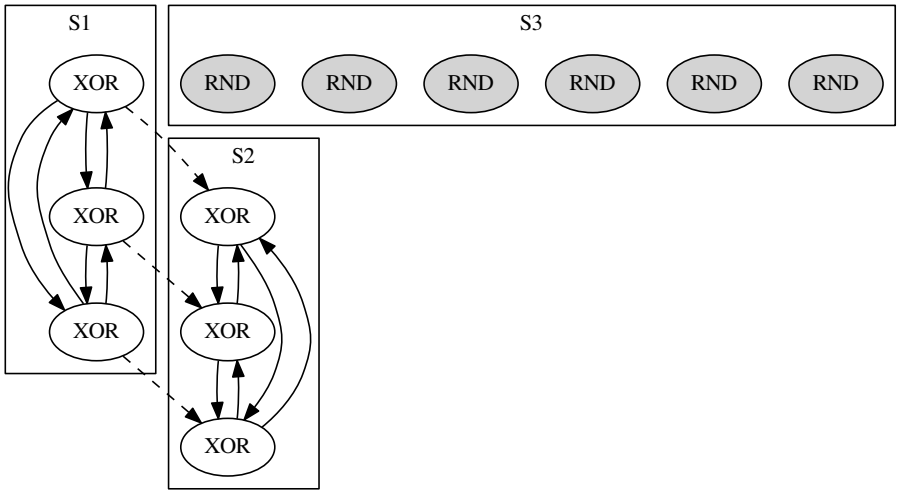


Figure 5.1: A system composed of three subsystems: S_1 and S_2 are two cliques with XOR functional dependencies. The nodes of S_2 depend on the nodes of S_1 with probability p. The third subsystem S_3 is composed of independent random Boolean nodes.

Synthetic data 2: system with chain of dependencies

We also produced a more elaborated variant of the previous system by introducing a further BN of the same kind as the previous ones. However, in this case, we imposed a gradual dependence among these three BNs according to the following procedure:

1. the first three values ($S_1 = \{x_1, x_2, x_3\}$) are set by choosing at random one among the fixed points of the BN;

2. the fourth to the sixth values ($S_2 = \{x_4, x_5, x_6\}$)) are set by choosing at random one among the fixed points of the BN with probability $1 - p$, while they are computed by XOR-ing two randomly chosen nodes in S_1 with probability p;

3. the same as for point 2 holds for $S_3 = \{x_7, x_8, x_9\}$, except that it depends upon

Table 5.3: *Test case:* RSs found in the main iterations of the sieving algorithm and corresponding T_c values with $p = 0$. At the end of each iteration, the RS with the highest T_c is grouped into a single variable for the next iteration. In iteration 1 we show also the second RS ranked. Note that, in iteration 3, the T_c of group $(S_1\ S_2\ x_{11})$ (lower than the chosen threshold of 3.0) makes it not relevant.

Synthetic data 1: independent cliques ($p = 0$)

Iteration	Relevant Set(s)	T_c
1	$x_1\ x_2\ x_3 \rightarrow (S_1)$	1517.01
	$x_4\ x_5\ x_6$	1454.52
2	$x_4\ x_5\ x_6 \rightarrow (S_2)$	684.923
3	$S_1\ S_2\ x_{11}$	1.644

the nodes in S_2;

4. the remaining nodes ($S_4 = \{x_{10}, x_{11}, \ldots, x_{18}\}$) are independently set to 0/1 with probability 0.5.

The results of the application of the sieving algorithm to these artificially-generated data deserve a detailed discussion. Table 5.5 summarizes the results of the case where variables are independent ($p = 0$). We can observe that the method first detects S_1, but the T_c of S_2 and S_3 is comparable. In the second iteration, it identifies S_2 and, subsequently, S_3. The algorithm is then able to identify the three subsystems and to distinguish them from the noisy environment because, in the fourth iteration, it groups all three systems together.

When a mild dependence is introduced ($p = 0.25$) we should observe a consequent dependence of S_2 upon S_1 and of S_3 upon S_2. This is indeed what the sieving algorithm returns, as shown in Table 5.6. It is important to remark that, with this low level of dependence, systems S_2 and S_3 are still detected as single entities, but are then grouped according to the chain of dependencies. This tendency starts to dilute for higher values of p up to $p = 1$, where there is a complete dependence of nodes in

Table 5.4: *Test case:* RSs found in the main iterations of the sieving algorithm and corresponding T_c values with $p = 0.75$. At the end of each iteration, the RS with the highest T_c is grouped into a single variable for the next iteration. Note that, in iteration 2, the T_c of the group $(S_1 + S_2\ x_9)$ (lower than the chosen threshold of 3.0) makes it not relevant.

Synthetic data 1: dependent cliques ($p = 0.75$)

Iteration	Relevant Set	T_c
1	$x_1\ x_2\ x_3\ x_4\ x_5\ x_6 \ \rightarrow\ (S_1 + S_2)$	1524.96
2	$S_1 + S_2\ \ x_9$	2.769

S_i from nodes in S_{i-1} (with $i = 2, 3$). Results for $p = 1$ are summarized in Table 5.7, where we can observe that S_1 is identified first; then, single nodes in S_2 are iteratively added until they form the set $S_1 + S_2$. Subsequently, the nodes in S_3 are added so as to group all nine variables in the BNs. We emphasize that the sieving algorithm correctly identifies and combine these groups according to the chained dependence among BNs we have introduced.

Finally, to assess the effectiveness of the method, we also analyzed these two sets of data by performing a hierarchical clustering based on the computation of the paired Pearson correlation between variables, as typically done when networks are analyzed. As expected, this approach did not discover any relevant set, because the main relations involve more than two variables. We are not claiming that our method outperforms any other method based on correlations, but just that, by its nature, it is able to capture relations involving multiple variables. Moreover, the nature of the two methodologies is completely different. Indeed, Pearson Correlation deals with quantitative variables, while the RI analysis is based on qualitative values (the values associated to the variables are represented as labels).

The successful outcome of the application of the sieving algorithm to these artificially-built datasets provides evidence to the effectiveness of the method. Nevertheless, its

Table 5.5: *Chain of dependencies case:* RSs found in the main iterations of the sieving algorithm and corresponding T_c values with $p = 0$. In iteration 1 we also show the RSs ranked second and third. At the end of each iteration, the RS with the highest T_c is grouped into a single variable for the next iteration.

Synthetic data 2: independent cliques ($p = 0$)

Iteration	Relevant Set(s)	T_c
1	$x_1\ x_2\ x_3 \ \rightarrow\ (S_1)$	838.024
	$x_4\ x_5\ x_6$	836.677
	$x_7\ x_8\ x_9$	831.635
2	$x_4\ x_5\ x_6 \ \rightarrow\ (S_2)$	664.485
3	$x_7\ x_8\ x_9 \ \rightarrow\ (S_3)$	576.078
4	$S_1\ S_2\ S_3$	7.013

Table 5.6: *Chain of dependencies case:* RSs found in the main iterations of the sieving algorithm and corresponding T_c values with $p = 0.25$. At the end of each iteration, the RS with the highest T_c is grouped into a single variable for the next iteration. Note that, in iteration 6, the T_c of the group detected (lower than the chosen threshold of 3.0) makes it not relevant.

Synthetic data 2: $p = 0.25$

Iteration	Relevant Set	T_c
1	$x_1\ x_2\ x_3 \ \rightarrow\ (S_1)$	815.723
2	$x_4\ x_5\ x_6 \ \rightarrow\ (S_2)$	325.067
3	$x_7\ x_8\ x_9 \ \rightarrow\ (S_3)$	152.263
4	$S_1\ S_2 \ \rightarrow\ (S_1+S_2)$	17.209
5	$S_1+S_2\ \ S_3 \rightarrow\ (S_1+S_2+S_3)$	9.330
6	$S_1+S_2+S_3\ \ x_{16}$	2.786

Table 5.7: *Chain of dependencies case:* RSs found in the main iterations of the sieving algorithm and corresponding T_c values with $p = 1$. At the end of each iteration, the RS with the highest T_c is grouped into a single variable for the next iteration.

Synthetic data 2: $p = 1$

Iteration	Relevant Set	T_C
1	$x_1\ x_2\ x_3 \ \rightarrow\ (S_1)$	900.645
2	$S_1\ x_4\ x_6$	267.988
3	$S_1 + x_4 + x_6\ \ x_5 \ \rightarrow\ (S_1 + S_2)$	122.419
4	$S_1 + S_2\ \ x_8$	50.414
5	$S_1 + S_2 + x_8\ \ x_7$	23.408
6	$S_1 + S_2 + x_8 + x_7\ \ x_9 \ \rightarrow\ (S_1 + S_2 + S_3)$	12.222

potential should be expressed on more complex cases, which are the subject of the following two sections in which data from a Catalytic Reaction Network and the Green Community data are analyzed.

Catalytic Reaction Network

As stated in section 3.1.3, the formation of groups of molecules able to collectively self-replicate is thought to be fundamental for the origin of life [58, 62, 63, 75, 112, 124, 196] and is likely to play an important role also in future bio-technological systems [221].

Many attempts have been made to determine the chemical arrangements that allow sustainable self-maintaining behaviors, one of the currently most sophisticated being probably Reflexive Autocatalytic Food-generated (RAF) sets [107, 108], recently utilized also in biochemical contexts [76, 107, 109, 234] or in protocell architectures [214].

On the other hand, efforts have been made to identify the relevant chemical species involved in real or artificial complex chemical reaction schemes [8, 233].
Typically, the systems we have analyzed are immersed in Continuous-flow Stirred-

Tank Reactors (CSTRs) [175], featuring a constant influx of feed molecules[1] and a continuous outgoing flux of all the molecular species proportional to their concentration. In this case, as the typical attractors are fixed points, which do not provide any useful information for computing the RI, a different approach has been followed, which consists in perturbing the fixed points and recording the transient.

By taking advantage of the aforementioned perturbative approach, we apply the iterative sieving to the data series coming from the perturbations imposed on the simulation of chemical arrangements. This allows us to investigate the effectiveness of the RI method in identifying RSs in systems where several reactions take place simultaneously, using only the concentrations of the various chemical species as input, and without any prior knowledge about the reaction graph. The simulations are based on a relatively simple system inspired by a model used in [70, 73, 74] and originally proposed by Kauffman [123, 124].
The analyzed scheme involves enzymatic condensations, whose process is considered as being composed of three steps: the first two create (reversibly) a temporary complex (composed by one of the two substrates and the catalyst) that can be used by a third reaction, which combines the complex and a second substrate to finally release the catalyst and the final product. The aforementioned three steps are summarized as follows:

1. Complex formation: A+C $\xrightarrow{Ccomp}$ A:C
2. Complex dissociation: A:C $\xrightarrow{Cdiss}$ A+C
3. Final condensation: A:C+B $\xrightarrow{Ccond}$ AB+C

where C_{comp}, C_{diss} and C_{cond} are respectively the reaction kinetic constants of complex formation, complex dissociation and final condensation. The dynamic of the systems is described adopting a deterministic approach, whereby the reaction scheme is translated into a set of Ordinary Differential Equations ruled by the mass

[1] The feed molecules in CSTRs are constantly present and therefore play the role of the "food" species constituting the base of RAF arrangements.

action law (see [214] for further details) and integrated by means of a custom Euler method with step-size control.

The main entities of the model are molecular species ("polymers"), represented by linear strings of letters (A, B, C, D). In the example of Figure 5.2, they form a catalytic reaction system composed of seven distinct condensation reactions, divided into two distinct RAF pathways: a chain of linear reactions (RAF1), the presence of whose root is guaranteed from the outside, and a RAF where two reciprocally catalyzing reactions are the roots of another linear reaction chain (RAF2).

As mentioned before, the asymptotic behavior of this kind of systems is a single fixed point [8], due to the system feedback structure. In order to apply our analysis, we need to observe the feedbacks in action. Therefore, we perturbed the concentration of some molecules in order to trigger a response in the concentration of (some) other species. Therefore, we temporarily lowered, one by one, by two orders of magnitude, the input concentrations of the food species (coloured ellipses in the example of Figure 5.2) after the system reached its stationary state[2]. In order to analyze the system response to perturbations, we used a three-level coding where, for each species, the digits $0-1-2$ stand for "concentration decreasing", "no change" and "concentration increasing", respectively[3]. In practice, the time series is obtained by computing (and then properly coding) the sign of the difference between two consecutive samples of the original data. Note that, in order to better observe the dependencies of the system, we set an observation frequency high enough to allow several observations during the transient situations[4].

In Figure 5.2 we report the most salient steps of the analysis, while the T_c values computed for the main groups are reported in Table 5.8.

[2]Note that we could also simulate the temporarily disappearance of the chemical species inside the CSTR vessel: in this case (i) the grouping process would be different (a consideration that highlights the fact that the perturbation itself is a dynamical process, with a significant influence on the final observations) and (ii) the identification of the chemical structures would clearly be easier. However, this procedure is not feasible in real experiments.

[3]In this experiment, we consider the concentration of a chemical species as being constant if it has not changed by more than 1% in a time period of 10 seconds.

[4]In other words, the transients are "smooth".

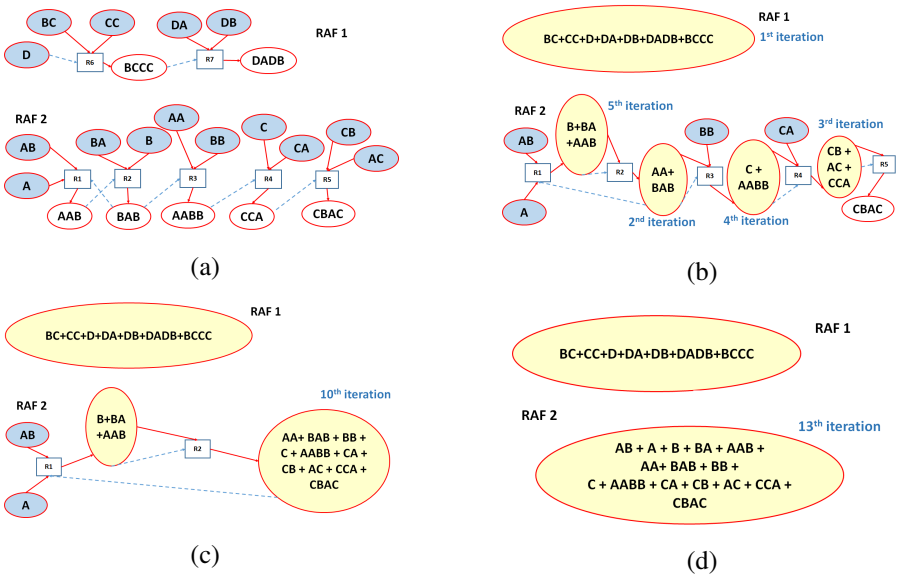


Figure 5.2: (a) The chemical system under analysis in the CSTR case study. Elliptic nodes represent chemical species: the ones filled in blue stand for those injected into the CSTR (food species), while the empty white ones are the more complex species built by specific condensation of the food species. Rectangular shapes represent reactions, where incoming arrows are oriented from substrates to reactions and outgoing arrows from reactions to products. Dashed lines indicate the catalytic role of a particular molecular species within the specific reaction context. The kinetic constants of all reactions which take place have the same value (C_{comp}=5000 $mol^{-1}s^{-1}$; C_{diss}=250 s^{-1}; C_{cond}=500 $mol^{-1}s^{-1}$); the incoming concentration of each food species is 0.01 *mol*, whereas, every second, 5% of the CSTR volume is renewed. (b) The five RSs found after the first five iterations of the iterated sieving algorithm. (c) The situation at the end of the iterated sieving algorithm. Note, however, that, if the method is further iterated (d), despite finding groups with significance below the chosen threshold (corresponding to a critical value of T_c close to 3.0), the iterative sieve detects the correct configuration anyway.

Table 5.8: RSs found in each iteration of the sieving algorithm and corresponding T_c values for the CSTR case study. The last three iterations are separated from the others because the T_c values of the detected RSs are lower than the threshold of 3.0.

Iteration	Detected Relevant Set	T_c
1	$[BC][CC][D][DA][DB][DADB][BCCC]$	305.711
2	$[AA][BAB]$	33.682
3	$[CB][AC][CCA]$	28.616
4	$[C][AABB]$	26.048
5	$[B][BA][AAB]$	24.293
6	$[C+AABB][CA]$	14.236
7	$[CB+AC+CCA][CBAC]$	10.762
8	$[C+AABB+CA][CB+AC+CCA+CBAC]$	11.535
9	$[AA+BAB][BB]$	5.213
10	$[AA+BAB+BB]$ $[C+AABB+CA+CB+AC+CCA+CBAC]$	4.484
11	$[B+BA+AAB][AA+BAB+BB+C+$ $AABB+CA+CB+AC+CCA+CBAC]$	2.953
12	$[A][B+BA+AAB+AA+BAB+BB+C+$ $AABB+CA+CB+AC+CCA+CBAC]$	0.957
13	$[AB][A+B+BA+AAB+AA+BAB+BB+$ $C+AABB+CA+CB+AC+CCA+CBAC]$	0.261

The entire linear reaction chain (RAF1) is immediately detected whereas, within the more complex RAF2 organization, the other groups highlight the strict relations among the reagents and the catalyzer of the same reaction (Figures 5.2.b and 5.2.c). The time series is too short to permit a complete detection of the whole RAF2 group (Figure 5.2.c) with sufficiently high significance; however, in the subsequent iterations of the method, although groups with significance below the chosen threshold (corresponding to a critical value of T_c equal to 3) are found, we can actually detect

the correct configurations. Indeed, we noticed that, if the time series length is artificially duplicated, an increased T_c value can also be computed for these groups, which confirms the correctness of their detection. All data are represented in Table 5.8, which displays the T_c values of the RS detected in each iteration of the sieving algorithm.

Note that the relatively long chain in RAF2 (composed by reactions R2, R3, R4 and R5) is "discovered" by our approach starting from the final part of the tail, and subsequently "going upward" towards its head (Figures 5.2.c and 5.2.d). This effect is due to the perturbations on the "last" food species of the chain (e.g., CA, CB or AC), which heavily affect the final part of this chain. However, their effects cannot propagate towards the species (e.g., BAB or AAB) that are located "upward" along the chain. On the contrary, perturbations on BA, B or AA heavily affect the initial part of this chain, as well as its final part — the higher the distance from its initial source, the weaker the effect. This attenuation process (observed also in [233]) induces a dynamical hierarchy on the chain system, which permits the fine subdivision highlighted in Figure 5.2. The same phenomenon is not observable in RAF1, on the one hand, because of the small size of the chain and, on the other hand, because the perturbations hit the root of the chain directly, causing strong and evident effects along the whole (short) structure. We remind that the "root" of RAF2 is composed of two reciprocally catalyzing reactions: indeed, this strong dynamical union permits interesting resilience effects [108, 214, 234].

Green Community

In this subsection, we examine the Green Community (GC) case study, which has been presented in section 3.1.4. This set of data comes from a project which initially involved only four mountain communities, dealing with a core topic about energy efficiency and renewable energy production-related issues. Later, it was extended to other social and organizational themes, gradually involving many heterogeneous stakeholders.

In order to observe the presence of (formal or informal) coalitions within the four mountain communities, we decided to extract from this database only the data

about the involved stakeholders' attendance (or absence) at some significant points-of-control (e.g., meetings) of the GC project. Therefore, following an indication presented also in section 4.1, we obtained a very sparse matrix, composed of 136 variables and 101 points-of-control (observations), as shown in the upper part of Figure 5.3.

Analyzing this matrix, we were able to infer some insightful indications about the formation of coalitions during the GC project.

However, considering the subject of the present section, we simply report the following observations:

- The iterated sieving procedure automatically stopped after 26 iterations[5], resulting in the final organisation shown in the second part of Figure 5.3.

- The groups exhibiting the simplest behaviors are composed of stakeholders (the system "variables" or "agents" in the following) present at only one event of the GC project. In fact, some events may have a restricted list of stakeholders that are allowed to participate in it: this piece of information is indeed a significant part of the process itself, and forces particular forms of coordination among agents (it excludes the agents that wish to participate in a particular event but do not have the correct permission, and could also force the presence of agents that would not participate). On the other hand, the detection of these "simple" groups is a confirmation of the correctness of the RI procedure when dealing with real-world data.

- Groups D and B, despite their illusory simplicity, are not so obvious: in particular, the explicit recognition of group D (and group B) indicates the existence of a distance between the variables belonging to these groups and other variables that, in spite of exhibiting similar behaviors, belong to other groups.

- The large group named G is composed of several large sub-groups which, in turn, may be composed of other smaller RSs. Some of these groups include very active variables (in particular group G7, which includes the head of the

[5] When a T_c lower than the threshold of 3.0 characterized the last detected RS.

GC project and the two social researchers involved in the observation of the whole project), whereas other groups, despite the relatively low activity of their members, are dynamically very heterogeneous (e.g., group G2).

Therefore, the detection of the most obvious groups, whose correctness was confirmed by the social researchers that collected the data, shows that the iterative RI procedure correctly works in analyzing the GC case. Moreover, the less obvious groups have been considered very "interesting" and sometimes "enlightening" by these specialists.

In any case, the final comment of the social specialists involved in the project was that "the RI methodology could constitute a very interesting *dashboard* potentially able to effectively support the fieldwork of the observers".

As a final observation, we can notice how even a measurement as simple as the recording of the presence/absence in (formal or informal) project meetings can result in the detection of very sophisticated groupings or hierarchies.

5.3.3 Final remarks

In this chapter, we have formally introduced a methodology able to support a wide application of the RI method. The technique we propose realizes a sieving action that is performed iteratively until a certain threshold in the T_c value is reached, and permits to group together variables (or sets of variables) of a complex system, which are detected as the most relevant by the RI method.

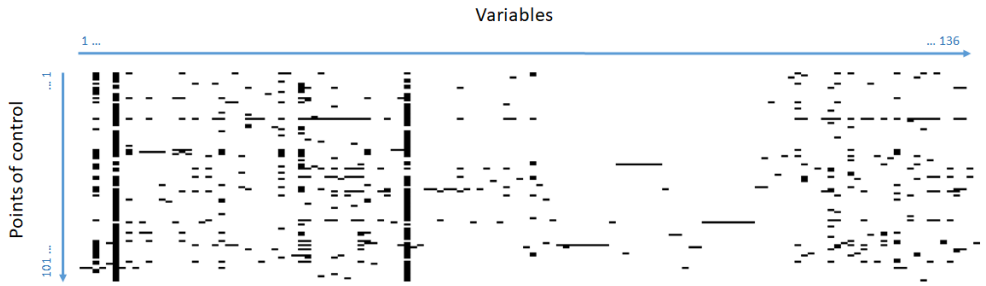
The iterated sieving algorithm aims at reducing the overall number of subsets found by the RI method. This is done by keeping only disjoint or partially overlapping subsets of variables, which means that only the subsets having the highest RI (e.g., T_c index) are taken into consideration in defining the architecture of the whole complex system. In the end, this appears to be built upon variable subsets that cannot be decomposed any further and represent the actual building blocks of the system.

The proposed approach has proven to be able to extract hidden information about the organization of the three complex systems we have analyzed.

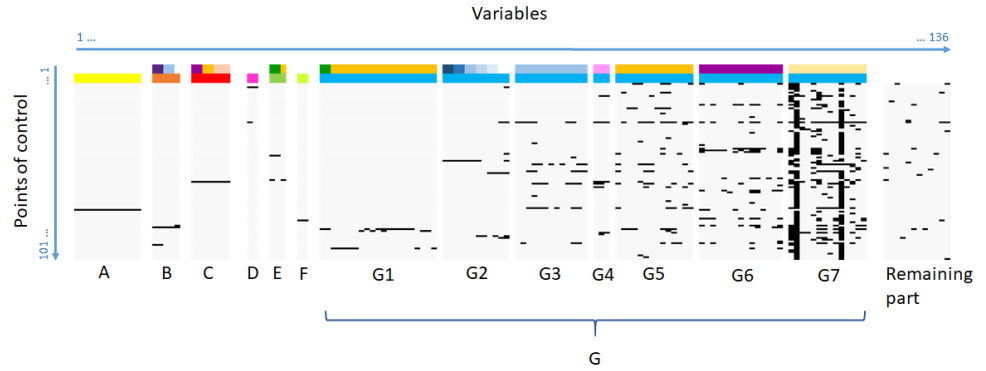
As we will show in chapter 6, the RI method, enriched with the sieving capabil-

ity, can be applied to several kinds of complex systems: social networks, biological networks, or socio-technological systems.

An extended description of the work described in this chapter can be found in [243].



(a)



(b)

Figure 5.3: (a): the temporal behavior of the 136 variables over the 101 points-of-control, which are distributed over time without a regular frequency. The involved stakeholders' attendance at the points-of-control of the GC project is marked in black (the absence is marked in white).
(b): the same variables, re-sorted in order to highlight the RSs found by the RI algorithm. The two colored bars highlight: (i) the different "final" groups (lower colored bar), and (ii) the internal subdivisions of the final groups (the smallest RSs detected during the 26 iterations of the sieving procedure). Group G was obtained by grouping smaller RSs: for easier graphical representation the exact multi-step sequence of the assembly has not been represented. The same observations holds for the assembly processes in general: for example, the steps leading to grouping RSs B, C, E, G1, G2, and G3 are intertwined, while groups G4 and G3 are formed "before" the final expression of group G1. The other variables (termed as the "remaining part") could not be assigned to any group with a sufficiently high degree of significance.

Chapter 6

Applications

As the previous chapters have shown, the RI methodology, combined with niching metaheuristics and with the sieving algorithm, is able to provide an effective description of different kinds of systems and can be applied to a broad range of non-stationary dynamical systems.

This chapter presents some applications of the RI methodology in different contexts. In particular, sections 6.1 and 6.2 describe the analysis of biological systems (T helper network and cancer evolution models, respectively), while section 6.3 presents an RI-based approach for community detection in social networks.

6.1 A RI Method to infer global properties of biological networks

This section shows how the application of the RI approach can lead to novel and effective interpretations of a biological network (T helper case study).

Nowadays, a plethora of molecular data results in a vast amount of pathways, networks of interactions and molecular scenarios. A large quantity of information is available on many biological systems, and researchers use it to infer global properties of biological networks [114, 189]. In spite of the strong representational power and flexibility of networks, there are, however, two major limitations which affect most

studies in the field [118, 214]:

- the information about the actual underlying interactions is often incomplete, so the inferred networks do not provide a complete picture of the interactions in the system under study;
- network studies are often concerned with "static" topological information, like connectivity and betweenness, whereas, in order to understand the functionality of a system, it is important to study its *dynamical properties*.

Modeling the dynamic behavior of such systems is difficult, due to the lack of kinetic data and to computational limitations. Among the methods for facing this problem, those based on steady-state approximations are widely used [104, 197]. Nevertheless, these kinds of analyses do not provide enough constraints to find a unique solution to the problem: thus researchers support these techniques by means of suitable hypotheses as, for example, minimization or maximization issues [197]. This drawback, in terms of modeling, has turned out to be particularly relevant when controlling the steady-state behavior of complex networked dynamical systems. In this respect, some efficient model-free methods based on multi-agent reinforcement learning [53] and on mean-field game theory [42] are rapidly emerging in several domains, such as telecommunications.

In order to overcome the aforementioned limitations of steady-state methods, it is worthwhile to resort to methods able to directly deal with the dynamical repertoire of the system. The RI approach seems suitable for this task, since it is characterized by the following features:

1. it is based on the observation of the dynamical states of the system (whether simulated or real), without requiring any *a priori* knowledge of the interactions among variables (whenever such knowledge is available, it can be used to complement the proposed method);
2. it can be applied to states coming from different steady state conditions, or even to states obtained from perturbation of these conditions (it does not require fixed asymptotic states);

3. it provides information about the organization of the system itself; indeed, complex systems often display complex organizational features that cannot be captured by a simple tree-like structure;

4. it is robust against noisy or incomplete data, being based on information-theoretic measures.

In this section we show that (i) the dynamically relevant groups of variables identified using the RI method in a biological network are extremely useful for describing the overall dynamics of the system and that (ii) this description can significantly enhance the explicative power of the graph description of a biological system, by highlighting the links that are actually effective.

Given the small size of the analyzed biological system, we use the NORTA method (presented in section 3.2) to take in account the non-zero pairwise correlations among the variables of the homogeneous system.

The following subsections describe the context of our analysis and the experimental results we obtained.

6.1.1 Context

In many natural systems, the most interesting recurrent patterns of interaction take place very often at levels that can be regarded as intermediate between pre-existing layers, which are, in turn, affected by the dynamics of these patterns. There are several examples of these "sandwiched" phenomena in physics and biology. Thus, the detection of intermediate-level structures and patterns is a very central issue to understand the dynamics of complex natural systems.

Many interesting systems can be represented, at least partially, by means of graphs. In this case, a widespread property is the presence of the so-called communities, portions of system elements within which the connections are dense, but between which they are sparser [161]. Their identification sometimes could detect groups that can be good relevant set candidates. Networks, however, have representational limits [118, 193]; moreover, we look for structures and patterns that can be observed while looking at the dynamics of the system, not at their static behavior.

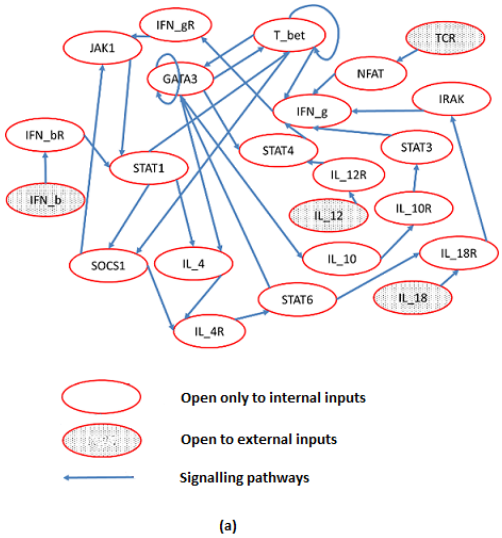
A method that mixes static and dynamical issues was proposed by Thomas et al. [229, 230] for regulatory networks, the focus of the present work, to capture the main qualitative features of the dynamics of such systems.

Works that use dynamical features in order to detect functional groups are not so frequent; many of them rely on similarity measures and clustering algorithms. This is what is done by Feldt et al. [72], for example.

An interesting dynamical approach uses methods introduced in information theory and applied in neurosciences by Edelman and Tononi in 1994 and 1998 [231, 232] to detect functional groups of brain regions (see section 2.1.1). This approach is the reference method from which the RI metrics derive, as described in section 2.1.2. The approach was extended to non-stationary dynamical regimes, in order to apply the method to a broad range of systems, including abstract models of gene regulatory networks and simulated chemical [240], and biological [241] systems.

The present section applies the RI methodology to the study of biological networks (regulatory networks). An interesting literature review about the reconstruction of gene regulatory networks and the development of mathematical models of how the patterns of activation and inhibition determine the state of activation of the network can be found in [50]. The T helper regulatory network considered in the present section is based on the one described in [149].

The T helper cell differentiation system The vertebrate immune system is composed of several cell populations, including antigen-presenting cells, natural killer cells, and B and T lymphocytes. There are two main kinds of T lymphocytes: the T cytotoxic cells that actively destroy virus-infected cells and tumor cells and the T helper cells (Th) that take part in cell- and antibody-mediated immune responses by secreting various cytokines, differently distributed in the two main T helper cell subtypes Th1 and Th2. Both sub-types derive from a common precursor Th0 through a rather complex differentiation path, modeled in [149, 190]. In this section, we use the discretization of an updated version of these paths described in [149] (Figure 6.1). The nodes TCR, IL_{18}, IFN_b and IL_{12} receive their input from outside the Th differentiation system and constitute the way the system is aware of its context (in other



$$GATA3_{t+1} = (GATA3_t + STAT6_t) \cdot T\_bet_t'$$
$$IFN\_bR_{t+1} = IFN\_b_t$$
$$IFN\_g_{t+1} = (IRAK_t + NFAT_t + STAT4_t + T\_bet_t) \cdot STAT3_t'$$
$$IFN\_gR_{t+1} = IFN\_g_t$$
$$IL\_10_{t+1} = GATA3_t$$
$$IL\_10R_{t+1} = IL\_10_t$$
$$IL\_12R_{t+1} = IL\_12_t$$
$$IL\_18R_{t+1} = IL\_18_t \cdot STAT6_t'$$
$$IL\_4_{t+1} = GATA3_t \cdot STAT1_t'$$
$$IL\_4R_{t+1} = IL\_4_t \cdot SOCS1_t'$$
$$IRAK_{t+1} = IL\_18R_t$$
$$JAK1_{t+1} = IFN\_gR_t \cdot SOCS1_t$$
$$NFAT_{t+1} = TCR_t$$
$$SOCS1_{t+1} = STAT1_t' + T\_bet_t$$
$$STAT1_{t+1} = IFN\_bR_t + JAK1_t$$
$$STAT3_{t+1} = IL\_10R_t$$
$$STAT4_{t+1} = IL\_12R_t + GATA3_t$$
$$STAT6_{t+1} = IL\_4R_t$$
$$T\_bet_{t+1} = (STAT1_t + T\_bet_t) \cdot GATA3_t'$$

(b)

Figure 6.1: (a) A graph representation of the Th differentiation system. Note that all the gray-filled nodes (TCR, IL_18, IFN_b, and IL_12) do not receive their input from the network regulating the differentiation system. Thus, in this representation, they do not have incoming links. (b) The dynamical rules of the Th differentiation system as described in [149].

words, they constitute the system "sensors"). Several signalling pathways are stimulated by their activation [111].

6.1.2 Experimental results

We simulated the gene regulatory network described in Figure 6.1 by means of a synchronous Boolean system. There are 2^{19} different initial conditions for each of the 2^4 different scenarios identified by the "sensor" nodes. However, we found only 33 different asymptotic behaviors (all fixed points). Three of these attractors coincide with the gene expression of Th0, Th1 and Th2 cells. These attractors are presented in [149] as the only really stable states, according to the information derived from the application of the so-called generalized logical analysis [230] to the Th differentiation system.

The limited number of different asymptotic behaviors should give us some information about the dynamical organization of the system[1]. Therefore, to extract this information, we tried to apply the RI methodology (i) to the mere juxtaposition of these attractors or (ii) by weighting their presence proportionally to the size of their basins of attraction, i.e., the width of the neighborhood from which the system converges into the state represented by the attractor under consideration.

In both cases the relevant subsets that were found are composed by TCR and NFAT nodes (Group1 in Figure 6.2) and all the other nodes (Group2 in Figure 6.2)[2]. This fact indicates that the Th differentiation machinery is indeed highly integrated. We can register the presence of these two first CRSs and successively filter them out,

[1] In this context we do not make hypotheses about the biological plausibility (or stability or biological function, if any) of these attractors, suggesting the interested readers to refer to Mendoza and Xenarios [19] and to the references quoted therein. Rather we highlight that, once a mathematical model has been established, its structure implies the presence of a well-defined set of attractors: so, an analysis that takes into account their presence (and therefore which highlights their interrelated dynamical relationships) should provide better results than a method that does not act in this way.

[2] The node JAK1 is constantly inactive in all attractors. Thus, its presence is useless for the purposes of a dynamical analysis and no CRS include it. Indeed, it is active in transient states, but this kind of analysis is out of the scope of the work described in this section (see [194] for a first comparison of the results of RI application to transients and asymptotic states).

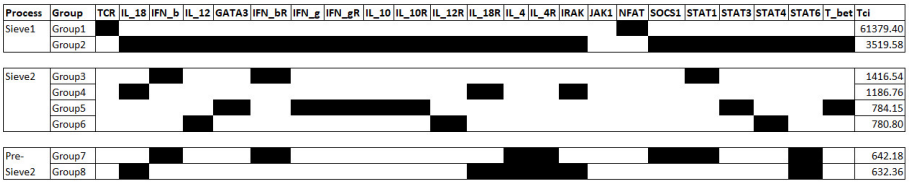


Figure 6.2: The groups detected by the application of the RI methodology followed by the sieving algorithm (groups 1-6): each group is represented as a row where black boxes denote the variables belonging to it. Group7 and Group8 have been discarded by the sieving algorithm, because they include the stronger relevant subsets indicated as Group3 and Group4: however, their observation is important, because it traces a significant coupling among Group3 and Group4 and the other system variables. Indeed, a second application of the iterated RI method fixes this strong association (data not shown).

in order to apply the sieving algorithm to all the remaining groups. In this case, the two approaches produce different results.

The simple attractor juxtaposition separates Group2 into two big subsets (see Figure 6.3, left), whereas the application of RI to an extended set of observations obtained by repeating input data related with the 33 attractors a number of times proportional to the width of their basins of attraction is able to identify (i) the four chains that transmit the external signals toward the inner core of the Th differentiation system (the TCR-NFAT chain, i.e., the Group1, already identified during the first RI application) and (ii) a "circle" of nodes that appears to be the "dynamical engine" of the Th differentiation system, denoted as Group5 (Figure 6.3, right).

It appears that nodes SOCS1, IL_4, IL_4R, and STAT6 do not belong to any relevant subsets (Figure 6.3, right), if we strictly adhere to the relevant subset definition. However, before the application of the sieving algorithm, the RI analysis reports two highly-ranked groups in the top positions, namely Group7 (composed by the aforementioned nodes and by Group3) and Group8 (composed of IL_4,IL_4R, STAT6, and by Group4). Indeed, these two groups are discarded by the sieving algorithm be-

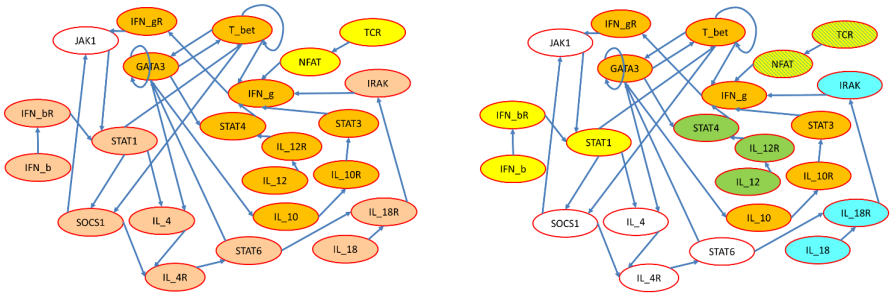


Figure 6.3: The main relevant subsets identified using the simple juxtaposition of the attractors of the Th differentiation system (left) or by weighting their presence proportionally to their basins of attraction (right). In the right part of the figure we highlight the presence of Group1, Group3, Group4, Group5, and Group6, respectively in striped, yellow, blue, orange, and green background.

cause they include two already identified and slightly stronger relevant subsets. Vice versa, we can use this information in order to identify the nodes influenced by (or influencing) Group3 and Group4. Thus, given the directions of the links of the Th system, it appears that the information acquired by Group3 (in particular by node IFN_b) is transmitted to the nodes belonging to the "white group", which, in turn, passes it to Group4. Therefore, the white group is composed by elements that seem to act as a sort of "transmission engine" for the Th differentiation system. Figure 6.4 highlights such an information flow from the "yellow" region (group 7) to the "blue" region (group 8).

The RI analysis therefore induces an interesting interpretation of the dynamical data which, when mapped on the already available topological knowledge, provides an expressive explanation of the system functioning. The same knowledge (the identification of groups of variables and of their relationships) is not derivable from the static analysis alone. The usual algorithms for the search of communities [120, 161] identify only the pair GATA3-T_bet. Moreover, only one of the identifiable 27 circuits is highlighted (Group5, which involves nodes T_bet, GATA3, IL_10, IL_10R,

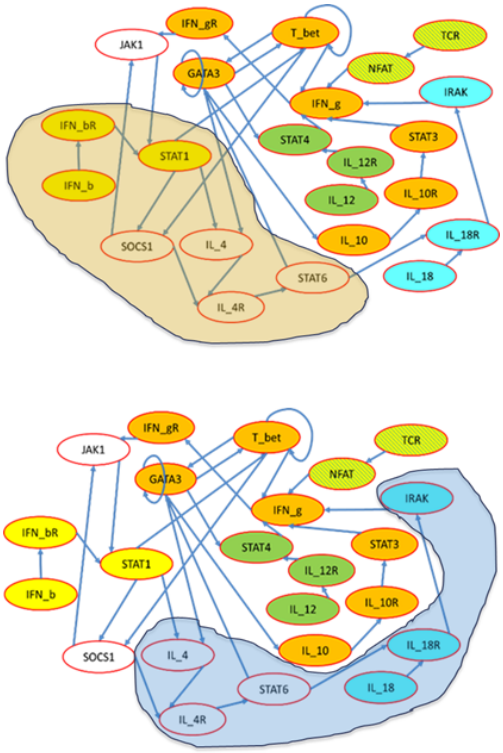


Figure 6.4: Same as Figure 6.3, but highlighting the correlation of Group3 and Group4 with other Th differentiation variables (SOCS1, IL_4, IL_4R and STAT6 – for brevity indicated in this caption as "WhiteGroup"). With reference to the table reported in Figure 6.2, one can see that, indeed, the first (and only) significant appearance of these variables as a block occurs along with Group3 and Group4, with which they compose Group7 and Group8, as shown by the third block of results in the table. Given the directions of the links, in this example assumed to be known, it appears that the graph structure of the system could enable the signal transmission from Group3 and Group5 to the WhiteGroup (first row). However, the RI index indicates as evident only the influence of Group3. In turn, the information acquired by the WhiteGroup from Group3 is transmitted to Group4, in such a way modulating the external signals coming from node IL_18 (second row).

STAT3, IFN_g, IFN_gR, JAK1 and STAT1).[3]

On the other hand, the usual dynamical analyses are mainly focused on the detailed reproduction or prediction of the system's behaviors [149] and therefore are not suitable for a highly abstracted and "global" vision of the system functioning. The same generalized logical analysis [230] that mixes topological and dynamical issues identifies chains of positive and negative feedbacks, eventually providing clues for the identification of stable attractors, but does not give the overall vision of the RI method, which identifies the genes involved in injecting information into the system (the groups 1, 3, 4 and 6) and the main circuit responsible for information processing (group5).

Obviously, this method cannot be used to reconstruct the detailed topology of the investigated system (though it could suggest useful groupings). It is worth mentioning, however, that the RI method can be applied directly to the experimental data, if these are available. In this respect, we can notice that, while the collection of time series is an experimentally difficult and costly task, the RI methodology can be applied by merely comparing different steady states (whose data could derive even from different beings), taking advantage of more common data sources. In case experimental data are available, the RI method can provide an effective idea of the dynamical organization of the observed system without requiring any knowledge of topology, dynamical rules, or parameters.

6.1.3 Final remarks

In this section, we proposed to use the RI method, improved through a novel technique for computing the correlation matrix of the homogeneous system (presented in section 3.2), as a mean to infer global properties of biological networks. The objective is twofold and encompasses both finding new insights about those systems and refining the method itself.

With respect to steady-state approximation approaches, the RI method, which is based on the observation of the dynamical states of the system, provides information

[3]Note that the node STAT1 participates in Group 3, one of the "sensors groups" of the Th differentiation system.

about the organization of the system itself and is robust against noisy or incomplete data, being based on information-theoretic measures. The RI method can be applied directly to the experimental data, if available. In this case, it can sketch an effective picture of the dynamical organization of the observed system. As a use case, we illustrated the analysis of the T helper network.

6.2 Detection of dynamical organization in cancer evolution models

This section describes the application of the RI methodology to the analysis of "cancer evolution" models, of which each individual patient represents a particular instance. The aim is to infer information on the dynamic organization of the tumor process starting from the observation of the damages undergone by a group of patients suffering from the same type of disease. In particular, we are interested in identifying relationships between genome mutations that lead to tumor progression, in order to recognize distinct independent cancer progression patterns in simulated patients, planning a road towards applications to real cases.

The following subsections describe the context of our analysis (i.e., the identification of independent progression patterns in cancer evolution models) and the experimental results we obtained by applying the RI method to detect cancer progressions. In particular, we used the iterative sieving algorithm, described in chapter 5, with the zI metric, described in section 3.3.

6.2.1 Context

Cancer is an evolutionary process according to which cancer cells progressively accumulate distinct (epi)genomic alterations (e.g., single-nucleotide variants, SNVs, and/or copy number alterations, CNAs, etc.) some of which – the so-called *drivers* – provide a selective advantage to the cells, which live and proliferate in a complex microenvironment with limited resources [150, 164]. As a result of this complex interplay that involves a large and varying number of competing cancer subpopula-

tions, high levels of inter- and intra-tumor heterogeneity are observed in most tumor (sub)types. Such heterogeneity is the major cause of drug resistance, treatment failure and relapses [30, 163, 244].

For this reason, in recent years a large number of statistical and machine-learning approaches is being developed to take full advantage of the impressive amount of *omics* data produced by massively parallel sequencing of cancer samples [14]. The goal is to decipher and characterize the somatic evolution of tumors, in order to identify possible regularities and differences between cancer (sub)types and to possibly isolate the molecular "weak points" in the evolutionary trajectories (e.g., bottlenecks), which may be targeted with ad-hoc therapeutic strategies [88].

In [35] some of the authors have introduced a computational pipeline (named PICNIC) for the reconstruction of cancer evolution models from cross-sectional somatic mutation profiles of multiple cancer patients. The theoretical framework combines Suppes' notion of *probabilistic causality* [224] and maximum likelihood estimation, in order to extract useful information on the underlying evolutionary process from noisy and often limited data [141, 187, 188]. The final outcome of the pipeline is a probabilistic graphical model which highlights the most likely accumulation paths of somatic mutations that characterize a specific cancer cohort. Not only such a model delivers an explanatory model of the evolution of a tumor type and of its heterogeneity, by highlighting the evolutionary paths that characterize distinct patients, but it also allows one to generate predictions, by identifying the most likely future stages of tumor progression which may be targeted in useful time. The application to various cancer datasets from The Cancer Genome Atlas (TCGA) allowed to automatically generate experimental hypotheses with translational relevance [160].

More in detail, one of the main steps of the pipeline is the stratification of patients in homogenous subgroups on the basis of their molecular makeup. On the one hand, a correct stratification should allow one to reduce the possible impact of confounding factors of heterogeneous cohorts on the inference accuracy [106]. On the other hand, a statistically robust stratification might be effective in defining personalized treatments, as the same therapy might even have very different efficacy on distinct patients, due to the molecular heterogeneity of the tumor composition.

In certain cases, stratification of patients can benefit from known clinical biomarkers [17], but in most cases it is necessary to employ unsupervised clustering techniques, such as non-negative matrix factorization (NMF) [84] or even classical methods such as k-means or Gaussian mixtures, usually on gene expression data [142]. In general, a successful stratification should identify disjoint sets of patients affected by distinct cancer subtypes, whose evolutionary history, i.e., the genomic alteration accumulation process, is significantly different among one another. Conversely, without stratification, inference methods such as PICNIC might struggle or even produce wrong outcome models, due to the confounding effects that are inherent of heterogeneous and intermixed cohorts (an effect commonly known as the Simpson paradox). The RI methodology allows one to identify groups of variables, therefore it could constitute a useful and interesting pre-processing tool.

6.2.2 Experimental results

The observed cancer progression can be considered as a "complex system" in which each patient is a particular realization of this dynamical process. Thus, the iterative sieving algorithm, described in chapter 5, can be applied along with the zI metric, described in section 3.3, to identify the "organized groups" of mutations, i.e., the distinct trees describing independent tumour progression, which might be used, in turn, to stratify the patients in homogenous subgroups.

Generation of test cases

In order to assess the efficacy and robustness of our method in identifying the sets of variables corresponding to independent cancer progressions, here we rely on simulated data in a variety of in-silico scenarios.
In this case, we sampled a large number of synthetic binary datasets describing cross-sectional mutational profiles of cancer patients from distinct generative topologies.
Each generative topology is described via a probabilistic graphical model, in which each node represents a genomic alteration and each edge corresponds to an evolutionary trajectory (i.e., the process of mutation accumulation) and is characterized by

a specific conditional probability (e.g., the edge $A \rightarrow B$ indicates that mutation B can occur with probability $P(B|A)$ and only if A has occurred).
In particular, we employed *forest* generative topologies – a forest being a disjoint union of trees, with multiple roots – to reflect the assumption of having distinct progression models corresponding to different patient subgroups/cancer subtypes, one per constituting tree. Each tree then describes a different pattern of *branching evolution* [49].

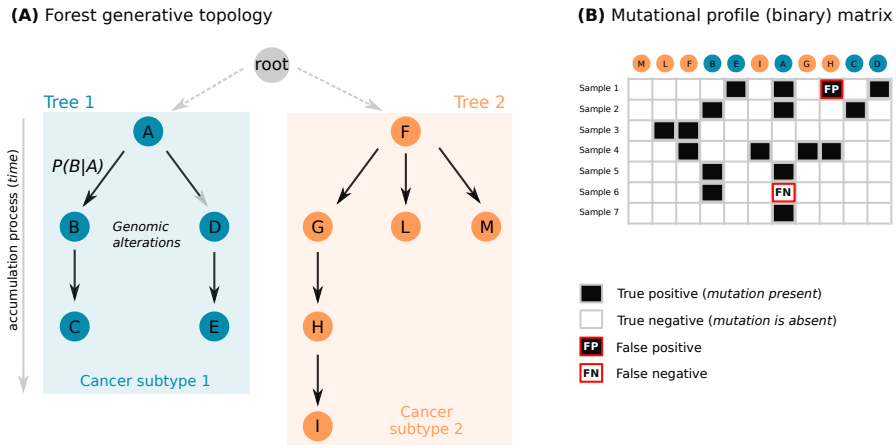


Figure 6.5: Synthetic data generation. (A) A number of forest topologies is randomly generated as probabilistic graphical models, in order to model independent tumor progressions (i.e., composed of disjoint sub-models involving different sets of events). Nodes in the graphical models are (epi)genomic alterations and edges represent the accumulation paths. Each edge is characterized by a randomly assigned conditional probability. A fictitious root is included for graphical purposes. The two distinct trees represent independent tumor progressions/subtypes. (B) A large number of independent binary datasets is sampled from the forest generative topologies (1: mutation is present, 0: mutation is absent). False positives and false negatives are included in the datasets depending on the parameters of the simulation.

Each binary dataset is represented by a data matrix having n (samples/patients) rows $\times$ m (genomic alterations) columns: each entry is equal to 1 if a certain muta-

tion is present in a sample, 0 otherwise. For each generative topology, binary datasets were sampled starting from the root using a recursive procedure and according to randomly assigned conditional probabilities (by ensuring that each mutation is observed at least once in the dataset).
In order to account for noise in the data, due to experimental and technical errors, we finally introduced a parameter $\nu \in (0,1)$ which represents the probability of each entry in the dataset to be random (i.e., false positives $(\varepsilon+)$ = false negatives $(\varepsilon-)$ = $\nu/2$). The noise level, even if uniformly distributed, complicates the inference problem.

To summarize our approach, we randomly generated 15 forest topologies with $m = 20$ nodes, and in particular: 5 topologies with 2 roots (i.e., 2 disjoint trees), 5 topologies with 3 roots and 5 topologies with 4 roots. For each topology, we sampled binary datasets with $n = 200$ samples and 3 noise levels: $\nu = 0$, 0.1, and 0.25.

The identification of the distinct trees composing the generative forest topology is extremely relevant, as each tree might be responsible for the evolutionary history of a specific subset of patients and, accordingly, might identify a specific cancer subtype and its evolutionary history.
By associating samples/patients to the trees, e.g., via maximum likelihood approaches, it would be finally possible to produce an effective stratification which might be, in turn, used in cancer evolution pipelines such as PICNIC to produce a reliable progression model for each distinct cancer subtype.

Analysis of test cases

We analyzed the forty-five datasets using the RI methodology (using the zI index and iterating the sieving algorithm until the index fell below the arbitrary reference threshold of 3.0).
The generative forests can be composed by very heterogeneous trees, and the length of the branches inside each tree could be very dissimilar: some trees are almost flat, others may have branches up to 5 levels deep. The identification of the trees of a forest is characterized by the presence of various types of possible errors:

1. at the single node level:
   (a) inability to assign a node to any tree (the data does not provide evidence high enough to allow the attribution);
   (b) attribution of a node to a branch to which it does not belong (even if the structure of the tree has been detected correctly);
   (c) attribution of a node to a tree to which it does not belong.
2. at the "branch" level (node groups):
   (a) correct identification of the individual branches, but no final connection between the various parts of the tree;
   (b) branches composed of heterogeneous parts (whole branches of the same tree, or subsets of other branches of the same tree);
   (c) fusion between two branches of two different trees.

In the list above, the errors are presented in increasing order of seriousness: the non-attribution of a node to a tree is less dangerous than its attribution to an incorrect branch which, in turn, is less dangerous than its attribution to an incorrect tree. In the same way, the correct identification of a branch permits its subsequent analysis using cancer evolution inference methods, which is impossible in the presence of the union of branches belonging to different trees.
Interestingly, in absence of noise, the RI methodology never inferred erroneous attributions of nodes, while the fraction of unassigned nodes was always extremely low (leading to an average node coverage close to 0.99 – see figure 6.6).

The trees have always been correctly identified, while the main branches of those classified as "not completely identified" have always been identified: indeed, when the identification is not complete, only the final fusion of the two already identified branches is missing. This phenomenon mainly occurs when it is necessary to identify large trees, as shown also in figure 6.6, where the number of complete identifications increases with the number of trees (because of the constant number of nodes in each experiment, the average size of the trees decreases with their number). Indeed, in order to be statistically significant, the complete identification of large objects requires

Size of trees					Noise 0.0			Noise 0.1				Noise 0.25			
Tree 4	Tree 3	Tree 2	Tree 1	Configuration	Incorrect attributions	Unassigned nodes	Trees / total trees	Incorrect attributions	Unassigned nodes	Trees / total trees	Branch merge	Incorrect attributions	Unassigned nodes	Trees / total trees	Branch merge
		3	17	F2T1	0	2	1/2	0	0	1/2	0	0	1	1/2	0
		1	19	F2T2	0	2	1/2	0	3	1/2	0	1	1	0/2	1
		1	19	F2T3	0	1	1/2	0	2	1/2	0	0	1	0/2	0
		3	17	F2T4	0	0	2/2	0	1	1/2	0	0	1	1/2	0
		8	12	F2T5	0	0	2/2	0	0	0/2	0	0	0	0/2	0
	1	6	13	F3T1	0	0	3/3	0	1	2/3	0	0	1	2/3	0
	1	3	16	F3T2	0	2	2/3	1	2	1/3	0	0	2	2/3	0
	2	3	15	F3T3	0	2	2/3	0	2	2/3	0	0	2	2/3	0
	2	4	14	F3T4	0	1	2/3	0	2	2/3	0	0	2	2/3	0
	1	7	12	F3T5	0	0	2/3	1	0	1/3	1	1	2	1/3	1
1	2	7	10	F4T1	0	0	4/4	0	1	1/4	0	1	1	0/4	1
1	2	7	10	F4T2	0	0	4/4	0	0	3/4	0	1	2	2/4	1
1	1	8	10	F4T3	0	1	3/4	0	1	2/4	0	1	2	1/4	1
3	4	4	9	F4T4	0	0	4/4	0	1	2/4	0	0	2	2/4	1
1	1	4	14	F4T5	0	0	3/4	0	0	3/4	0	0	1	2/4	0

Figure 6.6: Summary of the results of the analysis. For each configuration "FxTy" (where x stands for "number of trees in the forest" and y for "case identifier"), some relevant quantities are shown: on the right, the results of the analysis for different noise levels; on the left, the size of the trees. In case of absence of noise, the number of incorrect assignments, the number of unassigned nodes and the number of trees found compared to the number of total trees are shown. In the presence of noise, incorrect fusions of different branches are also present (there are no incorrect fusions of branches in the absence of noise). Note that an incorrect merger causes the inclusion of a tree consisting of a single element in a branch of a different tree (a situation corresponding to the only incorrect node assignment - see the text for a more detailed comment).

a high number of observations (a number that depends on the structure of the tree, and that is sometimes greater than the number of available patients).[4]
Anyhow, even when big progression trees are generated, the branches are always correctly identified: therefore, this information might be taken into consideration by a cancer evolution inference method without many concerns (a branch is actually a coherent tree). This allows us to say that the information resulting from the analysis of the noiseless cases can provide the information needed for a correct downstream processing.

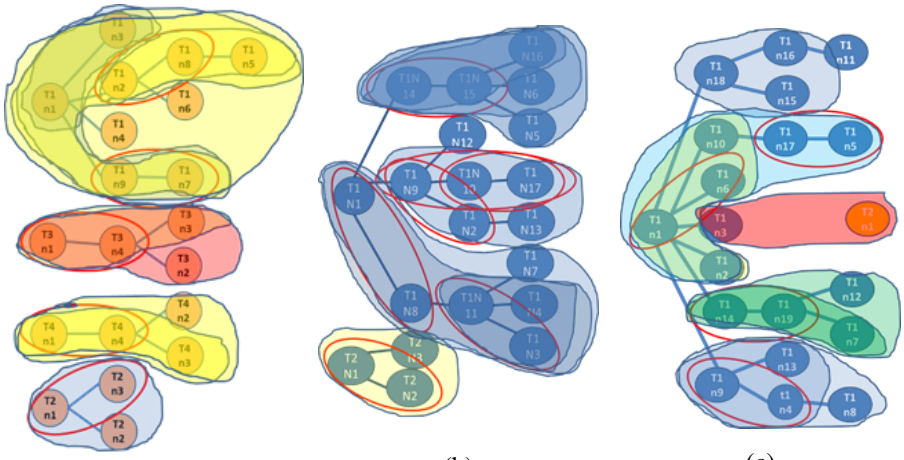
[4]Note that this observation is related to the number of observations that is possible to have in currently available clinical studies, rather than to the method we are applying.

Similar considerations can be made even when the noise level is considerably increased (figure 6.6). The number of node assignment errors remains extremely low, and the "merging" of different branches (which occurred only once with noise 0.1) are actually fusions of a corrected branch with a single element belonging to another branch. Indeed, these events are almost exclusively trees composed of a single node which, in these cases, are melted in a branch of a larger tree. Indeed, because of the presence of noise, the occurrence of a single mutation becomes the occurrence of more-than-one mutations: this causes the presence of a tree composed of a single element to be implausible[5]. Moreover, these mergers take place at very low zI values: always below 5.0, and mostly just near the 3.0 threshold.

6.2.3 Final remarks

In the work described in this section we have extended an interesting kind of analysis based on the juxtaposition of dynamical states belonging to different instantiations of a complex system, to a particular case in which different patients suffering from the same disease can be considered special "instances" of the same pathological dynamic process. In order to assess the accuracy of the approach, we simulated 200 patients, of which we therefore knew the division into classes. The RI methodology, based on entropic measurements, allowed us to identify distinct cancer subtypes (represented by different trees, each including successive accumulating mutations), potentially a very useful information for subsequent analyses or treatments. Vice versa - and in a very interesting way - this analysis confirmed the possibility of inferring information on the dynamic organization of a process starting from the observation of its distinct instances, under the simple hypothesis of a common structural condition. These results support further investigations, in order to identify suitable data from real cases that can be used to test the proposed approach in ordinary situations.

[5]The identification of a tree composed of a single element is a case that is strongly influenced by noise.



(a) (b) (c)

Figure 6.7: Results of the analysis on simulated cross-sectional cancer datasets sampled from forest topologies. The figures show all the groups gradually identified, up to the large final groups: the circles identify the nodes of each tree, connected by evolutionary paths. The fact that each small group resulting from the analysis is completely included in a larger group is a consequence of the use of the iterated sieve algorithm. (a) A perfect identification - F4T4 configuration without noise. (b) Even if the limited number of patients does not allow the identification of a very large tree, the RI algorithm still manages to identify the main branches of each tree; in this case there is no evidence which allows one to attribute the T1n12 node to one tree rather than to the other one - F2T1 configuration, two trees (one completely identified by the yellow set), with noise 0.25. (c) The F2T2 configuration is composed of two trees, respectively of size 1 and 19. When the noise level is equal to 0.25, one of the last groupings (at a very low evidence level) fuses a rare leaf of the largest tree with the single element of the second tree. Node T1n11 is not assigned to any tree.

6.3 Social RI for studying communities in a Facebook group of patients

Social networks can be considered as systems that exhibit complex dynamics, since they are characterized by complex and dynamic relationships among users. A challenging topic within this context regards the identification of communities or subsets of users, both within the whole network and within specific groups.

We applied the Relevance Index method to the study of communities of users in the Facebook group of the Italian association of patients affected by Hidradenitis Suppurativa.

Hidradenitis Suppurativa (HS), also known as Acne Inversa, is a chronic, underdiagnosed, often debilitating and painful disease that affects the folds of the skin.
It is estimated to affect between 1% and 4% of the world population having a considerable negative impact on the quality of life and on the emotional well-being. It is widely reported that the difficulties in obtaining a diagnosis, together with the lack of proper therapies, bring HS patients to develop an emotional closure, with the consequence that patients often do not talk about their condition with anybody, as reported by clinicians and by the Italian patients' association (Inversa Onlus) [19]. In the last few years, with the diffusion of social media, a considerable number of patients started to share a relevant amount of data related to their feelings, interacting with each other in a Facebook group using posts and comments. This support group represents an important container of clinical and social information about this condition and its impact on patients' lives. Starting from this consideration, the need to extract the unstructured information in this group has emerged, in order to give a better understanding of HS to doctors and clinical researchers.

The community detection has been performed using the T_c index computed by the HyReSS metaheuristic, which has been described in section 4.1. The results obtained by HyReSS have been compared to those obtained by an exhaustive search based on the same parallel implementation of the T_c computation.

The communities detected through the aforementioned method have been studied to search similarities in terms of number of posts, sentiments and number of Facebook

friendships. The results, shown in section 6.3.2, demonstrate that it is possible to detect significant subsets of users in the particular Facebook group we analyzed.

The following subsection describe the context of our analysis.

6.3.1 Context

The identification of the network structure and of communities in different types of network systems has gained great interest during the years, whatever the type of the considered network system: the Internet, particular social networks such as Facebook, Linkedin, Google+, etc., citation networks, e-mail networks, and the like [213]. These networks can be considered either static or dynamic complex systems and different methods have been devised to study their structures accordingly. The identification of a community structure in these networks could mirror the detection of highly interacting subsystems in complex systems. A community structure can be described as the gathering of nodes of the network into groups, which exhibit a higher density of edges within their members than with other groups. The edges, or links, may change over time and may refer to a different semantics according to the particular scenario behind the network. In a social network the nodes represent users (or better, user accounts) while the edges represent the connections between users that convey information, thoughts, news over time. These connections vary over time, as they can be established, removed and then established again, and can be of different types, according to the nature of the particular social network, since a connected user can be a friend, a relative, a colleague, a sport teammate, etc.

In this section, we focus onto community detection in a particular dynamic network complex system, i.e., a Facebook group.

Researchers have tried for years to employ genetic algorithms to tackle the problem of communities detection in complex networks: in [228], the authors proposed a scalable genetic method based onto the network modularity metric, demonstrating its effectiveness in a Karate club and a College Football scenarios. However, the work is based onto the *a-priori* knowledge of the number of edges connecting nodes in the network, and is characterized by an *ad-hoc* crossover phase as well as the initial assignment of a node to a random community.

The contribution in [177] presents GA-Net, a method to detect communities in social networks using a genetic algorithm, where the variation operators are tailored to take into consideration only the actual correlations among the nodes. The fitness function is called community score and is based on the locus-based adjacency representation, which requires the *a-priori* knowledge of the adjacency matrix of the nodes of the network.
More recently, in [134], the authors have employed an extended compact genetic algorithm to identify communities in complex networks. The method seems to address effectively the complex nonlinear optimization problem and the epistasis in the representation of chromosomes, both for benchmark and real-world networks, resulting more efficient and stable than other genetic algorithm-based solutions while, at the same time, requiring less time to reach convergence. However, it uses the modularity metric as a fitness function, and a marginal product model, which is quite complex, even if it is built through clusters based on mutual information, to model the probability distribution of the individuals.
In [100], the authors propose a novel generational genetic algorithm to solve the communities detection problem. The method is shown to outperform other genetic algorithms in terms of accuracy in finding communities; however, it is based onto the modularity metric and on *ad-hoc* initialization methods and search operators. Moreover, differently from our technique, it employs normalized mutual information only in the end of the process, to evaluate the performances of the algorithms studied, while providing a range of available solutions, i.e., a lowest and highest number of possible communities to be detected.

The application of genetic and evolutionary algorithms to complex social networks is not limited only to communities detection. For example, recently, the authors of [29] tried to maximize the influence of certain nodes, while, at the same time, minimizing their number. This was done by means of a particular Multi-Objective Evolutionary Algorithm, which outperformed two other state-of-art heuristics, and according to two different influence propagation models. Such a technique was effective in two complex network scenarios, namely ego-Facebook and ca-GrQc; however, it requires prior knowledge of the probabilities of influence propagation.

On the contrary, the solution we propose, thanks to the RI, does not require any *a-priori* knowledge on the structure of the social network, and allows one to dynamically study communities it detects from different points of view, such as sentiments, number of posts, etc., without the need to modify invasively crossover and mutation operations. To the best of our knowledge, the work described in this section has been the first that applied the RI method in combination with a genetic algorithm with the aim of identifying and studying, in a dynamical way, subsets of users in a social network.

The Facebook group we analyzed includes 612 users; we only focused on the most active ones, i.e., those who, on average, posted at least twice per year between 2010 and 2016 (32 users). This choice was made in order to analyze only significant data, and it is not due to a limitation of the proposed method, since it has been proven to scale with the dimension of the system under consideration, as shown in section 4.1. The group has been studied from the point of view of time and of sentiment. More precisely, the dynamic communities detected in the group by the RI method, based on the sentiment expressed in the posts during a certain period of time, are found to be correlated with the degree of the users, that is, the number of friend connections within the Facebook group itself.

The following subsection describes the dataset we collected from the Facebook group and the experimental results we obtained using the RI analysis.

6.3.2 Experimental results

In order to use the Relevance Index method to study a Facebook group of patients affected by Hidradenitis Suppurativa, we used the labeled data obtained in [139]. In that work, an automatic emotion detection and a social network analysis technique have been used to analyze the emotional states of each member of the group.

In collaboration with Inversa Onlus association, which created the support group, we collected data from the Facebook group using the official Facebook API with administration permissions. We retrieved all the metadata related to the posts, including content, author, date and time of publication. This kind of information has been carefully anonymized to preserve the privacy of the members.

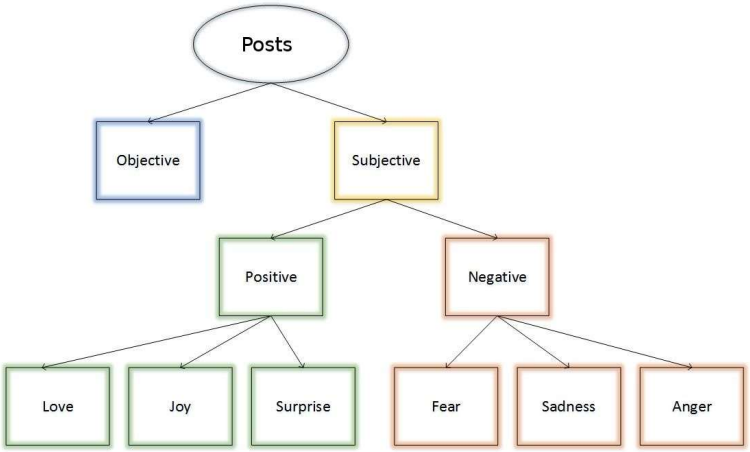


Figure 6.8: Structure of the hierarchical classifier.

The sentiments associated to the posts have been obtained using a manually annotated training set and a seven-output hierarchical classifier system based on Parrot's emotion categorization [170] (3 positive sentiments: joy, love, surprise, 3 negative sentiments: fear, anger, sadness and 1 neutral: objective), as shown in Figure 6.8. In particular, we used stopwords removal and stemming algorithms to preprocess the data [198, 12], the bag-of-words model (with values calculated using the TF-IDF weight function) to extract features from the sentences, the Information Gain algorithm for feature selection [254], and 7 Naive Bayes Multinomial classifiers to create the hierarchical classifier [6].

In [139, 140], we have estimated how these emotions are correlated with their own degree in the social network (the number of mutual connections in the group). The main results obtained in that work will be briefly presented in subsection 6.3.3, comparing the results with those obtained by the RI analysis.

The following subsection describes the data (variables and observations) we used for the RI analysis.

Dataset description

Each observation (sample) of the system represents a specific month within the period of interest. In particular, we focused onto two different case studies, based on two kinds of values:

1. Post-based value. This feature is set to 1 if the user has posted during the considered month, otherwise it is set to 0.

2. Sentiment-based value. This feature describes 4 different cases for each user in the considered month: no posts, positive sentiment, negative sentiment, neutral sentiment. The sentiment polarity has been obtained by looking at the overall emotion expressed by the user (counting his positive, negative and neutral emotions).

HyReSS performances

The datasets have been analyzed using both exhaustive search and HyReSS. Given the stochastic nature of HyReSS, 10 independent runs of the algorithm were executed to asses its performance. The results have been evaluated both in terms of quality and of speed-up with respect to the exhaustive search.

The quality of the results has been evaluated by comparing the list of highest-T_c subsets produced by HyReSS with the results of the exhaustive search. To let results be comparable, we relied on the same homogeneous system to compute the T_c values in both approaches.

The two algorithms have been compared also in terms of efficiency (execution time). It is to be noticed that the methods relied on the same GPU implementation of the fitness function, which means that the differences observed depend only on the efficiency and complexity of the algorithms and not on their implementation.

Tests were run on a Linux PC equipped with a 1.6 GHz Intel I7 CPU, 6 GB of RAM and a GeForce GTX 680 GPU by NVIDIA. The parameters regulating the behavior of HyReSS have been set as reported in Table 6.1.
Results are summarized in Table 6.2 and discussed in the following subsection.

Table 6.1: HyReSS parameter settings. The parameters are defined in section 4.1.

Dataset	P_{mut}	p	α_f	α_p	β	γ	δ	θ
Post-based	0.1	25600	256000	3	0.75	0.75	0.3	15
Sentiment-based	0.1	25600	256000	3	0.75	0.75	0.3	15

Table 6.2: Summary of HyReSS performances and comparison with the exhaustive search (ES). HyReSS execution time is expressed as the average and the standard deviation of the execution times measured over 10 runs. The speedup of HyReSS algorithm is related with the execution of a number of fitness evaluations which is much lower than the number of fitness evaluations of the exhaustive search.

Dataset	N. Variables	N. Samples	Time[s] (ES)	Time[s] (HyReSS)	Speedup
Post-based	32	84	1007.78	30.85 $\pm$ 0.61	32.67
Sentiment-based	32	84	1272.85	35.48 $\pm$ 0.55	35.88

The results obtained by HyReSS and by the exhaustive search are almost identical. Only occasionally at most one out of the top 50 sets, usually more than enough to understand the main dynamics of the systems we took into consideration, was not detected by HyReSS.

Social network results

This section presents the highest-T_c RSs identified by HyReSS using the post-based and the sentiment-based datasets.

Table 6.3 shows the users composing the highest-T_c subsets, together with their user-degree, which is defined as the number of Facebook friends in the considered

Facebook group. As one can notice the sentiment-based subset is composed, on average, by users with a higher degree compared with the post-based subset. For the sake of completeness, in Figure 6.9 we show the overall distribution of users according to their degree.

Table 6.3: The highest-T_c RSs identified on the post-based (left) and sentiment-based (right) datasets.

Highest-T_c RS (post-based)	
User	**Degree**
User 1A	120
User 2A	5
User 3A	0
User 4A	5
User 5A	8
User 6A	0
User 7A	13
User 8A	6
User 9A	0
User 10A	0
User 11A	0

Highest-T_c RS (sentiment-based)	
User	**Degree**
User 1B	42
User 2B	120
User 3B	38
User 4B	1
User 5B	31
User 6B	19
User 7B	1
User 8B	31
User 9B	23
User 10B	16
User 11B	3
User 12B	0
User 13B	9
User 14B	10
User 15B	5
User 16B	1

The bidirectional graphs in Figures 6.10 and 6.11 represent the Facebook group. Each node of a graph represents a user and each edge represents a friendship relationship between two users.
In order to better appreciate the results, the size of each node has been made proportional to the node degree.

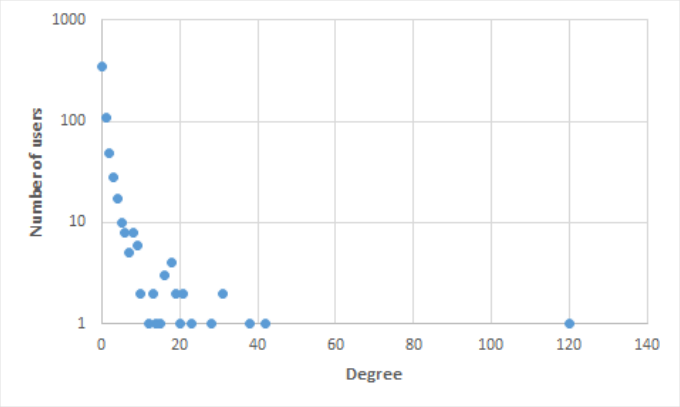


Figure 6.9: Graph representing the distribution of node degrees in a logarithmic scale.

Figure 6.10 shows the graph corresponding to the most active users (the ones who, on average, posted at least 2 posts per year). The red nodes in the left graph represent the highest-T_c RS identified by using the post-based dataset, while the blue nodes in the right graph represent the highest-T_c subset identified by using the sentiment-based dataset. The two communities shown in the figure are the most representative in terms of T_c; the other detected communities are simply subsets of these main communities and are not represented.

It is important to highlight that the sentiment-based analysis provides additional information with respect to the post-based one. In particular, it provides not only a timeline information about users' posts, but also an emotional information about the feeling expressed in the posts.

It is important to notice that the sentiment-based subset is composed by users with a higher degree. This result may represent an evidence of the influence of the group on the feelings expressed by users. Patients who developed many relationships with other users in the Facebook group express a similar dynamical behavior in terms of feelings, e.g., from negative posts to positive posts and vice versa.

Figure 6.11 shows the identified communities in the extended graph composed of all the 612 users of the Facebook group. It is to be noticed that the sentiment-based community, identified analyzing the most active users, is composed of the users who

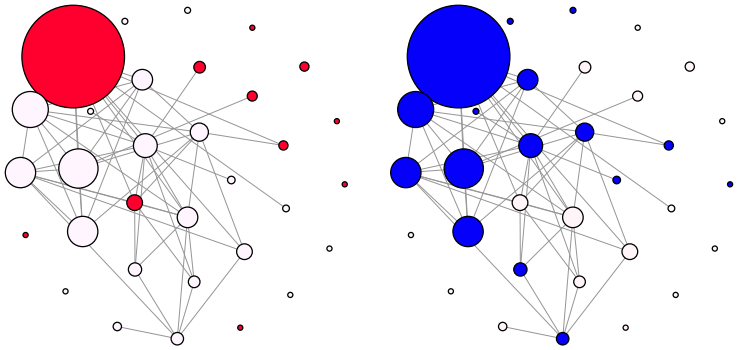


Figure 6.10: Graphs representing the 32 users who, on average, posted at least 2 posts per year in the considered period. The post-based community (red nodes, on the left) and the sentiment-based community (blue nodes, on the right) have been highlighted. The size of each node is proportional to its degree, and each edge represents a friendship relationship.

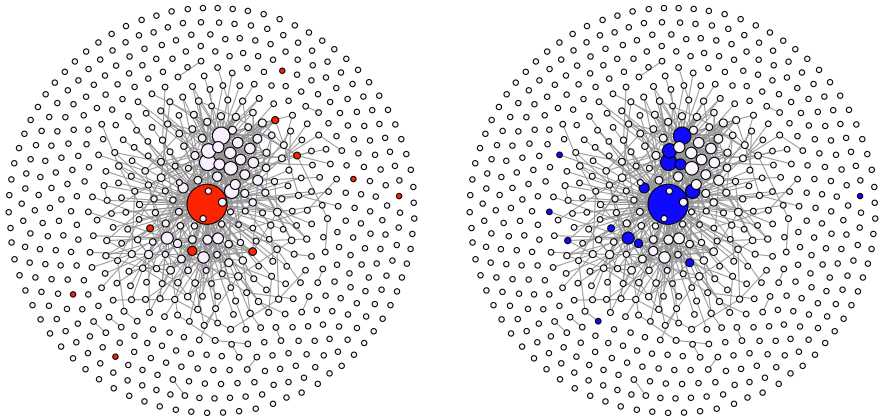


Figure 6.11: Graphs representing all 612 users of the Facebook group with the post-based community (red nodes, on the left) and the sentiment-based community (blue nodes, on the right). The size of each node is proportional to its degree, and each edge represents a friendship relationship.

have the highest degree also in the extended graph. It is to be noticed that the RI analysis has identified a community which is coherent with the network structure, using only the information about the posts published and the sentiments expressed over time.

The strong relation between the sentiments expressed by the users and their interactions inside the Facebook group will be further discussed in the following subsection, which will present similar results obtained using a different methodology.

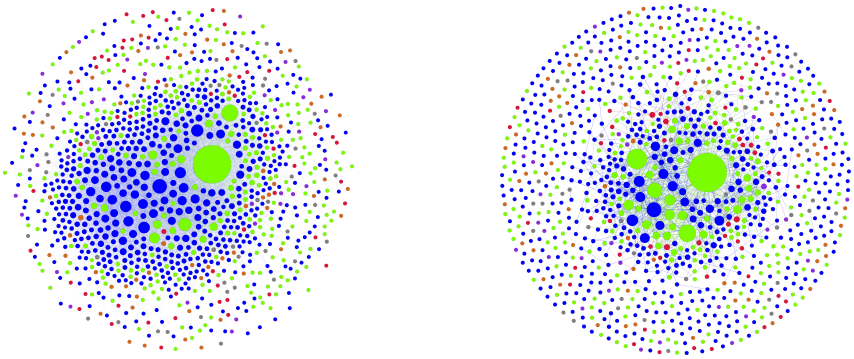
6.3.3 Comparison with conventional techniques and final remarks

This section has described the application of the RI method to the analysis of communities in a real social network: the closed Facebook group of users belonging to the Italian association of patients affected by Hidradenitis Suppurativa. The RI analysis has been performed using HyReSS metaheuristic.

The performance of HyReSS in searching relevant sets has been analyzed in two case studies (post-based and sentiment-based). HyReSS results have been compared to those obtained by an exhaustive search based on the same parallel code for computing the RI, evaluating both its quality and the obtained speed-up. HyReSS provided the same results as the exhaustive search, performing much fewer fitness evaluations in a significantly shorter time.

The application of the RI method to the Facebook group identified interesting communities. In particular, the emotional information highlighted a subset of users who developed many friendship connections with other users in the Facebook group. Starting from a dynamical analysis based only on the posts and on the sentiments (without any information about the network structure), the RI analysis was able to identify a sentiment-based community which is composed of users which are characterized by strong relationships in terms of Facebook friendship inside the network.

The strong relation between the sentiments expressed by the users and their interactions inside the Facebook group has been confirmed also by a different kind of analysis, combining machine learning and network theory. For the sake of completeness, Figure 6.12 shows the main results of the Social Network analysis we performed in [139, 140] studying the same Facebook group described in the present section. In



(a) (b)

Figure 6.12: Comparison between the interaction network (a) and the friendship network (b).

particular, the users' activities have been analyzed in terms of:

1. **Interaction network**, which is a directed and weighted graph. Each node represents a user and each link represents the number of comments that have been written by the source node user as a response to posts and comments written by the destination node user;

2. **Friendship network**, which is an undirected graph where each node represents a user and each undirected link represents the mutual Facebook friendship between two users (nodes).

In both networks, the size of a node represents its degree. It is to be noticed that the number of edges of friendship and interaction network are largely different. In fact, only 23.43% of the friendship relationships among the group members take place between users who are also connected in the interaction network.
Moreover, in both networks, the green central node with the highest degree represents the same user, who has a central role for the group.

The interaction network (Figure 6.12a) shows that the majority of the users who interact a lot with each other by writing posts and comments in the group expresses

sadness. This can derive from the support function of the group, in which the patients ask for advice and exchange opinions about their disease.

The friendship network (Figure 6.12b) shows that users who establish many relationships directly in the form of Facebook friendships express mainly joy. As a possible interpretation, this result may represent an evidence of the positive influence of the group.

Thus, the relationship between the expressed sentiments and the number of Facebook friendships, identified by the RI analysis, has been confirmed by a different kind of technique, based on machine learning and network theory. The RI methodology seems to provide useful information also in contexts which can be different from traditional complex systems analysis. The application of the RI methodology to machine learning tasks will be further investigated in the following chapter.

An extended description of the work described in this chapter can be found in [202, 204, 242].

Chapter 7

From complex systems analysis to machine learning and pattern recognition

The properties that the RI metrics highlight in time series, when analyzing the dynamics of complex systems, can be somehow assimilated to the properties of the multivariate distribution of the examples of a machine learning training set. This chapter aims at evaluating the applicability of the RI methodology to machine learning and pattern recognition problems. With no pretense to reach state-of-the-art performances, the following sections aim at showing that the hypothesis is reasonable and already competitive with existing machine learning approaches.

In particular, section 7.1 presents a *zI*-based approach to pattern clustering and classification, sections 7.2 and 7.3 propose, respectively, a supervised and unsupervised approach for feature extraction/selection. Finally, section 7.4 presents a *zI*-based preprocessing method for improving the accuracy of a classification task.

7.1 An integration-based approach to pattern clustering and classification

In this section, we use the RI methodology for an unusual application to a pattern clustering and classification problem. The results show that the centroids of the clusters of patterns identified by the method are effective for distance-based classification algorithms. We compare such a method with other conventional classification approaches to highlight its main features and to address future research towards the refinement of its accuracy and computational efficiency.

7.1.1 Context

As shown in chapters 2 and 3, the RI metrics rely on the analysis of a set of observations, represented as a matrix where each row represents the status of the system at a certain time, while columns represent the status variables of the system. Abstractly, one can say that such methods are able to find clusters of columns that exhibit some correlated properties or behaviors.

So, what would happen if one just transposed the matrix, to cluster events (status) instead of variables? And what would happen if one represented the status of the corresponding pixels in a set of patterns along the rows, and the patterns themselves along the columns?

In this section, we show that if we "turn the world upside down" (or, better, we rotate it by 90^o), the same method that could be employed to find groups of correlated variables can be used for clustering similar patterns. The work presented in this section aims at providing a proof of concept for such a theory and, with no pretense to reach state-of-the-art performances, to show that the hypothesis is reasonable and already competitive with existing classical clustering and classification approaches.

In particular, we consider the zI index (described in section 3.3), which is a normalized integration measure that accounts for the standardized distance of a group of variables from a reference condition of statistical independence.

As described above, using such an index to perform clustering (assigning a relevance score to each group of variables) can be somehow assimilated to some other

information theoretic approaches to clustering. In most such methods, cluster labels (represented by random variable Y) are assigned to data points (represented by random variable X), such that the mutual information between data and labels is maximized [68, 158, 245]. However, Ver Steeg *et al.* [235] demonstrated that this approach is fundamentally flawed, so that clustering performance deteriorates as the amount of data increases. The authors propose an alternative approach based on the optimization of the *Total Cluster Uncertainty*, which is a particular estimator of the conditional entropy $H(Y|X)$. Optimizing such a metric over all possible partitions is a difficult problem. The authors consider a heuristic approach that involves solving a tractable semidefinite problem. A radically different approach, which is probably the closest to the one we propose, was introduced by Aldana-Bobadilla *et al.* [4]. There, the best dataset partition $\Pi = \{C_1, C_2, .., C_k\}$ is obtained by maximizing the entropy $H(\Pi) = \sum_{C_i} H(X|C = C_i)$ and minimizing $\sum_i \sigma(C_i)$, at the same time. The partition space is explored by means of an evolutionary approach.

The following subsection presents our *zI*-based approach to clustering. Then, in subsection 7.1.3, we describe the application of the method to clustering and classification problems, reporting results obtained both on controlled synthetic data and on a real-world set of images of low-resolution digits extracted from license plates.

7.1.2 *zI* clustering and classification

To quickly find the most relevant subsets, we used HyReSS metaheuristic, described in section 4.1, combined with the iterative sieving algorithm, described in chapter 5.

When the *zI* is used for clustering, the iterative sieving operates by subsequently merging the highest-*zI* set of patterns into a new cluster in each step. By iterating the procedure, considering each time the merged variables as a new atomic entity, the final cluster set is composed by all the clusters, generated by such *mergers*, that have been detected at the time of the final iteration, i.e., the one in which the *zI* falls below a conventional value of 3.0 (i.e., 3 standard deviations from the reference condition of variable independence).

The experiments described in the following subsection, in which we have analyzed a synthetic and a more complex real-world dataset, have been aimed at proving

that, if the zI metric is applied to a matrix where columns represent patterns and rows represent values of corresponding features, it can detect clusters of patterns based on their similarity, which can be effective for the classification.

We considered two approaches to classification, one based on a distance criterion, and another based on the computation of the zI index. As concerns the first approach, considering that the resulting values of centroid pixels are an estimation of the *a priori* probability that the pixel is on when a pattern belongs to the cluster, we defined the distance $d(x, c_i)$, between the pattern x to be classified and each centroid c_i, as the pixel-wise sum of the absolute values of the differences between x and c_i. In the fully zI-based approach, instead, the classification relies on the computation of the same index zI used for computing the clusters. Substantially, the pattern x to be classified is subsequently included as an extra pattern into each cluster c_i, generating a new cluster $\tilde{c}_i$. Finally, we assign to x the label of the cluster c_i, from which the cluster $\tilde{c}_i$ having the highest zI has been generated.

7.1.3 Validation of the method

The main aim of the experiments we carried out has been, firstly, to validate the assumption that our method can detect groups of patterns that meet the requirements for being properly termed "clusters". Secondly, to demonstrate that such clusters can be effectively used as a basis for a minimum distance (1-Nearest Neighbor) classifier. Thus, in evaluating the classification performance of the cluster set we detected, and in comparing it with that of other commonly-used classifiers, we did not make any effort to optimize the number of clusters using, for instance, the classical cluster removal/splitting/merging strategies.

In the first part of this subsection we describe the results obtained in experiments made on a synthetic dataset. After that, we compare the classification, made using the clusters detected based on their zI values, with the performances: (i) of a baseline 1-Nearest Neighbor classifier based on a random selection of the training patterns, (ii) of other more "standard" distance-based classifiers of similar complexity, and (iii) of other commonly-used classifiers based on different principles.

When the zI is used to detect the subsets of a complex system, the analysis can

only use the values of the variables which describe the system status. When the goal is clustering, as in this case, and each instance of the training set is a pattern that we want to classify, most often a labeled dataset is available. The question, then, is whether one should approach the problem in an unsupervised or a supervised way.

The former option would imply extracting clusters (groups of patterns) from the whole dataset at the same time, regardless of the class to which they belong, and would represent the most natural and direct application of the method. Conversely, in the latter case, the clustering procedure would be repeated for each subset of patterns belonging to a certain class. In any case, the method is intrinsically unsupervised: the subdivision of the training set into distinct subsets corresponds to exploiting the knowledge about the problem, which allows one to move up one abstraction level and use the method to detect different nuances of the same class of patterns, instead of identifying generic clusters that would then need to be subsequently matched to class labels.

The unsupervised approach would have the obvious advantage of being applicable even in the absence of knowledge about the data. However, such an approach might have the drawback of highlighting correlations among feature sets which may, in turn, be totally uncorrelated with the actual classes to which the patterns belong. The supervised approach has the intrinsic advantage of avoiding *a priori* all possible hybridizations caused by the inclusion of patterns of different classes into the cluster.

The results described in the following have been obtained using the supervised approach, based on these considerations and on the nature of the datasets to which we have applied it.

Experiments on a synthetic dataset

In order to test our approach and permit an easier observation of its behavior, we created a synthetic set of patterns representing the ten digits from 0 to 9. To do so, we designed a "perfect" pattern for each of the ten digits (of size 13×8 pixels each), of which we then generated corrupted versions by adding noise at different levels, from 5% to 30%. In a first set of patterns, noise was more likely to be added to the foreground (the actual pixels composing the digit) than to the background. At the

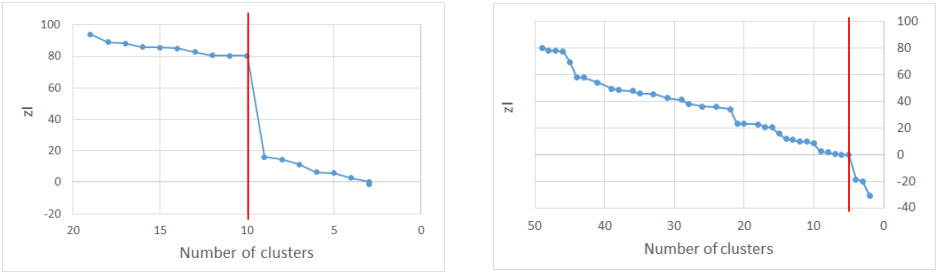
same time, we created other pattern sets by adding noise uniformly distributed all over the image, whose level was tuned to obtain the same average noise level as in the first set[1]. During the experiments, in each iteration of the sieving algorithm we considered all possible clusters having size lower than 5, merged the variables belonging to the highest-ranked group into a single cluster, and launched a new iteration where the new cluster was treated as a single entity, until the zI of the highest-ranked group was lower than 3.0.

Cluster detection In a first experiment, in which we considered only the ten "perfect" patterns, the method identified two clusters, including digits $0,9,6,8,3,5$ and $2,7,1,4$, respectively. This shows that a completely unsupervised strategy is unsuitable when too few training patterns are available.

The analysis of a moderately larger dataset, including six noisy patterns for each digit, provided some evidence in favor of the use of a totally unsupervised strategy. In fact, we could observe that: (i) the first mergers always include patterns representing the instances of the same digit, regardless the merged patterns are single patterns or larger clusters, and that (ii) the sequence of zI values associated to the new cluster obtained in subsequent mergers is characterized by a large drop, corresponding to the appearance of the first cluster including patterns representing different digits. In other words, as long as the dataset is "clean" enough, and possibly large enough but not too large, one can reach an ideal situation, in which each merger creates a new high-zI homogeneous cluster for a digit that has not been represented yet, until clusters representing all digits have been detected (Figures 7.1 and 7.2).

In fact, one can notice that, with the set we have taken into consideration, such a drop occurs typically when a number of homogeneous clusters equal to the number of classes has been reached. This coincidence shows that, in such a dataset, pattern homogeneity is preserved in spite of the addition of relevant noise levels.

[1] Patterns obtained by both strategies are represented with the tags 005, 010, 015, 020 and 030, even if these numbers represent the actual percent noise levels only for the first set.



(a) (b)

Figure 7.1: zI analysis of (a) a system composed of 20 patterns (two for each of the ten digits) and (b) a system composed of 50 patterns (ten for each digit from 0 to 4) with noise level equal to 10%. Notice that the zI suddenly drops after detecting a number of clusters equal to the number of digits.

Classification The analysis of the intra-cluster and inter-cluster distances confirmed this observation; moreover, it showed that the centroids also satisfied the requirement, critical for a classifier, that the distances among cluster centroids should be larger than the distances between each centroid and the patterns belonging to the corresponding cluster.

We tested this approach on a set of 200 patterns, generated by the same random process, obtaining a perfect classification also using the zI-based classification.

Experiments on a real-world dataset

The real-world dataset used in our experiments[2] was collected by Società Autostrade SpA at highway toll booths. It includes 11034 patterns representing the ten digits from 0 to 9, roughly uniformly distributed among the ten classes under consideration. The patterns have a size of 13×8 pixels, and have been trivially binarized pixel-wise using a threshold of 0.5 (considering pixel values normalized between 0 and 1). This resulted in strings of 104 binary features.

We used part of the full dataset to compute the relevant groups, building a training

[2] downloadable at ftp://ftp.ce.unipr.it/pub/cagnoni/license_plate.

set with 2000 patterns, 200 per digit, to compute the clusters, and a test set with 3000 patterns, 300 per digit, to assess the accuracy of the classifiers.

In the following, we describe and analyze in detail the results we have obtained using the supervised approach.

Cluster detection and distance-based classification By applying HyReSS to our dataset, we detected 296 clusters, well distributed among the ten classes, with an isolated minimum of 22 clusters for digit "7" and between 27 and 34 clusters for all others. Each cluster includes from 3 to 18 training patterns. The patterns comprised in the same cluster appear to be visually homogeneous and morphologically similar, which is particularly true for the "cleanest" patterns. We evaluated the quality of such clusters by applying to them a supervised classifier, and analyzing the performances

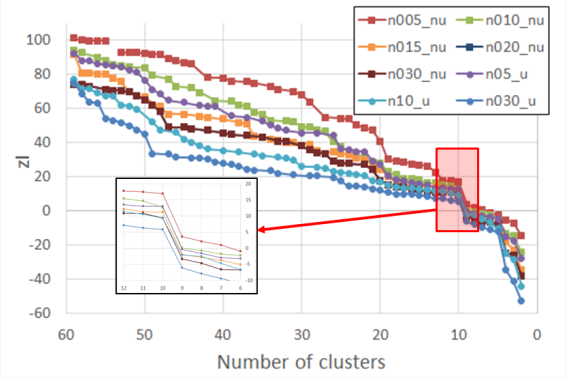


Figure 7.2: *zI* analysis of eight sets composed of 60 noisy patterns (six patterns for each digit). Five sets are affected by noise which is more likely to be added to foreground pixels (xxxx_nu in the figure legend), whereas the digits of the other three sets are affected by uniform noise (xxxx_u in the figure legend); nxxx tags indicate the level of noise that was added. The inset shows an enlargement of the iterations around the one in which heterogeneous clusters start appearing. All clusters detected before the drop correspond to homogeneous groups with the only exception of one very distorted pattern of set n030_u.

both of classical distance-based classifiers and of the classification strategy based on the zI.

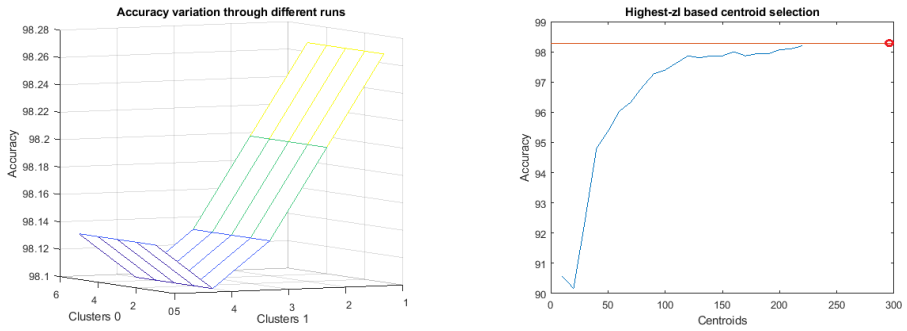
The most direct assessment of the quality of the centroids of a set of clusters is provided by the accuracy of the 1-Nearest Neighbor (minimum-distance or 1-NN) classifier, whose reference set includes the centroids of all clusters that have been detected by the zI method. Therefore, to classify a pattern x, after computing $d(x, c_i)$ as defined in section 7.1.2 for all centroids c_i, we assigned x to the class of the centroid $\hat{c}$ for which $d(x, \hat{c})$ was the smallest. Using the distance-based classifier, the set of 296 centroids was able to correctly classify 2948 out of the 3000 patterns in the test set, corresponding to an accuracy of 98.27%.
Adopting such a classification scheme, one should notice that a set of labeled data as the training set can itself represent a reference set of (possibly widely redundant) prototypes, which can be used as a basis for a 1-NN classifier. This means that a set of patterns, randomly selected from the training set, can be considered as a baseline reference for evaluating the quality of schemes, where protoypes are selected on the basis of some more sophisticated criteria.

Therefore, we considered as lower-end baselines the results of classifiers where the corresponding reference sets are: (i) 296 randomly extracted training patterns; (ii) 296 centroids computed as the average of a random set of patterns belonging to the same class.

Considering the stochasticity of the selection procedure, we repeated it 10 times, always obtaining accuracy values well below those yielded using the clusters detected by the zI clustering method for both the random pattern selector (average 96.70%, median 96.73%, worst 95.87%, best 97.43%) and the random centroid selector (average 96.17%, median 96.09%, worst 95.07%, best 97.10%).

Since we used HyReSS to compute the zI-based clusters in a reasonable time, we also analyzed the repeatability of the results of cluster detection. We computed for five times the clusters pertaining to class “0” and class “1” and compared the results we globally obtained on the whole test set for all the 25 different centroid sets which include all their possible combinations.

As can be observed in Figure 7.3(a), using the five different cluster sets for class



(a) (b)

Figure 7.3: (a) Variation of accuracy over different runs of the *zI*-based clustering algorithm. (b) Accuracy vs. size of the *zI*-based classifiers. The red line above highlights the reference accuracy obtained by the full 296-cluster classifier (circle on the top right-hand of the graph).

"0" does not affect accuracy at all, while some slight changes can be observed between the best and worst result for class "1", amounting to a difference in the number of correctly classified patterns of 5 or 0.17%. Considering the worst-case scenario, based on these data, we can estimate in about 0.4% the range of variability of the results we could expect if we generated five clusters for all the ten classes as well.

Finally, we compared the results of the 296 centroids detected by the *zI* method with the results obtained by Kohonen's Learning Vector Quantization (LVQ) [126] algorithm, which is a very commonly used centroid-based minimum-distance supervised classifier. Using LVQ, the user has to specify the number of centroids, so it is easy to set up experiments to compare classifiers of a certain size with similarly-sized LVQ classifiers.

We first compared the set of all 296 *zI*-based centroids with a 296-centroid LVQ classifier, running LVQ ten times. This "full" classifier exhibited an average accuracy of 98.37% (median 98.38%, worst 98.27%, best 98.50%). Then, we applied OSLVQ [32], an LVQ variant which optimizes the number of centroids during the training phase based on the results obtained on a validation set. This algorithm gave

us an idea of a reasonable optimal size and, possibly, performance, for such a classifier. Running OSLVQ ten times resulted in classifiers of size ranging from 71 to 81 centroids, with an average accuracy of 98.37% (median 98.40%, worst 98.13%, best 98.57%). Based on these results, we finally trained an 80-centroid LVQ classifier for 10 times, obtaining an average accuracy of 97.65% (median 97.77%, worst 97.13%, best 97.90%).

To obtain a classifier of arbitrary size, we needed a centroid selection strategy. Once again, we wanted an immediate implementation, not based on any *a posteriori* consideration on the distribution of the centroids. Therefore, we used directly the zI value associated to each cluster, building a $10 \times n$-cluster classifier by naïvely selecting the n highest-zI centroids for each class.

Doing so, one can observe (Figure 7.3(b)) that the performance of the classifiers increases smoothly with the classifier size, which proves that the centroids are well distributed over the pattern space, even if not as efficiently as possible. A less smooth accuracy increase, noticeable for the largest-size classifiers, is probably due both to some redundancy introduced in the centroid set, which may cause some partial overfitting effect, and to the way we selected the centroids. In fact, we kept adding one centroid per class in each step, while, certainly, the most efficient centroid distribution would correspond to a different number of centroids for each class.

This justifies with high probability the fact that a larger set of zI-based centroids seems to be needed to match the accuracy of a LVQ classifier. For instance, Figure 7.3(b) shows that we could approximately match the performance of the 80-centroid LVQ classifier (97.65%) building a 110-centroid classifier.

Fully zI-based classification

Operating as described in section 7.1.2, we applied to our test set the zI-based classifier including all 296 clusters, obtaining an accuracy level of 97.77%. As for the distance-based classifiers, we checked the performance of the zI-based clusters against random choices, choosing 296 random patterns and using them as reference clusters. This experiment was repeated ten times, with an average accuracy of 96.49% (median 96.70%, worst 94.30%, best 97.23%). Finally, as for the distance-based classification,

we created 296 clusters of random training patterns, obtaining an average accuracy of 92.08% (median 93.22%, worst 84.70%, best 94.93%).

Finally, we ranked the clusters in descending zI-value order, and tested the classification power of the subsets of these clusters, including the best-ranked ones (the best 1, 3, 5, 10, 15, 20, 22 clusters for each digit). A priori, the zI index simply evaluates the internal correlation level of each cluster and not its classification power. Nevertheless, also in this case (Figure 7.4) we can make similar remarks as for the results of the 1-NN classifier reported in Figure 7.3(b).

Other classifiers A final comparison was made with classifiers based on other principles, as provided by Weka [24]. Again, since our main goal was to confirm that the zI method is able to detect a set of clusters which can effectively be used for classification and not to reach the best possible performance, we just considered the default configurations of the classifiers we took into consideration. In particular, we classified the patterns in the test set using the following classifiers: Naïve Bayes Multinomial, J48, J48 Consolidated, SVM, Random Forest.

Table 7.1 summarizes all the results we have obtained on the real-world dataset, repeating the classification of the test set for ten times for the classifiers with a stochastic component, or repeating ten times a 10-fold cross-validation on the train-

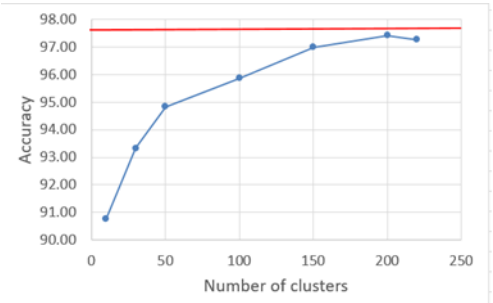


Figure 7.4: Accuracy vs. number of clusters. The red line above highlights the reference accuracy obtained by the 296-cluster classifier.

Table 7.1: Summary of the classification accuracy of the classifiers we used as references, assessed on ten runs on the test set. Above: distance-based classifiers (see Section 7.1.3); below: classifiers based on other principles. Results of classifiers marked with a star refer to ten executions of a ten-fold cross-validation.

Classifier	Mean	Median	Worst	Best
296-centroid LVQ	98.37%	98.38%	98.27%	98.50%
80-centroid LVQ	97.65%	97.77%	97.13%	97.90%
OSLVQ (71-81 centroids)	98.37%	98.40%	98.13%	98.57%
J48*	94.12%	94.15%	93.75%	94.35%
J48 Consolidated	95.72%	95.67%	95.00%	96.63%
Naïve Bayes Multinomial*	96.05%	96.07%	95.95%	96.15%
SVM*	98.54%	98.53%	98.40%	98.70%
Random Forest	99.02%	99.03%	98.93%	99.13%

ing set for the deterministic ones. The zI-based classifier reached a worse (average, if applicable) accuracy than SVM and Random Forest classifiers (98.77% and 99.02%, respectively), but better than J48, J48 Consolidated, and Naïve Bayes Multinomial (95.77%, 95.72% and 97.17%), respectively.

Even if we did not make any effort to tune the classifiers and optimize their performances, these results give an idea of the performance range that can be spanned by different classifiers on the dataset, and of the reasonably average rank of the classifiers obtained by the zI-method (98.27% and 97.77% accuracy for the 296-centroid distance-based and the fully zI-based classifiers, respectively).

7.1.4 Final remarks

The experiments reported in this section have proven that approaches derived from the RI metrics can also be used effectively as cluster detectors for pattern classification besides their "native" use for the analysis of the dynamical behavior of a complex

system.

In particular, the results of the experiments show that, without any further strategy aimed at selecting optimal cluster subsets and enhancing the classification performance, the accuracy of classifiers relying on clusters detected using the zI metric is competitive with other commonly-used classifiers.
Another application of the RI methodology to pattern clustering and classification will be presented in section 7.4 in a different context: the automatic classification of malicious users (trolls) in social networks.

The following sections investigate the applicability of the RI methodology to feature selection and feature extraction for pattern recognition. In particular, section 7.2 presents a supervised approach, while section 7.3 proposes an unsupervised approach.

7.2 Use of the Relevance Index for evolving relevant feature sets

In this section, we describe some early experiments to evaluate whether the RI methodology can be used to extract relevant feature subsets in binary pattern classification problems.

In particular, we used K-means Particle Swarm Optimization (PSO), described in section 4.2, to efficiently explore the RI search space and find the most relevant and discriminating feature subsets. We then turned such relevant subsets into a new smaller set of richer features, whose values depend on the values of the binary features they include.

This section reports the results we obtained in a simple character recognition task, comparing the performance of RI-based feature extraction and selection with other classical feature selection/extraction approaches.

7.2.1 Context

Intuitively, the property measured by the RI metrics seems not to be very different from the properties, namely relevance and non-redundancy, exhibited by the most discriminating feature sets describing patterns of interest in classification tasks. The work described in this section has been aimed at confirming this conjecture, by applying an evolutionary approach, detecting high-RI variable subsets to a simple binary character classification problem. In fact, the literature reports several examples in which evolutionary algorithms and information theory-derived criteria are the main components of effective feature-selection approaches [252].

Moreover, the integration, which is an important component of the RI metrics, can be seen as a measure of redundancy of the variables composing a feature set, as shown in [236].

In particular, considering a binary pattern classification task, we have applied the K-means PSO algorithm [173] to the exploration of the huge space of all possible feature subsets[3], looking for the subsets with the highest T_c index. Such subsets have then been transformed into a new representation, where the new features are obtained by considering each subset as a single discrete integer variable. Finally, we have compared the results obtained by applying the same configuration of a reference classifier to the original data and to the new representation, to gain some insights in support of the hypothesis that the latter is more effective and/or compact with respect to the original one.

Subsection 7.2.2 presents the experimental results obtained using the T_c index, but the same methodology can be applied also using other RI metrics. For the sake of completeness, at the end of the following subsection we will discuss also the feature sets obtained using the zI metric.

7.2.2 A case study: binary digit classification

The real-world dataset used in our experiments includes 6024 patterns representing the ten digits from 0 to 9, roughly uniformly distributed among the ten corresponding

[3] of size 2^N for patterns described by N features

classes. The patterns, extracted from the dataset of license plates described in section 7.1.3, have a size of 13×8 pixels and are represented as binary strings of 104 features.

In section 4.2, we have shown that K-means PSO is able to match the results of an exhaustive search almost perfectly for problem dimensions up to 30 variables, in fractions of the time required by the former. For larger-size problems, the results tend to be more variable, i.e., K-means PSO cannot detect all RSs in each run. However, for the application we are considering, this is less penalizing than for modeling problems, in which missing even a single component of the complex system being analyzed may totally invalidate the results.

The experiments aim at answering the following questions:

- *(Pre-processing)* How should one organize data to detect the relevant sets?
- *(Feature extraction)* How can one build new variables from the relevant sets detected?
- *(Feature selection)* How should one select the new variables?

Computing the relevant sets

When the RI method is used to detect subsystems of a complex system, the analysis cannot be based but on the values of the system status. When feature selection for classification is the goal, as in this case, most often a labeled data set is available. The question, then, is whether one should approach the problem as a plain filter approach, without taking the class information into account, or otherwise, opt for a "supervised" approach.

The former option would imply extracting RSs from all patterns in the dataset at the same time, regardless of the class to which they belong. Conversely, in the latter case, the computation of the RI metric would be repeated, class-wise, for each subset of patterns representing a certain digit.

The filter approach would have the obvious advantage of being universally applicable while unifying classical data clustering approaches with the search for RSs.

However, such an approach would have the drawback of highlighting correlations among feature sets, which may, in turn, be totally uncorrelated with the labels. This problem, to a certain extent, would be similar to the presence of a non-discriminating component, common to most data, when performing Principal Component Analysis.

However, there is also a low-level technicality, which, finally, forced us to opt for the supervised approach. What can usually be observed, in fact, is that feature sets that include the same RS tend to have T_c values that are often almost equally high. Therefore, if two feature sets that are relevant have significantly different T_c values, in a virtual ranking of all possible subsets, the distance between their rankings may be in the order of thousands.

When GPUs are used, the advantages of the fine-grained parallel processing allowed by graphic processors can be more than (negatively) compensated by the slow GPU/CPU communication, if processing is data-intensive and requires frequent data exchanges between the two processors. Because of this, to maximize computation speed, one needs to keep all candidate relevant sets in memory (at least, up to the quantity of free available GPU memory) while computing the T_c. Therefore, within the first few thousands of most relevant subsets that such a constraint allows one to keep in memory, in an extreme case, we could find only the most relevant set and all other thousands of groups that include it or differ from it just by a few elements. Moreover, the consideration that relevant feature sets that recur in different classes tend to have a higher T_c value, despite their poor discrimination power, clarifies why we finally had no better option but the supervised approach.

We ran $100,000$ iterations of PSO for each digit set, with a swarm size of 300 and 15 seeds for the K-means algorithm, executed at intervals of 500 iterations.

The experimental results confirmed that, while with the filter approach we could hardly detect a handful of sets having an independent stem within the first $10,000$ sets, we could easily detect a few dozens of independent sets using the supervised approach, keeping, in each run, only the best $3,000$ RSs in memory.

Feature extraction

After detecting the relevant sets for each class, one needs to transform them into a single variable, to remove redundancy and, possibly, reinforce their impact onto classification accuracy.

In these experiments, we have done so in the simplest and possibly most natural way, by directly turning each relevant set into a single binary string whose bits are the original binary variables belonging to the sets, and whose value is the decoding of the variable as an unsigned integer.

This choice, theoretically, has two main drawbacks. The first one is the sparse mapping of binary strings into integer values, which is such that strings, having a Hamming distance equal to one, may not preserve their neighborhood relationships when converted into integers. The other problem is the different scale of the new variables, which obviously depend on the size of the subset they represent. Both problems have rather straightforward solutions, consisting, on the one hand, using Gray coding, while, on the other hand, normalizing the values within the same range. In our experiments, however, we observed that the accuracies obtained with and without such a preprocessing were virtually the same. This is probably related with the discrete nature of such variables, which makes them substantially equivalent to nominal variables whose ordering is irrelevant, even if one uses classifiers which require numeric inputs.

Feature selection

The main goal of the experiments we have performed was to compare the results obtained using the new features with those obtained using the original data.

The main problem with the supervised approach is that, for each class, the obtained *RI* values are normalized with respect to a different homogeneous system, and therefore, despite having similar ranges, they are not rigorously comparable. Moreover, as explained above, even when retaining the best few thousands RSs extracted by the *RI* method, most RSs can be referred to a common "ancestor", i.e., the largest-*RI* RS which is included in them or includes them. Therefore, it is sometimes very

hard to identify "independent" sets of variables within such a huge number of nearly-identical or nearly-equivalent sets, even resorting to filtering out all related sets, according to the above-mentioned criterion, except the one having the largest *RI*.

In the end, this has prompted us, in this preliminary phase, to choose the relevant sets manually. In doing so, as a first step, we tried not to be too selective, and retained a rather large number of sets, almost comparable to the size of the patterns in the original datasets (79 vs. 104 features), selecting, for each class, the sets whose *RI* was within 10% of the best *RI* for each class, with a minimum of five sets per class.

A first qualitative assessment of the reasonability of the selected feature sets was done by checking where the pixels selected as relevant features were located. As expected, the features selected by the *RI* method appear to have been extracted from the regions of the patterns which are most visually distinctive for each class.

We then used the Weka [24] implementation of the Random Forest classifier (with the "default" settings of 100 random trees, each using $log_2(n_{features}) + 1$ random features) and 10-fold cross-validation on the license-plate dataset to classify the original and the transformed data. The results obtained by the two full sets of features are virtually equivalent, with slightly better accuracy obtained on the original data set (an average of 99.08% vs. 98.93% over five runs, with negligible standard deviations, possibly due to the rather large size of the dataset with respect to pattern variability).

In a subsequent experiment, we pooled together all features, new and original, and applied to the resulting 183-feature set the Weka implementation of the CfsSubsetEval feature selector with Genetic Search (population size=200; crossover rate=0.6; mutation rate=0.033; $200,000$ generations) to see which features would be selected. Two independent runs of the genetic selector resulted in two sets of 50 and 53 features, respectively, with virtually the same proportion of original and new features: $27/23$ and $28/25$, respectively. These ratios seem to speak slightly in favor of the new features, since these two ratios are closer to 1 than the ratio between the number of features in the two sets. Incidentally, the best results we obtained using the features thus selected were just slightly worse than those obtained with the full set of features: 98.77%.

When Weka InfoGainAttributeEval was used to select the attributes, we noticed

that, among the new variables, it tended to privilege features corresponding to larger sets. This is intuitively reasonable, since variables assuming more possible values may correspond to higher entropy "dynamics" with respect to "less complex" variables. Following this hint, we selected, among the new features, the 20 features which corresponded to sets of four or more original features.

After such a rather dramatic reduction of the representation size, we still obtained a reasonably good result (97.74%). However, what looks interesting, is that these features perform better (or much better) than the following feature sets:

1. The first 20 Principal Components derived from the original data (97.54%)
2. Full binary digits rescaled at 6×4 resolution (97.34%)
3. The 20 best original features according to the InfoGainAttributeEval filter (93.47%)

zI analysis

The identification of relevant feature sets can be performed also using the zI metric.

Figure 7.5 shows the pixels belonging to the highest-zI RSs, separately computed for the digits from 0 to 9. The yellow pixels correspond to the elements (pixels) that appear at least once in the first 500 positions of the RS ranking. To better appreciate the results, the image on the right side of Figure 7.6 highlights with the use of different colors some of the clusters identified for digit 2.

It can be seen that the most morphologically distinctive features of the digits are often clearly highlighted. Indeed, the more relevant subsets contain those elements of the images that define the shape of the number. These elements represent the borders of the numbers, i.e., those pixels who vary together in a "dynamical" way. The properties that the RI metrics highlight in time series, when analyzing the dynamics of complex systems, can be assimilated to the properties of the multivariate distribution of static patterns subject to translations, noise, etc.

Furthermore, we also compared the RSs identified using T_c and zI. Figure 7.7 shows the comparison between the results obtained on digit 2 and digit 5 using the zI metric (on the left) and the results obtained using the T_c index (on the right). We

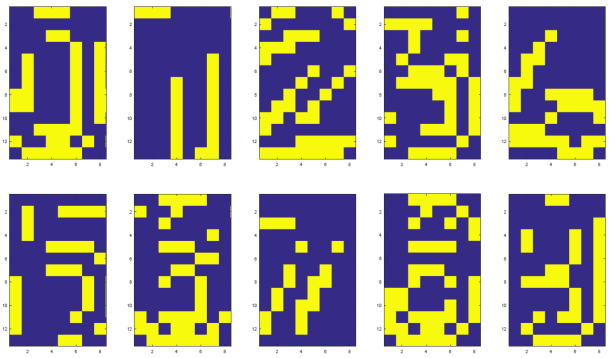


Figure 7.5: Visualization of the pixels of the RSs for all the digits from 0 to 9. The yellow pixels represent elements that appear at least in one of the first 500 RSs.

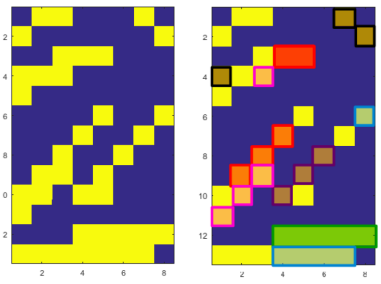


Figure 7.6: Visualization of the RSs identified for digit 2. The yellow pixels represent elements that appear at least in one of the first 500 RSs. On the right side, colors are used to highlight some of the RSs.

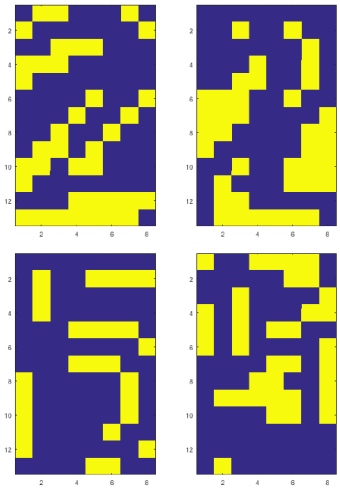


Figure 7.7: Visualization of the RSs of digit 2 (upper part) and digit 5 (bottom part). The yellow pixels represent elements that appear at least in one of the first 500 RSs. Left: zI index. Right: T_c index.

already discussed in section 3.3 the fact that these two indices are not equivalent and this comparison confirms that observation. In particular, the most significant groups identified by T_c also have to meet the requirement of being barely dependent on the complementary set of elements in the system. The RSs identified by the zI visually represent more morphologically distinctive features of the digits. Sections 7.3 and 7.4 will present the experimental results obtained in classification tasks, using zI based methodologies for feature extraction and preprocessing, respectively.

7.2.3 Final remarks

Answering the title question with a clear "Yes" would be probably over-optimistic, at the present stage of the research. However, the results seem to support the idea that feature extractors/selectors may be built using RI metrics as a criterion.

The supervised approach presented in this section is computationally expensive, since the RI analysis is repeated for each subset of patterns belonging to a certain

class.
The following section presents an unsupervised *zI*-based approach to feature extraction/selection, which allows one to analyze the whole dataset at the same time.

7.3 Experimental assessment of an unsupervised feature extraction method based on the RI metrics

The goal of the work described in this section is to apply the principles of the RI methodology to pattern recognition in an unsupervised way, checking whether the RI metrics can also identify, in a high-dimensional feature space, attribute subsets from which it is possible to build new features which can be effectively used for classification.
Preliminary results indicating that this is possible have been presented in section 7.2 using the RI metrics in a supervised way, i.e., by separately applying such metrics to homogeneous datasets comprising data instances which all belong to the same class, and iterating the procedure for all possible classes taken into consideration.

In this section, we check whether this would also be possible in a totally unsupervised way, i.e., by considering all data available at the same time, independently of the class to which they belong, under the hypothesis that the peculiarities of the variable sets that the RI metrics can identify correspond to the peculiarities by which data belonging to a certain class are distinguishable from data belonging to different classes. The results we obtained in experiments made with some publicly available real-world datasets, which will be presented in section 7.3.4, show that, especially when coupled to tree-based classifiers, the performance of a RI metrics-based unsupervised feature extraction method can be comparable to or better than other classical supervised or unsupervised feature selection or extraction methods.

7.3.1 Context

As shown in section 7.2, the properties that the *RI* metrics highlight in time series, when analyzing the dynamics of complex systems, can be somehow assimilated to the

properties of the multivariate distribution of static patterns subject to noise, translations, etc. Results indicating that this property can be exploited in pattern recognition applications, in particular for feature extraction, have been obtained in the previous section, where the attribute subsets which could be detected using the RI metrics were shown to be the base for extracting effective features for classification. In that case, the RI methodology was used in a "supervised" way. Despite yielding good results, such a supervised approach is impractical, since it requires that the already resource-demanding phase of RI analysis be repeated, at the same level of complexity, as many times as the number of classes [4]. However, if the analysis based on the RI metrics is able to detect the peculiarities characterizing the variations within data belonging to the same class, it may also be able to detect the peculiarities characterizing one class with respect to the others. This suggests that an "unsupervised" approach, where the method is applied to the whole dataset, independently of the classes to which data belong, may be as successful.

Based on the above considerations and building upon the positive indications of the results reported in section 7.2, the work presented in this section has been aimed at significantly extending that work from two points of view:

- Using an unsupervised approach to extract relevant features that can be used to solve supervised classification problems.
- Perform an in-depth analysis of the results that the zI-based approach to feature extraction can yield, by applying it to data from different real-world case studies.

Therefore, the main goal of this section is to transfer approaches based on the RI metrics from the domain of complex system analysis to feature extraction. Thus, the work presented in this section could be considered at least partially successful even if only the hypothesis of direct applicability of a method originally devised for

[4] Applying the RI analysis to a smaller number of data instances leads only to a negligible reduction of computation time as the intrinsic complexity of the RI analysis is $O(2^s)$, where s is the number of features describing the data and not the number of data samples.

complex system analysis to feature extraction could be experimentally demonstrated to be reasonable.

From now on, in this section, we will refer to the unsupervised *zI*-based method we have tested as ZIFF (*zI*-based Feature Finder). In ZIFF, the *zI* computation is carried out by HyReSS, which has been described in section 4.1, to counteract the "curse of dimensionality" and find the most relevant variable subsets in reasonable time. To further aggregate the variable sets and discard redundant ones, we adopt the iterative "sieving" procedure described in chapter 5.

Running experiments on three different datasets and using three different families of classifiers (tree-based, distance-based, and support vector machines), we could show that the compressed representations of data which can be obtained using ZIFF, even using possibly the simplest possible feature encoding can, in some cases, already yield results that are, on average, competitive with other feature extraction method, or even outperform them. As we will underline in section 7.3.4, all the feature extraction methods we used as references had, in principle, some *a priori* advantages over our method, in terms of the information available for constructing the reduced feature sets, being either supervised methods or methods in which each feature is a recombination of the whole set of features by which the original data are represented.

In the following subsection we introduce the background about feature extraction, focusing on unsupervised methods.

7.3.2 Background

Finding algorithms capable of effectively reducing the number of features (attributes) representing data in classification tasks is a central problem in pattern recognition. Theoretically, the presence of many features offers the opportunity to develop classifiers having better discriminating power. However, this is not always true in practice, because not all features are as important for understanding or representing the underlying phenomena of interest. Thus, reducing the number of attributes, or creating new more informative ones, is an essential pre-processing step in classification that may have several beneficial effects, such as lowering the complexity of the classifier model, reducing overfitting, increasing the interpretability of the results, reducing the

actual cost of feature collection and pre-processing, resisting noise, and, sometimes, improving the accuracy of a classifier.

There exist three possible approaches to modifying the number of features s, typically referred to as feature selection, feature extraction, and feature construction [157]:

- *Feature selection* chooses a subset of m features out of the original s ones;
- *Feature extraction* creates a set of $m(< s)$ new features from the original attributes through some function mappings;
- *Feature construction* augments the space of features by inferring or creating additional attributes.

These approaches can be further classified into supervised and unsupervised methods. The former include methods that take into account, somehow, information about the classifier or the class to which the training data belong. The latter do not rely on such information but just on the intrinsic properties of the dataset.

In the following we briefly review articles mainly regarding *unsupervised* feature extraction methods, with particular emphasis on those based on information theory.

Starting with methods which do not rely on Information Theory, the authors of [257] introduce a method for unsupervised feature extraction in time series clustering. This method relies on an orthogonal wavelet transform to perform feature extraction and determining also the appropriate dimension of the optimal feature set in an automated way. The sum of squared errors between the cluster centroids represented according to the new features and the original time series computed on five benchmarks is the criterion used to assess the quality of the resulting clustering.

More recently, in [183] a generic framework for both feature extraction and data classification of hyperspectral images has been presented. The technique tries to jointly optimize the features and the corresponding classifier. In [83], the authors employ Gaussian feature extraction for tracking particles in images produced by CMOS camera sensors. The method is verified on a channel flow, using a boosted regression tree ensemble, and tries to fit generalized Gaussian distributions to particle images.

The contribution in [256] uses a projection learning method, namely a low-rank and sparsity preserving embedding for unsupervised learning, which simultaneously learn the graph and the optimum projection by exploiting both global and local information of data. In [258], the authors apply an unsupervised deep-learning framework to reduce the dimensions of a dataset. This is achieved by aligning local features and then transferring them from local coordinates to global coordinates. Jiang et al. [115] try to apply a novel super Principal Component Analysis method to extract features from hyperspectral images in an unsupervised way, while in [208] a semi-supervised method, based on deep learning, is used to extract features from colon tissue images. The semi-supervision of the latter derives from the exploitation of domain-specific prior knowledge to identify salient subregions in an image, in order to pre-train a deep belief network.

Moving away from methods focused on image analysis, in [227] an extension of Principal Component Analysis, namely a tensor decomposition, is applied to miRNA transfection experiments in order to identify critical genes, whereas in [226] the same author applies a similar method, together with pure Principal Component Analysis, to represent gene expression in the brains of social insects.

The contribution in [116] compares different linear unsupervised feature extraction methods (Principal Component Analysis, Projection Pursuit and Band Subset Selection) for the classification of high-dimensional multispectral and hyperspectral data. The authors introduce also an optimized version of Projection Pursuit, called Optimized Information Divergence Projection Pursuit (OIPP), which maximizes the information divergence between the probability density function of the projected data and a Gaussian distribution. The aforementioned methods are compared by means of both supervised and unsupervised classifiers over different datasets where OIPP shows a good performance in contrasting the Hughes phenomenon as well as the curse of dimensionality.

The usage of Information Theory metrics for unsupervised feature extraction dates back to the nineteen-eighties, when unsupervised feature extraction was employed for face categorization by means of an image compression network [80]. Another example is the work of Fisher and Principe [79], which exploits mutual infor-

mation with the same aim. Since mutual information is composed of two entropy terms, the algorithm they propose tries to maximize one entropy term and, at the same time, minimize the other one, seeking a certain parameter that determines the distance of the observed distribution from a uniform distribution. To this aim the authors employ a Parzen window method as an estimator of the output distribution and the integrated square error as a distance method, as well as an indirect measure of the entropy. In such a way, even if the authors employ mutual information, the method is unsupervised, since it enables entropy manipulation modeled as local interaction between observations in the output space.

In [96], information-theoretic criteria are used for unsupervised image clustering. They exploit the information bottleneck principle and the mutual information between the clusters and the image content. The principle states that the desired clustering is the one that minimizes the loss of mutual information between the objects and the features extracted from them. The work exploits also a mixture of Gaussian distributions for modeling continuous images and, like our work, measures the quality of the resulting clustering by means of supervised classification.

In [105] the authors use, as a criterion, the maximization of an approximation of the mutual information between class labels and feature extractor with the aim to train a feature extractor independently of the classifier. The unsupervised characteristic results from the usage of Renyi's quadratic entropy in the computation of mutual information and from its nonparametric low-complexity estimation. The information-theoretic criterion employed for the estimation forces the feature extraction matrix to be a rotation matrix.

In [217] the authors study the problem of unsupervised domain adaptation, trying to detect domain-invariant features and, at the same time, construct the classifier model. This is obtained by optimizing mutual information, using discriminative clustering and simple gradient-based methods. The optimization both maximizes domain similarity (the negated mutual information between all data and their binary domain labels) and minimizes the expected classification error on the target domain (the negated mutual information between the target data and their cluster labels). The proposed solution performs better than other unsupervised methods such as Principal

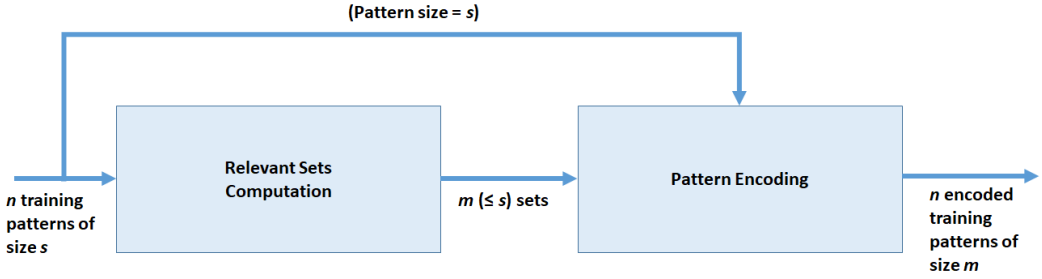


Figure 7.8: Data flow of the feature extraction method

Component Analysis, Structural Correspondence Learning and Transfer Component Analysis.

Furthermore, as shown in [236], the integration, which is the basis of the zI metric, can be seen as a measure of redundancy of the variables composing a feature set. The following subsection describes ZIFF, the unsupervised zI-based method we propose for feature extraction.

7.3.3 zI-based feature extraction

Let us consider a dataset $D \equiv \{\mathbf{X}_i(S) \in \{0,1\}^{k\times s} \times (\mathscr{L} \in \mathbb{N}), i = 1,\ldots,n\}$ whose elements are instances of a schema $S(a_1,\ldots,a_s,a_{s+1})$ in which data correspond to a set of k-bit binary features $A \equiv \{a_1,\ldots,a_s\}$. The class label a_{s+1}, if specified, is defined over a finite set of symbols, generalizable as a natural number subspace.

ZIFF derives m ($s \geq m$) new features from D. It only relies on A, i.e., it does not take the class label into account, and can therefore be termed as *unsupervised*, as opposed, for example, to the *supervised* approach we analyzed in section 7.2.

ZIFF is a two-stage method which produces as output a new representation $\hat{D} \equiv \{\mathbf{Y}_i(\hat{S}) \in \mathbb{R}^m \times (\mathscr{L} \in \mathbb{N}), i = 1,\ldots,n\}$ of the n input instances in D according to a new schema $\hat{S}$ defined over the m newly created features plus the class label.

The first stage of the method consists in the zI-based computation of the relevant variable sets $RS_j \subset A, j = 1,\ldots,m$ such that $RS_j \cap RS_t = \emptyset$ for $j \neq t$ and $\bigcup_{j=1}^{m} RS_j = A$.

The computation is made according to the sieving algorithm described in chapter 5, whose basic step is performed by the evolutionary algorithm described in 4.1. Each instance $\mathbf{X}_i$ of D is therefore "decomposed" into a set of binary strings $\mathbf{X}_{i,j}(RS_j)$ of size $k \times |RS_j|$ with $|z|$ representing the number of elements in z.

In the second stage we transform subsets $\mathbf{X}_{i,j}(RS_j)$, $i = 1,\ldots,n \;\; j = 1,\ldots,m$ of the original dataset into corresponding numeric features $\mathbf{Y}_{i,j}(RS_j) \;\; i = 1,\ldots,n \;\; j = 1,\ldots,m$. Therefore, the feature-construction process produces a new dataset, derived from the original one and of lower dimensionality, which is expected to retain as much as possible of the original information content which is useful for developing effective classifiers.

As the results obtained in our experiments will show, the feature-encoding phase is critical for the performance of the classifiers which can be generated using the transformed datasets as training and test sets. In the following section, we describe the encodings we have tested in our experiments. In section 7.3.4, while reporting the experimental results of our tests, we will discuss them more in depth, with particular regard to their compliance with the different classifiers we have taken into consideration.

Possible encoding schemata

The most direct (and, possibly, naïvest) way of encoding a binary string into a single numerical feature is transforming it into its corresponding integer representation.

The simplicity and easy applicability of this encoding is compensated by two main drawbacks:

1. It depends on the order in which bits are taken into consideration: bits that are remapped as more significant have a larger weight in the final encoding. The ratio between the weight of the most significant bit (MSB) and the least significant bit (LSB) increases exponentially with the number of bits, up to making the LSB virtually irrelevant.

2. Topological relationships are not preserved. Neighborhoods of the encoding of a given string S may not (and usually do not) include the encoding of the same

elements included in the neighborhood of S.

Despite these drawbacks, some kinds of classifiers, not surprisingly those which are able to "segment" the pattern space into a set of small intervals like tree-based classifiers do, seem to be mostly insensitive to such problems, as we will show in the next section. Because of this nice compliance between such an encoding and tree-based classifiers, in the experiments described therein we report results based on this encoding schema. In fact, the main goal of this section is showing that the iterated computation of the zI index is able to identify sets of variables by which one can build features that are relevant to classification rather than an exhaustive exploration of the whole space of possible encodings. In using this representation, we only take care of the order in which the variables/features are packed into a bit string, setting as most significant bits the attributes that have been grouped earlier, since, in each iteration, our sieving algorithm extracts the group that is deemed most relevant based on its zI value.

An immediate amendment of the previous schema consists in normalizing the features obtained as above by dividing each feature $Y_{.,j}$ by $2^{k\times|RS_j|}$, k being the length in bits of the binary representation of features X_i in the original dataset, such that all new features Y_i are bounded within the interval $[0,1]$. The more evident and usually positive effect of normalizing a dataset, especially when distance-based classifiers are being used, is mainly guaranteeing that no feature is more relevant than the others just because of its scale. In our approach, this effect can be appreciated i) as an equalizer between the transformed pattern components, whose scale, and therefore relative weight, depends on the number of elements in the relevant set from which it derives, but also ii) as an equalizer between each element in a relevant set, whose relative weight depends on its position within the bit string obtained by packing the values of all elements in the set. Finally, input data normalization is sometimes a requirement for certain kinds of classifiers.

Since normalization actually appears to bring a general improvement of the results, with virtually no negative side effects, we have normalized data representations by default in all our experiments.

7.3.4 Experimental evaluation on real-world datasets

In this subsection, we describe the experiments we made to evaluate whether the analysis based on the RI metrics can be effectively extended from the domain of complex system analysis to feature extraction in classification problems. In particular, we compare the results obtained by ZIFF to other supervised or unsupervised feature extraction and selection methods using real-world data from three test problems.

Considering the goals of our research, in setting up our experiments, we expected the results we would obtain to answer the following questions:

- Are the properties of the variable sets detected using the zI index, which have been already demonstrated to be effective in their original role of detecting functional blocks in complex systems, as relevant when such sets are used to build classification-oriented representations?
- How well do such representations compare with other representations which can be obtained by more classical feature selection/extraction methods?
- How much data- or classifier-dependent are the results provided by such representations?

In the rest of this subsection, we first describe the experimental setup and then we discuss the results of our tests, considering, in particular, the classification accuracy which can be obtained using representations of different dimension provided by ZIFF and by the other feature extraction/selection methods taken as references, when coupled with a selected sample of classifiers of different kinds, both as regards the data types they process and the classification criteria.

Experimental setup

The experimental setup has been the same for all three tests in which ZIFF and the reference methods have been applied to different datasets. In particular, for each set of features extracted by ZIFF in each iteration: i) a feature set of the same size has been extracted or selected using each reference method; ii) for each iteration of ZIFF

and for each feature extraction or selection method, a corresponding representation of the reference datasets (a training and a test set) has been computed; iii) each training set thus obtained has been used to train four classifiers of different kinds; iv) the classifiers have been finally validated on the corresponding test set by computing their classification accuracy.

Feature extraction/selection methods We selected the following four feature selection/extraction methods as the references to be compared to ZIFF:

- Principal Component Analysis (PCA) [21], which uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values representing the projection of the data onto an orthogonal base of linearly uncorrelated principal components;
- Information Gain (IG) [184], which is the conditional expected value of the Kullback–Leibler divergence of the univariate probability distribution of one variable from the conditional distribution of such a variable given the other ones;
- Chi-squared test (X^2) [98], a statistical hypothesis test where the sampling distribution of the test statistic is a chi-squared distribution when the null hypothesis is true;
- Univariate feature selection (F_VAL) [67], a feature selection method based on univariate statistical tests.

One should notice, on the one hand, that, except for PCA, all these methods are feature selection methods. This fact offers PCA (and ZIFF) the advantage to be able to create new features that derive from more than one of the original ones, thus having higher potential information content. On the other hand, one should also consider that, while PCA and ZIFF are unsupervised methods, the feature selection methods taken into consideration are all supervised and aimed explicitly at maximizing classification accuracy, which often more than compensates for the above drawback.

Classifiers As a criterion to compare the five feature selection/extraction methods, we used the classification accuracy of the following four classifiers, trained and tested using, as data representation, the feature sets extracted by each of the above methods and by ZIFF:

Tree-based

- Random Forest (RF) [26], an ensemble learning method that constructs a multitude of decision trees at training time and outputs as its final decision (i.e., class) the mode of all the decisions taken by the trees;
- Decision trees (C4.5) [185], a tree classifier where nodes hierarchically partition the pattern space, each node based on the values of a single feature, until a leaf node, representing a decision, is reached.

Distance-based

- K-Nearest Neighbors (KNN) [21], a non-parametric method where an object is classified by applying a majority vote strategy to the decisions of its neighbors in the training set, resulting in the object being assigned to the class most frequently represented by its K nearest neighbors.

Support Vector Machines (SVM)

- Radial Basis Function–SVM [21], a supervised learning model where the examples, represented as points in the pattern space, are mapped using a Radial Basis Function kernel, such that the margin (distance) between boundary examples representing different classes is maximized.

Datasets The three different real-world datasets, onto which we focused our attention, are described in the following.

The first dataset (DS_1)[5] is the same used in sections 7.2 and 7.1. It contains low-resolution (13×8) binary patterns, representing digits from 0 to 9. It includes $11,034$

[5] ftp://ftp.ce.unipr.it/pub/cagnoni/license_plate

Name	Data Type	# Input Features	# Classes	Training Data	Test Data
DS_1	B	104	10	6,024	5,010
DS_2	N/B	64	10	3,823	1,797
DS_3	N	60	3	2,230	956

Table 7.2: Description of the three datasets used in our tests (B=binary N=Nominal)

patterns composed of strings of 104 binary features, roughly uniformly distributed among the ten classes under consideration. We used part of the full dataset as a training set with $6,024$ patterns, and the remaining $5,010$ patterns, exactly 501 per digit, as a test set.

The second dataset (DS_2) is the *Optical Recognition of Handwritten Digits* dataset from the UCI machine learning repository[6]. The dataset contains $5,620$ samples of bitmaps of hand-written digits from 0 to 9: 32×32 bitmaps have been divided into non-overlapping blocks of size 4×4 and the number of pixels has been counted in each block. This quantization has produced input patterns of 8×8 pixels, each of which may have an integer value in the range $[0,16]$. In our experiments, we considered a 4-level discretization of the original 16 values; as a consequence, each image of the dataset is an instance with 64 numerical features assuming values in the range $[0,3]$, and one class attribute with values in $[0,9]$. The training set comprises $3,823$ instances, while the test set $1,797$ instances.

The third dataset (DS_3) regards splice-junctions in gene sequences of the DNA[7]. It consists of $3,186$ data points (*instances*) described by 180 indicator binary variables (*nominal features*) and by one class label assuming three possible values (*ei, ie, neither*). The classes represent the boundaries between exons (the parts of the DNA sequence retained after splicing) and introns (the parts of the DNA sequence that are spliced out). This dataset is a processed version of the one contained also in the Irvine

[6]https://archive.ics.uci.edu/ml/datasets/optical+recognition+of+handwritten+digits
[7]https://www.openml.org/d/40670

database[8] in which the names of the examples, along with the ambiguous instances (only four, actually) were removed. In the dataset we used in our experiments, the four nitrogenous bases (Adenine, Guanine, Thymine, Cytosine) were replaced by an integer label from the set 0,1,2,3 to make it easier to obtain the numeric encoding of the features extracted by ZIFF. We considered $2,230$ sequences as a training set and the remaining 956 as a test set.

Evaluation procedure The test procedure we adopted to compare the performance of ZIFF to the reference methods can be summarized as follows:

1. *Compute the Relevant Sets by iteratively applying HyReSS to the training set*. Starting from an initial representation in which all *RSs* correspond to a single feature of the original dataset, in each iteration we merge into a new set the *RSs* from the previous iteration whose union has the largest possible *zI* value. From then on, and until it is involved in a new merger, HyReSS will treat the newly-created variable set as a single variable (group variable) in the representation of data, which will substitute its components in the *RS* list. Thus, after each iteration, the number of variables (*RSs*) used to represent data decreases while the average size of the *RSs* increases.

2. *Encode the training and test data*, as described in 7.3.3, according to the feature sets extracted up to the current iteration of HyReSS.

3. For each reference method, and for each iteration of ZIFF, *consider the feature sets*, extracted or selected by the method under consideration, having the same size as the number of *RSs* found by ZIFF, *and compute the corresponding representation* of the benchmark datasets.

4. *Apply the classifiers to such representations* for testing the effectiveness of the corresponding feature set using the classification accuracy as a quality criterion.

[8] https://archive.ics.uci.edu/ml/datasets/Molecular+Biology+(Splice-junction+Gene+Sequences)

5. *Stop iterating* when the *zI* of the new group variable deriving from the merger "proposed" by HyReSS falls below a pre-set threshold.

To perform the *zI* computation as efficiently as possible, HyReSS was implemented in C++ and the corresponding GPU-based fitness function, which computes the *zI* index, in CUDA C [1]. Instead, the code we used for our tests was written in Python to take advantage of the scikit-learn[9] Python package implementation of all the necessary classifiers and reference feature extraction/selection methods[10].

Since our goal was to compare the effectiveness of the representations obtained with different feature extraction/selection methods under controlled conditions, and not to obtain the highest possible performance, when applying a classifier to our benchmarks we used the default classifier configuration proposed by scikit-learn. Therefore, when analyzing the results, it is important to mainly focus on the relative ranking between the different methods, independently of the absolute quality of the results.

To take into consideration the intrinsic stochastic nature of Random Forest and a heuristic that has been introduced in the scikit-learn version of C4.5, which makes the results of that classifier stochastic as well, we repeated all experiments using such classifiers five times and computed the average accuracy over the five independent runs.

In this regard, it should also be noticed that the outcome of HyReSS, which is a stochastic optimization algorithm, was stable over the five repetitions, i.e., it always found the same variable sets.

Finally, when considering the KNN classifier, we preliminary tested K values equal to 1, 3, and 5. Since no substantial differences could be observed which would affect the comparison between ZIFF and the references, we only report the graphs related to $K = 1$.

[9] `https://scikit-learn.org`

[10] Using scikit-learn, the C4.5 algorithm is implemented as an object belonging to the DecisionTreeClassifier (`DecisionTreeClassifier(criterion=entropy,...)`) class.

Results and comparisons

A preliminary interesting observation which can provide some hints on the sensibility of the features that ZIFF extracts is that, according to the structure of the data from which they have been extracted, they tend to correspond to contiguous regions, and seem to have the same role as focus-of-attention algorithms have in computer vision. This is true for both the data representing 2-dimensional patterns (DS_1 and DS_2) and 1-dimensional DNA sequences (DS_3). Figure 7.9 highlights the variable sets identified by ZIFF after the 70^{th} iteration of the sieving algorithm has been applied to DS_1. Different colors represent different groups of pixels identified by ZIFF. As can be seen, the groups represent connected subparts of a digit.

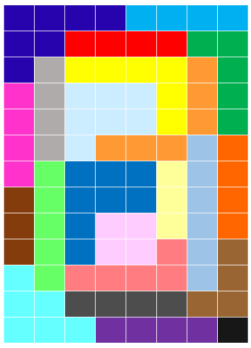
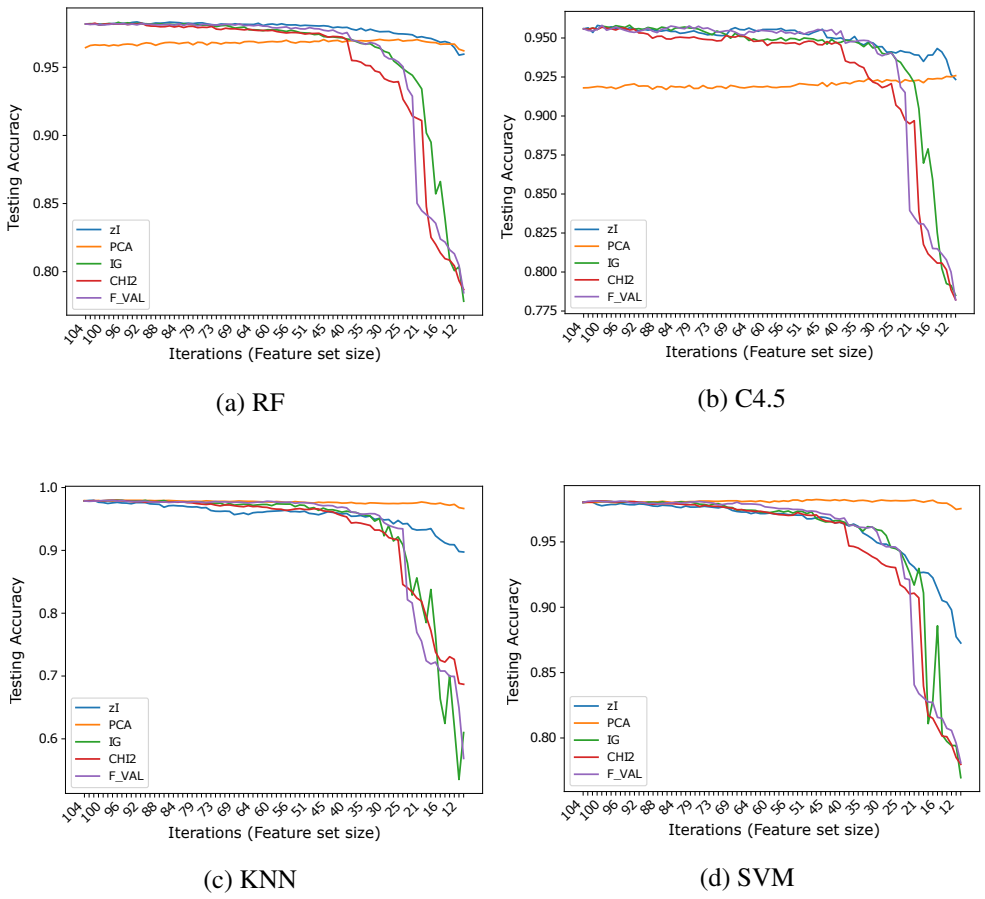


Figure 7.9: Relevant variable (pixel) sets identified in the DS_1 after the 70^{th} iteration of ZIFF. Pixels of the same color belong to the same relevant set.

As regards the quantitative assessment of the quality of the feature sets extracted by ZIFF and by the reference methods, following the procedure described above, the results obtained by ZIFF and by the reference methods can be summarized plotting a curve $C_{i,j,f}$ for each feature extractor/classifier pair. A point of $C_{i,j,f}$ represents the accuracy value which is obtained when classifier i is applied to dataset j represented by the feature set extracted by method f, having a size equal to the number of features extracted by ZIFF in each iteration.

In particular, for each iteration, we collect and plot the classification accuracy on a 2D graph having the size of the feature set (decreasing) or the iteration number

(increasing) along the x axis and the classification accuracy along the y axis.



(a) RF

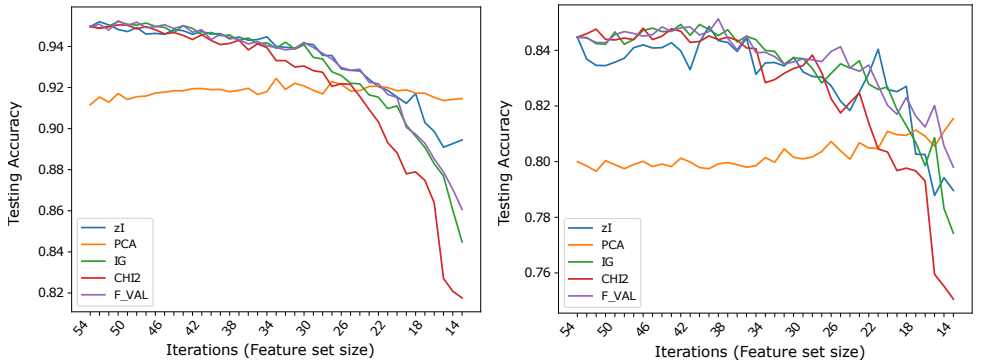
(b) C4.5

(c) KNN

(d) SVM

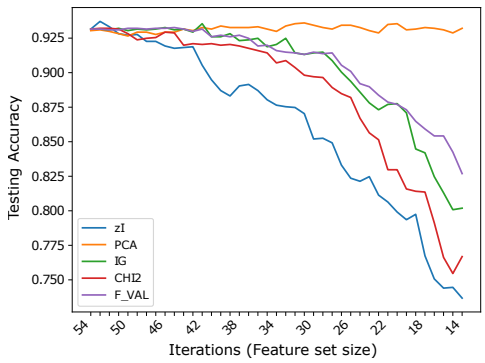
Figure 7.10: Accuracy vs number of iterations (and number of features) for different classifiers using DS_1.

Figures 7.10, 7.11, and 7.12 show the results (accuracies) we have obtained in our tests, subdivided by dataset. Ticks on the x axis represent ZIFF iterations, increasing from left to right, while the corresponding numerical labels represent the number of features extracted in each iteration (the first iteration includes all the original features). These graphs allow one to compare our method and the reference methods

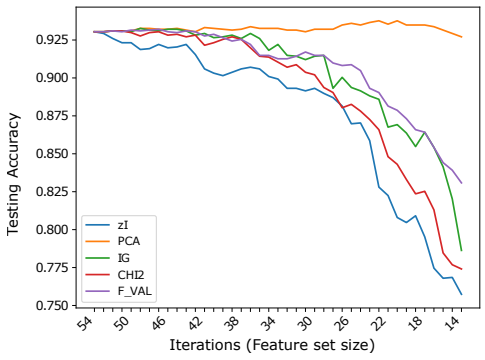


(a) RF

(b) C4.5

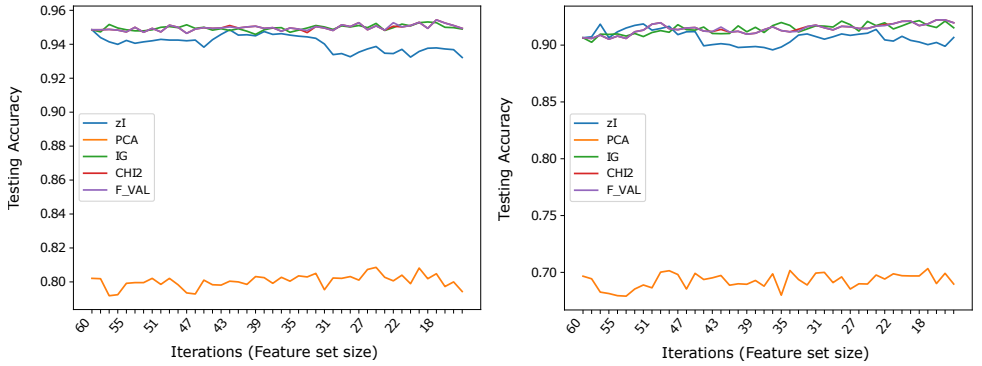


(c) KNN

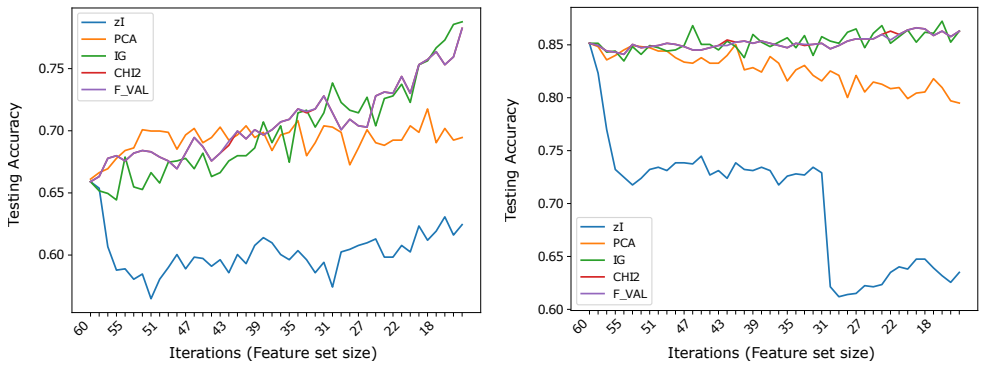


(d) SVM

Figure 7.11: Accuracy vs number of iterations (and number of features) for different classifiers using DS_2.



(a) RF (b) C4.5



(c) KNN (d) SVM

Figure 7.12: Accuracy vs number of iterations (and number of features) for different classifiers using DS_3.

over the same data, assessing the capacity of each method to produce representations which preserve the information that is most relevant to classification, as the dimensionality of the representation decreases.

Thus, each figure embeds four graphs; each graph within each figure corresponds to a dataset/classifier pair, and each plot within each graph corresponds to a different data representation, given by the specific feature extraction/selection method.

We will first provide a straightforward description of the results obtained on each dataset and then comment on them, trying to justify - with regard to ZIFF's limitations, the nature of data, and the nature of the classifiers - the rather large differences between the performance obtained using different feature extraction and selection methods as well as by different classifiers.

The results obtained on DS_1 seem to strongly support the hypothesis that the variable (pixel) sets identified by the *zI* index actually correspond to features that are relevant for classification. When RF is used as a classifier, ZIFF is the best performing method with virtually all feature set sizes except for the very last few iterations (feature set sizes approximately below 15), where PCA slightly outperforms it (by less than 1% in the very last iteration). The feature-selection methods taken as references perform very closely to ZIFF down to a feature set size of about 40, but show a dramatic decrease soon after that threshold. This decrease is probably "physiological", considering that the features they select are single bits, with respect to real numbers which condense the information carried by a relevant number of original feature as happens with PCA, whose features combine all the input features, and, to a minor extent, with ZIFF. The results obtained on the other tree-based classifier, C4.5, are very similar, even if generally worse than those obtained by RF, with a performance decrease which is particularly remarkable (4.8% on average on all iterations) for PCA.

Regarding the results which can be obtained using K-NN or SVM classifiers on DS_1, the same considerations can be made about the performance of feature extractors versus feature selectors. The accuracy of IG, X^2, and F_VAL is virtually the same as with RF and C4.5 and the critical thresholds under which their performance decreases is also the same. PCA also works as well as before, yielding a performance that is

virtually constant with all feature set sizes, up to the minimum number we have taken into consideration, which corresponds to the number of sets found in the last iteration of ZIFF. ZIFF, with these classifiers, has a behavior which is intermediate between PCA and the supervised feature selectors, still showing better robustness with respect to feature set size than the latter, but a worse performance than the former.

On DS_2, once again, even if less markedly, the features found by ZIFF are the best-performing with RF except for the last iterations (feature size below 20) where PCA is better (of about 2% in the very last iteration), and are very similar to the performance of the feature-selection methods with C4.5. However, on this dataset, RF is much more accurate than C4.5. This is most probably caused by the higher intrinsic difficulty of this task (i.e., to the higher variability of data), since RF is capable to create a much finer segmentation of the input domain than C4.5.

ZIFF's performance is definitely the worst with the classifiers (1-NN and SVM) which usually (or preferably) deal with continuous inputs which reflect data properties for which the computation of Euclidean distances or other topological relationships makes sense.

The inappropriateness of the encoding of ZIFF when used along with K-NN classifiers using the Euclidean distance or SVM is even more clearly highlighted by the results obtained on DS_3. In this case, the results obtained using K-NN as a classifier can probably be totally neglected for the incompatibility of such a classifier with data represented by nominal attributes, as we have already remarked. We report such results just for the sake of completeness. Regarding the results obtained with the tree-based classifiers on DS_3, the supervised feature selectors are the best representations here, slightly outperforming ZIFF by about 1% on average (roughly 95% versus 94%). ZIFF still exhibits a remarkable robustness with respect to the feature set size, showing that, when coupled with tree-based classifiers, it is also very robust with respect to the representation of the input data. This is not the case for PCA, whose performance, as could be expected, is very negatively affected by a representation based on nominal features. Only with SVM, PCA accuracy is systematically above 80%, which is still more than 10% worse than the performance of the feature selectors and zI with RF.

A few observations can be made after analyzing the results reported above, regarding, in particular, the datasets, ZIFF's encoding, and the classifiers.

A first observation is that results obtained on DS_1 are generally more "stable" and reproducible. This is most probably related to the intrinsic nature of the data, which makes their classification an easier task than classifying the two other datasets, as the best classification results which can be obtained seem to confirm. In fact, quite obviously, although both DS_1 and DS_2 represent the same kind of data (2-dimensional patterns representing the figures from 0 to 9), DS_2 represents handwritten characters which are characterized by an intrinsic variability which is much larger than for data in DS_1. Considering also that DS_1 includes a larger number of samples, our observation finds a rather straightforward justification.

Regarding the intrinsic properties of data, representations based on nominal attributes affect negatively the performance of distance-based classifiers like K-NN. Applying the PCA to such a kind of data by treating the nominal indices as integers actually makes little sense.

In particular, if we consider ZIFF's problems with K-NN and SVM with respect to the good performance with tree-based classifiers, they may be caused by the lower effectiveness of the features selected by ZIFF and/or by the possible problems related to the compatibility of the encoding scheme with classifiers usually applied to real number representations, which we have highlighted in section 7.3.3.

The analysis of the results obtained on DS_2 and DS_3 seem to support the second hypothesis quite clearly. In fact, ZIFF's encoding does not preserve distances, and tends to scatter compact neighborhoods in the original dataset into a myriad of small intervals in the 1-D representation onto which ZIFF maps the original data. This problem affects minimally, if not at all, the performance of tree-based classifiers, whose way of working actually corresponds to a partitioning of the input space into a large number of regions which are labeled after one of the possible classes. The fact that RF performs much better than C4.5 is most probably related with the much higher capacity of RF to identify small regions, owing to the large number of trees ($1,000$ as default value in our tests) by which it is composed with respect to the single, even if larger, decision tree generated by C4.5.

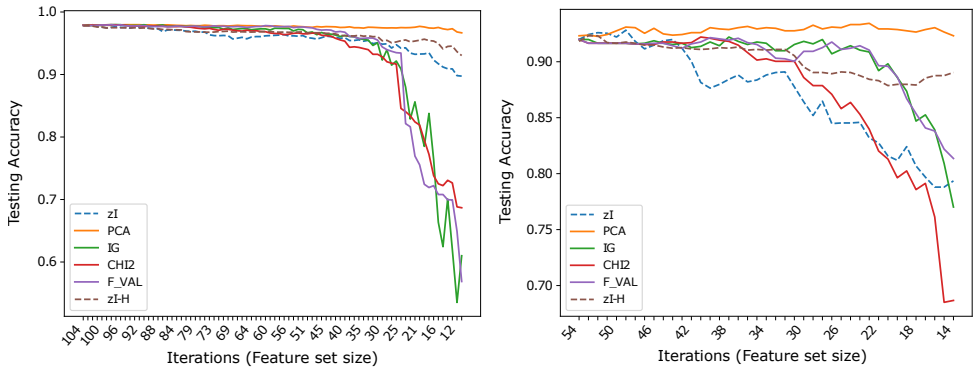


Figure 7.13: Results obtained by a 1-NN classifier applied to DS_1 (left) and to DS_2 after data binarization (right). The two dashed curves in each plot highlight the improvement that can be obtained computing a generalized Hamming distance to classify zI-based representations of binary data (zI-H), with respect to using the Euclidean distance (zI).

Regarding the problems that ZIFF has with K-NN classifiers based on the Euclidean distance, they are mostly attributable to the distance function adopted, owing to the mentioned lack of topological consistency between the original data representation and the encoding used in ZIFF. This can be shown if we consider datasets that are originally binary, and their K-NN classification based on ZIFF's encoding and on a distance that can be considered a generalization of the Hamming distance, defined as follows:

$$\hat{H}(\mathbf{x},\mathbf{y}) \;=\; \sum_{i=1}^{N} \delta_{x_i,y_i}$$

where x and y are N-dimensional vectors and $\delta_{x_i,y_i} = 1$ if $x_i \neq y_i$, 0 otherwise. The distance $\hat{H}(\mathbf{x},\mathbf{y})$ therefore corresponds to the number of corresponding elements of the two vectors which are different. Therefore, if x and y are bit strings, it actually corresponds to the Hamming distance between them, but it can be applied also to integer representations of binary variable sets like the ones used as encodings by ZIFF.

Using 1-NN and $\hat{H}$ as a distance to classify the data representation obtained by applying ZIFF to DS_1 and, after binarizing the original data, to DS_2, we can observe a dramatic improvement of the classification accuracy curves with respect to using the Euclidean distance. In Figure 7.13, the two dashed lines represent the results of ZIFF using the two distances. Notice that, while for DS_1 the plots corresponding to the reference methods are the same as in Figure 7.10, for DS_2 the results of the reference methods are different from those in Figure 7.11, since all reference feature extractors/selectors which, in that figure, had been applied based on a four-level discrete representation of the original data, have been applied here to the binary representation of the same data.

In principle, the $\hat{H}$ distance and, consequently, any K-NN classifier based on it, could be applied to any discrete representation. However, the number of training data necessary for a 1-NN classifier to achieve the same reliability (i.e., necessary for the probability of finding a match between two corresponding components to be the same) increases exponentially with the size of the new feature. Considering, for instance, the representation of a feature derived from M original pixels, the number of possible values that the feature can assume if the original data are binary is 2^M, while it becomes 2^{2M} if only the original data have a discrete four-value representation. This consideration is the most likely reason for the poor results that can be obtained if we apply 1-NN to the four-level representation of DS_2 and DS_3. It is also almost certainly the reason for the performance decline, with respect to PCA and to some of the other reference methods, which can be observed - even if, on DS_1, it is moderate - as the number of iterations increases and, thus, the number of new features decreases and their size, in terms of number of original features included in each of them, increases. Therefore, it is unsurprising that DS_1 is the dataset on which this effect is less evident, as it contains many more instances than the other datasets, while also being the dataset within which the variance of data within the same class is the lowest (data are printed characters, even if noisy and acquired at a very low resolution).

Therefore we can say that the $\hat{H}$ distance fits ZIFF's encoding of features much better than the Euclidean distance for use with distance-based classifiers. The results obtained even suggest that it is optimal for such a representation, when the original

features assume discrete values, provided the dataset is large enough. As demonstrated by our tests, this is often the case with binary data representations, but may soon require too many training instances when the number of possible values assumed by the original discrete-valued features increases.

7.3.5 Discussion and final remarks

The work described in this section has been aimed at verifying whether the procedure that uses the zI metric for detecting relevant functional blocks in complex systems could be applied to classification tasks as well. ZIFF represents possibly the most straightforward adaptation of such a procedure to feature extraction.

In particular, we wanted to assess ZIFF's performance as an unsupervised extractor of relevant features by which the dimensionality of a dataset representation could be reduced significantly but with minimum loss of information and, even more importantly, of discrimination power between the different target classes of a supervised classification problem.

Using random forest and decision trees as classifiers, ZIFF obtained results that are competitive with, when not better than, the other feature extractor/selectors taken into consideration. We also showed that, provided that the training dataset size is large enough, ZIFF can also achieve good results using K-NN classifiers based on an appropriate distance. On the contrary, its performance is generally poor when a SVM is used as a classifier.

The results of the tests we have performed on three datasets of real-world data (printed characters, handwritten characters, and DNA sequences) suggest that, even if at the conjecture level, but consistently with the outcome of the tests, we can actually attribute the good results that ZIFF has obtained using tree-based classifiers to the relevance of the variable sets ZIFF could identify, the disappointing results obtained using SVMs being most probably related with the inability of our encoding scheme to preserve relationships (such as scale, distance, etc.) which are essential for such classifiers.

A further observation that is worth making is that, in its present implementation, which is exactly the same which was developed for complex system analysis for

which such a property was not reasonable, ZIFF's first stage does not allow one to create groups that share the same variable since, in every iteration, the variable sets that are merged are substituted by the new group variable thus obtained. Even if ZIFF has obtained good results when applied to binary pattern classification, this can be a rather severe limitation which can be trivially highlighted taking as an example a simple classification problem in which 'x' symbols are to be classified against '+' symbols. Considering also the example given in figure 7.9, one would expect the relevant variable sets to be represented by horizontal, vertical and oblique segments, possibly having a pixel in common, as happens, for example, with the vertical and horizontal segment of the plus sign. Trivially considering the features to be equivalent to the segments into which the symbols can be decomposed and the plus symbol as an example, if a feature set represents the vertical bar then the horizontal bar needs to be split into two segments, one to the right and one to the left of the vertical bar, which, by themselves, are obviously much less relevant than the whole horizontal bar.

In spite of these limitations, the results we have obtained in our tests confirm that ZIFF, in its version as described in this section, can represent a viable approach to feature extraction when a good trade-off between classification accuracy and representation size is the goal.

Open issues

Such limitations can be the stimulus and target for future research, summarized in the following.

Firstly, since a mapping between a discrete space into a topologically equivalent continuous one may imply the creation of a higher-dimensional space, which is opposite to the goals with which ZIFF has been devised, the design of new data-dependent encodings seems to be inevitable, to take advantage of local properties of the datasets under consideration, to achieve the goal of limiting the dimension of the representation. In particular, we are going to implement and test methods that are specifically aimed at minimizing the distance between adjacent representations, such as, for instance, Kohonen's self-organizing maps (SOM) [127].

Similarly, using global optimization approaches like, for instance, genetic pro-

gramming [180], it would be possible to find such a (generally non-linear) data-induced mapping satisfying as well as possible the requirements for the preservation of topological relationships when moving from the pattern space to the new feature-representation space. Such a property is essential if one wants to apply distance-based classifiers, such as K-NN [57] and Learning Vector Quantization [127] or transformation-based classifiers like SVM [211] to the new reduced-size representation.

Finally, the feature extraction stage of ZIFF can be modified to enable the re-use of the same original attributes in different feature sets, as happens for example with the PCA, in which all new features are represented by a transformation of the whole set of attributes in the original data representation. We expect this improvement to lead to better final results even if the extension of the number of variable sets to be taken into consideration by HyReSS could lead to a further increase of the already high demand of computation resources of ZIFF itself.

The following section presents a different application of the *zI* metric for another kind of machine learning task (training set preprocessing).

7.4 A RI method for improved detection of malicious users in social networks

The phenomenon of "trolling" in social networks is becoming a very serious threat to the online presence of people and companies, since it may affect ordinary people, public profiles of brands, as well as popular characters.

In this section, we present a novel method to preprocess the temporal data describing the activities of possible troll profiles on Twitter, with the aim of improving automatic troll detection. The proposed method is based on the zI, which is used to group different user profiles, detecting different behavioral patterns for standard users and trolls. The proposed methodology combines HyReSS metaheuristic, which has been described in section 4.1, and the iterative sieving algorithm, described in chapter 5, to identify the most relevant groups of trolls and legitimate users.

The comparison of the results, obtained on data preprocessed using this novel

method and on the original dataset, demonstrate that the proposed technique generally improves the classification performance of troll detection, virtually independently of the classifier that is used.

7.4.1 Context

The definition of an "Internet troll" is still debated, since the acknowledgment of the associated behaviors is largely subjective. This lack of a single and precise definition has contributed to making the phenomenon more vague, attracting little interest by the researchers' community. Nevertheless, it has very serious implications for the online presence of both ordinary people and public profiles of brands, as well as popular characters. As a matter of fact, quite recently, this problem has received much more attention by the general public, largely as a response to events that have caused a stir about the "toxicity" of some social media sites [195]. Moreover, popular magazines and specialized press agencies have begun to address the issue and to write articles with large resonance as well.

As a general understanding, an "Internet troll" can be acknowledged as an individual adopting an antisocial behavior, who provokes and irritates other users of an online social platform with the intended effect to derail the normal evolution of an online discussion and possibly to stop it abruptly. To this aim, he/she can adopt an aggressive or offensive language, with prolonged arguments and unsolicited personal attacks. Thus, the term "trolling" refers to "a specific type of malicious online behavior, intended to disrupt interactions, aggravate interactional partners and lure them into fruitless argumentation" [45].

To address this problem, online platforms like Twitter periodically release new services and features, such as the practical functionalities that allow one to report anti-social behaviors or to "mute" annoying users. However, all attempts used till now have not been able to eradicate the problem, while the need of dealing with it is becoming more and more urgent. Hence, it is fundamental to create automated methods to manage the complexity of the issue, possibly leveraging on artificial intelligence, data mining and social network analysis.

Although the concept of Internet trolling may seem tied to the meaning of some

words like rudeness, arrogance, impertinence and vulgarity, the latter do not provide an accurate description of the phenomenon, since typically trolling also implies keeping the real intent of causing problems hidden. It is clear that trolling is a more complex problem than just provocative attacks. Accordingly, the task of automatic troll detection needs to take many different aspects into account [81].

Several studies have been conducted to demonstrate the applicability of various analytical tools and methods for automatic troll detection. Among the different viewpoints from which the phenomenon can be studied, some research works show that temporal patterns of the online behavior can provide a valuable contribution to automatic troll detection. For example, in [81], a set of objective features extracted from social network logs has been identified as characterizing malicious trolling behaviors. Among these, features related with time and frequency of actions appeared to be the most relevant for troll detection.

Various studies have related the frequency of publication of posts and comments to the quality of online discussions. The behavior of users who would be later banned from some large websites has been analyzed in [40], where the features characterizing such behaviors have been highlighted. In addition to the kind of text produced by users before being banned, patterns of activities are also observed. In particular, among the most useful features for detecting users likely to be banned in the future, an important role is played by the frequency of some activities, like the number of posts and comments per day. In [54], the authors use newsroom interviews, reader surveys and moderators' choices for characterizing the comments published on a newspaper website. In particular, they note that the frequency of commenting is a valuable indicator of low-quality discourse.

In [154], two classifiers are described, which detect two types of trolls: "paid trolls", who try to manipulate a user's opinion, and "mentioned trolls", who offend users and provoke anger. Many features regarding sentiment and text analysis are considered, based either on lexicon models or on bag-of-words models. Moreover, some metadata are considered, including the publication time. In particular, the analysis distinguishes a worktime period and a nighttime period; it also distinguishes workdays (Monday-Friday) and weekend days (Saturday and Sunday). Quite inter-

estingly, the analysis demonstrates that these features provide the largest impact on accuracy.

Troll Pacifier, the holistic system we proposed in [81], analyses many different groups of features, including temporal features which represent a breakdown of the daily activity into four-hour long time windows. This time window interval has been chosen after an extensive comparison, in which we have trained automatic classifiers based on different algorithms (K-Nearest Neighbors (KNN), Naïve Bayes (NB), Sequential Minimal Optimization (SMO), J48, etc.) using different time window lengths. Features measuring the activities in four-hour long intervals provided consistently the best classification results. In TrollPacifier, the time intervals represent single days (from Sunday to Saturday). In addition to these metrics, additional features consider the frequency of actions in the recent timeline and during the whole time of a user's presence on the platform.

This section aims at deepening the analysis of the impact of temporal features on troll detection. The final goal of our study is to improve the classification accuracy achieved by the time-related features by themselves, by clustering training data and substituting them with the centroid of their cluster. Clustering is performed using a method, based on the zI metric, for preprocessing the temporal data that have been used as the training set from which Troll Pacifier has been derived in [81]. Subsection 7.4.2 briefly describes the architecture of Troll Pacifier and the experimental results we obtained, focusing on the different kinds of features used, including the temporal features.

In section 7.1 we have demonstrated that the Relevance Index metrics, which were originally developed for the analysis of the structure of complex systems, could also be used effectively as a data clustering tool. While that section was mainly a proof of concept, in this section we evaluate the RI approach on a practical case which we show can benefit from its application. In particular 7.4.3, subsection presents a practical case in which we apply the zI-based clustering as a preprocessing step for troll detection

7.4.2 Troll Pacifier

This section describes Troll Pacifier, a holistic system for troll detection, which analyses many different features of trolls and legitimate users on the popular Twitter platform [81]. In this system, the best known and promising approaches and research lines are applied, along with original new ideas, in a form that fits such a large public platform. In particular, we have identified six groups of features, based respectively on the analysis of writing style, sentiment, behaviors, social interactions, linked media, and publication time. After a systematic collection and grouping of features, the different groups have been compared using a machine learning approach.

Data acquisition

The creation of a dataset of troll users is a crucial point of the analysis.
In order to collect our training set we used two cascaded approaches. The first one is based on distant supervision [93, 31] and allows one to obtain a raw dataset. The second one consists in manually filtering the previous dataset in order to obtain a more accurate training set.

In the first approach, we adapted an idea described by Mihaylov et al. [153] that defines a troll as a user that is called in this way at least N times by N different users. Twitter provides a series of official accounts, to which members can report their problems (e.g., @Twitter, @Support, @Safety, @TwitterUK, @TwitterAU, @TwitterSA, etc.). In particular, whenever a common user feels annoyed or even threatened by another, he/she can report the incident to one of these accounts, via a tweet containing a mention to the harasser, in the hope that Twitter administrators take the necessary measures. However, moderators are not always able to take appropriate countermeasures and often many users continue their online activities without being removed.

Therefore, we used the Twitter "Advanced Search" function to select the users which have been reported by other users that accuse them to be trolls (in messages containing words such as "troll", "ban", "harass", "block", "stalk", or some common derivations). In this way we built a raw dataset of trolls composed by users mentioned in these messages, but not yet banned by administrators.

For the non-troll class, we used the same approach and we selected users starting from general tweets containing common words such as “a”, “an”, “the”, “and”.

We finally obtained a dataset composed by 3000 troll users and 3000 non trolls. By manual inspection of a hundred of instances of this training set, we found many errors. In fact, our estimation is that more than a quarter of the users mentioned to the support channels do not behave actually in an anti-social manner, according to the tracts discussed in section 7.4.1.

Therefore, we decided to manually filter this raw dataset in order to obtain a more accurate training set, composed of 500 troll and 500 non-troll users. This final dataset has been validated by multiple independent human judges, through the manual inspection of users reported to the official support channels. In particular, users have been selected after inspecting both their recent timelines and their role and attitude in prolonged discussions, where they were repeatedly mentioned as trolls.

Groups of features

In the literature, we have identified six main types of troll detection approaches. For each type of approach, we have defined a group of features, using ideas proposed in previous researches as well as new ones. In particular, for building TrollPacifier we have identified 6 groups of features, which are listed in the following paragraphs.

- **Sentiment analysis (*SENT*).** This group includes 26 features, to distinguish positive, negative and objective posts, but also to associate them with more precise emotions. About “sentic computing” [33], TrollPacifier includes the main results of the SenticNet library [34]: sensitivity; polarity; trollness; attention; pleasantness; attitude. Moreover, it takes into account the results of lexicon and rule-based sentiment analysis, using the VaderSentiment library [87]. In particular, from this analysis TrollPacifier gathers values representing the maximum, minimum and average levels of positive, negative and neutral sentiments; polarity; trollness. Additionally, TrollPacifier includes a whole hierarchical emotion detection system, as described in section 6.3.2. In particular, the output of each level of classification is used to obtain a feature for the user-level anal-

ysis. Thus, collected features are: number of objective and subjective tweets; number of positive and negative tweets; number of tweets expressing one of the six basic emotions of Parrott's model [170]: fear, anger, sadness, love, joy, and surprise. Finally, TrollPacifier includes an ad-hoc text-based classifier for evaluating the overall abusiveness of a text, i.e., provoking others to finally report the author as a troll. The classifier is trained with two classes of texts: those written by alleged trolls and those written by normal users. More details of this classifier are provided in the next section.

- **Time and frequency of actions (*TIME*).** This group includes 57 features, to identify the most active day hours and the time dedicated to each post. Considering the results presented in [40, 54, 154], after an optimization process, we have included in TrollPacifier features for representing the activity in daily intervals of 4 hours. We have chosen this time length after a thorough comparison, in which we have trained automatic classifiers based on different algorithms (K-nearest neighbors, Naive Bayes, Sequential Minimum Optimization, C4.5) [152] using different interval lengths. Features measuring the activities in intervals of four hours provided consistently the best classification results. In TrollPacifier, the time intervals are distinguished by single day (from Sunday to Saturday), and also grouped together for generic workdays and weekends. In addition to these metrics, additional features consider the frequency of actions in the recent timeline and during the whole user's presence on the platform.

- **Text content and style (*TEXT*).** This group includes 31 features, to measure the grammatical correctness and the kind of language used in posts. TrollPacifier includes some features for taking into account the readability grades, based on various metrics [40, 52, 55, 153]. TrollPacifier also includes the following other features in this class: average word length and sentence length, by number of characters, syllables, or words; number of long words and complex words; number of verbs in general and some auxiliary verbs in particular; number of conjunctions, pronouns, prepositions, articles, subordinations, either in the middle or at the beginning of a phrase; number of hapaxes and rare words.

- **User behaviors (*BEHA*).** This group includes 38 features, to distinguish users participating more actively, i.e., contributing with original messages and media objects. Taking into consideration the experiences of relevant research works [40, 48, 138, 192], we have introduced into TrollPacifier a number of features to characterize a user's online behavior, including: total number of tweets, retweets, replies, favorites, citations and quotes in the timeline; proportion between active actions (original tweets and replies) and passive actions (retweets and quotes); count of various actions associated with a single item (e.g., number of replies to a single tweet); maximum repetitions of a single action.

- **Interactions with the community (*COMM*).** This group includes 34 features, to highlight a user's integration within his group of followers and followees. To represent a user's relationships within his own community, TrollPacifier includes the following features [130, 131, 167, 192]: number of followers and followees; ratio of these numbers; ratio of tweets per follower; number of posts retweeted or favorited by other users; counts of given and received mentions; number of different users mentioned; counts of different actions, including retweets, replies, mentions, related to a single user or to a single tweet; h-index based on retweets, likes, and their sum.

- **Advertisement of external content (*ADVE*).** This group includes 38 features, to count the number of references to diverse external content and other channels of discussion. To evaluate the possible usefulness of external links and other forms of advertisement for troll detection [9, 46, 113], we have added the following features in TrollPacifier: number and frequency of URLs in posts and comments, as well as in the profile information provided to the platform; number and frequency of published or advertised videos, images and other media; number and frequency of hashtags.

Architecture

Troll Pacifier has been implemented using ActoDES, which is a software framework which adopts the actor model for simplifying the development of complex distributed

systems [18]. The services developed for this project provide additional functionalities to actors, for the continuous analysis of various social streams. In particular, a Twitter service allows other actors to send various kinds of requests:

- User timeline, to obtain the recent tweets of a specified user and save them in a local storage system.
- Content query, to similarly obtain recent tweets published on Twitter and selected according to some constraints specified by the actor.
- Stream, to continuously receive messages published on the platform during the execution; tweets are obtained in the form of JSON objects, which are then stored in a NoSQL repository (namely a MongoDB database [41]).

Apart from the system-level ActoDES actors, TrollPacifier includes additional actors, as shown in Figure 7.14. They are dedicated to (*i*) basic tasks, like acquiring streaming data and users' profile information from Twitter; (*ii*) direct feature extraction tasks, with different actors for the six different groups of features; (*iii*) specialized classification tasks, aimed at calculating additional features through intermediate steps; and (*iv*) final automatic classification, based on different machine learning algorithms. Features are extracted by these actors in both the initialization stage, for creating the training set, and the online operation stage, for evaluating streaming content. Three final classification algorithms have been included: Naive Bayes (NB), Sequential Minimal Optimization (SMO) and Random Forest (RF). The system can also be easily configured to encapsulate any other classification algorithm. As an additional feature, it is also possible to create an online learning loop, thus periodically feeding the training set with newly automatically classified instances, above a certain threshold of confidence [82].

In particular, for the SENT group, some features are obtained through some automatic classifiers that are implemented by few specialized actors integrated into TrollPacifier. One subsystem is dedicated to emotion detection and is built as a hierarchy of classifiers. Another subsystem is dedicated to evaluating the "abusiveness" of a text, through an ad-hoc trained classifier.

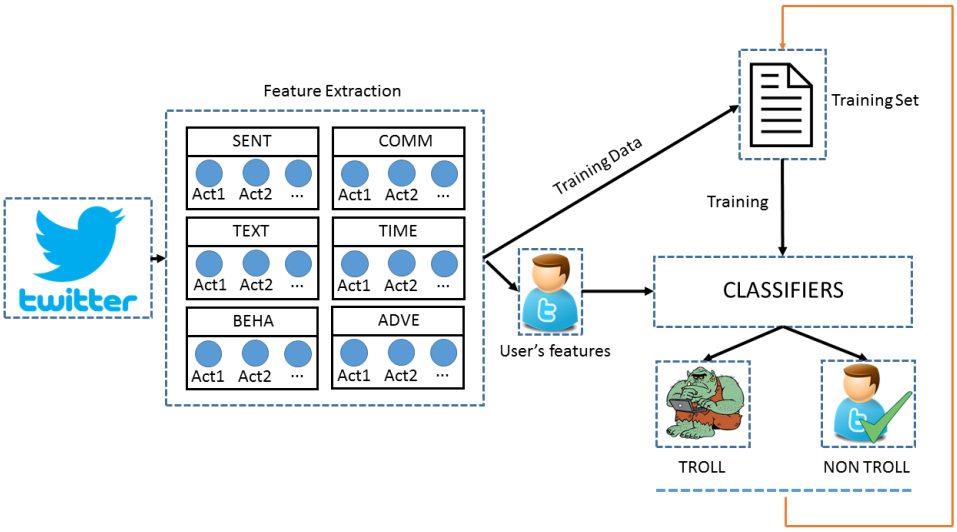


Figure 7.14: Representation of the actor-based system architecture.

Troll Pacifier results

The experimental results highlight the effectiveness of the six considered groups of features (COMM, TEXT, BEHA, SENT, TIME, ADVE) for troll detection. We also analyzed the results obtained by considering a dataset containing all the features of the six considered groups (TOT dataset). Figure 7.15 shows the accuracies obtained with 10 runs of 10-fold cross validation on the different datasets corresponding to the different group of features, using different classification algorithms. We performed 10 runs of 10-fold cross validation in order to obtain more reliable results.

It is to be noticed that the combination of all the groups of features allowed us to obtain a very high accuracy (95.5% using Random Forest Classifier).

At a finer level of analysis, it is possible to assess which features have the largest influence on the results. In particular, we use the Information Gain algorithm to find the most relevant features. IG evaluates the worth of an attribute by calculating the reduction in entropy for each feature. The result is that features that perfectly discriminate the class give maximal information and unrelated features give low information.

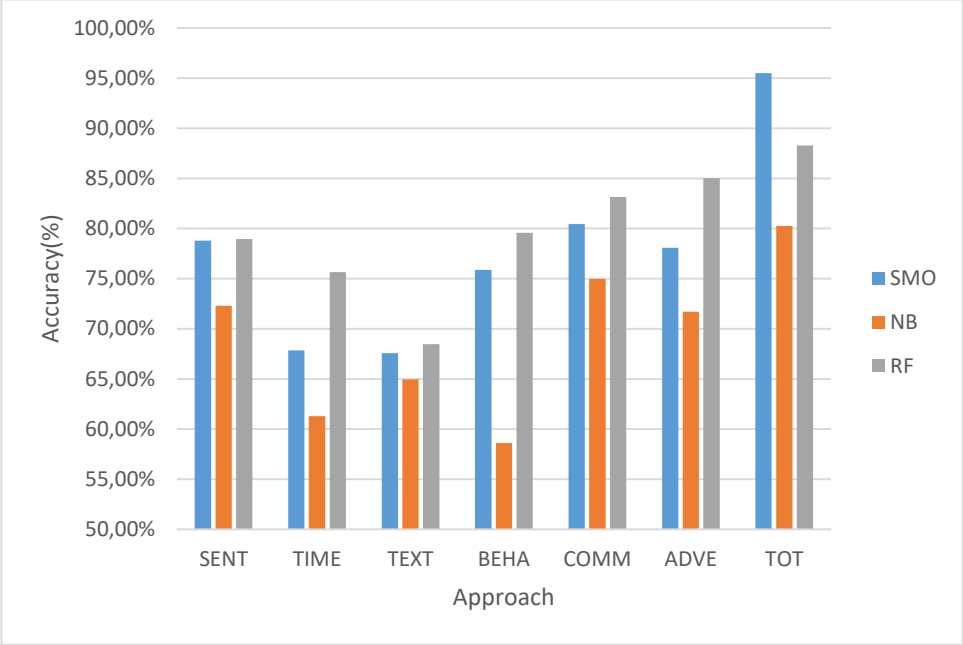


Figure 7.15: Accuracy obtained with different groups of features and different algorithms.

Table 7.3 describes the first 10 features in decreasing order of Information Gain, together with the corresponding group.

Features from the COMM group are the most discriminative ones (4 of the best 10 features belong to this group). However, it is to be noticed that features from different groups provide important contributions for automatic classification and also provide useful insights about diverse aspects of online trolling. In particular, it is quite interesting that the simple analysis based only on the daily and hourly frequency of messages provide quite good results. In fact, a troll produce many more tweets than a normal account, in particular deep in the night. This can be related to availability of time and to prolonged arguments, but it can also be related to personality traits of online trolls which would deserve further studies [121].

The group of temporal features express a dynamical behavior. Thus, the follow-

Table 7.3: The first 10 features in decreasing order of Information Gain.

Description	Group	IG
#Users mentioned in the quoted tweets + #Users mentioned in the tweets	COMM	0.327
The results of an ad-hoc text-based classifier for evaluating the "abusiveness" of a text (described in 7.4.2)	SENT	0.275
#Answers to the user's tweets + #Retweets + #Shares	COMM	0.250
#Public lists the user belongs to	ADVE	0.235
#Followers of the user	COMM	0.228
The frequency of user's messages on Mondays from 00:00 to 04:00 (7.4.2)	TIME	0.139
#Urls in posts and comments	ADVE	0.191
#Replies to the user	COMM	0.178
The frequency of user's messages on Thursdays from 08:00 to 12:00 (7.4.2)	TIME	0.167
The frequency of user's messages on Fridays from 08:00 to 12:00 (7.4.2)	TIME	0.164

ing subsection describes the analysis of this dynamical behavior using the zI metric, aiming at improving the effectiveness of this group of features for troll detection.

7.4.3 zI-based preprocessing for troll detection

Starting from Troll Pacifier temporal features, the problem has been approached in a supervised way, applying the iterative sieving algorithm and computing the zI index from data regarding trolls and legitimate users separately, to derive two specific models of the two classes. Thus, we analyzed two dynamical systems, including 300 trolls and 300 standard users, respectively. For each system, the zI analysis is intrinsically unsupervised and aims at detecting different behavioral patterns of the same class of users, finding clusters of trolls and clusters of non-trolls characterized by a similar

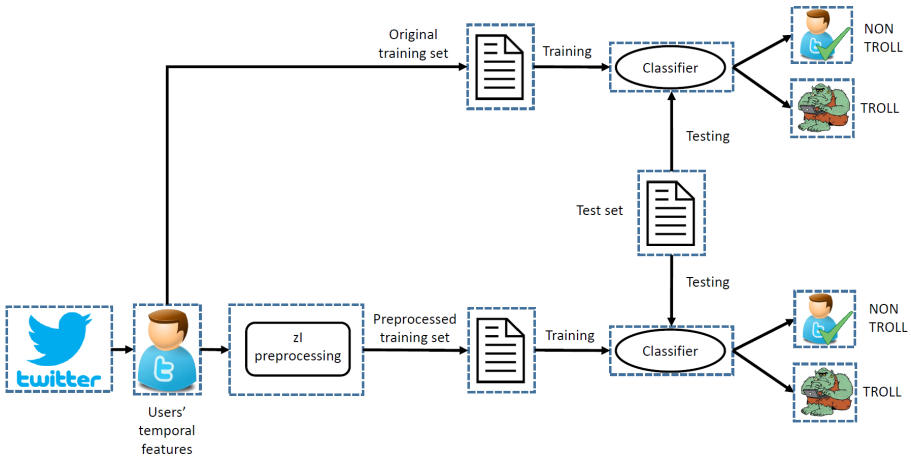


Figure 7.16: Troll detection on Twitter using the original dataset used in Troll Pacifier (above) and the zI-preprocessed dataset as training sets for automatic classification (below).

dynamical behavior.

Each observation (data matrix row) represents the frequency of the tweets posted by each user (data matrix column) within a specific time interval during the day (e.g., the frequency of user's tweets posted on Monday from 00:00 to 04:00). The four-hour long ranges have been chosen in order to create non-overlapping intervals distinguishing worktime and nighttime periods, but also working days and weekend days.

Given the large number of variables which describe the two systems, we used HyReSS combined with the iterative sieving algorithm to identify the most relevant groups of trolls and legitimate users. Each group of trolls (or legitimate users) corresponds to a behavioral prototype. Based on this assumption, we created a new preprocessed dataset composed of the centroids of the groups of trolls and legitimate users discovered by our method. As will be shown in section 7.4.4, this preprocessing phase allowed us to improve the classification accuracy in the automatic troll detection task.

The main building blocks of our method are shown in Figure 7.16, which also

illustrates the data flow of the procedure by which we compared the classifications obtained from the raw data and those obtained from the zI-preprocessed ones.

7.4.4 Experimental results

In order to apply the *zI* method to the analysis of the dynamical behavior of trolls and legitimate users on Twitter, we used the temporal features of the labeled dataset of Troll Pacifier. In this section we describe the dataset we preprocessed using the zI approach and we show the accuracy improvement that could be obtained using the new dataset thus obtained to train different classifiers.

Dataset description

Starting from the dataset described in subsection 7.4.2, we focused on the group of features related to time and frequency of actions, after we observed that this set of features can give a significant contribution to troll detection.

In particular, we performed a dynamical analysis of the frequency of users' posts, to improve classification accuracy. We analyzed the frequency of the tweets posted by a user within a specific time interval during each day (i.e., Sunday from 00:00 to 04:00, Sunday from 04:00 to 08:00, Sunday from 08:00 to 12:00, ..., Saturday from 20:00 to 00:00) to detect some behavioral prototypes of trolls and legitimate users. In our experiments, we considered a four-level discretization of the original frequency values (low, medium-low, medium-high and high frequency of users' posts in each time interval).

The dataset is the same used for Troll Pacifier (500 trolls and 500 non-trolls). The instances have been split into a training set composed of 300 trolls and 300 non-trolls and a test set composed of the remaining 200 trolls and 200 non-trolls.

Troll detection

In this subsection we compare the results of troll detection obtained through classifiers trained using the original dataset with those obtained after *zI*-based preprocessing of the same data.

The zI analysis identified 161 groups of trolls and 143 groups of non-trolls. Given the stochastic nature of the HyReSS, it should be noticed that the outcome of HyReSS was stable over five repetitions, repeatedly finding the same groups. The preprocessed dataset is composed of the centroids of each group of trolls and non-trolls detected using the zI-based approach. This preprocessing step aims at polishing the original dataset by substituting groups of users behaving similarly with the prototype which is closest to their dynamical behavior. After preprocessing, the final dataset contains 304 instances (the 161 centroids corresponding to the groups representing trolls, along with the 143 centroids of groups representing ordinary users).

The effect of this preprocessing on the detection of troll and non-troll users has been tested using four different classifiers: Random Forest (RF), Naïve Bayes (NB), Sequential Minimal Optimization (SMO) and K-Nearest Neighbors (KNN). Table 7.4 shows the accuracies obtained by training the classifiers on the original and on the preprocessed dataset.

To take into consideration the intrinsic stochastic nature of Random Forest and Sequential Minimal Optimization, we repeated all experiments using such classifiers five times; in the table we report the average accuracy over the five runs.

Table 7.4: Troll detection accuracies obtained by training different classifiers on the original dataset and on the preprocessed one.

Dataset	RF	NB	SMO	KNN (k=1)	KNN (k=5)	KNN (k=10)
Original	76.95%	60.00%	65.00%	65.75%	70.75%	70.75%
Preprocessed	77.10%	63.25%	69.25%	69.25%	71.25%	72.75%

As shown in Table 7.4, the introduction of the zI-based preprocessing allowed us to improve the accuracy of all the classifiers.

Since the preprocessed dataset is not perfectly balanced (161 trolls and 143 non-trolls corresponding to the centroids of the troll and non-troll groups), in table 7.5 we also show the F-measure of the classifiers for a fairer comparison.

Table 7.5: F-measure obtained by training different classifiers on the original dataset and on the preprocessed one.

Dataset	RF	NB	SMO	KNN (k=1)	KNN (k=5)	KNN (k=10)
Original	0.769	0.546	0.622	0.656	0.704	0.704
Preprocessed	0.771	0.594	0.678	0.689	0.709	0.725

The classifiers trained after the zI-based preprocessing tend to perform better than the standard ones. The accuracy and F-measure improvements highlight the importance of the dynamical analysis of users' behavior for automatic troll detection.

7.4.5 Final remarks

In this section we applied a dynamical analysis based on the zI metric to preprocess the temporal data describing the users' behavior, aimed at improving the classification accuracy of troll detection. The dynamical analysis we propose is based on the observation of the frequency of users' tweets within different time windows.

We compared the results of troll detection based on the raw dataset previously used with the same goal with those obtained after zI-based preprocessing of the same data. The tests have been performed using four different classifiers.

The classifiers trained using the zI-based preprocessing performed better than the standard ones, leading to an improvement in the classification accuracy and F-measure. The results show that the zI analysis is able to detect significant behavioral patterns for troll detection, finding clusters of trolls and clusters of non-trolls characterized by a similar dynamical behavior.

The experimental results confirmed the applicability of the RI methodology to a context which is different from complex systems analysis, improving the performance of a machine learning task.

An extended description of the work described in this chapter can be found in [203, 205, 206, 207].

Chapter 8

Conclusions

The work described in this thesis is related to the analysis of complex systems and has mainly regarded the identification of the internal organization of systems whose components are usually well-known and can be defined in terms of variables. The methodology we have proposed allows one to overcome a merely topological approach, which cannot be applied when the interactions among the system components are not known a priori. In particular, the use of the RI metrics allows one to detect the main interacting structures of a dynamical system, based on the observation of the status assumed by its variables over time.

From a methodological point of view, in Chapter 3 we have proposed some improvements to the RI metrics, in terms of both computational efficiency and effectiveness for complex systems analysis. In Chapter 4 we have presented two niching metaheuristics for complex systems analysis, which can compute the RI metrics providing results comparable to an exhaustive search in a much shorter time. These metaheuristics have allowed us to extend our research on the detection of RSs to dynamical systems of much larger sizes than previously possible. Finally, in Chapter 5 we have proposed an iterative algorithm aimed at revealing hierarchical dynamical structures in complex systems.

The proposed methodology can be applied to a broad range of complex systems, both natural and artificial, since it depends only on the dynamical observation of the

variables' states. In this thesis, the proposed algorithms have been applied to some artificially designed systems, in order to test the effectiveness of the technique, as well as to interesting real-world data, representing chemical, biological, and social systems, obtaining interesting results.

Focusing on the applications, in Chapter 6 we have presented three case studies related to different contexts, within which the proposed methodology has been successfully applied. The first case study is related to biology and has led to novel and effective interpretations of the interactions occurring in a regulatory network (T-helper network). The second one deals with the detection of the dynamical organization in cancer evolution models. The third case study is related to social network analysis and has allowed us to study the dynamical behavior of users in a Facebook group and to identify interesting communities. The results obtained have been confirmed by a different kind of analysis, which combines machine learning and network theory.

Since the proposed methodology has provided useful information also in contexts which are different from traditional complex systems analysis, we have further investigated the applicability of the RI methodology to machine learning and pattern recognition tasks (clustering, classification, feature extraction and preprocessing). In particular, the properties that the RI metrics highlight in time series, when analyzing the dynamics of complex systems, can be somehow assimilated to the properties of the multivariate distribution of the examples of a machine learning training set.
In chapter 7 we have presented some RI-based algorithms for machine learning and pattern recognition tasks. With no pretense to reach state-of-the-art performances, the test we have performed have shown that the proposed methodology is reasonable and already competitive with existing machine learning algorithms.

In a classification context, the RI analysis can be applied according to two different approaches:

- "supervised" approach: the RI method is applied iteratively to homogeneous subsets of samples belonging to the same class, to extract features capable of "summarizing" the information content of each class;

- "unsupervised" approach: the RI method is applied to the whole dataset, inde-

pendently of the classes to which data belong.

Despite yielding good results, the supervised approach is impractical, since it requires that the already resource-demanding phase of RI analysis be repeated, at the same level of complexity, as many times as the number of classes. Nevertheless, since the analysis based on the RI metrics can detect the peculiarities characterizing the variations within data belonging to the same class, it is also able to detect the peculiarities characterizing a class/cluster of data with respect to the rest of the dataset. In fact, results obtained by the unsupervised approach are almost as good.

The experimental results have shown that the approaches derived from the RI metrics, originally devised for complex systems analysis, can actually be applied also to machine learning and pattern recognition. However, there are still some limitations, which can be a stimulus for future research. In particular, from the implementation point of view, the efficiency of the computation of the RI metrics could be further optimized, to limit the computational overhead of the RI analysis. From the methodological point of view, the main open issues are:

- further improving the effectiveness of the proposed metaheuristics for RS detection, possibly devising some self-adapting mechanisms to automatically fit the parameter values to the system under investigation;
- allowing the sieving algorithm to generate RSs sharing the same variables (currently we are only considering mutually exclusive groups). We expect this improvement to lead to better final results, especially in feature extraction tasks, enabling the re-use of the same original attributes in different feature sets;
- finding an optimal encoding for the RSs in the feature extraction context. In particular, we are going to implement and test methods that are specifically aimed at preserving the relative distances between patterns in the original and in the transformed domain, such as, for instance, Kohonen's self-organizing maps (SOMs);
- further improving the discretization methods to deal with continuous variables. As shown in the previous chapters, a proper discretization allows one to apply

the RI methodology to the dynamical analysis of systems described by continuous variables, e.g., using a four-level discretization to describe the frequency of users' posts (low, medium-low, medium-high and high frequency) in a time interval (see sections 6.3 and 7.4), or using a three-level encoding to represent the dynamical behavior of the concentrations of different chemical species in a Catalytic Reaction System (see section 3.1), where "0", "1" and "2" stand respectively for "decreasing concentration", "no change" and "increasing concentration". The discretization method could be enriched by a more detailed analysis of the statistical distribution of the system observations, in order to find an optimal k-level encoding.

Candidate's publications related with this thesis

Extended description of the work described in this thesis can be found in:

- L. Sani, R. Pecori, M. Mordonini, and S. Cagnoni. From complex system analysis to pattern recognition: Experimental assessment of an unsupervised feature extraction method based on the relevance index metrics. *Computation*, 7(3):39, 2019.

- G. Lombardo, P. Fornacciari, M. Mordonini, L. Sani, and M. Tomaiuolo. A combined approach for the analysis of support groups on Facebook-the case of patients of hidradenitis suppurativa. *Multimedia Tools and Applications*, 78(3):3321–3339, 2019.

- M. Villani, L. Sani, R. Pecori, M. Amoretti, A. Roli, M. Mordonini, R. Serra, and S. Cagnoni. An iterative information-theoretic approach to the detection of structures in complex systems. *Complexity*, 2018:1–15, 2018.

- P. Fornacciari, M. Mordonini, A. Poggi, L. Sani, and M. Tomaiuolo. A holistic system for troll detection on Twitter. *Computers in Human Behavior*, 89:258–268, 2018.

- L. Sani, R. Pecori, P. Fornacciari, M. Mordonini, M. Tomaiuolo, and S. Cagnoni. A relevance index-based method for improved detection of malicious users in

social networks. In *International Workshop on Artificial Life and Evolutionary Computation*. Springer, 2019.

- L. Sani, G. D'Addese, A. Graudenzi, and M. Villani. The detection of dynamical organization in cancer evolution models. In *International Workshop on Artificial Life and Evolutionary Computation*. Springer, 2019.

- L. Sani, G. D'Addese, R. Pecori, M. Mordonini, M. Villani, and S. Cagnoni. An integration-based approach to pattern clustering and classification. In C. Ghidini, B. Magnini, A. Passerini, and P. Traverso, editors, *AI*IA 2018 – Advances in Artificial Intelligence*, pages 362–374, Cham, 2018. Springer International Publishing.

- L. Sani, G. Lombardo, R. Pecori, P. Fornacciari, M. Mordonini, and S. Cagnoni. Social relevance index for studying communities in a Facebook group of patients. In K. Sim and P. Kaufmann, editors, *Applications of Evolutionary Computation*, pages 125–140. Springer International Publishing, 2018.

- L. Sani, R. Pecori, E. Vicari, M. Amoretti, M. Mordonini, and S. Cagnoni. Can the relevance index be used to evolve relevant feature sets? In K. Sim and P. Kaufmann, editors, *Applications of Evolutionary Computation*, pages 472–479, Cham, 2018. Springer International Publishing.

- G. Lombardo, A. Ferrari, P. Fornacciari, M. Mordonini, L. Sani, and M. Tomaiuolo. Dynamics of emotions and relations in a facebook group of patients with hidradenitis suppurativa. In *International Conference on Smart Objects and Technologies for Social Good*, pages 269–278. Springer, 2017.

- L. Sani, M. Amoretti, E. Vicari, M. Mordonini, R. Pecori, A. Roli, M. Villani, S. Cagnoni, and R. Serra. Efficient search of relevant structures in complex systems. In *AI*IA 2016 Advances in Artificial Intelligence: XVth International Conference of the Italian Association for Artificial Intelligence*, Genova, Italy, November 29 – December 1, 2016, Proceedings, pages 35–48. Springer International Publishing, 2016.

- L. Sani, A. Bononi, R. Pecori, M. Amoretti, M. Mordonini, A. Roli, M. Villani, S. Cagnoni, and R. Serra. An improved relevance index method to search important structures in complex systems. In *Italian Workshop on Artificial Life and Evolutionary Computation*, pages 3–16. Springer, 2018.

- M. Villani, L. Sani, M. Amoretti, E. Vicari, R. Pecori, M. Mordonini, S. Cagnoni, and R. Serra. A relevance index method to infer global properties of biological networks. In M. Pelillo, I. Poli, A. Roli, R. Serra, D. Slanzi, and M. Villani, editors, *Artificial Life and Evolutionary Computation*, pages 129–141, Cham, 2018. Springer International Publishing.

- G. Silvestri, L. Sani, M. Amoretti, R. Pecori, E. Vicari, M. Mordonini, and S. Cagnoni. Searching relevant variable subsets in complex systems using K-means PSO. In M. Pelillo, I. Poli, A. Roli, R. Serra, D. Slanzi, and M. Villani, editors, *Artificial Life and Evolutionary Computation*, pages 308–321. Springer International Publishing, 2018.

- E. Vicari, M. Amoretti, L. Sani, M. Mordonini, R. Pecori, A. Roli, M. Villani, S. Cagnoni, and R. Serra. GPU-based parallel search of relevant variable sets in complex systems. In *Italian Workshop on Artificial Life and Evolutionary Computation*, pages 14–25. Springer, 2016.

Bibliography

[1] CUDA Toolkit. `http://developer.nvidia.com/cuda-toolkit`. [Online; accessed January 12, 2020].

[2] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[3] R. Albert, H. Jeong, and A.-L. Barabási. Internet: Diameter of the World-Wide Web. *nature*, 401(6749):130–131, 1999.

[4] E. Aldana-Bobadilla and A. Kuri-Morales. A Clustering Method Based on the Maximum Entropy Principle. *Entropy*, 17(1):151–180, 2015.

[5] M. Amoretti and C. Gershenson. Measuring the complexity of adaptive peer-to-peer systems. *Peer-to-Peer Networking and Applications*, 9(6):1031–1046, 2016.

[6] G. Angiani, S. Cagnoni, N. Chuzhikova, P. Fornacciari, M. Mordonini, and M. Tomaiuolo. Flat and Hierarchical Classifiers for Detecting Emotion in Tweets. In *AI* IA 2016 Advances in Artificial Intelligence*, pages 51–64. Springer, 2016.

[7] V. Anzoise and S. Sardo. Dynamic systems and the role of evaluation: The case of the Green Communities project. *Evaluation and Program Planning*, 54:162–172, 2016.

[8] A. Arkin, P. Shen, and J. Ross. A test case of correlation metric construction of a reaction pathway from measurements. *Science*, 277(5330):1275–1279, 1997.

[9] J. Aro. The cyberspace war: propaganda and trolling as warfare tools. *European View*, 15(1):121–132, Jun 2016.

[10] T. Bäck and H.-P. Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary computation*, 1(1):1–23, 1993.

[11] R. Badii and A. Politi. *Complexity – Hierarchical structures and scaling in physics*. Cambridge Nonlinear Science Series. Cambridge University Press, 1997.

[12] Y. Bao, C. Quan, L. Wang, and F. Ren. The role of pre-processing in twitter sentiment analysis. In *International Conference on Intelligent Computing*, pages 615–624. Springer, 2014.

[13] Y. Bar-Yam. *Dynamics of complex systems*. Studies in nonlinearity. Addison–Wesley, Reading, MA, 1997.

[14] N. Beerenwinkel, R. F. Schwarz, M. Gerstung, and F. Markowetz. Cancer evolution: mathematical models and computational inference. *Systematic biology*, 64(1):e1–e25, 2014.

[15] S. Behbahani and C. W. de Silva. Niching Genetic Scheme With Bond Graphs for Topology and Parameter Optimization of a Mechatronic System. *IEEE/ASME Transactions on Mechatronics*, 19(1):269–277, Feb 2014.

[16] G. Beni. The concept of cellular robotic system. In *Proceedings IEEE International Symposium on Intelligent Control 1988*, pages 57–62. IEEE, 1988.

[17] J. M. Bennett, D. Catovsky, M.-T. Daniel, G. Flandrin, D. A. Galton, H. R. Gralnick, and C. Sultan. Proposals for the classification of the acute leukaemias French-American-British (FAB) co-operative group. *British journal of haematology*, 33(4):451–458, 1976.

[18] F. Bergenti, A. Poggi, and M. Tomaiuolo. An Actor Based Software Framework for Scalable Applications. *Lecture Notes in Computer Science (LNCS)*, 8729:26–35, 2015. Proc. 7th International Conference on Internet and Distributed Computing Systems (IDCS 2014); Calabria; Italy; 2014-09-22/24 [MT].

[19] V. Bettoli, S. Pasquinucci, S. Caracciolo, D. Piccolo, S. Cazzaniga, F. Fantini, L. Binello, G. Pintori, and L. Naldi. The hidradenitis suppurativa patient journey in Italy: current status, unmet needs and opportunities. *Journal of the European Academy of Dermatology and Venereology*, 30(11):1965–1970, 2016.

[20] S. Bird and X. Li. Adaptively Choosing Niching Parameters in a PSO. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, pages 3–10, New York, NY, USA, 2006. ACM.

[21] C. M. Bishop. *Pattern Recognition And Machine Learning*. Springer, 2006.

[22] C. Blum and D. Merkle. *Swarm Intelligence: Introduction and Applications*. Springer Publishing Company, Incorporated, 1 edition, 2008.

[23] E. Bonabeau, D. d. R. D. F. Marco, M. Dorigo, G. Theraulaz, et al. *Swarm intelligence: from natural to artificial systems*. Number 1. Oxford university press, 1999.

[24] R. R. Bouckaert, E. Frank, M. Hall, R. Kirkby, P. Reutemann, A. Seewald, and D. Scuse. *WEKA manual for version 3-7-8*. University of Waikato, NZ, 2013.

[25] A. Brabazon, M. O'Neill, and S. McGarraghy. *Natural computing algorithms*. Springer, 2015.

[26] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, Oct 2001.

[27] R. Brits, A. P. Engelbrecht, and F. Van den Bergh. A niching particle swarm optimizer. In *Proceedings of the 4th Asia-Pacific conference on simulated*

evolution and learning, volume 2, pages 692–696. Singapore: Orchid Country Club, 2002.

[28] R. Brits, A. P. Engelbrecht, and F. van den Bergh. Solving systems of unconstrained equations using particle swarm optimization. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 3, pages 6 pp. vol.3–, Oct 2002.

[29] D. Bucur, G. Iacca, A. Marcelli, G. Squillero, and A. Tonda. Multi-objective Evolutionary Algorithms for Influence Maximization in Social Networks. In *European Conference on the Applications of Evolutionary Computation*, pages 221–233. Springer, 2017.

[30] R. A. Burrell, N. McGranahan, J. Bartek, and C. Swanton. The causes and consequences of genetic heterogeneity in cancer evolution. *Nature*, 501(7467):338–345, 2013.

[31] S. Cagnoni, P. Fornacciari, J. Kavaja, M. Mordonini, A. Poggi, A. Solimeo, and M. Tomaiuolo. Automatic Creation of a Large and Polished Training Set for Sentiment Analysis on Twitter. In *International Workshop on Machine Learning, Optimization, and Big Data*, pages 146–157. Springer, 2017.

[32] S. Cagnoni and G. Valli. OSLVQ: a training strategy for optimum-size Learning Vector Quantization classifiers. In *IEEE International Conference on Neural Networks, IEEE WCCI 1994*, volume 2, pages 762–765 vol.2, 1994.

[33] E. Cambria, P. Chandra, A. Sharma, and A. Hussain. Do not feel the trolls. In *CEUR Workshop Proceedings*, volume 664, pages 1–12, 2010.

[34] E. Cambria, S. Poria, D. Hazarika, and K. Kwok. SenticNet 5: discovering conceptual primitives for sentiment analysis by means of context embeddings. In *AAAI*, pages 1795–1802, 2018.

[35] G. Caravagna, A. Graudenzi, D. Ramazzotti, R. Sanz-Pamplona, L. De Sano, G. Mauri, V. Moreno, M. Antoniotti, and B. Mishra. Algorithmic methods to

infer the evolutionary trajectories in cancer progression. *Proceedings of the National Academy of Sciences*, 113(28):E4025–E4034, 2016.

[36] M. C. Cario and B. L. Nelson. Modeling and Generating Random Vectors with Arbitrary Marginal Distributions and Correlation Matrix. Technical report, 1997.

[37] D. Chang, Y. Zhao, and C. Zheng. A Real-Valued Quantum Genetic Niching Clustering Algorithm and its Application to Color Image Segmentation. In *International Conference on Intelligent Computation and Bio-Medical Instrumentation (ICBMI)*, pages 144–147, Dec 2011.

[38] A. Chaos, M. Aldana, C. Espinosa-Soto, B. Ponce de Leon, A. Garay Arroyo, and E. Alvarez-Buylla. From Genes to Flower Patterns and Evolution: Dynamic Models of Gene Regulatory Networks. *J Plant Growth Regul*, 25:278–289, 2006.

[39] X. Chen, Y. S. Ong, M. H. Lim, and K. C. Tan. A Multi-Facet Survey on Memetic Computation. *IEEE Transactions on Evolutionary Computation*, 15(5):591–607, Oct 2011.

[40] J. Cheng, C. Danescu-Niculescu-Mizil, and J. Leskovec. Antisocial behavior in online discussion communities. *arXiv preprint arXiv:1504.00680*, 2015.

[41] K. Chodorow. *MongoDB: the definitive guide*. " O'Reilly Media, Inc.", 2013.

[42] F. Cimorelli, F. D. Priscoli, A. Pietrabissa, L. R. Celsi, V. Suraci, and L. Zuccaro. A distributed load balancing algorithm for the control plane in software defined networking. In *2016 24th Mediterranean Conference on Control and Automation (MED)*, pages 1033–1040, June 2016.

[43] M. Clerc. *Particle swarm optimization*, volume 93. John Wiley & Sons, 2010.

[44] M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, Feb 2002.

[45] B. A. Coles and M. West. Trolling the trolls: Online forum users constructions of the nature and properties of trolling. *Computers in Human Behavior*, 60:233–244, 2016.

[46] J. Cordi. *Social Media Revolution: Political and Security Implications*. NATO Parliamentary Assembly, 2017.

[47] T. Cover and A. Thomas. *Elements of information theory, 2nd Edition*. Wiley-Interscience, New York, 2006.

[48] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi. DNA-inspired online behavioral modeling and its application to spambot detection. *IEEE Intelligent Systems*, 31(5):58–64, 2016.

[49] A. Davis, R. Gao, and N. Navin. Tumor evolution: Linear, branching, neutral or punctuated? *Biochimica et Biophysica Acta (BBA)-Reviews on Cancer*, 1867(2):151–161, 2017.

[50] H. De Jong. Modeling and simulation of genetic regulatory systems: a literature review. *Journal of Computational Biology*, 9(1):67–103, 2002.

[51] K. A. De Jong. Analysis of the behavior of a class of genetic adaptive systems. 1975.

[52] J. de-la Pena-Sordo, I. Santos, I. Pastor-López, and P. G. Bringas. Filtering Trolling Comments through Collective Classification. In *International Conference on Network and System Security*, pages 707–713. Springer, 2013.

[53] F. Delli Priscoli, A. Di Giorgio, F. Lisi, S. Monaco, A. Pietrabissa, L. Ricciardi Celsi, and V. Suraci. Multi-agent quality of experience control. *International Journal of Control, Automation and Systems*, 15:892–904, 01 2017.

[54] N. Diakopoulos and M. Naaman. Towards quality discourse in online news comments. In *Proceedings of the ACM 2011 conference on Computer supported cooperative work*, pages 133–142. ACM, 2011.

[55] I. O. Dlala, D. Attiaoui, A. Martin, and B. B. Yaghlane. Trolls identification within an uncertain framework. In *Tools with Artificial Intelligence (ICTAI), 2014 IEEE 26th International Conference on*, pages 1011–1015. IEEE, 2014.

[56] M. Dorigo and G. Di Caro. Ant colony optimization: a new meta-heuristic. In *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*, volume 2, pages 1470–1477. IEEE, 1999.

[57] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification*. John Wiley & Sons, 2012.

[58] F. J. Dyson. *Origins of Life*. Cambridge, UK: Cambridge University Press, 1985.

[59] H. Ebel, L.-I. Mielsch, and S. Bornholdt. Scale-free topology of e-mail networks. *Physical Review E*, 66, 2002.

[60] R. Eberhart and J. Kennedy. Particle swarm optimization. In *Proceedings of the IEEE international conference on neural networks*, volume 4, pages 1942–1948. Citeseer, 1995.

[61] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer Publishing Company, Incorporated, 2nd edition, 2015.

[62] M. Eigen and P. Schuster. The hypercycle: A principle of natural self-organisation, part A. *Naturwissenschaften*, 64:541–565, 1977.

[63] M. Eigen and P. Schuster. The hypercycle: A principle of natural self-organization, part B: The abstract hypercycle. *Naturwissenschaften*, 65:7–41, 1978.

[64] C. Emmeche, S. Køppe, and F. Stjernfelt. Explaining emergence: towards an ontology of levels. *Journal for General Philosophy of Science*, 28(1):83–117, 1997.

[65] A. P. Engelbrecht. *Computational Intelligence: An Introduction*. Wiley Publishing, 2nd edition, 2007.

[66] A. P. Engelbrecht, B. Masiye, and G. Pampard. Niching ability of basic particle swarm optimization algorithms. In *Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005.*, pages 397–400. IEEE, 2005.

[67] B. Everitt. *The Cambridge Dictionary of Statistics*. Cambridge University Press, 1996.

[68] L. Faivishevsky and J. Goldberger. A Nonparametric Information Theoretic Clustering Algorithm. In *Proceedings of the 27th International Conference on Machine Learning ICML'10*, pages 351–358, 2010.

[69] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the Internet topology. *SIGCOMM Comput. Commun. Rev.*, 29(4):251–262, Aug. 1999.

[70] J. Farmer, S. A. Kauffman, and N. H. Packard. Autocatalytic replication of polymers. *Physica D: Nonlinear Phenomena*, 22(1):50 – 67, 1986. Proceedings of the Fifth Annual International Conference.

[71] G. Febres and K. Jaffé. Calculating entropy at different scales among diverse communication systems. *Complexity*, 2015.

[72] S. Feldt, J. Waddell, V. Hetrick, J. Berke, and M. Żochowski. Functional clustering algorithm for the analysis of dynamic network data. *Physical Review E*, 79(5):056104, 2009.

[73] A. Filisetti, A. Graudenzi, R. Serra, M. Villani, D. De Lucrezia, R. M. Füchslin, S. A. Kauffman, N. Packard, and I. Poli. A stochastic model of the emergence of autocatalytic cycles. *Journal of Systems Chemistry*, 2(1):2, Jun 2011.

[74] A. Filisetti, A. Graudenzi, R. Serra, M. Villani, R. M. Füchslin, N. Packard, S. A. Kauffman, and I. Poli. A stochastic model of autocatalytic reaction networks. *Theory in Biosciences*, 131(2):85–93, Jun 2012.

[75] A. Filisetti, R. Serra, T. Carletti, M. Villani, and I. Poli. Non-linear protocell models: synchronization and chaos. *The European Physical Journal B*, 77(2):249–256, Sep 2010.

[76] A. Filisetti, M. Villani, C. Damiani, A. Graudenzi, A. Roli, W. Hordijk, and R. Serra. On RAF sets and autocatalytic cycles in random reaction networks. In C. Pizzuti and G. Spezzano, editors, *Advances in Artificial Life and Evolutionary Computation*, pages 113–126, Cham, 2014. Springer International Publishing.

[77] A. Filisetti, M. Villani, A. Roli, M. Fiorucci, I. Poli, and R. Serra. On some properties of information theoretical measures for the study of complex systems in advances in artificial life and evolutionary computation. *Communications in Computer and Information Science*, 445:140–150, 2014.

[78] A. Filisetti, M. Villani, A. Roli, M. Fiorucci, and R. Serra. Exploring the organisation of complex systems through the dynamical interactions among their relevant subsets. In P. Andrews and et al., editors, *Proceedings of the European Conference on Artificial Life 2015, ECAL 2015*, pages 286–293. The MIT Press, 2015.

[79] J. W. Fisher and J. C. Principe. A methodology for information theoretic feature extraction. In *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98CH36227)*, volume 3, pages 1712–1716 vol.3, May 1998.

[80] M. K. Fleming and G. W. Cottrell. Categorization of faces using unsupervised feature extraction. In *1990 IJCNN International Joint Conference on Neural Networks*, pages 65–70 vol.2, June 1990.

[81] P. Fornacciari, M. Mordonini, A. Poggi, L. Sani, and M. Tomaiuolo. A holistic system for troll detection on Twitter. *Computers in Human Behavior*, 89:258–268, 2018.

[82] P. Fornacciari, M. Mordonini, A. Poggi, and M. Tomaiuolo. Software actors for continuous social media analysis. *CEUR Workshop Proceedings*, 1867:84–89, 2017.

[83] S. Franchini, A. Charogiannis, C. N. Markides, M. J. Blunt, and S. Krevor. Calibration of astigmatic particle tracking velocimetry based on generalized Gaussian feature extraction. *Advances in Water Resources*, 124:1 – 8, 2019.

[84] Y. Gao and G. Church. Improving molecular cancer class discovery through sparse non-negative matrix factorization. *Bioinformatics*, 21(21):3970–3975, 2005.

[85] C. Gershenson and N. Fernandez. Complexity and Information: Measuring Emergence, Self-organization, and Homeostasis at Multiple Scales. *Complex.*, 18(2):29–44, Nov. 2012.

[86] S. Ghosh and S. G. Henderson. Behavior of the NORTA method for correlated random vector generation as the dimension increases. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 13(3):276–294, 2003.

[87] C. H. E. Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth International Conference on Weblogs and Social Media (ICWSM-14)*, pages 216–225, 2014.

[88] R. J. Gillies, D. Verduzco, and R. A. Gatenby. Evolutionary dynamics of carcinogenesis and why targeted therapy does not work. *Nature Reviews Cancer*, 12(7):487, 2012.

[89] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5):533–549, 1986.

[90] F. Glover and M. Laguna. Tabu search. In *Handbook of combinatorial optimization*, pages 2093–2229. Springer, 1998.

[91] F. Glover, M. Laguna, and R. Martí. Scatter search. In *Advances in evolutionary computing*, pages 519–537. Springer, 2003.

[92] F. W. Glover and G. A. Kochenberger. *Handbook of metaheuristics*, volume 57. Springer Science & Business Media, 2006.

[93] A. Go, R. Bhayani, and L. Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12):1–6, 2009.

[94] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[95] D. E. Goldberg, J. Richardson, et al. Genetic algorithms with sharing for multimodal function optimization. In *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49. Hillsdale, NJ: Lawrence Erlbaum, 1987.

[96] J. Goldberger, S. Gordon, and H. Greenspan. Unsupervised image-set clustering using an information theoretic framework. *IEEE Transactions on Image Processing*, 15(2):449–458, Feb 2006.

[97] S. Goudarzi, W. H. Hassan, M. H. Anisi, A. Soleymani, M. Sookhak, M. K. Khan, A.-H. A. Hashim, and M. Zareei. ABC-PSO for vertical handover in heterogeneous wireless networks. *Neurocomputing*, 256(Supplement C):63 – 81, 2017. Fuzzy Neuro Theory and Technologies for Cloud Computing.

[98] C. Greenwood and M. S. Nikulin. *A guide to chi-squared testing*. Wiley, 1996.

[99] J. J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Transactions on systems, man, and cybernetics*, 16(1):122–128, 1986.

[100] M. Guerrero, F. G. Montoya, R. Baños, A. Alcayde, and C. Gil. Adaptive community detection in complex networks using genetic algorithms. *Neurocomputing*, 266(Supplement C):101 – 113, 2017.

[101] W. J. Gutjahr. *Convergence Analysis of Metaheuristics*, pages 159–187. Springer US, Boston, MA, 2010.

[102] H. Haken. Introduction to synergetics. In *Synergetics*, pages 9–19. Springer, 1973.

[103] G. R. Harik. Finding Multimodal Solutions Using Restricted Tournament Selection. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 24–31, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[104] M. J. Herrgård, M. W. Covert, and B. Ø. Palsson. Reconstruction of microbial transcriptional regulatory networks. *Current opinion in biotechnology*, 15(1):70–77, 2004.

[105] K. E. Hild, D. Erdogmus, K. Torkkola, and J. C. Principe. Feature extraction using information-theoretic learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1385–1392, Sep. 2006.

[106] M. Hofree, J. P. Shen, H. Carter, A. Gross, and T. Ideker. Network-based stratification of tumor mutations. *Nature methods*, 10(11):1108, 2013.

[107] W. Hordijk, J. Hein, and M. Steel. Autocatalytic Sets and the Origin of Life. *Entropy*, 12(7):1733–1742, 2010.

[108] W. Hordijk and M. Steel. Detecting autocatalytic, self-sustaining sets in chemical reaction systems. *Journal of Theoretical Biology*, 227(4):451 – 461, 2004.

[109] W. Hordijk and M. Steel. A formal model of autocatalytic sets emerging in an RNA replicator system. *Journal of Systems Chemistry*, 4(1):3, Feb 2013.

[110] X. M. Hu, J. Zhang, Y. Yu, H. S. H. Chung, Y. L. Li, Y. H. Shi, and X. N. Luo. Hybrid Genetic Algorithm Using a Forward Encoding Scheme for Lifetime Maximization of Wireless Sensor Networks. *IEEE Transactions on Evolutionary Computation*, 14(5):766–781, Oct 2010.

[111] Y. Huang and R. L. Wange. T cell receptor signaling: beyond complex complexes. *Journal of Biological Chemistry*, 279(28):28827–28830, 2004.

[112] S. Jain and S. Krishna. Autocatalytic Sets and the Growth of Complexity in an Evolutionary Model. *Phys. Rev. Lett.*, 81:5684–5687, Dec 1998.

[113] M. Jaitner. Exercising power in social media. *The fog of cyber defence*, page 57, 2013.

[114] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A.-L. Barabási. The large-scale organization of metabolic networks. *Nature*, 407(6804):651–654, 2000.

[115] J. Jiang, J. Ma, C. Chen, Z. Wang, Z. Cai, and L. Wang. SuperPCA: A Superpixelwise PCA Approach for Unsupervised Feature Extraction of Hyperspectral Imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 56(8):4581–4593, Aug 2018.

[116] L. O. Jimenez-Rodriguez, E. Arzuaga-Cruz, and M. Velez-Reyes. Unsupervised Linear Feature-Extraction Methods and Their Effects in the Classification of High-Dimensional Data. *IEEE Transactions on Geoscience and Remote Sensing*, 45(2):469–483, Feb 2007.

[117] H. John. Adaptation in natural and artificial systems, 1992.

[118] J. Johnson. *Hypernetworks in the science of complex systems*, volume 3. World Scientific, 2013.

[119] S. Johnson. *Emergence: The connected lives of ants, brains, cities, and software*. Simon and Schuster, 2002.

[120] H. Johnston. Cliques of a graph-variations on the Bron-Kerbosch algorithm. *International Journal of Parallel Programming*, 5(3):209–238, 1976.

[121] P. K. Jonason, A. Jones, and M. Lyons. Creatures of the night: Chronotypes and the Dark Triad traits. *Personality and Individual Differences*, 55(5):538 – 541, 2013.

[122] S. Kauffman. Homeostasis and differentiation in random genetic control networks. *Nature*, 224(5215):177, 1969.

[123] S. A. Kauffman. *At home in the universe: the search for laws of self-organization and complexity*. Oxford, UK: Oxford University Press, 1993.

[124] S. A. Kauffman. *The origins of order*. Oxford, UK: Oxford University Press, 1993.

[125] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[126] T. Kohonen. Learning Vector Quantization. In M. A. Arbib, editor, *The handbook of brain theory and neural networks*, pages 537–540. MIT Press, 1998.

[127] T. Kohonen. *Self-organizing maps, 3rd edition*. Springer Berlin, Heidelberg, 2001.

[128] J. R. Koza et al. *Genetic programming II*, volume 17. MIT press Cambridge, 1994.

[129] O. Kramer. *Genetic algorithm essentials*, volume 679. Springer, 2017.

[130] S. Kumar, F. Spezzano, and V. Subrahmanian. Accurately detecting trolls in slashdot zoo via decluttering. In *Advances in Social Networks Analysis and Mining (ASONAM), 2014 IEEE/ACM International Conference on*, pages 188–195. IEEE, 2014.

[131] J. Kunegis, A. Lommatzsch, and C. Bauckhage. The slashdot zoo: mining a social network with negative edges. In *Proceedings of the 18th international conference on World wide web*, pages 741–750. ACM, 2009.

[132] S. E. Lacy, M. A. Lones, and S. L. Smith. Forming classifier ensembles with multimodal evolutionary algorithms. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 723–729, May 2015.

[133] D. Lane, D. Pumain, S. E. van der Leeuw, and G. West. *Complexity perspectives in innovation and social change*, volume 7. Springer Science & Business Media, 2009.

[134] J. Li and Y. Song. Community detection in complex networks using extended compact genetic algorithm. *Soft Computing*, 17(6):925–937, Jun 2013.

[135] X. Li. Adaptively Choosing Neighbourhood Bests Using Species in a Particle Swarm Optimizer for Multimodal Function Optimization. In *in GECCO-2004*, pages 105–116. Springer-Verlag, 2004.

[136] Y. Li, Y. Chen, J. Zhong, and Z. Huang. Niching particle swarm optimization with equilibrium factor for multi-modal optimization. *Information Sciences*, 494:233–246, 2019.

[137] B. Liu and Z. Li. Study on the automatic recognition of hidden defects based on Hilbert Huang transform and hybrid SVM - PSO model. In *2017 Prognostics and System Health Management Conference (PHM-Harbin)*, pages 1–7, July 2017.

[138] A. Lökk and J. Hallman. Viability of Sentiment Analysis for Troll Detection on Twitter: A Comparative Study Between the Naive Bayes and Maximum Entropy Algorithms, 2016.

[139] G. Lombardo, A. Ferrari, P. Fornacciari, M. Mordonini, L. Sani, and M. Tomaiuolo. Dynamics of emotions and relations in a Facebook group of patients with hidradenitis suppurativa. In *International Conference on Smart Objects and Technologies for Social Good*, pages 269–278. Springer, 2017.

[140] G. Lombardo, P. Fornacciari, M. Mordonini, L. Sani, and M. Tomaiuolo. A combined approach for the analysis of support groups on Facebook-the case of patients of hidradenitis suppurativa. *Multimedia Tools and Applications*, 78(3):3321–3339, 2019.

[141] L. O. Loohuis, G. Caravagna, A. Graudenzi, D. Ramazzotti, G. Mauri, M. Antoniotti, and B. Mishra. Inferring tree causal models of cancer progression with probability raising. *PloS one*, 9(10):e108358, 2014.

[142] J. Lu, G. Getz, E. A. Miska, E. Alvarez-Saavedra, J. Lamb, D. Peck, A. Sweet-Cordero, B. L. Ebert, R. H. Mak, A. A. Ferrando, et al. MicroRNA expression profiles classify human cancers. *nature*, 435(7043):834, 2005.

[143] R. D. Luce. The theory of selective information and some of its behavioral applications. In R. D. Luce, editor, *Developments in Mathematical Psychology*, pages 5–119. Free Press, Glencoe, 1960.

[144] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

[145] S. W. Mahfoud. Crowding and preselection revisited. In *PPSN*, volume 2, pages 27–36, 1992.

[146] S. W. Mahfoud. *Niching methods for genetic algorithms*. PhD thesis, Citeseer, 1995.

[147] S. Mansy, J. Schrum, M. Krishnamurthy, S. Tobe, D. Trecol, and J. Szostak. Template-directed synthesis of a genetic polymer in a model protocell. *Nature*, 454, 2008.

[148] J. Marull, C. Font, R. Padró, E. Tello, and A. Panazzolo. Energy–Landscape Integrated Analysis: A proposal for measuring complexity in internal agroecosystem processes (Barcelona Metropolitan Region, 1860–2000). *Ecological Indicators*, 66:30 – 46, 2016.

[149] L. Mendoza and I. Xenarios. A method for the generation of standardized qualitative dynamical systems of regulatory networks. *Theoretical Biology and Medical Modelling*, 3(13), 2006.

[150] L. M. Merlo, J. W. Pepper, B. J. Reid, and C. C. Maley. Cancer as an evolutionary and ecological process. *Nature reviews cancer*, 6(12):924, 2006.

[151] P. Mesejo, O. Ibáñez, O. Cordón, and S. Cagnoni. A survey on image segmentation using metaheuristic-based deformable models: state of the art and critical analysis. *Applied Soft Computing*, 44:1–29, 2016.

[152] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell. *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013.

[153] T. Mihaylov, G. Georgiev, and P. Nakov. Finding Opinion Manipulation Trolls in News Community Forums. In *CoNLL*, pages 310–314, 2015.

[154] T. Mihaylov and P. Nakov. Hunting for troll comments in news community forums. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 399–405, 2016.

[155] G. A. Miller. Note on the bias of information estimates. In H. Quastler, editor, *Information Theory in Psychology*, pages 95–100. Free Press, Glencoe, 1955.

[156] G. A. Miller and W. G. Madow. On the maximum likelihood estimate of the Shannon-Wiener measure of information. Technical Report 54-75, Air Force Cambridge Research Center, 1954.

[157] H. Motoda and H. Liu. Feature Selection Extraction and Construction. 2002.

[158] A. C. Müller, S. Nowozin, and C. H. Lampert. Information Theoretic Clustering Using Minimum Spanning Trees. In *Pattern Recognition*, pages 205–215. Springer Berlin Heidelberg, 2012.

[159] U.-U. Narantsatsralt and S. Kang. Social Network Community Detection Using Agglomerative Spectral Clustering. *Complexity*, 2017.

[160] C. G. A. Network et al. Comprehensive molecular characterization of human colon and rectal cancer. *Nature*, 487(7407):330, 2012.

[161] M. E. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[162] H. B. Nguyen, B. Xue, and P. Andreae. Mutual information for feature selection: estimation or counting? *Evolutionary Intelligence*, 9(3):95–110, Sep 2016.

[163] S. Nik-Zainal, P. Van Loo, D. C. Wedge, L. B. Alexandrov, C. D. Greenman, K. W. Lau, K. Raine, D. Jones, J. Marshall, M. Ramakrishna, et al. The life history of 21 breast cancers. *Cell*, 149(5):994–1007, 2012.

[164] P. C. Nowell. The clonal evolution of tumor cell populations. *Science*, 194(4260):23–28, 1976.

[165] E. Nuño and N. Cutululis. A heuristic for the synthesis of credible operating states in the presence of renewable energy sources. In *2016 International Conference on Probabilistic Methods Applied to Power Systems (PMAPS)*, pages 1–7, Oct 2016.

[166] C. K. Oei, D. E. Goldberg, and S.-J. Chang. Tournament selection, niching, and the preservation of diversity. *Illigal report*, 91011, 1991.

[167] F. J. Ortega, J. A. Troyano, F. L. Cruz, C. G. Vallejo, and F. EnríQuez. Propagation of trust and distrust for the detection of trolls in a social network. *Computer Networks*, 56(12):2884–2895, 2012.

[168] A. Owen. Empirical Likelihood Ratio Confidence Regions. *Ann. Statist.*, 18(1):90–120, 03 1990.

[169] A. Papoulis and S. U. Pillai. *Probability, random variables, and stochastic processes*. Boston: McGraw-Hill, 2015.

[170] W. G. Parrott. *Emotions in social psychology: Essential readings*. Psychology Press, 2001.

[171] K. Parsopoulos, V. P. Plagianakos, G. Magoulas, and M. Vrahatis. *Improving Particle Swarm Optimizer by Function "Stretching"*, pages 445–457. 01 2001.

[172] K. Parsopoulos and M. Vrahatis. Modification of the particle swarm optimizer for locating all the global minima. In *Artificial Neural Nets and Genetic Algorithms*, pages 324–327. Springer, 2001.

[173] A. Passaro and A. Starita. Particle Swarm Optimization for Multimodal Functions: A Clustering Approach. *Journal of Artificial Evolution and Applications*, 2008.

[174] M. W. Pereira, G. S. Neto, and M. Roisenberg. A topological niching covariance matrix adaptation for multimodal optimization. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 2562–2569, July 2014.

[175] R. Perry and D. Green. *Perry's Chemical Engineers' Handbook*. McGraw-Hill, 8^{th} edition, 2007.

[176] A. Pétrowski. A clearing procedure as a niching method for genetic algorithms. In *Proceedings of IEEE international conference on evolutionary computation*, pages 798–803. IEEE, 1996.

[177] C. Pizzuti. *GA-Net: A Genetic Algorithm for Community Detection in Social Networks*, pages 1081–1090. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[178] R. Poli. Analysis of the publications on the applications of particle swarm optimisation. *Journal of Artificial Evolution and Applications*, 2008, 2008.

[179] R. Poli, J. Kennedy, and T. Blackwell. Particle swarm optimization. *Swarm Intelligence*, 1(1):33–57, 2007.

[180] R. Poli, W. B. Langdon, N. F. McPhee, and J. R. Koza. *A field guide to genetic programming*. Lulu. com, 2008.

[181] M. Preuss. *Multimodal optimization by means of evolutionary algorithms*. Springer, 2015.

[182] M. Prokopenko, F. Boschetti, and A. J. Ryan. An information-theoretic primer on complexity, self-organization, and emergence. *Complexity*, 15(1):11–28, 2009.

[183] T. Qiao, Z. Yang, J. Ren, P. Yuen, H. Zhao, G. Sun, S. Marshall, and J. A. Benediktsson. Joint bilateral filtering and spectral similarity-based sparse representation: A generic framework for effective feature extraction and data classification in hyperspectral imaging. *Pattern Recognition*, 77:316 – 328, 2018.

[184] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, Mar 1986.

[185] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[186] G. R. Raidl and J. Gottlieb. Empirical analysis of locality, heritability and heuristic bias in evolutionary algorithms: A case study for the multidimensional knapsack problem. *Evolutionary Computation*, 13(4):441–475, 2005.

[187] D. Ramazzotti, G. Caravagna, L. Olde Loohuis, A. Graudenzi, I. Korsunsky, G. Mauri, M. Antoniotti, and B. Mishra. CAPRI: efficient inference of cancer progression models from cross-sectional data. *Bioinformatics*, 31(18):3016–3026, 2015.

[188] D. Ramazzotti, A. Graudenzi, L. De Sano, M. Antoniotti, and G. Caravagna. Learning mutational graphs of individual tumour evolution from single-cell and multi-region sequencing data. *BMC bioinformatics*, 20(1):210, 2019.

[189] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A.-L. Barabási. Hierarchical organization of modularity in metabolic networks. *Science*, 297(5586):1551–1555, 2002.

[190] E. Remy, P. Ruet, L. Mendoza, D. Thieffry, and C. Chaouiya. From logical regulatory graphs to standard Petri nets: dynamical roles and functionality of

feedback circuits. In *Transactions on Computational Systems Biology VII*, pages 56–72. Springer, 2006.

[191] R. Righi, A. Roli, M. Russo, R. Serra, and M. Villani. New Paths for the Application of DCI in Social Sciences: Theoretical Issues Regarding an Empirical Analysis. In F. Rossi, S. Piotto, and S. Concilio, editors, *Advances in Artificial Life, Evolutionary Computation, and Systems Chemistry*, pages 42–52, Cham, 2017. Springer International Publishing.

[192] F. Riquelme and P. González-Cantergiani. Measuring user influence on Twitter: A survey. *Information Processing & Management*, 52(5):949–975, 2016.

[193] R. Roberto Serra and M. Villani. *Modelling Protocells*. Springer Netherlands Science+Business Media Dordrecht,, 2017.

[194] A. Roli, M. Villani, R. Caprari, and R. Serra. Identifying Critical States through the Relevance Index. *Entropy*, 19(73):1–15, 2017.

[195] S. Rosenbaum. Is Twitter Toxic? Can Social Media Be Tamed? *Forbes*, 2016(Sep 9), 2016.

[196] K. Ruiz-Mirazo, C. Briones, and A. de la Escosura. Prebiotic Systems Chemistry: New Perspectives for the Origins of Life. *Chemical Reviews*, 114(1):285–366, 2014.

[197] E. Ruppin, J. A. Papin, L. F. De Figueiredo, and S. Schuster. Metabolic reconstruction, constraint-based analysis and game theory to probe genome-scale metabolic networks. *Current opinion in biotechnology*, 21(4):502–510, 2010.

[198] H. Saif, M. Fernández, Y. He, and H. Alani. On stopwords, filtering and data sparsity for sentiment analysis of twitter. 2014.

[199] M. Sales-Pardo, R. Guimera, A. A. Moreira, and L. A. N. Amaral. Extracting the hierarchical organization of complex systems. *Proceedings of the National Academy of Sciences*, 104(39):15224–15229, 2007.

[200] L. Sani, M. Amoretti, E. Vicari, M. Mordonini, R. Pecori, A. Roli, M. Villani, S. Cagnoni, and R. Serra. Efficient search of relevant structures in complex systems. In *AI*IA 2016 Advances in Artificial Intelligence: XVth International Conference of the Italian Association for Artificial Intelligence, Genova, Italy, November 29 – December 1, 2016, Proceedings*, pages 35–48. Springer International Publishing, 2016.

[201] L. Sani, A. Bononi, R. Pecori, M. Amoretti, M. Mordonini, A. Roli, M. Villani, S. Cagnoni, and R. Serra. An improved relevance index method to search important structures in complex systems. In *Italian Workshop on Artificial Life and Evolutionary Computation*, pages 3–16. Springer, 2018.

[202] L. Sani, G. D'Addese, A. Graudenzi, and M. Villani. The detection of dynamical organization in cancer evolution models. In *International Workshop on Artificial Life and Evolutionary Computation*. Springer, 2019.

[203] L. Sani, G. D'Addese, R. Pecori, M. Mordonini, M. Villani, and S. Cagnoni. An integration-based approach to pattern clustering and classification. In C. Ghidini, B. Magnini, A. Passerini, and P. Traverso, editors, *AI*IA 2018 – Advances in Artificial Intelligence*, pages 362–374, Cham, 2018. Springer International Publishing.

[204] L. Sani, G. Lombardo, R. Pecori, P. Fornacciari, M. Mordonini, and S. Cagnoni. Social relevance index for studying communities in a facebook group of patients. In K. Sim and P. Kaufmann, editors, *Applications of Evolutionary Computation*, pages 125–140. Springer International Publishing, 2018.

[205] L. Sani, R. Pecori, P. Fornacciari, M. Mordonini, M. Tomaiuolo, and S. Cagnoni. A relevance index-based method for improved detection of malicious users in social networks. In *International Workshop on Artificial Life and Evolutionary Computation*. Springer, 2019.

[206] L. Sani, R. Pecori, M. Mordonini, and S. Cagnoni. From complex system analysis to pattern recognition: Experimental assessment of an unsupervised

feature extraction method based on the relevance index metrics. *Computation*, 7(3):39, 2019.

[207] L. Sani, R. Pecori, E. Vicari, M. Amoretti, M. Mordonini, and S. Cagnoni. Can the relevance index be used to evolve relevant feature sets? In K. Sim and P. Kaufmann, editors, *Applications of Evolutionary Computation*, pages 472–479, Cham, 2018. Springer International Publishing.

[208] C. T. Sari and C. Gunduz-Demir. Unsupervised Feature Extraction via Deep Learning for Histopathological Classification of Colon Tissue Images. *IEEE Transactions on Medical Imaging*, 38(5):1139–1149, May 2019.

[209] I. Schoeman and A. Engelbrecht. Using vector operations to identify niches for particle swarm optimization. In *IEEE Conference on Cybernetics and Intelligent Systems, 2004.*, volume 1, pages 361–366. IEEE, 2004.

[210] I. L. Schoeman. *Niching in Particle Swarm Optimization*. PhD thesis, School of Engineering, University of Pretoria, 2010.

[211] B. Scholkopf and A. J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[212] H.-P. P. Schwefel. *Evolution and optimum seeking: the sixth generation*. John Wiley & Sons, Inc., 1993.

[213] J. Scott. *Social Network Analysis*. Sage Publications, 2017.

[214] R. Serra and M. Villani. *Modelling Protocells: The Emergent Synchronization of Reproduction and Molecular Replication*. Springer Netherlands, Dordrecht, 2017.

[215] R. Serra, M. Villani, and A. Semeria. Genetic network models and statistical properties of gene expression data in knock-out experiments. *Journal of Theoretical Biology*, 227(1):149 – 157, 2004.

[216] C. Shalizi, M. Camperi, and K. Klinkner. Discovering Functional Communities in Dynamical Networks. In Airoldi, E. et al., editor, *Statistical Network Analysis: Models, Issues, and New Directions: ICML 2006 Workshop on Statistical Network Analysis, Pittsburgh, PA, USA, June 29, 2006, Revised Selected Papers*, pages 140–157, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[217] Y. Shi and F. Sha. Information-theoretical Learning of Discriminative Clusters for Unsupervised Domain Adaptation. In *Proceedings of the 29th International Conference on Machine Learning*, ICML'12, pages 1275–1282, USA, 2012. Omnipress.

[218] P. Siarry. *Metaheuristics*, volume 23. Springer, 2016.

[219] G. Silvestri, L. Sani, M. Amoretti, R. Pecori, E. Vicari, M. Mordonini, and S. Cagnoni. Searching relevant variable subsets in complex systems using k-means pso. In M. Pelillo, I. Poli, A. Roli, R. Serra, D. Slanzi, and M. Villani, editors, *Artificial Life and Evolutionary Computation*, pages 308–321. Springer International Publishing, 2018.

[220] H. Simon. The architecture of complexity. *Proc. Amer. Phil. Soc.*, 106(6), 1962.

[221] R. V. Solé, A. Munteanu, C. Rodriguez-Caso, and J. Macía. Synthetic protocell biology: from reproduction to computation. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 362(1486):1727–1739, 2007.

[222] O. Sporns, G. Tononi, and G. Edelman. Theoretical Neuroanatomy: Relating Anatomical and Functional Connectivity in Graphs and Cortical Connection Matrices. *Cerebral Cortex*, 10(2):127–141, 2000.

[223] Q. Sun, Y. Wang, Y. Jiang, L. Shao, and D. Chen. Fault diagnosis of SEPIC converters based on PSO-DBN and wavelet packet energy spectrum. In *2017 Prognostics and System Health Management Conference (PHM-Harbin)*, pages 1–7, July 2017.

[224] P. Suppes. A probabilistic theory of causality. 1973.

[225] M. Tabor. *Chaos and integrability in nonlinear dynamics*, volume 3. Wiley New York, 1989.

[226] Y.-H. Taguchi. Tensor decomposition-based and principal-component-analysis-based unsupervised feature extraction applied to the gene expression and methylation profiles in the brains of social insects with multiple castes. *BMC Bioinformatics*, 19(4):99, May 2018.

[227] Y.-H. Taguchi. Tensor Decomposition-Based Unsupervised Feature Extraction Can Identify the Universal Nature of Sequence-Nonspecific Off-Target Regulation of mRNA Mediated by MicroRNA Transfection. *Cells*, 7(6), 2018.

[228] M. Tasgin, A. Herdagdelen, and H. Bingol. Community detection in complex networks using genetic algorithms. *arXiv preprint arXiv:0711.0491*, 2007.

[229] R. Thomas and M. Kaufman. Multistationarity, the basis of cell differentiation and memory. I. Structural conditions of multistationarity and other nontrivial behavior. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 11(1):170–179, 2001.

[230] R. Thomas, D. Thieffry, and M. Kaufman. Dynamical behaviour of biological regulatory networks—I. Biological role of feedback loops and practical use of the concept of the loop-characteristic state. *Bulletin of Mathematical Biology*, 57(2):247–276, 1995.

[231] G. Tononi, A. McIntosh, D. Russel, and G. Edelman. Functional clustering: Identifying strongly interactive brain regions in neuroimaging data. *Neuroimage*, 7:133–149, 1998.

[232] G. Tononi, O. Sporns, and G. M. Edelman. A measure for brain complexity: relating functional segregation and integration in the nervous system. *Proceedings of the National Academy of Sciences*, 91(11):5033–5037, 1994.

[233] W. Vance, A. Arkin, and J. Ross. Determination of causal connectivities of species in reaction networks. *Proceedings of the National Academy of Sciences*, 99(9):5816–5821, 2002.

[234] V. Vasas, C. Fernando, M. Santos, S. Kauffman, and E. Szathmáry. Evolution before genes. *Biology Direct*, 7(1):1, Jan 2012.

[235] G. Ver Steeg, A. Galstyan, F. Sha, and S. DeDeo. Demystifying Information-theoretic Clustering. In *Proceedings of the 31st International Conference on International Conference on Machine Learning ICML'14*, pages I–19 – I–27, 2014.

[236] J. R. Vergara and P. A. Estévez. A review of feature selection methods based on mutual information. *Neural computing and applications*, 24(1):175–186, 2014.

[237] E. Vicari, M. Amoretti, L. Sani, M. Mordonini, R. Pecori, A. Roli, M. Villani, S. Cagnoni, and R. Serra. Gpu-based parallel search of relevant variable sets in complex systems. In *Italian Workshop on Artificial Life and Evolutionary Computation*, pages 14–25. Springer, 2016.

[238] M. Villani, A. Barbieri, and R. Serra. A dynamical model of genetic networks for cell differentiation. *PloS one*, 6(3):e17703, Jan. 2011.

[239] M. Villani, S. Benedettini, A. Roli, D. Lane, I. Poli, and R. Serra. Identifying emergent dynamical structures in network models. In *Recent Advances of Neural Network Models and Applications*, pages 3–13. Springer, 2014.

[240] M. Villani, A. Filisetti, S. Benedettini, A. Roli, D. Lane, and R. Serra. The detection of intermediate-level emergent structures and patterns. In Miglino, O. et al., editor, *Advances in Artificial Life, ECAL 2013*, pages 372–378. The MIT Press, 2013.

[241] M. Villani, A. Roli, A. Filisetti, M. Fiorucci, I. Poli, and R. Serra. The Search for Candidate Relevant Subsets of Variables in Complex Systems. *Artificial Life*, 21(4), 2015.

[242] M. Villani, L. Sani, M. Amoretti, E. Vicari, R. Pecori, M. Mordonini, S. Cagnoni, and R. Serra. A relevance index method to infer global properties of biological networks. In M. Pelillo, I. Poli, A. Roli, R. Serra, D. Slanzi, and M. Villani, editors, *Artificial Life and Evolutionary Computation*, pages 129–141, Cham, 2018. Springer International Publishing.

[243] M. Villani, L. Sani, R. Pecori, M. Amoretti, A. Roli, M. Mordonini, R. Serra, and S. Cagnoni. An iterative information-theoretic approach to the detection of structures in complex systems. *Complexity*, 2018:1–15, 2018.

[244] B. Vogelstein, N. Papadopoulos, V. E. Velculescu, S. Zhou, L. A. Diaz, and K. W. Kinzler. Cancer genome landscapes. *science*, 339(6127):1546–1558, 2013.

[245] M. Wang and F. Sha. Information Theoretical Clustering via Semidefinite Programming. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 761–769, 2011.

[246] D. J. Watts and S. H. Strogatz. Collective dynamics of "small-world" networks. *Nature*, 393(6684):440–442, 1998.

[247] S. S. Wilks. The Large-Sample Distribution of the Likelihood Ratio for Testing Composite Hypotheses. *The Annals of Mathematical Statistics*, 9:60–62, 1938.

[248] A. Will, J. Bustos, M. Bocco, J. Gotay, and C. Lamelas. On the use of niching genetic algorithms for variable selection in solar radiation estimation. *Renewable Energy*, 50:168 – 176, 2013.

[249] S. Wolfram. Statistical mechanics of cellular automata. *Reviews of modern physics*, 55(3):601, 1983.

[250] W. Xie, B. L. Nelson, and R. R. Barton. Statistical uncertainty analysis for stochastic simulation with dependent input models. In *Proceedings of the Winter Simulation Conference 2014*, pages 674–685, Dec 2014.

[251] X. Xu and Z. Yan. Probabilistic load flow evaluation with hybrid Latin Hypercube Sampling and multiple linear regression. In *2015 IEEE Power Energy Society General Meeting*, pages 1–5, July 2015.

[252] B. Xue, M. Zhang, W. Browne, and X. Yao. A Survey on Evolutionary Computation Approaches to Feature Selection. *IEEE Transactions on Evolutionary Computation*, 20(4):606–626, 2016.

[253] X.-S. Yang, Z. Cui, R. Xiao, A. H. Gandomi, and M. Karamanoglu. *Swarm intelligence and bio-inspired computation: theory and applications*. Newnes, 2013.

[254] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Icml*, volume 97, page 35, 1997.

[255] V. Yannibelli and A. Amandi. A deterministic crowding evolutionary algorithm to form learning teams in a collaborative learning context. *Expert Systems with Applications*, 39(10):8584 – 8592, 2012.

[256] S. Zhan, J. Wu, N. Han, J. Wen, and X. Fang. Unsupervised feature extraction by low-rank and sparsity preserving embedding. *Neural Networks*, 109:56 – 66, 2019.

[257] H. Zhang, T. B. Ho, Y. Zhang, and M. S. Lin. Unsupervised Feature Extraction for Time Series Clustering Using Orthogonal Wavelet Transform. *Informatica (Slovenia)*, 30:305–319, 2006.

[258] J. Zhang, J. Yu, and D. Tao. Local Deep-Feature Alignment for Unsupervised Dimension Reduction. *IEEE Transactions on Image Processing*, 27(5):2420–2432, May 2018.

Acknowledgments

The work described in this thesis originates from a collaboration between the University of Parma, the University of Modena and Reggio Emilia, and the University of Bologna.

I would like to thank my advisor Stefano Cagnoni, who gave me the opportunity to work on this project. I would like to express my gratitude to him for his support, for his dedication, for the revision of this thesis, and for the inspiring conversations we had.

I would like to thank Marco Villani, Andrea Roli and Roberto Serra, who involved us in the RI project. Their useful advice has been an important guide for the development of this work. It has been a pleasure to learn from them and to exchange opinions with them. I would also like to thank them for their support during my first conference as a speaker, just before the beginning of my PhD. They made me feel at home and they welcomed me to the inspiring "Wivace" community.

I would like to thank the professors and the colleagues of the SoWide research group: Monica Mordonini, Michele Tomaiuolo, Agostino Poggi, Paolo Fornacciari, Gianfranco Lombardo, Giulio Angiani, Alberto Ferrari and Eleonora Iotti. They gave me the opportunity to work on a lot of different projects, expanding my knowledge on different topics.

I would like to thank Riccardo Pecori for his help in the RI project, for his enthusiasm and for his support.

I would like to thank Michele Amoretti for his support in the RI project, for the implementation of the first version of the GPU parallel code for the T_c index

computation, and for his useful advice.

I would like to thank everyone who has collaborated with our group in the RI project, in particular, Luca La Rocca, Gianluca D'Addese, Alberto Bononi, Emilio Vicari, Gianluigi Silvestri, Chiara Lasagni and Alex Graudenzi.

I would like to thank all the colleagues and friends who have been by my side during these years. In particular, a special thank goes to Marina Raineri and Andrea Minari, who shared this "journey" with me since the beginning.

I would like to thank all the Computer Engineering professors of the University of Parma, from whom I learned a lot, and Paolo Federici for his technical help.

I would like to thank the students who worked with me during these years on their thesis and projects and who attended my classes. They gave me the opportunity to learn how to become a "teacher". I feel I have "grown up" with them.

I would also like to express my gratitude to my family for being always by my side, helping me, and believing in me. I would like to thank them for being an example and a reference in my life. This thesis is dedicated to them.