BABEŞ-BOLYAI UNIVERSITY
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
EÖTVÖS LORÁND UNIVERSITY
FACULTY OF INFORMATICS

Agent Based Pattern Recognition

PhD Thesis Abstract

PhD student: Radu D. Găceanu
Scientific supervisor UBB: Prof. Dr. Horia F. Pop
Scientific supervisor ELTE: Assoc. Prof. Dr. Habil. László Kozma

2012

Acknowledgments

I would like to express my gratitude to my supervisors, Prof. Dr. Horia F. Pop and Assoc. Prof. Dr. Habil. László Kozma for their guidance, suggestions and support. I would also like to thank to all my collaborators.

The author would like to thank for the financial support provided from programs co-financed by The Sectorial Operational Programme Human Resources Development, Contract POSDRU 6/1.5/S/3 "Doctoral studies: through science towards society".

List of publications

[CDG07] C. Chira, D. Dumitrescu, and **R. D. Găceanu**. Stigmergic agent systems for solving NP-hard problems. *Studia Informatica*, Special Issue KEPT-2007: Knowledge Engineering: Principles and Techniques (June 2007):177–184, June 2007. **(indexed MathSciNet, Zentralblatt MATH, EBSCO Publishing).**

[GP10] **R. D. Găceanu** and H. F. Pop. An adaptive fuzzy agent clustering algorithm for search engines. In *MACS2010: Proceedings of the 8th Joint Conference on Mathematics and Computer Science*, pages 185–196. Komarno, Slovakia, 2010. **(indexed MathematicalReviews).**

[GP11a] **R. D. Găceanu** and H. F. Pop. A context-aware ASM-based clustering algorithm. *Studia Universitatis Babes-Bolyai Series Informatica*, LVI(2):55–61, 2011. **(indexed MathSciNet, Zentralblatt MATH, EBSCO Publishing).**

[GO11] **R. D. Găceanu** and G. Orbán. Using rsl to describe the stock exchange domain. In *microCAD International Scientific Conference*. University of Miskolc, Hungary, 31 March – 1 April 2011. **(International conference).**

[GP11b] **R. D. Găceanu** and H. F. Pop. A fuzzy clustering algorithm for dynamic environments. In *KEPT2011: Knowledge Engineering Principles and Techniques, Selected Papers, Eds: M. Frentiu, H.F. Pop, S. Motogna*, pages 119–130. Babes-Bolyai University, Cluj-Napoca, Romania, July 4–6 2011. **(ISI — Conference Proceedings Citation Index).**

[GP11c] **R. D. Găceanu** and H. F. Pop. An incremental ASM-based fuzzy clustering algorithm. In *Informatics'2011, Slovakia, i'11:Proceedings of the Eleventh International Conference on Informatics, Informatics 2011, Eds: V. Novitzká, Štefan Hudák*, pages 198–204. Slovak Society for Applied Cybernetics and Informatics, Rožňava, Slovakia, November 16–18 2011. **(indexed MathematicalReviews).**

[Găc11] **Radu D. Găceanu**. A bio-inspired fuzzy agent clustering algorithm for search engines. *Procedia Computer Science*, 7(0):305 – 307, 2011. Proceedings of the 2nd European Future Technologies Conference and Exhibition 2011 (FET 11), 10.1016/j.procs.2011.09.060. http://www.sciencedirect.com/science/article/pii/S187705091100620X. **(ISI — Conference Proceedings Citation Index).**

[CCGa] Gabriela Czibula, Istvan Czibula, and **Radu D. Găceanu**. Intelligent data structures selection using neural networks. *Knowledge and Information Systems*, Springer London, pages 1–22. 10.1007/s10115-011-0468-3. **(ISI — Science Citation Index Expanded) IF = 2.0** (2010)

[GP12] R. D. Găceanu and H. F. Pop. An incremental approach to the set covering problem. *Studia Universitatis Babes-Bolyai Series Informatica*, LVIII(2), 2012. — under review. **(indexed MathSciNet, Zentralblatt MATH, EBSCO Publishing).**

[CCGb] Gabriela Czibula, Istvan Czibula, and **Radu D. Găceanu**. A Support Vector Machine Model For Intelligent Selection of Data Representations. *Applied Soft Computing* — under review. **(ISI — Science Citation Index Expanded) IF = 2.1** (2010)

Contents

Contents of the thesis

Introduction

This work is the result of my research in the field of Pattern Recognition, particularly Agent Based Pattern Recognition, research conducted under the supervision of both Prof. Dr. Horia F. Pop (starting from 2008) and of Assoc. Prof. Dr. Habil. László Kozma (starting from 2009).

The research topic is about using several types of software agents in pattern recognition. We will investigate in the thesis the use of agents in NP-hard optimization problems as well as in hybrid data analysis.

The rapid growth of data comes with the natural need for extracting and analysing meaningful information and knowledge from this data. This information and knowledge could be used in different applications, ranging from fraud detection, to production control, market basket analysis, customer analytics and so on. Data analysis can be viewed as a step forward in the information technology evolution. It is the process of inspecting, transforming, and modelling data with the goal of uncovering patterns, associations and anomalies and thus support decision making.

An important step in data mining is pattern recognition which deals with assigning a label to a given input data. Classification and clustering are examples of pattern recognition. Classification and clustering can be applied in many fields like in marketing (for finding groups of customers with similar behaviour), biology (classification of plants and animals given their features), fraud detection, and document classification.

Classification is the process of assigning a label to a piece of input data based, for example, on a predefined model. Since the class label of each training data item is provided, classification is a supervised learning problem. On the other hand, clustering is an unsupervised learning problem and it deals with finding a structure in a collection of unlabelled data. Classification and clustering together with a general overview of data analysis are presented in Chapter 1.

In both classification and clustering object data belonging to the same class or cluster have to be similar with each other and items from different classes or clusters have to be as dissimilar as possible. This implies a great deal of imprecision and uncertainty and a way to handle this is by using soft computing methods. Soft computing deals with imprecision and uncertainty in the attempt to achieve robustness and low cost solutions. This multidisciplinary field was introduced by Lotfi A. Zadeh and its main goal is to develop intelligent systems and to solve mathematically unmodelled problems [Zad97, CMR$^+$07, VSP09].

Soft Computing opens the possibility of solving complex problems for which a mathematical model is not available. Moreover, it introduces human knowledge like cognition, recognition, learning into the field of computing. This opens the way for constructing intelligent, autonomous, self-tuning systems.

In order to design autonomous and intelligent systems software agents are employed. An agent is an entity that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors. An agent that always tries to optimize an appropriate performance measure is called a rational agent. Agents exhibit several characteristics ([SP04, Ser06]) from which the most interesting one is self-organization. It is the capability of an entity to organize and improve its behaviour without being guided or managed. Agents seldom reside alone in the environment. Instead they coexist and interact forming multi-agent systems. Chapter 1 presents several types of interactions in a multi-agent system.

The thesis is structured in four chapters as follows.

Chapter 1, **Theoretical background**, introduces the field of data analysis and agent-based data analysis. Data analysis is becoming increasingly popular, due to the rapid growth of data amounts and the natural need for extracting meaningful information and knowledge from this data. The information and knowledge could be used in different applications ranging from intrusion detection systems, to production control, pattern recognition and so on. Data analysis can be viewed as a step forward in the information technology evolution. This chapter presents some of the most important problems in data analysis like clustering and classification and also the use of software agents.

The Chapters 2, 3 and 4 contain our original contribution in the field of agent-based pattern recognition. For each original approach that we propose, we outline possibilities for improvement and future research directions. Chapter 5 outlines the conclusions of the thesis.

Chapter 2, **Contributions to NP optimization problems**, begins with a short overview of NP completeness and NP optimization problems in Section 2.1. The rest of the chapter is entirely original and presents our contribution to NP optimization problems, focusing on two well-known NP-hard problems: Travelling Salesman Problem (TSP) and Set Covering Problem (SCP). In Section 2.1 a short overview of NP completeness is made. In Section 2.2 the travelling salesman problem is approached using the stigmergic agent model. The Stigmergic Agent System (SAS) combines the strengths of Multi-agent Systems (MAS) and Ant Colony Systems (ACS). Stigmergy provides a general mechanism that relates individual and colony level behaviours: individual behaviour modifies the environment, which in turn modifies the behaviour of other individuals. The stigmergic agent mechanism employs several agents able to interoperate in order to solve problems by using both direct communication and indirect (stigmergic) communication. The algorithm was evaluated on several standard datasets outlining the potential of the method. In Section 2.3 the soft agent model is introduced. A soft agent is an intelligent agent that may deal with imprecision, uncertainty, partial truth and approximation during its execution as a reactive agent or goal oriented agent or both. This new agent model is used in Section 2.4 where a new incremental clustering approach to the Set Covering Problem is presented. Experiments on standard datasets suggest that the approach is promising. Section 2.5 outlines the conclusions of the chapter and indicates future research directions.

Chapter 3, **New approaches to unsupervised learning**, begins with a short overview of various agent-based clustering approaches in Section 3.1. The rest of the chapter is entirely original and presents our contribution to agent-based clustering, particularly in two main directions: ASM-based batch clustering and incremental clustering [EKS+98, Kam10, LKC02, LLLH10, DL11]. We are focusing on developing clustering algorithms that allow the discovery and analysis of hybrid data. In Section 3.1 a short overview of various agent-based clustering approaches is presented. We approach the idea of agent-based cluster analysis in Section 3.2. Each data is represented by an agent placed in a two dimensional grid. The agents will group themselves into clusters by making simple moves according to some local environment information and the parameters are selected and adjusted adaptively. This behaviour based on ASM (Ant Sleeping Model [CXC04]) where an agent may be either in an active state or in a sleeping state. In order to avoid the agents being trapped in local minima, they are also able to directly communicate with each other. Furthermore, the agent moves are expressed by fuzzy IF-THEN rules and hence hybridization with a classical clustering algorithm is needless. The proposed fuzzy ASM-based clustering algorithm is presented in Section 3.2.1. In this model data items to be clustered are represented by agents that are able to react according to the changes in the environment, namely the number of neighbouring agents. However a change in the data item itself is not handled at runtime. An extension to a context-aware system would be beneficial in many practical situations. In general, context-aware systems could greatly change the way we interact with the world — they could anticipate our needs and advice us when taking some decisions. In a changing environment context-awareness is undoubtedly beneficial. Such systems could make much more relevant recommendations and support decision making. An extension to a context-aware approach is presented in Section 3.2.2. Case studies for both approaches including experiments on standard datasets [Iri88, Win91] are presented in Section 3.2.3. The idea behind incremental clustering is that it is possible to consider one instance at a time and assign it to one of the already built clusters without significantly affecting the already existing structures. Section 3.3 presents an incremental clustering approach based on ASM. In incremental clustering only the cluster representations need to be kept in memory so not the entire dataset and thus the space requirements for such an algorithm are very small. Whenever a new instance is considered an incremental clustering algorithm would basically try to assign it to one of the already exiting clusters. Such a process is not very complex and therefore the time requirements for an incremental clustering algorithm are also small. The fuzziness of the approach allows the discovery of hybrid data. Experimental evaluation on standard datasets [Iri88, Win91] are presented in Section 3.3.3. Section 3.4 outlines the conclusions of the chapter and indicates some research directions that will be followed.

Chapter 4, **New supervised learning approaches to software development**, is entirely original and it focuses on the problem of dynamically selecting, using supervised learning approaches, the most suitable representation for an abstract data type, according to the software system's current execution context. In this direction, a neural network approach and a support vector machine approach are proposed. Selecting and creating the appropriate data structure for implementing an abstract data type (ADT) can greatly influence the performance of a software system. It is not a trivial problem for a software developer, as it is hard to anticipate all the usage scenarios of the deployed application. It is not clear how to select a good implementation for an abstract data type when access patterns to it are highly variant, or even unpredictable. Due to this fact, the software system may choose the appropriate data representation, at runtime, based on the effective data usage pattern. This dynamic selection can be achieved using machine learning techniques, which can assure complex and adaptive systems development. In this chapter we approach the problem of dynamically selecting, using supervised learning approaches, the most suitable representation for an abstract data type according to the software system's current execution context. In this direction, a neural network model and a support vector machine model are proposed. The considered problem arises from practical needs, it has a major importance for software developers. Improper use of data structures in software applications leads to performance degradation and high memory consumption. These problems can be avoided by properly selecting data structures for implementing ADTs, according to the nature of the manipulated data. In Section 4.1 the problem of dynamic data structure selection is presented. It is explained that this is a complex problem because each particular data structure is usually more efficient for some operations and less efficient for others and that is why a static analysis for choosing the best representation can be inappropriate, as the performed operations can not be statically predicted. A practical example is presented and an experiment is performed in order to motivate our approach. In Section 4.2 we present our first proposal of using supervised learning for dynamically selecting the implementation of an abstract data type from the software system, based on its current execution context. For this purpose, a neural network model will be used. In fact, selecting the most appropriate implementation of an abstract data type is equivalent to predicting, based on the current execution context, the type and the number of operations performed on the ADT, on a certain execution scenario. In Section 4.3 we evaluate the accuracy of the technique proposed in Section 4.2, i.e. the ANN model's prediction accuracy. Starting from a data set given at [For10], we have simulated an experiment for selecting the most appropriate data structure for implementing the *List* ADT. Experimental results suggest that our approach provides optimized data structure selection and reduces the computational time by selecting the data structure implementation which provides a minimum overall complexity for the operations performed on a certain abstract data type on a given execution scenario. Section 4.4 presents a comparison to related work. In Section 4.5 the problem of data representation selection problem (DRSP) is approached using support vector machines. Computational experiments from Section 4.6 confirm a good performance of the proposed model and indicates the potential of our proposal. The advantages of our approach in comparison with similar approaches are also emphasized in Section 4.7.

Chapter 5, **Conclusions**, draws the conclusions of the thesis.

The original contributions introduced by this thesis are contained in Chapters 2, 3 and 4 and they are as follows:

- A stigmergic agent system algorithm for solving the travelling salesman problem (Section 2.2) [CDG07].
- A new model for software agents: the soft agent model (Section 2.3) [GP12].
- An incremental clustering algorithm for solving the set covering problem (Section 2.4) [GP12].

- Experimental evaluation of both algorithms on standard datasets (Section 2.2 and Section 2.4) [CDG07, GP12].
- A fuzzy ASM-based clustering algorithm (Section 3.2.1) [GP10, Găc11].
- A context-aware fuzzy clustering algorithm (Section 3.2.2) [GP11a, GP11b].
- An incremental fuzzy clustering algorithm (Section 3.3) [GP11c].
- Experimental evaluation of the algorithms on standard datasets (Section 3.2.3 and Section 3.3.3) [GP10, Găc11, GP11a, GP11b, GP11c].
- The discovery and analysis of hybrid data (Section 3.2.3 and Section 3.3.3) [GP11a, GP11b, GP11c].
- The applicability of the fuzzy ASM-based methods in clustering web search results (Section 3.2.3) [GP10, Găc11].
- A supervised learning approach for the dynamic selection of abstract data types implementations during the execution of a software system, in order to increase the system's efficiency (Section 4.2) [CCGa, CCGb].
- A neural networks approach to the considered problem (Section 4.2.2) [CCGa].
- Accuracy evaluation of the proposed neural network based technique on a case study (Section 4.3) [CCGa].
- A support vector machines approach to the considered problem (Section 4.5.3) [CCGb].
- Accuracy evaluation of the proposed support vector machine based technique on a case study (Section 4.6) [CCGb].
- A comparison of the advantages of the proposed supervised learning approaches to DRSP with existing similar approaches (Section 4.4 and Section 4.7) [CCGa, CCGb].

Chapter 1

Theoretical background

Data analysis is becoming increasingly popular due to the rapid growth of data amounts and the natural need for extracting meaningful information and knowledge from this data. Various applications ranging from production control to intrusion detection systems and pattern recognition may benefit from the extracted information and learned knowledge. Data analysis may be seen as a step forward in the evolution of information technology. In this chapter some of the most important problems in pattern recognition are presented, namely, clustering and classification. The field of soft computing is briefly presented in Section 1.3 and the topic of multi agent interactions is presented in Section 1.4.

1.1 Data analysis and data mining

The process of gathering, modelling and transforming data in the attempt to extract relevant information that may support decision making is called data analysis. Data mining focuses on modelling predictive rather than purely descriptive purposes and it is a particular technique of data analysis. Business intelligence refers to data analysis techniques applied mainly in analysing business data aiming to increase decision making support. Data analysis may be divided into exploratory data analysis and confirmatory data analysis. Exploratory data analysis deals with discovering new features in the data. On the other hand, confirmatory data analysis deals with confirming or falsifying given hypotheses. However there are several data analysis varieties. Predictive analytics, for example, applies statistical models in forecasting future events. Text analytics focuses on applying various techniques to extract and classify information contained in sources of textual data.

1.2 Pattern recognition

Pattern recognition is the problem of assigning a label to a given input data. According to the type of the involved learning procedure, algorithms in pattern recognition may be categorized in supervised learning, unsupervised learning and semi-supervised learning algorithms. In the following we will refer to clustering which is an unsupervised learning problem and to classifications which is a supervised learning problem.

According to [Mar09], clustering is "the most important unsupervised learning problem". Given a collection of unlabelled data the goal of a clustering process is to find a structure in the considered dataset. In [Mar09] clustering is defined as "the process of organizing objects into groups whose members are similar in some way". So a cluster is a group of objects which are "similar" between them and are "dissimilar" to the objects belonging to other clusters [Mar09].

Classification is the process of assigning a piece of input data (instance or data item) described by a vector of features to a given category (class). Initially a classifier is built based on a given dataset. This step is the training (learning) phase and at this point a classifier is built by learning from a training dataset containing a set of data items with their features and the associated class labels. Because the class label of each training data item is provided, classification is a supervised learning problem. More formally, classification may be seen as learning a mapping, $y = f(X)$, such that given a data item from X the class label y can be predicted. This model is then used for classification.

The performance or the quality of a classifier can be evaluated by computing its accuracy, i.e., the percentage of test set data items that are correctly classified. Other performance measures could be the computation speed, robustness (the ability to handle noisy or missing data), scalability and interpretability (the level of understanding and insight that is provided by the classifier) [HK06].

1.3 Soft computing

Proposed by Lotfi A. Zadeh [Bla94b], soft computing [Zad94] is a multidisciplinary field that deals with imprecision, uncertainty, approximation, and partial truth in order to achieve robustness and low cost solutions. The main objective of soft computing is the development of intelligent systems and solving non-linear and mathematically complicated to model problems [Zad97]. The main advantages of soft computing are:

- it opens the possibility of solving complex problems, in which mathematical models are not available
- it introduces the human knowledge such as cognition, recognition, understanding, learning, and others into the field of computing and hence opens the way for constructing intelligent, autonomous, self-tuning systems.

Soft computing comprises, but is not limited to, the following components: fuzzy systems, neural networks, swarm intelligence, evolutionary computing. Fuzzy sets [Zad65] represent a mathematical theory for modelling imprecision and they are central to soft computing. They were introduced by Zadeh, having a major success initially in Japan and China and then in the whole world [Bla94a].

1.4 Multi agent interactions

A multi agent system (MAS) is a system composed of several interacting agents. Multi-agent systems may be used for solving problems which are difficult or impossible for an individual agent or a monolithic system to solve. Communication is crucial in MAS. In general direct communication is assumed in a classical MAS and in this case we deal with intelligent agents. But communication could also be done indirectly, through the environment. In this case we deal with formations of simple creatures like ant colonies or bird flocks which collectively lead to the emergence of intelligent global behaviour, to what is known as swarm intelligence.

An agent is an entity that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors [SP04]. An agent that always tries to optimize an appropriate performance measure is called a rational agent. Such a definition of a rational agent is fairly general and can include human agents (having eyes as sensors, hands as actuators), robotic agents (having cameras as sensors, wheels as actuators), or software agents (having a graphical user interface as sensor and as actuator).

Ant colony optimization (ACO) [DS04] is a nature-inspired metaheuristic that addresses combinatorial optimization (CO) problems [PS82]. Some inherently hard problems can be addressed using metaheuristics [BR03]. Examples of metaheuristics are: ACO, tabu search, simulated annealing and evolutionary computation. It is important to note that these are approximation algorithms, i.e., they are used for obtaining good enough solutions in an acceptable amount of time [Glo89, Glo90, KJV83].

Chapter 2

Contributions to NP optimization problems

The chapter begins with a short overview of NP completeness and NP optimization problems in Section 2.1. The rest of the chapter is entirely original and presents our contribution to NP optimization problems, focusing on two well-known NP-hard problems: Travelling Salesman Problem (TSP) and Set Covering Problem (SCP).

The approaches presented in this chapter represent original works published in [CDG07, GP12].

The chapter is structured as follows. In Section 2.1 a short overview of NP completeness is made. In Section 2.2 the travelling salesman problem is approached using the stigmergic agent model. The Stigmergic Agent System (SAS) combines the strengths of Multi-agent Systems (MAS) and Ant Colony Systems (ACS). Stigmergy provides a general mechanism that relates individual and colony level behaviours: individual behaviour modifies the environment, which in turn modifies the behaviour of other individuals. The stigmergic agent mechanism employs several agents able to interoperate in order to solve problems by using both direct communication and indirect (stigmergic) communication. The algorithm was evaluated on several standard datasets outlining the potential of the method. In Section 2.3 the soft agent model is introduced. A soft agent is an intelligent agent that has to deal with imprecision, uncertainty, partial truth and approximation during its execution as a reactive agent or goal oriented agent or both. This new agent model is used in Section 2.4 where a new incremental clustering approach to the Set Covering Problem is presented. Experiments on standard datasets suggest that the approach is promising. Section 2.5 outlines the conclusions of the chapter and indicates future research directions.

The original contributions of this chapter are:

- A stigmergic agent system algorithm for solving the travelling salesman problem (Section 2.2) [CDG07].
- A new model for software agents: the soft agent model (Section 2.3) [GP12].
- An incremental clustering algorithm for solving the set covering problem (Section 2.4) [GP12].
- Experimental evaluation of both algorithms on standard datasets (Section 2.2 and Section 2.4) [CDG07, GP12].

2.1 NP-completeness

In computational complexity theory, the complexity class NP-complete is a class of decision problems for which the required time for solving using any currently known algorithm increases very quickly as the size of the problem grows. Even though a method for computing the solutions to NP-complete problems using a reasonable amount of time remains unknown, it is still necessary to deal with these problems. In many situations approximation algorithms are used in order to address NP-complete problems [CLRS09].

2.2 Stigmergic agents

In [CDG07] a Stigmergic Agent System (SAS) combining the strengths of Ant Colony Systems and Multi-Agent Systems concepts is proposed. The agents from the SAS are using both direct and indirect (stigmergic) communication. Stigmergy occurs as a result of individuals interacting with and changing an environment [DS04]. Stigmergy was originally discovered and named in 1959 by Grasse, a French biologist studying ants and termites. Grasse was intrigued by the idea that these simple creatures were able to build such complex structures. The ants are not directly communicating with each other and have no plans, organization or control built into their brains or genes. Nevertheless, ants lay pheromones during pursuits for food, thus changing the environment. Even though ants are not able to directly communicate with each other, they do communicate however — indirectly — through pheromones.

Stigmergy provides a general mechanism that relates individual and colony level behaviours: individual behaviour modifies the environment, which in turn modifies the behaviour of other individuals.

The SAS mechanism employs several agents able to interoperate on the following two levels in order to solve problems:

- direct communication: agents are able to exchange different types of messages in order to share knowledge and support direct interoperation; the knowledge exchanged refers to both local and global information
- indirect (stigmergic) communication: agents have the ability to produce pheromone trails that influence future decisions of other agents within the system.

2.3 Soft agents

An architecture based on agents using indirect communication is proposed in [Ste90]. In [HSP08] social relationships are modelled using a fuzzy-agent model. In [CXC04] an ant-based clustering algorithm is presented. It is based on the ASM (Ants Sleeping Model) approach. In [CDG07] a Stigmergic Agent System (SAS) combining the strengths of Ant Colony Systems and Multi-Agent Systems concepts is proposed. The agents from the SAS are using both direct and indirect communication. By using direct communication the risk of getting trapped in local optima is lower. However, as showed in [SCCK04], most ant-based algorithms can be used only in a first phase of the clustering process because of the high number of clusters that are usually produced. In a second phase a k-means-like algorithm is often used. In [SCCK04], an algorithm in which the behaviour of the artificial ants is governed by fuzzy IF-THEN rules is presented. Like all ant-based clustering algorithms, no initial partitioning of the data is needed, nor should the number of clusters be known in advance. The ants are capable to make their own decisions about picking up items. Hence the two phases of the classical ant-based clustering algorithm are merged into one, and k-means becomes superfluous. In the approaches from [GP10, GP11a] fuzzy agents are employed for solving the clustering problem. Agent moves are expressed by fuzzy IF-THEN rules and hence hybridization with a classical clustering algorithm is needless.

A soft agent is an intelligent agent that has to deal with imprecision, uncertainty, partial truth and approximation during its execution as a reactive agent or goal oriented agent or both. An important property of the soft agents is that they only can sense their local environment. They can communicate with remote agents, but their vision is limited to a local neighbourhood and they maintain little information about their state. Nevertheless they can act on the global environment. A soft agent perceives its local environment though the *Perception* layer. This layer is also responsible for listening to messages from other, possibly remote, agents. The *Controller* layer is responsible for deciding which of the following layers should have control over the agent. The control layer can be implemented as a set of control rules which can also act as a filter suppressing information from sensors. The reactive layer provides an immediate response to changes that occur in the local environment. Roughly speaking, it implements a mapping *situation* → *action*. The *Procative* layer achieves the agent's proactive behaviour, it ensures that the agent reaches its goal. The *Action* layer is responsible for executing the selected action on the environment (local or global) and with dispatching messages to other agents.

2.4 A new approach to the set covering problem

The set covering problem is a classical problem in computer science and complexity theory and it serves as a model for many real-world applications especially in the resource allocation area. In an environment where the demands that need to be covered change over time, special methods are needed that adapt to such changes. We reformulate the set covering problem as a clustering problem where the within cluster sum of squared errors to be minimized corresponds to the cost associated to a certain set covering that needs to be minimal. We have developed an incremental clustering algorithm in order to address the set covering problem. The algorithm continuously considers new items to be clustered. Whenever a new data item arrives it is encapsulated by an agent which will autonomously decide to be included in a certain cluster in the attempt to either maximize its cover or minimize the cost. We have introduced the soft agent model in order to encapsulate this behaviour. Initial tests suggest the potential of our approach.

The Set Covering Problem (SCP) is a classical problem in computer science and complexity theory and it serves as a model for many applications in the real world like: facility location problem, airline crew scheduling, resource allocation, assembly line balancing, vehicle routing, information retrieval etc. Let us consider a set X and a family F of subsets of X such that every element from X belongs to at least one subset from F. The set covering problem is the problem of finding a minimum number of subsets from F (or subsets of minimum cost) such that their union is the set X.

In our model the input is an $m \times n$ incidence matrix A, where $m = \mid X \mid$ and each column corresponds to a set S_j with $j \in \{1, \ldots, n\}$. Each column j has a corresponding cost $c_j > 0$. We say that a column j covers a row i if $a_{ij} = 1$. In our incremental approach, whenever a new data item arrives it is encapsulated by an agent which will autonomously decide to join a certain cluster in the attempt to either maximize the cluster cover or minimize its the cost. We have used soft agents in order to deal with the two conflicting objectives: maximize cover and minimize cost. As in any approximation algorithm an optimal solution is not guaranteed to be found, the purpose being to find reasonably good solutions fast enough. Ongoing tests on large datasets [Bea] suggest promising results.

2.5 Conclusions and future work

In this chapter we have presented our contributions to NP optimization problems namely to the travelling salesman problem and to the set covering problem. These approaches have been presented in our original papers [CDG07, GP12].

We have seen that the proposed SAS approach for solving TSP is a powerful optimization technique that combines the advantages of two models: Ant Colony Systems and Multi-Agent Systems. Interoperation between agents is based on both indirect communication — given by pheromone levels — and direct knowledge sharing, greatly reducing the risk of falling into the trap of local minima. Experimental results on standard datasets outline the advantage of the approach over the classical Ant Colony Systems. We have also developed an incremental clustering algorithm in order to address the set covering problem. The algorithm continuously considers new items to be clustered. Whenever a new data item arrives it is encapsulated by an agent which will autonomously decide to join a certain cluster in the attempt to either maximize the cluster cover or minimize its the cost. We have used soft agents in order to deal with the two conflicting objectives: maximize cover and minimize cost. As in any approximation algorithm an optimal solution is not guaranteed to be found, the purpose being to find reasonably good solutions fast enough. Ongoing tests on large datasets [Bea] suggest promising results.

For each original approach proposed in this chapter we have emphasized improvement possibilities and possible future extensions.

As future research directions we intend to improve the approaches presented in this chapter, to extend the evaluation of the proposed techniques and to investigate and develop other computational models for addressing NP-hard problems.

Ongoing research focuses on numerical experiments to demonstrate the robustness of the proposed model. The SAS method has to be further refined in terms of types of messages that agents can directly exchange. Furthermore, other metaheuristics are investigated with the aim of identifying additional potentially beneficial hybrid models.

Chapter 3

New approaches to unsupervised learning

This chapter begins with a short overview of various agent-based clustering approaches in Section 3.1. The rest of the chapter is entirely original and presents our contribution to agent-based clustering, particularly in two main directions: ASM-based batch clustering and incremental clustering. We are focusing on developing clustering algorithms that allow the discovery and analysis of hybrid data.

The unsupervised learning approaches presented in this chapter are original works published in [GP10, GP11a, GP11b, Găc11, GP11c].

The chapter is structured as follows. In Section 3.1 a short overview of various agent-based clustering approaches is presented. We approach the idea of agent-based cluster analysis in Section 3.2. Each data is represented by an agent placed in a two dimensional grid. The agents will group themselves into clusters by making simple moves according to some local environment information and the parameters are selected and adjusted adaptively. This behaviour based on ASM (Ant Sleeping Model) where an agent may be either in an active state or in a sleeping state. In order to avoid the agents being trapped in local minima, they are also able to directly communicate with each other. Furthermore, the agent moves are expressed by fuzzy IF-THEN rules and hence hybridization with a classical clustering algorithm is needless. The proposed fuzzy ASM-based clustering algorithm is presented in Section 3.2.1. In this model data items to be clustered are represented by agents that are able to react according to the changes in the environment, namely the number of neighbouring agents. However a change in the data item itself is not handled at runtime. An extension to a context-aware system would be beneficial in many practical situations. In general, context-aware systems could greatly change the way we interact with the world — they could anticipate our needs and advice us when taking some decisions. In a changing environment context-awareness is undoubtedly beneficial. Such systems could make much more relevant recommendations and support decision making. An extension to a context-aware approach is presented in Section 3.2.2. Case studies for both approaches including experiments on standard datasets [Iri88, Win91] are presented in Section 3.2.3. The idea behind incremental clustering is that it is possible to consider one instance at a time and assign it to existing clusters without significantly affecting the already existing structures. Section 3.3 presents an incremental clustering approach based on ASM. In incremental clustering only the cluster representations need to be kept in memory so not the entire dataset and thus the space requirements for such an algorithm are very small. Whenever a new instance is considered an incremental clustering algorithm would basically try to assign it to one of the already exiting clusters. Such a process is not very complex and therefore the time requirements for an incremental clustering algorithm are also small. The fuzziness of the approach allows the discovery of hybrid data. Experimental evaluation on standard datasets [Iri88, Win91] are presented in Section 3.3.3. Section 3.4 outlines the conclusions of the chapter and indicates some research directions that will be followed.

The original contributions of this chapter are:

- A fuzzy ASM-based clustering algorithm (Section 3.2.1) [GP10, Găc11].
- A context-aware fuzzy clustering algorithm (Section 3.2.2) [GP11a, GP11b].

- An incremental fuzzy clustering algorithm (Section 3.3) [GP11c].
- Experimental evaluation of the algorithms on standard datasets (Section 3.2.3 and Section 3.3.3) [GP10, Găc11, GP11a, GP11b, GP11c].
- The discovery and analysis of hybrid data (Section 3.2.3 and Section 3.3.3) [GP11a, GP11b, GP11c].
- The applicability of the fuzzy ASM-based methods in clustering web search results (Section 3.2.3) [GP10, Găc11].

3.1 Agent-based unsupervised learning

Several clustering algorithms exist each with its own strengths and weaknesses. Some algorithms need an initial estimation of the number of clusters (k-means, fuzzy c-means); others could often be too slow (agglomerative hierarchical clustering algorithms). Ant-based clustering algorithms often require hybridization with a classical clustering algorithm such as k-means.

3.2 ASM-based clustering

In ASM (Ants Sleeping Model), an agent located on a two-dimensional grid may be in any of the following states: active or sleeping. When the agent's fitness is low, it has a higher probability to wake up and start searching for a more secure and comfortable position to sleep in. When such a position in located, the agent has a higher probability to move in a sleeping sate until the surrounding environment becomes less hospitable and activates it again. At the beginning of every considered ASM-based clustering approach the agents are randomly scattered on the grid in active state. In each loop, after the agent moves to a new position, it will recalculate its current fitness f_{disim} and probability p_a so as to decide whether it needs to continue moving. While the p_a is high the agent is likely in active state and it continues to move on the grid. If the current p_a becomes small, the agent has a lower probability of continuing to move on the grid and it may stop at the current position and switch to sleeping state. With increasing number of iterations, such movements gradually increase, eventually, making similar agents gathered within a small area and different types of agents located in separated areas. Thus, the corresponding data items are clustered.

The agents decide upon the way they move on the grid according to their similarity with the neighbours, using fuzzy IF-THEN rules. Thus two agents can be similar (S), different (D), very different (VD). If two agents are similar they would get closer to each other. If they are different or very different they will get away from each other. The number of steps they do each time they move depend on the similarity level. So if the agents are VD they would jump many steps away from each other; if they are D they would jump less steps away from each other. In the end the ants which are S will be in the same cluster. The parameter α is the average distance between agents and this changes at each step further influencing the fitness function. The parameter λ influences the agents' activation pressure and it may decrease over time. The parameter t is used for the termination condition which could be something like $t < t_{max}$. The parameters s_x, s_y, the agent's vision limits, may also be updated in some situations.

The skeleton of the context aware approach is based on the ASM-like algorithm from [CXC04] embellished with features from [CDG07, GP10, SCCK04]. The agents decide upon the way they move on the grid according to their similarity with the neighbours, using fuzzy IF-THEN rules. Thus two agents can be similar (S), different (D), very different (VD). If two agents are similar or very similar they would get closer to each other. If they are different or very different they will get away from each other. The number of steps they do each time they move depend on the similarity level. So if the agents are VD they would jump many steps away from each other; if they are D they would jump less steps away from each other. In the end the ants which are S will be in the same cluster. The similarity computation is taking into account the actual structure of the data or the data density from the agent's neighbourhood; a bigger change from one agent to another translates into a certain similarity which then affects the agent's movement on the grid.

Computational experiments showing the potential of the proposed method are presented. In the first case study a custom dataset is considered and comparison with the k-means clustering is done suggesting the strength of the proposed algorithm. In the second case study the algorithm is tested on a larger dataset. Comments regarding the performance together with idea for further improvements are presented. The third case study is presenting a possible application of this clustering approach in a real-life scenario — clustering web search results [GP10]. The advantages and disadvantages of the proposed techniques over similar approaches are also discussed in the thesis.

3.3 Incremental clustering

The idea behind incremental clustering is that it is possible to consider one instance at a time and assign it to existing clusters without significantly affecting the already existing structures. The incremental approach to clustering is also applicable in online situations like wireless sensor networks or data streams. Ongoing research is done in the area sensor data and data stream mining [SdLFdCG09, HZK+09, GKS09]. In [SdLFdCG09], a new approach to novelty detection in data streams is presented. The ability to detect new concepts is an important aspect in machine learning systems. The approach presented in this paper [SdLFdCG09] takes novelty detection beyond one-class classification, by detecting emerging cohesive and representative clusters of examples, and then further by merging similar concepts. The proposed method goes in the direction of constructing a class structure that aims at reproducing the real one in an unsupervised continuous learning fashion. The paper [HZK+09] presents a general approach for context-aware adaptive mining of data streams that tries to dynamically and autonomously adjust data stream mining parameters according to changes in context and situations. Data stream processing adaptation to variations of data rates and resource availability is crucial for consistency and continuity of running applications like health care systems. In [GKS09] a new data model called Spatio-Temporal Sensor Graphs (STSG), which is designed to model sensor data on a graph by allowing the edges and nodes to be modelled as time series of measurement data is presented. It is shown how this model could be applied in finding patterns like growing hotspots in sensor data. The case studies and the related study show that the presented model is less memory expensive. At the beginning of the algorithm from [CXC04], the agents are randomly scattered on the grid in active state. They randomly move on the grid. In each loop, after the agent moves to a new position, it will recalculate its current fitness f and probability p_a so as to decide whether it needs to continue moving.

In order to test the algorithm in a real-world scenario, the Iris dataset [Iri88] was considered for a first test case. For the second case study the wine dataset [Win91] was considered. This dataset contains the results of a chemical analysis of wines grown in the same region in Italy but derived from three different wine growers. The analysis determined the quantities of 13 constituents found in each of the three types of wines. In [Win91] it is mentioned that the initial dataset had 30 attributes. So the current dataset has 13 attributes plus the class. There are 178 instances grouped in three classes corresponding to the three wine growers. Items ranging from 1 to 59 belong to the first class, items from 60 to 130 belong to the second class and items from 131 to 178 belong to the third class.

3.4 Conclusions and future work

We have introduced in this chapter new agent-based unsupervised learning approaches based on our original papers [GP10, GP11a, GP11b, Găc11, GP11c].

The algorithms presented in Section 3.2 are based on the adaptive ASM approach from [CXC04]. The major improvement is that, instead to moving the agents at a randomly selected site, we are letting the agents choose the best location. Agents can directly communicate with each other — similar to the approach from [CDG07]. In [SCCK04], the fuzzy IF-THEN rules are used for deciding if the agents are picking up or dropping an item. In our model we are using the fuzzy rules for deciding upon the direction and length of the movement. Moreover, in the approach from Section 3.2.2 the agents are able to adapt their movements if changes in the environment would occur. Case studies for these approaches have been performed in Section 3.2.3. In order to test the algorithm in a real-world scenario, the Iris and Wine datasets have been considered [Iri88, Win91]. Experiments outline the ability of our approaches to discover hybrid data. In Section 3.3 an incremental clustering algorithm is introduced. Incremental clustering is used to process sequential, continuous data flows or data streams and in situations in which cluster shapes change over time. Such algorithms are well fitted in real-time systems, wireless sensor networks or data streams because in such systems it is difficult to store the datasets in memory. The algorithm considers one instance at a time and it basically tries to assign it to one of the existing clusters. Only cluster representations need to be kept in memory so computation is both fast and memory friendly. We have seen in the tests from the incremental approach (Section 3.3.3) that most of the apparently classification errors were actually items that have high membership degrees to more than one cluster. Nevertheless, in our opinion, it is again clear that we are dealing with hybrid data. Actually the hybrid nature of the data is suggested in [Iri88] and in [Win91] and this is the main reason for choosing these datasets for our analysis. By using fuzzy methods such features of the data are easy to be observed. The fact that there are hybrid items could be an indication of the quality of data.

For each approach proposed in this chapter we have outlined the advantages and drawbacks and emphasised improvement possibilities and directions for further extension.

As future research directions we intend to improve the approaches presented in this chapter, to extend the evaluation of the proposed techniques and to investigate the use of various metaheuristics in unsupervised learning.

Chapter 4

New supervised learning approaches to software development

This chapter is entirely original and it focuses on the problem of dynamically selecting, using supervised learning approaches, the most suitable representation for an abstract data type according to the software system's current execution context. In this direction, a neural network approach and a support vector machine approach are proposed.

The supervised learning approaches for the problem of automatic selection of data representations presented in this chapter are original works published in [CCGa] and under review in [CCGb].

Selecting and creating the appropriate data structure for implementing an abstract data type (ADT) can greatly impact the performance of a software system. It is not a trivial problem for a software developer, as it is hard to anticipate all the usage scenarios of the deployed application. It is not clear how to select a good implementation for an abstract data type when access patterns to it are highly variant, or even unpredictable. Due to this fact, the software system may choose the appropriate data representation, at runtime, based on the effective data usage pattern. This dynamic selection can be achieved using machine learning techniques, which can assure complex and adaptive systems development.

In this chapter we approach the problem of dynamically selecting, using supervised learning approaches, the most suitable representation for an abstract data type according to the software system's current execution context. In this direction, a neural network model and a support vector machine model are proposed. The considered problem arises from practical needs, it has a major importance for software developers. Improper use of data structures in software applications leads to performance degradation and high memory consumption. These problems can be avoided by properly selecting data structures for implementing ADTs, according to the nature of the manipulated data.

To our knowledge, so far, there are no existing machine learning approaches for the problem of automatic selection of data representations.

The chapter is structured as follows. In Section 4.1 the problem of dynamic data structure selection is presented. It is explained that this is a complex problem because each particular data structure is usually more efficient for some operations and less efficient for others and that is why a static analysis for choosing the best representation can be inappropriate, as the performed operations can not be statically predicted. A practical example is presented and an experiment is performed in order to motivate our approach. In Section 4.2 we present our first proposal of using supervised learning for dynamically selecting the implementation of an abstract data type from the software system, based on its current execution context. For this purpose, a neural network model will be used. In fact, selecting the most appropriate implementation of an abstract data type is equivalent to predicting, based on the current execution context, the type and the number of operations performed on the ADT, on a certain execution scenario. In Section 4.3 we evaluate the accuracy of the technique proposed in Section 4.2, i.e. the ANN model's prediction accuracy. Starting from a data set given at [For10], we have simulated an experiment for selecting the most appropriate data structure for implementing the *List* ADT. Experimental results suggest that our approach provides optimized data structure selection and reduces the computational time by selecting the data structure implementation which provides a minimum overall complexity for the operations performed on a certain abstract data type on a given execution scenario. Section 4.4 presents a comparison to related work. In Section 4.5 the problem of data representation selection problem (DRSP) is approached using support vector machines. Computational experiments from Section 4.6 confirm a good performance of the proposed model and indicates the potential of our proposal. The advantages of our approach in comparison with similar approaches are also emphasized in Section 4.7.

The original contributions of this chapter are:

- To introduce a supervised learning approach for the dynamic selection of abstract data types implementations during the execution of a software system, in order to increase the system's efficiency (Section 4.2) [CCGa, CCGb] .
- To approach the considered problem using neural networks (Section 4.2.2) [CCGa].
- To evaluate the accuracy of the proposed neural network based technique on a case study (Section 4.3) [CCGa].
- To approach the considered problem using support vector machines (Section 4.5.3) [CCGb].
- To evaluate the accuracy of the proposed support vector machine based technique on a case study (Section 4.6) [CCGb].
- To emphasize the advantages of the proposed supervised learning approaches to DRSP in comparison with existing similar approaches (Section 4.4 and Section 4.7) [CCGa, CCGb] .

4.1 The problem of dynamic data structure selection

Abstract data types (ADTs) [WB01] are used in software applications to model real world entities from the application domain. An ADT can be implemented using different data structures. The study of data structures and the algorithms that manipulate them is among the most fundamental topics in computer science [Mou01]. Most of what computer systems spend their time doing is storing, accessing, and manipulating data in one form or another. There are numerous examples from all areas of computer science where a relatively simple application of good data structure techniques resulted in massive savings in computation time and, hence, money.

Let us consider that in a software application a *Collection* ADT (also known as *Bag*) is used. The main operations supported by a *collection* of elements are: **insertion** of an element into the collection, **deletion** of an element from the collection and **searching** an element in the collection. In order to better motivate our approach, we performed an experiment considering the *List* ADT and three data structures for implementing a *List*: **vector** (dynamic array), **linked list** and **balanced search tree**. The main operations supported by a *list* of elements are: **insertion** of an element into the list (at the beginning, at the end, at a certain position), **deletion** of an element from the list (a given element or from a given position), **searching** an element in the list, **iterating** through the list, **accessing** an element from the list at a certain position and **updating** an element from a certain position.

4.2 Automatic selection of data representations using ANN

Data structures [WB01] provide means to customize an abstract data type according to a given usage scenario. The volume of the processed data and the data access flow in the software application influence the selection of the most appropriate data structure for implementing a certain abstract data type. During the execution of the software application, the data flow and volume is fluctuating due to external factors (such as user interaction), that is why the data structure selection has to be dynamically adapted to the software system's execution context. This adaptation has to be made during the execution of the software application and it is hard or even impossible to predict by the software developer. Consequently, in our opinion, machine learning techniques would provide a better selection at runtime of the appropriate data structure for implementing a certain abstract data type.

Artificial neural networks are emerging as the technology of choice for many applications, such as pattern recognition, speech recognition [SH07], prediction [LB05], system identification and control. We will use a *feedforward* neural network that will be trained using the *backpropagation-momentum* learning technique [RN02].

4.3 Experimental evaluation

In this section we aim at evaluating the accuracy of the technique proposed in Section 4.2, i.e. the ANN model's prediction accuracy.

As there is no publicly available case study for the problem of automatic selection of data representations, nor a case study in the related literature that can be reproduced, we consider our own case study. We describe in this section simulation results of applying our learning based approach to a selection problem that will be described below. Starting from the data set given at [For10], we have simulated an experiment for selecting the most appropriate data structure for implementing the *List* ADT. The considered data set consists of the results of a chemical analysis of wines grown in the same region in Italy but derived from different cultivars. The analysis determined the quantities of 13 constituents found in each types of wines [Win91].

The data set for evaluating the ANN classification model presented in Section 4.2 consists of (input, output) samples collected and pre-processed as we have described in Subsection 4.2.2. An input represents an *execution context* and the target output is the *most suitable implementation* for the *List* ADT (1, 2 or 3 according to the selected implementation). In our case study, as the instantiation of the *List* ADT occurs in the $Wine$ class, an *execution context* will contain the values of the attributes of this class (13 attributes corresponding to the wine constituents described at [Win91]). The collected data set consists of 178 input-output samples and will be denoted by $\mathcal{D}$.

Considering the experimental results presented above, we can conclude that our approach provides optimized data structure selection and reduces the computational time by selecting the data structure implementation which provides a minimum overall complexity for the operations performed on a certain abstract data type on a given execution scenario.

4.4 Comparison to related work

In this section we aim at providing a brief comparison of our approach with several existing approaches for the problem of automatic selection of data representations. To our knowledge, so far, there are no existing machine learning approaches for the considered problem, and, moreover, there are no publicly available case studies for it.

4.5 Automatic selection of data representations using SVM

The design and implementation of efficient abstract data types are important issues for software developers. Selecting and creating the appropriate data structure for implementing an abstract data type is not a trivial problem for a software developer, as it is hard to anticipate all the use scenarios of the deployed application. Moreover, it is not clear how to select a good implementation for an abstract data type when access patterns to it are highly variant, or even unpredictable. The problem of automatic data structure selection is a complex one because each particular data structure is usually more efficient for some operations and less efficient for others, that is why a static analysis for choosing the best representation can be inappropriate, as the performed operations can not be statically predicted. Therefore, we propose a predictive model in which the software system learns to choose the appropriate data representation, at runtime, based on the effective data usage pattern. This paper describes a new attempt to use a Support Vector Machine model in order to dynamically select the most suitable representation for an aggregate according to the software system's execution context. Computational experiments confirm a good performance of the proposed model and indicates the potential of our proposal. The advantages of our approach in comparison with similar approaches are also emphasized.

The study of data structures and the algorithms that manipulate them is among the most fundamental topics in computer science [Mou01]. Most of what computer systems spend their time doing is storing, accessing, and manipulating data in one form or another. There are numerous examples from all areas of computer science where a relatively simple application of good data structure techniques resulted in massive savings in computation time and, hence, money. Software applications use abstract data types (ADTs) [WB01] to model real world entities from the application domain. An ADT can be implemented using different data structures.

Let us consider that in a software application a *Collection* ADT (also known as *Bag*) is used. The main operations supported by a *collection* of elements are: **insertion** of an element into the collection, **deletion** of an element from the collection and **searching** an element in the collection. In order to better motivate our approach, we performed an experiment considering the *List* ADT and three data structures for implementing a *List*: **vector** (dynamic array), **linked list** and **balanced search tree**. The main operations supported by a *list* of elements are: **insertion** of an element into the list (at the beginning, at the end, at a certain position), **deletion** of an element from the list (a given element or from a given position), **searching** an element in the list, **iterating** through the list, **accessing** an element from the list at a certain position and **updating** an element from a certain position.

In this section we present several existing approaches for the problem of automatic selection of data repesentations. To our knowledge, so far, there are no existing machine learning approaches for the considered problem, and, moreover, there are no publicly available case studies for it.

Data structures [WB01] provide means to customize an abstract data type according to a given usage scenario. The volume of the processed data and the data access flow in the software application influence the selection of the most appropriate data structure for implementing a certain abstract data type. During the execution of the software application, the data flow and volume is fluctuating due to external factors (such as user interaction), that is why the data structure selection has to be dynamically adapted to the software system's execution context. This adaptation has to be made during the execution of the software application and it is hard or even impossible to predict by the software developer. Consequently, in our opinion, machine learning techniques would provide a better selection at runtime of the appropriate data structure for implementing a certain abstract data type.

First, the software system S is monitored during the execution of a set of scenarios that include the instantiation of the abstract data type $\mathcal{T}$. The result of this supervision performed by a software developer is a set of *execution contexts*, as well as the type and the number of operations from $\mathcal{O}$ performed on $\mathcal{T}$ saved in a log file. The software developer will analyze the resulted log file and will decide, for each *execution context* (input) , the *most suitable implementation* for $\mathcal{T}$ given the *execution context* (output). This decision will be based on computing the global computational complexity of the operations performed on $\mathcal{T}$ during the scenario given by the *execution context* for each possible implementation of $\mathcal{D}_i$ of $\mathcal{T}$ and then selecting the implementation that minimizes the overall complexity.

SVMs use a technique known as the "kernel trick" to apply linear classification techniques to non-linear classification problems. Using a Kernel function [Vap00], the data points from the input space are mapped into a higher dimensional space. Constructing (via the Kernel function) a separating hyperplane with maximum margin in the higher dimensional space yields a non-linear decision boundary in the input space separating the tuples of one class from another.

In our current implementation, we have considered *execution contexts* of radius 0 (i.e. $\mathcal{R} = 0$). This means that the execution context contains only the state of the object that uses the abstract data type $\mathcal{T}$ considered for optimisation.

4.6 Computational experiments

In this section we aim at evaluating the accuracy of the technique proposed in Section 4.2, i.e. the SVM classification model's prediction accuracy. As there is no publicly available case study for the problem of automatic selection of data representations, nor a case study in the related literature that can be reproduced, we consider our own case studies. We describe in this section simulation results of applying our classification approach to two selection problems that will be described in the following.

Starting from the data set given at [For10], we have simulated an experiment for selecting the most appropriate data structure for implementing the *List* ADT. The considered data set consists of the results of a chemical analysis of wines grown in the same region in Italy but derived from different cultivars. The analysis determined the quantities of 13 constituents found in each types of wines. More details about this data set can be found at [Win91].

The data set for evaluating the SVM classification model presented in Section 4.2 consists of (input, output) samples collected and pre-processed. An input represents an *execution context* and the target output is the most suitable implementation for the *List* ADT (**vector**, **linked list** or **balanced search tree**) within the input *execution context*. The data set consists of 178 samples. An overall *learning acuracy* of 0.9625 was obtained.

We will consider a real software system as a case study for evaluating the learning accuracy of the SVM. It is a DICOM (*Digital Imaging and Communications in Medicine*) [DICiM11] and HL7 (*Health Level 7*) [HL0] compliant PACS (*Picture Archiving and Communications System*) system, facilitating medical images management, offering quick access to radiological images, and making the diagnosing process easier. The analyzed application is a large distributed system, consisting of several subsystems in form of stand-alone and web-based applications. We have considered as our case study one of the subsystems from this application. The analyzed subsystem is a stand-alone Java application used by physicians in order to interpret radiological images. The application fetches clinical images from an image server (using DICOM protocol) or from the local file system (using DICOM files), displays them, and offers various tools to manage radiological images.

We have used for evaluation a set of 96 image series samples which were obtained from publicly available DICOM image files [Osi10, RiplsDis10, cir10, oPDsf10, hp10]. The images are real images from real patients, but anonymized for confidentiality reasons. For managing the DICOM image files, an open source implementation of the DICOM standard was used [sciom11].

The results are stable, a standard deviation of 0.040024407 on the classification accuracies was obtained. The low value of the standard deviation indicates a good precision of the proposed approach.

Considering the experimental results presented in Section 4.6, we can conclude that our approach provides optimized data structure selection and reduces the computational time by selecting the data structure implementation which provides a minimum overall complexity for the operations performed on a certain abstract data type on a given execution scenario.

4.7 Comparison to related work

In this section we aim at providing a brief comparison of our approach with the existing approaches for the problem of automatic selection of data representations.

4.8 Conclusions and future work

In this chapter we have presented our model for dynamically selecting the most suitable implementation of an abstract data type from a software application based on the system's execution context. For predicting, at runtime, the most appropriate data representation, a neural network and a support vector machine classification model were used. We have also illustrated the accuracy of both proposed approaches on case studies.

Considering the results presented in Section 4.3 and in Section 4.6, we can conclude that the approaches introduced in this paper for a dynamic selection of data representations have the following advantages:

- They are general, as they can be used for determining the appropriate *implementation* for any abstract data type, and with arbitrary number of data structures that can be chosen for implementing the ADT.

- They reduce the computational time by selecting the data structure implementation which provides a minimum overall complexity for the operations performed on a certain abstract data type on a given execution scenario. Consequently the efficiency of the software system during its evolution is increased.

- They are is scalable, as even if the considered software system is large, the abstract data types are locally optimized, considering only the current execution context. The size of the execution context does not depend on the size of the software system as shown in the thesis.

However, the main drawback of both approaches is that it is hard to supervise the learning process, as the supervision of an expert software developer is required for inspecting the collected execution contexts.

Further work will be focused on:

- Improving the proposed classification model by adding to it the capability to adapt itself using a feed-back received when inappropriate data representations are selected.
- Applying other machine learning techniques [KTL11, ZDYZ11] , self-organizing feature maps [SK99], or other modelling techniques [RKY10, Ngu10, TD10] for solving the problem of automatic selection of data representations during the execution of a software system.
- Studying the applicability of other learning techniques, like semi-supervised learning [ZL10] or reinforcement learning [SB98] in order to avoid as much as possible the supervision during the training process.
- Evaluating our approach on other case studies and real software systems.

Chapter 5

Conclusions

It has been that seen pattern recognition is central in data analysis tasks. As data is often characterised by a great deal of imprecision and uncertainty, intelligent, autonomous systems need to be developed that can handle such complex problems.

In Chapter 4 we have presented our model for dynamically selecting the most suitable implementation of an abstract data type from a software application based on the system's execution context. For predicting, at runtime, the most appropriate data representation, a neural network and a support vector machine classification model were used. We have also illustrated the accuracy of both proposed approaches on case studies.

Considering the results presented in Section 4.3 and in Section 4.6, we can conclude that the approaches introduced in this paper for a dynamic selection of data representations have the following advantages:

- They are general, as they can be used for determining the appropriate *implementation* for any abstract data type, and with arbitrary number of data structures that can be chosen for implementing the ADT.
- They reduce the computational time by selecting the data structure implementation which provides a minimum overall complexity for the operations performed on a certain abstract data type on a given execution scenario. Consequently the efficiency of the software system during its evolution is increased.
- They are is scalable, as even if the considered software system is large, the abstract data types are locally optimized, considering only the current execution context. The size of the execution context does not depend on the size of the software system as shown in the corresponding section from the thesis.

In Chapter 3 we have presented our contribution to agent-based clustering, particularly in two main directions: ASM-based batch clustering and incremental clustering. We have focused on developing clustering algorithms that allow the discovery and analysis of hybrid data. The algorithms presented in Section 3.2 are based on the adaptive ASM approach from [CXC04]. The major improvement is that, instead to moving the agents at a randomly selected site, we are letting the agents choose the best location. Agents can directly communicate with each other — similar to the approach from [CDG07]. In [SCCK04], the fuzzy IF-THEN rules are used for deciding if the agents are picking up or dropping an item. In our model we are using the fuzzy rules for deciding upon the direction and length of the movement. Moreover, in the approach from Section 3.2.2 the agents are able to adapt their movements if changes in the environment would occur. Case studies for these approaches have been performed in Section 3.2.3. In order to test the algorithm in a real-world scenario, the Iris and Wine datasets have been considered [Iri88, Win91]. Experiments outline the ability of our approaches to discover hybrid data. In Section 3.3 an incremental clustering algorithm is introduced. Incremental clustering is used to process sequential, continuous data flows or data streams and in situations in which cluster shapes change over time. Such algorithms are well fitted in real-time systems, wireless sensor networks or data streams because in such systems it is difficult to store the datasets in memory.

The algorithm considers one instance at a time and it basically tries to assign it to one of the existing clusters. Only cluster representations need to be kept in memory so computation is both fast and memory friendly. We have seen in the tests from the incremental approach (Section 3.3.3) that most of the apparently classification errors were actually items that have high membership degrees to more than one cluster. Nevertheless, in our opinion, it is again clear that we are dealing with hybrid data. Actually the hybrid nature of the data is suggested in [Iri88] and in [Win91] and this is the main reason for choosing these datasets for our analysis. By using fuzzy methods such features of the data are easy to be observed. The fact that there are hybrid items could be an indication of the quality of data.

In Chapter 2, we have presented our contribution to NP optimization problems, focusing on two well-known NP-hard problems: Travelling Salesman Problem (TSP) and Set Covering Problem (SCP). In Section 2.1 a short overview of NP completeness is made. In Section 2.2 the travelling salesman problem is approached using the stigmergic agent model. The Stigmergic Agent System (SAS) combines the strengths of Multi-agent Systems (MAS) and Ant Colony Systems (ACS). Stigmergy provides a general mechanism that relates individual and colony level behaviours: individual behaviour modifies the environment, which in turn modifies the behaviour of other individuals. The stigmergic agent mechanism employs several agents able to interoperate in order to solve problems by using both direct communication and indirect (stigmergic) communication. The algorithm was evaluated on several standard datasets outlining the potential of the method. In Section 2.3 the soft agent model is introduced. A soft agent is an intelligent agent that has to deal with imprecision, uncertainty, partial truth and approximation during its execution as a reactive agent or goal oriented agent or both. This new agent model is used in Section 2.4 where a new incremental clustering approach to the Set Covering Problem is presented. Experiments on standard datasets suggest that the approach is promising.

As future research directions, we intend to improve the proposed approaches, to extend the evaluation of the techniques that were proposed in this thesis and to investigate the use and to develop other computational models in pattern recognition.

Future work will be conducted in the following directions:

- investigating other metaheuristics with the aim of identifying additional potentially beneficial hybrid models
- using our models for solving other NP-optimization problems
- extending our methods in order to handle categorical data
- applying the incremental clustering approach in Intrusion Detection Systems
- improving the proposed classification model for DRSP by adding to it the capability to adapt itself using a feedback received when inappropriate data representations are selected
- applying other machine learning techniques like self-organizing feature maps or other modelling techniques for solving the problem of automatic selection of data representations during the execution of a software system
- studying the applicability of other learning techniques like semi-supervised learning or reinforcement learning in order to avoid as much as possible the supervision during the training process
- evaluating our techniques on other case studies and real software systems.

Bibliography

[AAA03] Noga Alon, Baruch Awerbuch, and Yossi Azar. The online set cover problem. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, STOC '03, pages 100–105, New York, NY, USA, 2003. ACM.

[ABKS99] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. In *SIGMOD Conference*, pages 49–60, 1999.

[AM10] Laurent Alfandari and Jérôme Monnot. Approximation of the clustered set covering problem. *Electronic Notes in Discrete Mathematics*, 36:479–485, 2010.

[AY01] Charu C. Aggarwal and Philip S. Yu. Outlier detection for high dimensional data. In *SIGMOD Conference*, pages 37–46, 2001.

[AZAY10] Moh'd Belal Al-Zoubi, Al-Dahoud Ali, and Abdelfatah A. Yahya. Fuzzy clustering-based approach for outlier detection. In *Proceedings of the 9th WSEAS international conference on Applications of computer engineering*, ACE'10, pages 192–197, Stevens Point, Wisconsin, USA, 2010. World Scientific and Engineering Academy and Society (WSEAS).

[BC96] J.E Beasley and P.C Chu. A genetic algorithm for the set covering problem. *European Journal of Operational Research*, 94(2):392 – 404, 1996.

[Bea] J E Beasley. OR-Library. http://people.brunel.ac.uk/∼mastjjb/jeb/orlib /scpinfo.html.

[BG08] Irad Ben-Gal. *Bayesian Networks*. John Wiley & Sons, Ltd, 2008.

[Bla94a] Betty Blair. Interview with zadeh, creator of fuzzy logic. *Azerbaijan International*, 2(4):46–47, Winter 1994.

[Bla94b] Betty Blair. Short biographical sketch. *Azerbaijan International*, 2(4):4, Winter 1994.

[BR03] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3):268–308, 2003.

[BW94] Aart J. C. Bik and Harry A. G. Wijshoff. On automatic data structure selection and code generation for sparse computations. In *Proceedings of the 6th International Workshop on Languages and Compilers for Parallel Computing*, pages 57–75, London, UK, 1994. Springer-Verlag.

[BW96] Aart J. C. Bik and Harry A. G. Wijshoff. Automatic data structure selection and transformation for sparse matrix computations. *IEEE Trans. Parallel Distrib. Syst.*, 7:109–126, February 1996.

[CCFM97] Moses Charikar, Chandra Chekuri, Tomás Feder, and Rajeev Motwani. Incremental clustering and dynamic information retrieval. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, STOC '97, pages 626–635, New York, NY, USA, 1997. ACM.

[CCGa] Gabriela Czibula, Istvan Czibula, and Radu Găceanu. Intelligent data structures selection using neural networks. *Knowledge and Information Systems*, pages 1–22. 10.1007/s10115-011-0468-3.

[CCGb] Gabriela Czibula, Istvan Czibula, and Radu Găceanu. A support vector machine model for intelligent selection of data representations. *Applied Soft Computing*. under review.

[CDG07] C. Chira, D. Dumitrescu, and R. D. Găceanu. Stigmergic agent systems for solving NP-hard problems. *Studia Informatica*, Special Issue KEPT-2007: Knowledge Engineering: Principles and Techniques (June 2007):177–184, June 2007.

[CH96] Tyng-Ruey Chuang and Wen L. Hwang. A probabilistic approach to the problem of automatic selection of data representations. *SIGPLAN Not.*, 31:190–200, June 1996.

[cir10] Patient contributed image repository. http://www.pcir.org/, 2010.

[CL11] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms (3. ed.)*. MIT Press, 2009.

[CM01] Maria Stella Fiorenzo Catalano and Federico Malucelli. Practical parallel computing. chapter Parallel randomized heuristics for the set covering problem, pages 113–132. Nova Science Publishers, Inc., Commack, NY, USA, 2001.

[CMP06] C. Chira C. M. Pintea, P. Pop. Reinforcing ant colony system for the generalized traveling salesman problem. In *Volume of Evolutionary Computing, International Conference Bio-Inspired Computing - Theory and Applications (BIC-TA)*, pages 245–252, New York, NY, USA, September 18–22, 2006. Wuhan, China.

[CMR+07] Oscar Castillo, Patricia Melin, Oscar Montiel Ross, Roberto Seplveda Cruz, Witold Pedrycz, and Janusz Kacprzyk. *Theoretical Advances and Applications of Fuzzy Logic and Soft Computing*. Springer Publishing Company, Incorporated, 1st edition, 2007.

[coNop] A compendium of NP optimization problems. http://www.nada.kth.se/ viggo/problemlist/compendium.html.

[CSM+11] Broderick Crawford, Ricardo Soto, Eric Monfroy, Fernando Paredes, and Wenceslao Palma. A hybrid ant algorithm for the set covering problem. *International Journal of the Physical Sciences*, 6(19):4667–4673, September 16 2011.

[CST00] Nello Cristianini and John Shawe-Taylor. *An introduction to support Vector Machines: and other kernel-based learning methods*. Cambridge University Press, New York, NY, USA, 2000.

[CXC04] L. Chen, X. H. Xu, and Y. X Chen. An adaptive ant colony clustering algorithm. In *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on, Vol. 3*, pages 1387–1392, 2004.

[DB05] Marco Dorigo and Christian Blum. Ant colony optimization theory: A survey. *Theor. Comput. Sci.*, 344(2-3):243–278, 2005.

[DDC99] Marco Dorigo and Gianni Di Caro. *The ant colony optimization meta-heuristic*, pages 11–32. McGraw-Hill Ltd., UK, Maidenhead, UK, England, 1999.

[DGF+91] J. L. Deneubourg, S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, and L. Chretien. The dynamic of collective sorting robot-like ants and ant-like robots. In *SAB90 - 1st Conf. On Simulation of Adaptive Behavior: From Animals to Animats*, pages 356–365. MIT Press, 1991.

[DICiM11] Digital Imaging and Communications in Medicine. http://medical.nema.org/, 2011.

[DL11] Steven Simske Dalong Li. Training set compression by incremental clustering. *Journal of Pattern Recognition Research*, 6:56–64, 2011.

[Dor07] M. Dorigo. Ant colony optimization. *Scholarpedia*, 2(3):1461, 2007.

[DP95] Dan Dumitrescu and Horia Florin Pop. Degenerate and non-degenerate convex decomposition of finite fuzzy partitions — I. *Fuzzy Sets and Systems*, 73:365–376, 1995.

[DP98] Dan Dumitrescu and Horia Florin Pop. Degenerate and non-degenerate convex decomposition of finite fuzzy partitions — II. *Fuzzy Sets and Systems*, 96:111–118, 1998.

[DS04] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, 2004.

[EKS+98] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Michael Wimmer, and Xiaowei Xu. Incremental clustering for mining in a data warehousing environment. In *Proceedings of the 24rd International Conference on Very Large Data Bases*, VLDB '98, pages 323–333, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[EpKSX96] Martin Ester, Hans peter Kriegel, Jrg S, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.

[Est09] Martin Ester. Density-based clustering. In *Encyclopedia of Database Systems*, pages 795–799. 2009.

[Etz96] O. Etzioni. Moving up the information food chain: deploying softbots on the world wide web. In *Proceedings of the 13rd national Conference on Artificial Intelligence (AAAI- 96)*, pages 4–8. Portland, OR, 1996.

[fIPA] Foundation for Intelligent Physical Agents. http://www.fipa.org/.

[FLM97] T. Finin, Y. Labrou, and J. Mayfield. *Kqml as an Agent Communication Language*. Software Agents, B.M. Jeffrey, MIT Press, 1997.

[FMPS00] Kilian Foth, Wolfgang Menzel, Horia Florin Pop, and Ingo Schröder. An experiment on incremental analysis using robust parsing techniques. In *The 18th International Conference on Computational Linguistics*, pages 1026–1030. Universität des Saarlandes, Saarbrücken, Germany, July-August 2000.

[For10] M Forina. http://archive.ics.uci.edu/ml, 2010.

[Găc11] Radu D. Găceanu. A bio-inspired fuzzy agent clustering algorithm for search engines. *Procedia Computer Science*, 7(0):305 – 307, 2011. Proceedings of the 2nd European Future Technologies Conference and Exhibition 2011 (FET 11).

[GC10] Serge Guillaume and Brigitte Charnomordic. Interpretable fuzzy inference systems for cooperation of expert knowledge and data in agricultural applications using fispro. In *FUZZ-IEEE*, pages 1–8, 2010.

[Gei93] S. Geisser. *Predictive inference: an introduction*. Monographs on statistics and applied probability. Chapman & Hall, 1993.

[GKP02] Mihaela Gordan, Constantine Kotropoulos, and Ioannis Pitas. A support vector machine-based dynamic network for visual speech recognition applications. *EURASIP J. Appl. Signal Process.*, 2002:1248–1259, January 2002.

[GKS09] Betsy George, James M. Kang, and Shashi Shekhar. Spatio-temporal sensor graphs (stsg): A data model for the discovery of spatio-temporal patterns. *Intell. Data Anal.*, 13(3):457–475, 2009.

[Glo89] Fred Glover. Tabu search - part I. *INFORMS Journal on Computing*, 1(3):190–206, 1989.

[Glo90] Fred Glover. Tabu search - part II. *INFORMS Journal on Computing*, 2(1):4–32, 1990.

[GO11] R. D. Găceanu and G. Orbán. Using rsl to describe the stock exchange domain. In *microCAD International Scientific Conference*. University of Miskolc, Hungary, 31 March – 1 April 2011.

[Gol89] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[GP08] Darwin Gouwanda and S. G. Ponnambalam. Evolutionary search techniques to solve set covering problems. In *World Academy of Science, Engineering and Technology*, pages 20–26. WASET, March 2008.

[GP10] R. D. Găceanu and H. F. Pop. An adaptive fuzzy agent clustering algorithm for search engines. In *MACS2010: Proceedings of the 8th Joint Conference on Mathematics and Computer Science*, pages 185–196. Komarno, Slovakia, 2010.

[GP11a] R. D. Găceanu and H. F. Pop. A context-aware ASM-based clustering algorithm. *Studia Universitatis Babes-Bolyai Series Informatica*, LVI(2):55–61, 2011.

[GP11b] R. D. Găceanu and H. F. Pop. A fuzzy clustering algorithm for dynamic environments. In *KEPT2011: Knowledge Engineering Principles and Techniques, Selected Papers, Eds: M. Frentiu, H.F. Pop, S. Motogna*, pages 119–130. Babes-Bolyai University, Cluj-Napoca, Romania, July 4–6 2011.

[GP11c] R. D. Găceanu and H. F. Pop. An incremental ASM-based fuzzy clustering algorithm. In *Informatics'2011, Slovakia, i'11:Proceedings of the Eleventh International Conference on Informatics, Informatics 2011, Eds: V. Novitzká, Štefan Hudák*, pages 198–204. Slovak Society for Applied Cybernetics and Informatics, Rožňava, Slovakia, November 16–18 2011.

[GP12] R. D. Găceanu and H. F. Pop. An incremental approach to the set covering problem. *Studia Universitatis Babes-Bolyai Series Informatica*, LVIII(2), 2012.

[Har75] J.A. Hartigan. *Clustering algorithms.* Wiley series in probability and mathematical statistics. Applied probability and statistics. Wiley, 1975.

[HCL00] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification, 2000.

[HFH+09] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *SIGKDD Explorations*, 11(1):10–18, 2009.

[HK06] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques, 2nd ed.* Morgan Kaufmann, 2006.

[HKT01] J. Han, M. Kamber, and A. K. H. Tung. *Spatial Clustering Methods in Data Mining: A Survey.* Taylor and Francis, 2001.

[HL0] Health Level 7. www.hl7.org/, 0.

[hp10] DICOM home page. ftp://medical.nema.org/medical/dicom/datasets/, 2010.

[HSP08] Samer Hassan, Mauricio Salgado, and Juan Pavón. Friends forever: Social relationships with a fuzzy agent-based model. In *HAIS*, pages 523–532, 2008.

[HZK+09] Pari Delir Haghighi, Arkady B. Zaslavsky, Shonali Krishnaswamy, Mohamed Medhat Gaber, and Seng Wai Loke. Context-aware adaptive data stream mining. *Intell. Data Anal.*, 13(3):423–434, 2009.

[Ins] MP-TESTDATA The TSPLIB Symmetric Traveling Salesman Problem Instances. http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html.

[Iri88] Machine Learning Repository Iris. http://archive.ics.uci.edu/ml/datasets/iris, 1988.

[Kam10] A. Kamble. Incremental clustering in data mining using genetic algorithm. *International Journal of Computer Theory and Engineering*, 2(3):1793–8201, 2010.

[KH01] Gregor Kiczales and Erik Hilsdale. Aspect-oriented programming. *SIGSOFT Softw. Eng. Notes*, 26:313–, September 2001.

[KHK99] George Karypis, Eui-Hong Han, and Vipin Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *IEEE Computer*, 32(8):68–75, 1999.

[KJV83] Scott Kirkpatrick, D. Gelatt Jr., and Mario P. Vecchi. Optimization by simmulated annealing. *Science*, 220(4598):671–680, 1983.

[KKvdSS96] Ben Krse, Ben Krose, Patrick van der Smagt, and Patrick Smagt. An introduction to neural networks, 1996.

[Kot07] Sotiris B. Kotsiantis. Supervised machine learning: A review of classification techniques. *Informatica (Slovenia)*, 31(3):249–268, 2007.

[KS05] Vikas Kumar and Marta Schuhmacher. Fuzzy uncertainty analysis in system modelling. In *European Symposium on Computer Aided Process Engineering 15 L. Puigjaner and A. Espua (Editors)*, 2005.

[KS08] P. R. Kumar K. Sreelakshmi. Performance evaluation of short term wind speed prediction techniques. *IJCSNS International Journal of Computer Science and Network Security*, 8(8):162–169, 2008.

[KTL11] Suzan Köknar-Tezel and Longin Jan Latecki. Improving svm classification on imbalanced time series data sets with ghost points. *Knowl. Inf. Syst.*, 28:1–23, July 2011.

[KY95] GEORGE J. Klir and BO Yuan. *FUZZY SETS AND FUZZY LOGIC Theory and Applications*. Prentice Hall, 1995.

[LB05] Kristopher R. Linstrom and A. John Boye. A neural network prediction model for a psychiatric application. In *Proceedings of the Sixth International Conference on Computational Intelligence and Multimedia Applications*, pages 36–40, Washington, DC, USA, 2005. IEEE Computer Society.

[LF94] E. Lumer and B. Faieta. Diversity and adaptation in populations of clustering ants. In *J.-A.Meyer, S.W.Wilson(Eds.), Proceedings of the Third International Conference on Simulation of Adaptive Behavior: From Animats, Vol.3*, pages 501–508. MIT Press/Bradford Books,Cambridge, MA, 1994.

[LKC02] Kyung-Soon Lee, Kyo Kageura, and Key-Sun Choi. Implicit ambiguity resolution using incremental clustering in korean-to-english cross-language information retrieval. In *Proceedings of the 19th international conference on Computational linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[LLLH10] Zhenhui Li, Jae-Gil Lee, Xiaolei Li, and Jiawei Han. Incremental clustering for trajectories. In Hiroyuki Kitagawa, Yoshiharu Ishikawa, Qing Li, and Chiemi Watanabe, editors, *Database Systems for Advanced Applications*, volume 5982 of *Lecture Notes in Computer Science*, pages 32–46. Springer Berlin / Heidelberg, 2010.

[Low78] James R. Low. Automatic data structure selection: an example and overview. *Commun. ACM*, 21:376–385, May 1978.

[LR76] James Low and Paul Rovner. Techniques for the automatic selection of data structures. In *Proceedings of the 3rd ACM SIGACT-SIGPLAN symposium on Principles on programming languages*, POPL '76, pages 58–67, New York, NY, USA, 1976. ACM.

[MA75] E. H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7(1):1–13, 1975.

[Mar09] Stephen Marsland. *Machine Learning: An Algorithmic Perspective*. Chapman & Hall/CRC, 1st edition, 2009.

[Mat] Fuzzy Logic Toolbox MathWorks. http://www.mathworks.com/help/toolbox /fuzzy/fp351dup8.html.

[Mit97] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[MK91] S. Miyake and F. Kanaya. A neural network approach to a bayesian statistical decision problem. *IEEE Trans. Neural Networks*, 2:538–540, 1991.

[Mou01] D.M. Mount. Lecture notes CMSC 420, Data Structures, 2001.

[NGS11] K Venkatramaiah Deepak P C Navneet Goyal, Poonam Goyal and Sanoop P S. An efficient density based incremental clustering algorithm in data warehousing environment. In *2009 International Conference on Computer Engineering and Applications IPCSIT vol.2 (2011), IACSIT Press, Singapore*, pages 482 – 486, 2011.

[Ngu10] Nam Nguyen. A new svm approach to multi-instance multi-label learning. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 384–392, Washington, DC, USA, 2010. IEEE Computer Society.

[NMM06] Giuseppe Narzisi, Venkatesh Mysore, and Bud Mishra. Multi-objective evolutionary optimization of agent-based models: An application to emergency response planning. In *Computational Intelligence*, pages 228–232, 2006.

[oPDsf10] Washington State University College of Pharmacy DICOM sample files. http://info.betaustur.org/, 2010.

[Osi10] Advanced Imaging in 3D/4D/5D Sample DICOM Image Sets Osirix. http://pubimage.hcuge.ch:8080/, 2010.

[OSM] ObjectWeb: Open Source Middleware. http://asm.objectweb.org/.

[PS82] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, 1982.

[PS96] Horia Florin Pop and Costel Sârbu. A new fuzzy regression algorithm. *Anal. Chem.*, 68:771–778, 1996.

[PSHD96] Horia Florin Pop, Costel Sârbu, Ossi Horowitz, and Dan Dumitrescu. A fuzzy classification of the chemical elements. *J. Chem. Inf. Comput. Sci.*, 36:465–482, 1996.

[RHW86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning internal representations by error propagation*, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.

[RiplsDis10] J.-P.: Signal Roux and image processing lab sample DICOM image sets. http://www.creatis.insa-lyon.fr/ jpr/public/gdcm/gdcmsampledata/, 2010.

[RKY10] Vikas C. Raykar, Balaji Krishnapuram, and Shipeng Yu. Designing efficient cascaded classifiers: tradeoff between accuracy and cost. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 853–860, New York, NY, USA, 2010. ACM.

[RN02] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2002.

[Roj96] Raúl Rojas. *Neural networks: a systematic introduction*. Springer-Verlag New York, Inc., New York, NY, USA, 1996.

[Rov78] P. Rovner. *Automatic representation selection for associative data*. Managing Requirements Knowledge, International Workshop, 1978.

[SB98] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[SC08] Ingo Steinwart and Andreas Christmann. *Support Vector Machines*. Springer Publishing Company, Incorporated, 1st edition, 2008.

[SCCK04] S. Schockaert, M. De Cock, C. Cornelis, and E. E. Kerre. Fuzzy ant based clustering. In *Ant Colony Optimization and Swarm Intelligence, 4th International Workshop (ANTS 2004), LNCS 3172*, pages 342–349, 2004.

[sciom11] Open source clinical image and object management. http://www.dcm4che.org/, 2011.

[SdLFdCG09] Eduardo J. Spinosa, André Carlos Ponce de Leon Ferreira de Carvalho, and João Gama. Novelty detection with application to data streams. *Intell. Data Anal.*, 13(3):405–422, 2009.

[Ser06] Gabriela Serban. *Sisteme Mutiagent in inteligenta artificiala distribuita. Arhitecturi si aplicatii*. Ed. Risoprint, Cluj-Napoca, 2006.

[sf] Apache software foundation. http://www.apache.org/.

[SGT+02] Ingo Schröder, K. Giannakoglou, D. Tsahalis, J. Periaux, K. Papailiou, T. Fogarty (eds, Horia F. Pop, A Fachbereich Informatik, Wolfgang Menzel, and Kilian A. Foth. Learning weights for a natural language grammar using genetic algorithms, 2002.

[SH07] Mark D. Skowronski and John G. Harris. Automatic speech recognition using a predictive echo state network classifier. *Neural Networks*, 20(3):414 – 423, 2007. Echo State Networks and Liquid State Machines.

[SK99] Panu Somervuo and Teuvo Kohonen. Self-organizing maps and learning vector quantization forfeature sequences. *Neural Process. Lett.*, 10:151–159, October 1999.

[SP00] Costel Sârbu and Horia Florin Pop. Fuzzy clustering analysis of the first 10 meic chemicals. *Chemosphere*, 40:513–520, 2000.

[SP04] Gabriela Serban and Horia Florin Pop. *Tehnici de Inteligenta Artificiala. Abordari bazate pe Agenti Inteligenti*. Ed. Mediamira, Cluj-Napoca, 2004.

[Spe87] C. Spearman. The proof and measurement of association between two things. By C. Spearman, 1904. *The American journal of psychology*, 100(3-4):441–471, 1987.

[SS01] Bernhard Scholkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[SSS79] Edmond Schonberg, Jacob T. Schwartz, and Micha Sharir. Automatic data structure selection in setl. In *Proceedings of the 6th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, POPL '79, pages 197–210, New York, NY, USA, 1979. ACM.

[SSS81] Edmond Schonberg, Jacob T. Schwartz, and Micha Sharir. An automatic technique for selection of data representations in setl programs. *ACM Trans. Program. Lang. Syst.*, 3:126–143, April 1981.

[Ste90] Luc Steels. Components of expertise. *AI Magazine*, 11(2):28–49, 1990.

[TD10] Balint Takacs and Yiannis Demiris. Spectral clustering in multi-agent systems. *Knowl. Inf. Syst.*, 25:607–622, December 2010.

[Vap00] V.N. Vapnik. *The nature of statistical learning theory*. Statistics for engineering and information science. Springer, 2000.

[VSP09] K. R. Venugopal, K. G. Srinivasa, and L. M. Patnaik. *Soft Computing for Data Mining Applications*. Springer Publishing Company, Incorporated, 1st edition, 2009.

[Wat89] C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, Cambridge, England, 1989.

[WB01] D.A. Watt and D.F. Brown. *Java collections: an introduction to abstract data types, data structures, and algorithms*. John Wiley, 2001.

[WD92] Christopher J. C. H. Watkins and Peter Dayan. Technical note q-learning. *Machine Learning*, 8:279–292, 1992.

[Win91] Machine Learning Repository Wine. http://archive.ics.uci.edu/ml/datasets/wine, 1991.

[WLZ00] G Wahba, Y Lin, and H Zhang. Generalized approximate cross validation for support vector machines, or, another way to look at margin-like quantities. *Advances in large margin classifiers*, (1006):297309, 2000.

[Woo99] Michael Wooldridge. *Intelligent Agents, An Introduction to Multiagent Systems*. Ed. G. Weiss, 1999.

[Woo09] Michael J. Wooldridge. *An Introduction to MultiAgent Systems (2. ed.)*. Wiley, 2009.

[WYM97] Wei Wang, Jiong Yang, and Richard R. Muntz. Sting: A statistical information grid approach to spatial data mining. In *VLDB*, pages 186–195, 1997.

[Yel03] D. M. Yellin. Competitive algorithms for the dynamic selection of component implementations. *IBM Syst. J.*, 42:85–97, January 2003.

[Zad65] Lotfi Askar Zadeh. Fuzzy sets. *Inf. Control*, 8:338–353, 1965.

[Zad94] Lotfi A. Zadeh. Fuzzy logic, neural networks, and soft computing. *Commun. ACM*, 37:77–84, March 1994.

[Zad97] Lotfi A. Zadeh. The roles of fuzzy logic and soft computing in the conception, design and deployment of intelligent systems. In *Software Agents and Soft Computing*, pages 183–190, 1997.

[Zad02] Lotfi A. Zadeh. Toward a perception-based theory of probabilistic reasoning with imprecise probabilities. *Journal of Statistical Planning and Inference*, 105(1):233 – 264, 2002. Imprecise Probability Models and their Applications.

[Zad08] Lotfi Askar Zadeh. Is there a need for fuzzy logic. *Information Sciences*, 178(13):2751–2779, July 2008.

[ZDYZ11] Xingquan Zhu, Wei Ding, Philip Yu, and Chengqi Zhang. One-class learning and concept summarization for data streams. *Knowledge and Information Systems*, 28:523–553, 2011. 10.1007/s10115-010-0331-y.

[Zha00] G. P. Zhang. Neural networks for classification: a survey. *IEEE Trans. Systems, Man and Cybernetics*, 30(4):451–462, November 2000.

[ZL10] Zhi-Hua Zhou and Ming Li. Semi-supervised learning by disagreement. *Knowl. Inf. Syst.*, 24:415–439, September 2010.

[ZRL97] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: A new data clustering algorithm and its applications. *Data Min. Knowl. Discov.*, 1(2):141–182, 1997.