Babeş-Bolyai University of Cluj-Napoca
Faculty of Mathematics and Informatics

Evolutionary Methods for Solving Optimization Problems

PhD Candidate:
Flavia Zamfirache

Supervisor:
Prof. Dr. Militon Frenţiu

Summary

2011

Abstract

Evolutionary methods are successfully used to solve static optimization problems, where the aim of the method is to find the global best solution. In the case of optimization problems in dynamic and uncertain environments the aim of the method is to follow the changes in the environment or to find solutions with a robust behavior. In this case some problems can appear because of the premature convergence of the evolutionary methods. Different mechanisms are proposed in order to solve the problems in uncertain and dynamic environments, methods that allow to maintain the population diversity during the run, memory archive usage and multi-populations.

Since many optimization problems in real world are dynamic, their properties vary/change over time. This thesis is focused on three real world problems: grid scheduling, data clustering and medical rules mining by analyzing their behavior in uncertain and dynamic environments. Uncertainty in optimization problem can arise from several sources: the evaluation of the fitness function may be subject to noise, the design variables may be subject to perturbations, the fitness function may be approximated or may change over time.

In this thesis we proposed and analyzed some algorithms' behavior in uncertain and dynamic environments. We proposed a simple perturbation of particles position in the case of a particle swarm optimization heuristic and the introduction of random elements in the case of a differential evolution heuristic and we analyzed the behavior of the two mechanisms on two dynamic optimization problems benchmarks. For the grid scheduling problem we proposed and studied the behavior of an evolutionary scheduler in the case of online scheduling. For the evolutionary scheduler and an ant scheduler, the robustness and the behavior in a dynamic environment are analyzed. In the case of clustering problem we studied the influence of noise and missing values for `AntClust` heuristic and proposed a density property

in order to improve the noisy data clustering. For rule mining problem we proposed a multi-objective evolutionary algorithm that incorporates human experts evaluation for medical data and we analyzed the influence of missing values in data and human interaction in the search process. Also a hybrid classifier based on non-nested generalized exemplars and an evolutionary selection of attributes and exemplars is proposed in order to offer an alternative to exhaustive search and to a switch strategy used for selection of a task scheduling algorithm for a grid scheduler.

Contents

Chapter 1

Introduction

1.1 Key Words

Evolutionary Algorithms, Particle Swarm Optimization, Ant Colony Optimization, Grid Scheduling, Medical Data Rule Mining, Clustering, AntClust, Scheduling Data Rule Mining

1.2 Motivation

Many optimization problems found in real world are subject to change. These changes could be the result of simulation, measurement or approximation errors, that implies noise or uncertainty in objective evaluation. In addition, the design variables or environmental conditions may also be perturbed or change over time. For example, in a task scheduling problem new tasks can appear or disappear from schedulers, machines can function slower or material quality change, traffic jams can appear in a routing problems, etc. So, we can say that these optimization problems should be solved tacking into account the fact that they correspond to uncertain and dynamic environments. In this cases, problem objective is not to find the global optima, but to follow the changes in dynamic environment or to find solution with robust behavior when the environment is uncertain.

Evolutionary methods have been used so far with success to solve optimization problem. They are an alternative to deterministic algorithms that find suboptimal solutions of the problem. Among their advantages we can count the robustness regarding problem size (a good alternative in the case of large problems), problem instance and random variables. The quality of evolutionary methods depends on the balance between search space exploitation

and exploration; the exploitation allows the improvement of the solution while exploration encourage the coverage of new region of the search space. Various search techniques belong to evolutionary methods: genetic algorithms, evolutionary strategies, genetic algorithms, particle swarm optimization, ant colony optimization, artificial immune systems, etc. Evolutionary methods proved to be suited for solving optimization problems in uncertain and dynamic environments, because there are different mechanisms that can be added in order to obtain good results, without having to restart the heuristic.

This thesis is focused on analyzing the ability of some evolutionary methods to solve different optimization problems in uncertain and dynamic environments. This is a challenging problem because many problems that can be found in practice are dynamic and various factors can influence the solution quality and the solution feasibility can depend of multiple criteria.

1.3 Thesis Structure

The thesis is structured as follows.

Chapter 2 makes a general presentation of the concepts that are discussed in this thesis: optimization problems, uncertain and dynamic environments. This chapter also contains a motivation of the fact that clustering and rule mining problems can be formulated as optimization problems.

Chapter 3 contains a general presentation of the heuristics used in this thesis: evolutionary algorithms, ant colony optimization and particle swarm optimization. Some mechanisms needed to adapt the nature inspired heuristics enumerated earlier into dynamic environments are presented. This chapter also contains an example of two nature inspired metaheuristics: differential evolution and particle swarm optimization applied into a dynamic environment for two optimization problems benchmarks.

Chapter 4 addresses the problem of grid scheduling into a static and dynamic environments. The chapter contains a review of evolutionary techniques applied to grid scheduling problem, the formalization of grid scheduling problem and describes the particularities of the two nature inspired heuristics: evolutionary algorithm and ant colony optimization needed in order to map the heuristics to the grid scheduling problem. The numerical results discussed in this chapter are regarding the robustness and behavior of the two heuristics in a static and dynamic environment. Various types of scheduling (batch, pseudo-batch and online) are

analyzed. Results of a comparative analysis between the evolutionary scheduler and other heuristics for online scheduling are proposed.

Chapter 5 contains an application of an ant clustering algorithm, `AntClust`, in the case of noisy and medical datasets. A review of ant clustering approaches can be found in Section 5.1. The presentation of `AntClust` algorithm (which is inspired by the chemical recognition of ants belonging to the same nest), parameters settings recommendation and variants is done. An experiment of applying `AntClust` algorithm on noisy data and an adaptation of the algorithm in order to improve its performance is discussed. Also an analysis of the behavior of the `AntClust` algorithm on incomplete medical data is done.

Chapter 6 presents a multiobjective interactive evolutionary approach for rules mining from medical data. A presentation of the rules mining problem, of accuracy and interestingness measures that can be used to evaluate the rules and some techniques that allow evaluation of the rules without a prepocessing step and a review of evolutionary approaches used in rule mining is done. Simulation results for finding association rules for some UCI Machine Learning Repository datasets and a set of obstetrical data collected during one year in a regional Obstetric-Gynecology hospital are done. Simulation results regarding the influence of different correction techniques for missing values in data has on rule miner are also analyzed. This chapter also presents a hybrid classifier based on non-nested generalized exemplars and an evolutionary selection of attributes and exemplars and analysis its classification accuracy compared with others (un)supervised classifiers in order to find a good scheduling algorithm.

Chapter 7 presents the conclusions and some directions for future research

1.4 Contributions

This thesis addresses the problem of applying evolutionary methods to optimization problems. It is focused on discussing and applying the following evolutionary methods: evolutionary algorithms, differential evolution, ant colony optimization and particle swarm optimization applications in uncertain environments. Since uncertainties in optimization problems can come from different sources we analyzed: the influence of noise in clustering problem, robustness in the case of scheduling problems, the behavior of some evolutionary schedulers in dynamic environments and the discovery of association rules in an interactive environment.

The main contributions of the thesis relative to the application of nature inspired algorithms to optimization problems in uncertain environments can be summarized as follows:

- in order to slow down the premature convergence of Differential Evolution heuristics in dynamic environments we proposed to use random elements (Chapter 3 page 15);
- in order to avoid the premature convergence of Particle Swarm Optimization heuristic in dynamic environments we proposed a perturbation that affects the particle position (Chapter 3 page 16);
- a Simple Population Scheduler is proposed. Its effectiveness for on online scheduling is investigated (Chapter 4 page 20) and the influence of perturbation strategies (Chapter 4 page 22) and structure of initial population (Chapter 4 page 20) on the solution quality is analyzed;
- for an Evolutionary Scheduler and an Ant Scheduler we done an analysis: of their behavior in static and dynamic environments (Chapter 4 page 22) and of heuristics robustness (Chapter 4 page 23);
- the ability of `AntClust` heuristic to cluster noisy data is analyzed and a density based parameter is proposed in order to improve the clustering results (Chapter 5 page 29);
- numerical results regarding `AntClust` heuristic to cluster incomplete medical data are discussed (Chapter 5 page 31);
- a multi-objective evolutionary rule mining algorithm for medical data that involves not only automatic evaluation, but also the user evaluation of the rules (Chapter 6 page 38) is proposed. The particularity of our approach consists in introducing a crowding distance between rules, that acts as a diversity criteria so that the Pareto set is being stimulated to select elements from less crowded regions in ordered to be added into an archive (Chapter 6 page 36);
- the influence of several methods of handling the missing values when searching for rules in data are analyzed (Chapter 6 page 40);
- a evolutionary selection mechanism for classification rules generated using NNGE was proposed and was applied in order to automatic select the scheduling strategy (Chapter 6 page 37).

Chapter 2

Optimization Problems in Dynamic and Uncertain Environments

The problem's objective in an dynamic environment is to follow the optima during the environment changes and to find robust solutions. The problem's objective in an uncertain environment is to find the best of all possible solutions that will also be the most suited when the environment is changing.

2.1 Optimization Problems in Uncertain Environments

There are different causes that can produce uncertainty into an environment:

- *Noise.* The fitness evaluation can be subject to noise.
- *Perturbations.*The design variables could be subject to perturbations or changes after the optimal solution has been determined.
- *Fitness Approximation.* Fitness function is very expensive to evaluate, or an analytical fitness function is not available, fitness functions are often approximated based on data generated from experiments or simulations.
- *Time Varying Fitness Function.* The fitness function is deterministic at any point in time, but is dependent on time.

2.2 Optimization Problems in Dynamic Environments

The identification of optima's that are changing is a problem for algorithms because of the premature convergence. So, the algorithms must find a balance between the exploitation and exploration of the search space. If the exploitation property is too much encouraged, then it is possible to freeze into a local optimum and transform the search into a local search. Otherwise if the exploration property is too much encouraged then it is possible that the search to become random. The particularities of a dynamic optimization problem are determined by: problem representation, the characteristics of the environment and the quality measures used to evaluate the performance of the algorithm.

2.2.1 Continuous Optimization Problems

Continuous optimization studies optimization problems that handle functions defined on continuous domains, that are restricted on the definition domain and are optimized (maximized or minimized). In this case, the optimum can follow a *continuous* or *discontinuous* trajectory in the search space.

2.2.2 Combinatorial optimization problems

The combinatorial optimization problems (like resource allocation, design, scheduling, etc) handle the efficient allocation of finite resources to achieve an objective when some of resources are restricted.

2.3 Clustering as an Optimization Problem

Clustering is a technique whose aim is to identify groups (clusters) of similar data without using any prior information on these groups but only by measuring some similarities (or dissimilarities) between data. There are a lot of clustering methods that differ one from the other by: (i) the information they required about the problem (e.g. number of clusters); (ii) the way of identifying and constructing the clusters; (iii) the way of treating the data (numerical, categorical or both types of data); (iv) the similarity measure which they use. With respect to the way the clusters algorithms interpret the clusters there are two main approaches: (i)clusters are distant compact sets (this implementation leads to a *global op-*

timization problem); (ii) clusters correspond to dense regions sets separated by low density regions sets (this implementation leads to a *local optimization algorithms*). Clustering algorithms can be also classified regarding the number of criteria they try to optimize in single and multi-objective optimization techniques.

2.3.1 Optimization Approaches in Clustering

Clustering problem can be seen like a:

- **Global Optimization.**
- **Local Optimization.**
- **Multi-objective Optimization.** In [ZZN+07] we analyzed different quality criteria for unsupervised clustering of documents. The comparative analysis is based on a differential evolution algorithm that allows the estimation both of the number of clusters and of their representatives. The results illustrate the particularities of different clustering criteria and the ability of the proposed approach to identify both the number of clusters and their representatives, suggesting that the best behavior is obtained by combining the connectedness and separability measures.

2.3.2 Clustering on Noisy Data

Noise in data can come from random errors (unknown encoding, out of range variables, inconsistent entries) or variance in measured variables. In order to remove noise several methods can be used: consistency checks, domain knowledge, statistical methods, etc. In the case of clustering techniques a method based on the measure of the data density can be used. With respect to this criterion, clusters could be seen as dense regions of data while the noise corresponds to regions with low density of data. Algorithms based on density are: DBSCAN, DENCLUE and UNC (Unsupervised Niche Clustering). We proposed in [ZZ05b] a density mechanism in order to improve the `AntClust` algorithm performance to deal with noisy data (see Chapter 5).

2.4 Rule Mining as an Optimization Problem

Data mining is an automated or semi-automated process for extracting previously unknown and potentially useful knowledge and patterns from large databases. One of the data mining problem that can be formulated like an optimization problem is association rule discovery that seeks to discover associations among items encoded within a dataset. An association rule has the form $(A \Rightarrow C)$ where A (antecedent) and C (consequent) are disjoint conjunctions of item subset pairs. Since rules are combinations of attributes, the association rules problem may be seen as a combinatorial optimization problem. In this case its goal is to optimize different quality measures of the rules (e. g. support, confidence) thus the association rule mining can be considered as multi-objective optimization problem. A detailed multi-objective evolutionary algorithm that discovers association rules in medical data in presented in Chapter 6.

2.5 Conclusions

In this chapter we introduced the context on which this thesis is developed: optimization problems, uncertain environments, clustering and rule mining problems. We started with a presentation of the properties that can induce uncertainty into environment: the evaluation of the fitness function that might be subject to noise, the design variables that might be subject to perturbations, fitness function that might be approximated, fitness functions may change over time. After that we detailed the dynamic environments properties, measures, types of problems and benchmarks used for testing. Then, we presented the clustering and rules process from an optimization perspective.

Chapter 3

Nature Inspired Metaheuristics Used in Dynamic Optimization

Optimization problems in uncertain and dynamic environments are complex and difficult, and often classical algorithmic approaches based on mathematical and dynamic programming are able to solve only very small problem instances. For this reason, in recent years, metaheuristics such as Ant Colony Optimization, Evolutionary Computation, Simulated Annealing, Tabu Search and others, emerged as successful alternatives to classical approaches.

3.1 Nature Inspired Metaheuristics

In this thesis we manly used three metaheuristics inspired from nature: evolutionary algorithms, ant colony optimization and particle swarm optimization. We have chosen these metaheuristics because of their ability to adapt to changing environmental conditions, fact that makes them suitable for numerical experiments described in the next chapters.

3.1.1 Evolutionary Algorithms

Evolutionary algorithms (EA) are heuristic search techniques based on the idea of natural evolution (selection, survival of the best). The process of collective learning of the population, auto-adaptation and robustness are some of the advantages of evolutionary algorithms over other global optimization techniques. Among the most known evolutionary algorithms are: *genetic algorithms* (Holland 1975), *evolutionary programming* (Fogel 1966), *evolutionary strategies*(Rechenberg 1973, Schwefel 1981), *genetic programming* (Koza 1999). Applications

of evolutionary algorithms can be found in problems like: data clustering, scheduling, design, simulation and identification, control, etc.

3.1.2 Ant Colony Optimization

Ant Colony Optimization (ACO) is a metaheuristic inspired by the behavior of real ants colonies which enables ants to find shortest paths between food sources and their nest. This behavior is the basis for a cooperative interaction that leads to the emergence of the shortest paths. ACO was proposed by Dorigo and has been widely used to resolve combinatorial optimization problems (as traveling sales-man, sequential ordering problem, network routing problem, graph coloring, etc) and rapidly became a popular optimization technique.

3.1.3 Particle Swarm Optimization

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Eberhart and Kennedy in 1995, inspired by the social behavior of bird flocking or fish schooling. PSO is a population-based search procedure where the individuals, referred as particles, are grouped into a swarm. Each particle in the swarm represents a candidate solution to the optimization problem, that flies through the problem space by following the current optimum particles. PSO is applicable to different domains like: training of neural networks, extraction of rules from fuzzy networks, image recognition, optimization of electric power distribution networks, structural optimization (optimal shape and sizing design, topology optimization), process biochemistry, system identification in biomechanics, etc.

3.2 Adaptation Mechanisms for Dynamic Environments

Dynamic optimization problems are problems of which constraints, objectives or representations change over time during the optimization process. In stationary optimization, the only goal of optimization algorithms is to find the global optimum as fast as possible, while in dynamic optimization when the problem changes, the goal of an algorithm turns from finding the global optimum to detecting the changes and tracking the movement of optimum

over time. Nature inspired heuristics encounter problems regarding premature convergence in dynamic environments that does not allow them to discover the new optima, because the population loses the diversity in the iterative process. The simplest solution to resolve the premature convergence problem would be to restart the algorithm when a change in the environment is noticed, but this approach has the disadvantage that all the information gained by the population are lost when the search is restarted. So, different techniques were proposed to resolve this problem.

3.2.1 Evolutionary Algorithms

There are different techniques that preserve the population property of reacting at environment changes without restarting the search process like [JB05]: maintaining population diversity, memory mechanism and multi-populations.

There are two main approaches in enhancing the population diversity necessary for an optima tracking process: a reactive one and a proactive one. The first approach consists in reacting to a change by triggering a diversity increasing mechanism (e.g. hypermutation). The proactive approach consists in maintaining the population diversity throughout the entire run of the algorithm. This means avoiding the convergence by decreasing the selection pressure (through sharing or crowding) or by directly stimulating the diversity (through random immigrants in each generation).

In the case of memory based approaches, the EA contains a memory that allows to reuse information from past generations, which is especially useful when optima repeatedly returns to previous locations. Memory based approaches can be divided into explicit and implicit memories. In the case of explicit memory, a specific strategy for storing and retrieving information is defined (e.g. reinsert individuals from the memory that have been successfully in similar environments). In the case of implicit memory, EA keeps a redundant representation of the population (e.g diploidy).

Another diversity preserving mechanism is based on multiple populations. If each population maintain information about a different promising area of the search space the search process can easily follow the optimum. This approach can be of reactive type if a population is reinitialized when a change is detected or of proactive type when populations are always enforced to search different areas.

3.2.2 Ant Colony Optimization

In the case of dynamic problem, different strategies were proposed in order to improve ACO algorithm performance: population ACO, pheromone matrix update strategy, new ant properties (sensitivity).

3.2.3 Particle Swarm Optimization

Many strategies are proposed for application of PSO in dynamic environments because PSO outdated memory and diversity loss that can happen in the case of dynamic changes. The outdated memory problem happens because the information stored in each element may no longer be valid and may misguide the search. In the case of outdated memory, most of the approaches assume that the change time in environment is known or can be detected and the elements are reevaluated. In order to discover the environment changes, some randomly chosen *sentry* particle are used to re-evaluated the environment at the beginning of each iteration. Beside random particles the use of the global best particle and the second-best particle as *sentry* particles is proposed. For avoiding population convergence, different strategies were proposed: induce diversity after a change, maintain diversity during the search and multi-populations.

3.3 Numerical Experiments

The existence of so many different mechanisms for maintaining diversity in populations, raises the problem of choosing the right mechanism for a given problem. Besides its effectiveness, a mechanism should be as simple as possible in order not to increase the computational cost of the algorithm. In the same time, a given mechanism could have different impact on different population-based optimization methods. So, in [ZZ06] we analyzed some simple diversity enhancing mechanisms applied to two population-based optimization algorithms: differential evolution (DE) [SP95] heuristic and particle swarm optimization (PSO) [KE95] heuristic.

3.3.1 Differential Evolution

Differential evolution (DE) introduced in [SP95] is a population-based optimization method based on a recombination operator which uses randomly selected parents and a simple selection procedure based on a competition between a parent and its offspring.

3.3.2 Adaptation Mechanisms to a Dynamic Environment

To avoid premature convergence of the two analyzed algorithms, DE and PSO, in the dynamic environment we analyzed the impact of the diversity enhancing schemes. In the case of DE, we analyzed a parameter adaptation scheme, random elements and a multi-population approach; for PSO, a simple perturbation scheme and a multi-population approach were tested.

Adaptation Mechanisms for DE

In [ZZ06], as diversity measure we used the variance. In order to keep the diversity to almost the same level during the evolution process, at each generation, t, the ratio $c_j = \gamma \cdot Var(x^i)/Var(z^i)$ is computed. Another mechanism we tested in [ZZ06] is a multi-population approach that is characterized by the fact that in each population an adaptive DE algorithm is applied and periodically a random migration process is started: any element of each subpopulation is, with a given migration probability, interchanged with a randomly selected element from an arbitrary sub-population. Despite their benefits for avoiding premature convergence in the case of static problems, these mechanisms are not powerful enough for tracking changing optima.

The problem which we analyzed in [ZZ06] is to find the simplest mechanism which is still powerful enough to allow for tracking a changing optimum. In the case of continuous changes of the optimum position (without large jumps) it should be enough to perturb the current best element in order to arrive in the region of the new position of the optimum. Thus, using a cloud of elements wandering around the currently best element could ensure the ability of the algorithm to track the changing optimum.

In the DE with random elements the population is divided into two parts: a set of elements which evolves according to DE rules and another set $\{x_1, \ldots, x_\mu\}$ of elements which

just wander around the best element, x_* , of the previous generation:

$$y_i^j = x_*^j + K_j \cdot N(0,1),\ i = \overline{\mu+1, m},\ j = \overline{1, n} \tag{3.1}$$

where $N(0,1)$ is a random value with a standard normal distribution and $K_j > 0$ is a parameter controlling the amplitude of the perturbation.

Adaptation Mechanisms for PSO

The PSO algorithm also has premature convergence problems when the velocity, v_i becomes too small and the particles are frozen. In [ZZ06], we analyzed two ideas in order to resolve the premature convergence problem: increase the particle speed and perturbing particle position. In [LA05], a variant of perturbing the particle velocity is proposed: if the velocity values smaller that a threshold value v_c, are replaced with a random value $U(-1,1)V_{max}/\rho$. The variant proposed in [LA05] is called Turbulent Particle Swarm Optimization (TPSO). We proposed the use of a simpler approach based on perturbing not the velocities but directly the position of frozen particles:

$$x_i^j(t+1) = \begin{cases} x_i^j(t) + v_i^j(t+1), & \text{if } v_i^j(t+1) \geqslant v_c \\ x_i^j(t) + K_j \cdot N(0,1), & \text{if } v_i^j(t+1) < v_c \end{cases} \tag{3.2}$$

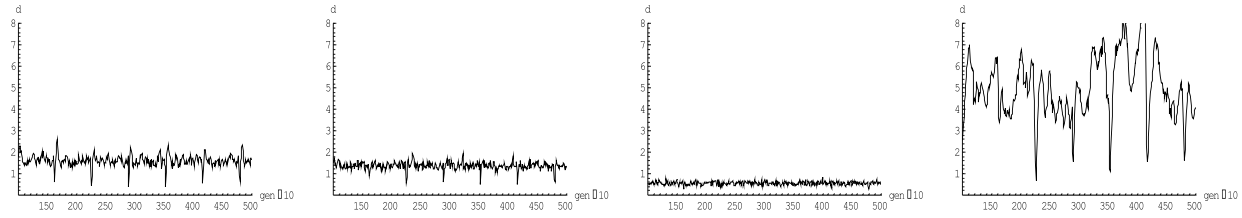
where N(0,1) is a random value with a standard normal distribution and $K_j > 0$ is a parameter controlling the perturbation. Since any change in the position has an influence also on velocity, the particle can escape from the local optimum.

3.3.3 Numerical Results

To analyze the impact of the diversity enhancing schemes for dynamic optimization problems we used as test functions a dynamic variant of Ackley's function and a test function from the Moving Peaks Benchmark [Bra].

The evolution of the averaged distance between the best element in the population and the current optimum is illustrated in Figure 3.1 for some variants of the DE algorithm. For smaller values of γ (e.g. $\gamma = 1$) the adaptive DE is not able to track the optimum (Figure 3.1 (d)). On the other hand, using a percent between 25% and 50% of random elements the behavior of the algorithm is significantly improved. It should be remarked that in order

to track the optimum it is necessary to reevaluate at each generation the elements of the current population.



(a) (b) (c) (d)

Figure 3.1: Behaviour of DE for dynamic Ackley function. (a) DE with fixed parameters ($p = 0.5$, $F = 0.5$); (b) Adaptive DE with $\gamma = 1.25$; (c) DE with fixed parameters ($p = 0.5$, $F = 0.5$) and 50% random elements; (d) Adaptive DE with $\gamma = 1$

For DE algorithms, we analyzed the impact of parameters adaptation and of random elements. As is illustrated in Figure 3.2 the adaptive DE has an acceptable behavior but not as good as if a small percent of random elements is used.

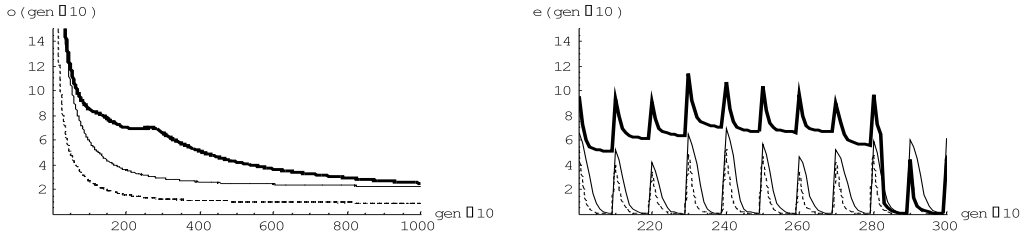


Figure 3.2: Behaviour of DE for one moving peak: offline error (left), current error (right). DE variants: adaptive DE ($\gamma = 1$) + 50% random elements (thick line), adaptive DE ($\gamma = 1$) + 25% random elements (dashed line), adaptive DE (($\gamma = 1.25$)) without random elements (thin line).

Since the convergence of PSO algorithms is slower than the convergence of DE algorithms it would be expected that classical PSO algorithms are able to track slowly moving optima. Figure 3.3 suggests that for simple dynamic problems there is no need for supplementary diversity enhancing schemes (as position/velocity perturbation or random elements).

3.4 Conclusions

Both the Turbulent PSO introduced in [LA05] and the simple perturbed PSO that we proposed in [ZZ06] proved to be effective in avoiding premature convergence. Concerning the

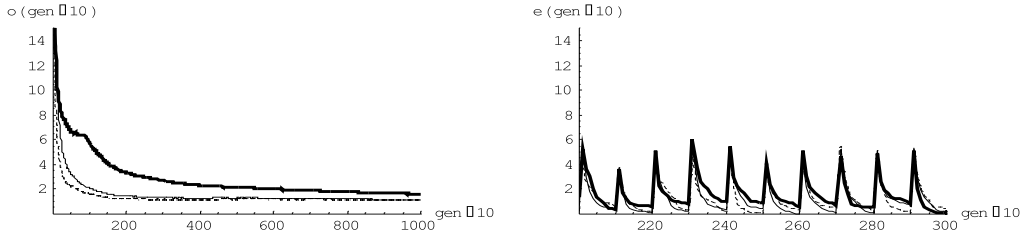


Figure 3.3: Behaviour of PSO for one moving peak. offline error (left), current error (right). PSO variants: classical (dashed line), perturbed PSO (thin line), PSO + 25% random elements (thick line).

multi-population approach, the numerical experiments suggest that it slows down the convergence of PSO but it is not able to avoid premature convergence by itself.

In the case of relatively slowly changing optima good results are obtained using simple diversity enhancing mechanisms (e.g. parameter adaptation which favor large values of the variance or random elements in the case of DE algorithms) or even no supplementary mechanisms (as in the case of PSO algorithms). However, when the severity of change is higher, supplementary schemes (e.g. exclusion [MM05], charged PSO [BB02]) should be used.

Chapter 4

Task Scheduling in Dynamic Distributed Environments

The task scheduling problem has attracted a lot of attention lately and as a consequence, currently there are a lot of scheduling algorithms addressing different variants of the problem.

4.1 Grid Scheduling Problem

The grid scheduling problem is a NP-hard combinatorial optimization problem. Different variants of the problem with respect to the tasks properties, to the computing environment characteristics and to the scheduling process particularities can be found.

Formalization. The task scheduling problem aims is to find an allocation of resources to tasks such that the tasks requirements (hardware, software), their precedence constraints are satisfied and some quality of service criteria are optimized. The assignment of tasks is based on estimations of the execution times of the tasks on various resources. Let us consider a set of n tasks, $\{t_1, \ldots, t_n\}$, to be scheduled on a set of $m < n$ processors, $\{p_1, \ldots, p_m\}$. Let us suppose that for each pair (t_i, p_j) we know an estimation $ET(i, j)$ of the time needed to execute the task t_i on the processor p_j. A schedule is an assignment of tasks to resources, $S = (p_{j_1}, \ldots, p_{j_n})$, where $j_i \in \{1, \ldots, m\}$ and p_{j_i} denotes the processor to which the task t_i is assigned. If $\mathcal{T}_j$ denotes the set of tasks assigned to processor p_j and T_j^0 denotes the time moment since the processor j is free then the completion time corresponding to this processor will be $CT_j = T_j^0 + \sum_{i \in \mathcal{T}_j} ET(i, j)$. The makespan is the maximal completion time over all processors, i.e. $makespan = \max_{j=\overline{1,m}} CT_j$. The problem to be solved is that of finding the schedule with the minimal makespan value.

4.2 Approaches in Static Environment

Since near-optimal solutions for task scheduling are acceptable in practice, we tested two nature inspired heuristics for task scheduling: an evolutionary algorithm (simple population-based scheduler) and an ant colony optimization scheduler. The test results we done are for both static and dynamic environments.

4.2.1 Simple Population-based Scheduler

In [ZFZ11] we proposed a scheduler named *SPS - SimplePopulationScheduler*. The key operator of the algorithm is the perturbation (mutation) operator, which is the module that defines the schedules improvement strategy. Another element that influence the schedulers performance is the initialization step.

Population initialization. The construction of a (sub)-optimal schedule is usually based on creating an initial schedule which is then iteratively improved. When constructing an initial schedule there are two decisions to take: (i) the order in which the tasks are assigned to processors; (ii) the criterion used to select the processor corresponding to each task. Depending on these elements, there are several strategies [BSB+01] as: random, OLB, MET, MCT, Max-Min and Min-Min. Each of these strategies generates initial schedules with a specific potential of being improved. Therefore, it would be beneficial to use not just one strategy, but to use a population of initial schedules constructed through different strategies. The use of some seed schedules in the initial population has also been emphasized by many authors [BSB+01, PKN10, Xha07].

Perturbation. The initial schedules created by the scheduling heuristics are usually non-optimal and thus they can be improved by moving or swapping tasks between resources, that is equivalent with a mutation strategy. Depending on the criteria used to select the source and destination resources, and the tasks to be relocated, there can be designed a lot of strategies to perturb a schedule [XA10]. Most perturbation operators involved in the scheduling heuristics used in task scheduling are based on two typical operations: "move" one task from a resource to another one and "swap" two tasks between their resources. The strategies presented in Table 4.1 were selected based on their simplicity, efficiency and randomness/greediness balance. The "random move" corresponds to the "local move" operator [Xha07] and is similar to the mutation operator used in evolutionary algorithms. The "greedy move" operator is related to the "steepest local move" in [Xha07] but with a higher greediness

Source		Destination		Strategy
Processor	Task	Processor	Task	
Random	Random	Random	-	*Random Move*
Most loaded (max CT)	Random	Best improvement	-	*Greedy Move*
Most loaded (max CT)	Random	Least Loaded (min CT)	Random	*Greedy Swap*

Table 4.1: Characteristics of the strategies used to perturb the schedules

since it always involves the most loaded processor. The "greedy swap" is similar to "steepest local swap" in [Xha07] but it is less greedy and less expensive since it does not involve a search over the set of tasks. If we apply at one step just one of the perturbations (random move, greedy move or greedy swap) we will name it *SimplePerturbation* and if we apply combination of the three types of the perturbations we will name it *HybridPerturbation*.

Since one perturbation step does not necessarily lead to an improvement in the quality of a schedule, we consider an iterated application of the perturbation step until either n iterations were executed (each task has the chance to be moved) or until a maximal number, g_p, of unsuccessful perturbations is reached.

4.2.2 Ant Colony Optimization

This section presents the particularities of the Ant Scheduler, inspired by the algorithm introduced in [RL04].

Pheromone initialization. In order to help the construction of good schedules we used the idea of incorporating information corresponding to a schedule generated by the Min-Min heuristic. This has been done by reinforcing the elements in the pheromone matrix which correspond to the Min-Min schedule. The idea to seed some information obtained from greedy heuristics is a common approach in evolutionary scheduling [CX06] and it also improves the behavior of the ant colony algorithm.

Local Search Step. Following the conclusions of previous studies [RL04], that local search can significantly improve the behavior of an ant-based scheduler, we applied an improvement step to the best schedule found at each epoch. This improvement step is based on a rebalancing operator. More specifically, in the rebalancing step the same actions take place

like in the case of mutation for the population based scheduler (*HybridPerturbation*). This operation is repeated for a given number of times or until no improvement can be observed.

4.2.3 Numerical Results

In the case of Simple Population-based Scheduler (SPS) we analyzed the influence of perturbation strategies [ZFZ11]. We also analyzed the ability of the simple population based scheduler and ant colony optimization scheduler to construct schedules in the static case for three types of environments: consistent (C), semi-consistent (S) and inconsistent (I) [ZZC10a, ZZC10b].

Test Environment. The test data we used are from the benchmark introduced in [BSB+01], which provides matrices containing values of the expected time for computation (ET) generated based on different assumptions related to task heterogeneity, resource heterogeneity and consistency.

Analysis of the perturbation strategies. The aim of the numerical study for static environments in [ZFZ11] was to analyze the influence of the perturbation strategies on the performance of a Simple Population-based Scheduler (SPS). We also analyzed some initialization strategies: (i) random initialization; (ii) use of some scheduling heuristics and randomly initialization for the other elements; (iii) use random perturbations of some scheduling heuristics; (iv) use of Min-Min heuristic and random perturbations of it. As expected, the best results were obtained when the initial population contains seeds obtained by using scheduling heuristics, while the worst behaviour corresponds to purely random initialization.

We compared four perturbation variants (random, move-based, swap-based and the hybrid one) with a state of the art memetic algorithm hybridized with Tabu Search (MA+TS) [Xha07]. Even if based on simpler operators, the algorithm proposed in this work provides schedules close in quality to those generated by MA+TS, in the case of inconsistent test cases ("u_i_**" problems), the proposed scheduler using the hybrid perturbation operator provides better results.

Analysis of SPS and AS for Static Environments. In [ZZC10a, ZZC10b] we analyzed the behavior of the Simple Population-based Scheduler (SPS) and the Ant Scheduler (AS) in a static environment. In SPS algorithm case, we used for tests a simple perturbation strategy - GreadyMove. In our analysis we used the data characterized by highly heterogeneous tasks and processors, with different levels of consistency. We used the first files from classes "u_c_hihi" (consistent computing environment), "u_i_hihi" (inconsistent computing

environment), "u_s_hihi" (semi-consistent computing environment). The results obtained by GA [CX06] are slightly better than those obtained by our SPS and AS in the case of consistent and semi-consistent problems but, since the standard deviation values for GA are not provided it is hard to evaluate the statistical significance of this difference. On the other hand, both SPS and AS behave better in the inconsistent case.

4.3 Heuristics Robustness Analysis

In [ZZC10b] we analyzed the robustness of the two previously presented scheduling algorithms. A good schedule should satisfy three properties: (i) it should be obtained in a small amount of time; (ii) it should be easily adapted to a slightly different context; (iii) it should be robust, meaning that it is affected as little as possible by run time changes [CJSZ08]. The obtained results suggest that for consistent and semi-consistent environments, the Min-Min heuristic leads to the most robust schedules while, in the case of inconsistent environments the SPS and ACO schedulers lead to more robust solutions.

4.4 Online Scheduling

In the case of online scheduling, the tasks arrive sequentially and they are scheduled as they arrive, that differs from the previous approach, where task were scheduled in batches. In [ZFZ11] several dynamic scheduling heuristics with aging have been tested against their corresponding population based versions (the specific heuristic was used as perturbation operator in SPS). Their behaviour has also been compared with the SPS algorithm using the non-iterated hybrid perturbation (at each generation, the perturbation is applied once). Among the dynamic algorithms we tested a flavour of DMECT as described in [Fr9], the MinQL heuristics [FMC09] and the classic Min-Min and Max-Min with aging.

Test Environment. For online scheduling, we considered a simulation model where tasks execution times (ET) follow a Pareto distribution. The arrival rate is simulated using an 8 order polynomial fitted to some real world traces [Fei10]. A total number of 500 tasks were generated for every test. Rescheduling was done every 250 time units given a minimal ET of 1000 units. All tests were repeated 20 times.

Numerical Results. Several dynamic scheduling heuristics with ageing (DMECT, MinQL) have been tested against their corresponding population based versions which were

constructed by using the specific scheduling heuristics as perturbation operators in SPS (pDMECT, pMinQL). Their behaviour has also been compared with the SPS algorithm based on a non-iterated hybrid perturbation (at each perturbation step the hybrid perturbation is applied only once).

Table 4.2 presents the main benefits of population based scheduling heuristics (pDMECT and pMinQL) when used in online scheduling. Both pDMECT and pMinQL obtained significantly better results than their non-populational variants, with pDMECT having a behavior similar to SPS (the best values in Table 4.2 were validated using a t-test with 0.05 as level of significance) . The only notable difference was that of speed. pDMECT required almost 30 seconds to build a schedule while the simple population-based schedules needed only three seconds on average.

	DMECT	pDMECT	MinQL	pMinQL	SPS	Max-Min	Min-Min
MS	66556.20	**49409.11**	76564.40	54332.89	**46996.76**	61165.15	68774.87
	±15097.85	±9522.13	±18114.51	±9891.15	±8812.87	±11936.19	±15101.05
Time	66.56	28343.04	3.06	2254.64	2777.70	684.49	669.21
(ms)	± 15.50	±10702.15	±2.52	± 314.45	± 578.22	±242.15	±209.99

Table 4.2: Best average makespan (MS) obtained by online scheduling heuristics and their population based variants

4.5 Adaptation to Dynamic Environment

In this section we present the analysis done in [ZZC10a, ZZC10b] concerning the behavior of several mechanisms of exploiting information from previous scheduling stages applied to two nature inspired schedulers and we discuss some experimental results.

4.5.1 Simple Population-based Scheduler

In the case of a dynamic environment, the scheduling process consists of consecutive scheduling events. At each scheduling event, there is a list of available processors which is partially different from the list corresponding to the previous step. If the difference between the list of available processors is not too large, then one can exploit the schedules constructed at the previous scheduling event.

In order to use information from the evolutionary process corresponding to previous scheduling events in [ZZC10a, ZZC10b] we analyzed the behaviour of two proposed variants: use of good schedules obtained in the previous scheduling event and use of an archive of good schedules obtained in previous scheduling events.

4.5.2 Ant Colony Optimization

The pheromone matrix ensures the communication between ants, as it is shared by all ants at each epoch of a scheduling event. Thus, it seems natural to use it also for communication between the scheduling events. This means that when a schedule is constructed for a new list of available processors, the pheromone matrix is not reinitialized. The values computed at the previous scheduling event are used instead. The main particularity of this approach is that the pheromone values corresponding to unavailable processors are just kept unchanged during the construction of a new schedule.

4.5.3 Numerical Results in a Dynamic Environment

The experimental analysis is conducted on the benchmark data provided in [BSB+01] and the dynamic character of the computing environment is simulated by randomly marking some resources as unavailable.

Test Environment. In order to simulate the dynamic character of the environment, we generated at each scheduling event a new list of available processors by just randomly removing a given percent of processors from the initial list of 16 resources. The percent of unavailable processors we used in our experiments was 10%, 20% and 40%.

Numerical Results. In the case of the simulated dynamic environment we analyzed the behavior of the dynamic SPS described and of the dynamic ant scheduler. For the evolutionary scheduler we used the simple perturbation strategy, *Greedy Move*.

The aim of these experiments was to analyze the impact of the memory mechanisms (use of previous schedules or previous values of the pheromones) on the ability of the evolutionary and ant schedulers to adapt to dynamic environments. Table 4.3 contains the ratio of scheduling events when the dynamic variants outperformed the static variants and also the ratio of events when the static ones outperformed the dynamic ones. In all other cases the static and dynamic variants behaved similarly. The main remark is that the behavior of the two schedulers is different. In the SPS case as long as the difference between the lists

of available machines corresponding to consecutive events is in average at most 12% then using the archive of previous schedules is beneficial for all environment types (consistent, semi-consistent, inconsistent). However the benefit of using a memory mechanism is larger in the case of consistent environments and smaller in the case of inconsistent ones. In the case of the ant based scheduler the memory mechanism based on the preservation of the pheromone matrix does not lead to improvements over the static AS variant in the case of consistent environments. On the other hand in the case of an inconsistent environment the dynamic AS variant does not only have a better behavior than the static AS but it also leads to schedules better than those obtained by SPS.

Problem	Unavailable machines (%)	SPS		AS	
		Dynamic	Static	Dynamic	Static
C	10	42/50	0/50	0/50	49/50
C	20	5/50	3/50	2/50	39/50
C	40	1/50	3/50	0/50	30/50
S	10	35/50	1/50	0/50	49/50
S	20	5/50	2/50	5/50	43/50
S	40	0/50	3/50	12/50	35/50
I	10	31/50	0/50	49/50	0/50
I	20	4/50	1/50	39/50	1/50
I	40	0/50	1/50	20/50	2/50

Table 4.3: Ratio of scheduling events when there are statistically significant differences between the dynamic and static variants of the SPS and AS.

4.6 Conclusion

In this chapter we presented two scheduling algorithms: an evolutionary scheduler (Simple Population-based Scheduler) and an ant colony optimization approach regarding scheduling problem. We also analyzed their behavior in a static environment [**Z**ZC10a, **Z**ZC10b, **Z**FZ11], dynamic environment [**Z**ZC10a, **Z**ZC10b] and for online scheduling [**Z**FZ11].

Chapter 5

Nature Inspired Clustering

Beside the classical hierarchical and partitional clustering techniques, the techniques inspired from nature, like evolutionary clustering or swarm clustering techniques are successfully applied to datasets clustering.

5.1 Ants and Clustering

The behavior of ant colonies inspired the development of various clustering techniques. Different approaches are based on different aspects of real ants behavior: (i) cemetery organization and larval sorting; (ii) chemical recognition of nestmates [LMV02]; (iii) self-assembling behaviour of ants; (iv) Pachycondyla apicalis ants strategy for prey searching.

5.2 `AntClust` Algorithm. Variants

The `AntClust` algorithm proposed in [LMV02], simulates the so-called "colonial closure" phenomenon in ants colonies. This phenomenon is based on some chemical odors the ants posses, which allow them to recognize the difference between nestmates and intruders. Each ant has its own view on the colony odor, which is continuously updated. Starting from these ideas, Labroche et al. proposed a model of an artificial ant able to participate to the clustering of a set of data.

Each ant, i, has the following characteristics: (1) An associated data, x_i. This is the unique element which is not modified during the clustering process; (2) A label, L_i which is a natural number that identifies the cluster; (3) A similarity threshold, T_i , which is used

to establish if two ants are sufficiently similar to be nestmates; (4) The age A_i, which is in fact a counter which counts the number of meetings to which the ant has participated and it is used for computing some mean values; (5) An adaptive parameter, M_i which measures the ant perception of its nest size; (6) An adaptive parameter, M_i^+ which measures the ant perception of the acceptance degree by the other members of its nest. The better the ant is integrated in its nest, the larger M_i^+ is.

The clustering process consists of three main phases: threshold learning phase, random meetings phase and clusters refining phase. In the meeting phase five rules are applied based on ants properties: the creation of a new nest, the inclusion of an ant into an existing nest, the positive meeting between two nestmates, the negative meeting between two nestmates and meeting between ants belonging to different nests.

5.3 Dealing with Noise in Ant Clustering

In [ZZ05b] we analyzed the behavior of `AntClust` for two synthetic noisy datasets. The first set consists of 6 ellipsoidal clusters, generated by using a bi-dimensional normal distribution, superposed with an uniform noise (see Figure 5.1(a)). The second set consists of 2050 points grouped in 4 clusters having different geometric shapes (the points have been uniformly generated in the interior of these geometric shapes) and 750 points uniformly distributed in the exterior of the geometric shapes, representing the noise (see Figure 5.1(d))

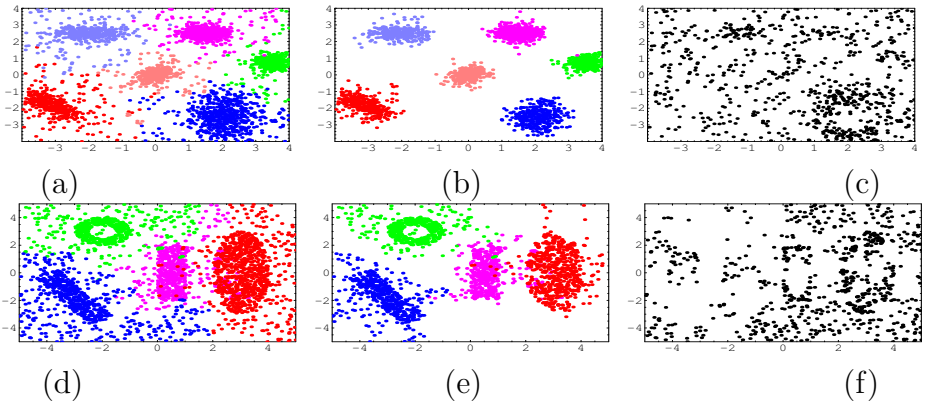


Figure 5.1: (a),(d) Results obtained by applying the original `AntClust`; (b),(e) Clusters identified by ignoring the points for which the acceptance parameter (M^+) is lower than the quartile of values of M^+ for all data; (c),(f) The ignored points (estimation of the noise)

.

`AntClust` (applied with $k_M = m^2/10$ for the first data set and $k_M = m^2/2$ for the second data set) identified 6 and 4 clusters respectively as can be seen in Figure 5.1(d). Since M^+ is a measure of the acceptance degree of an ant by its nestmates it seems natural to interpret it also as a level of its significance for the cluster. This means that data having low values for M^+ could be considered as candidates to be noise. The critical issue here is how to choose a threshold on M^+ values. Since all M^+ values are in $[0, 1)$ an absolute value could be used (e.g. 0.1). On the other hand a relative value based on some order statistics could be also applied. The results obtained by using as threshold the quartile value of M^+s are illustrated in Figures 5.1 (b), (c) and 5.1 (e), (f). In the case of ellipsoidal clusters, the result is acceptable but in the case of geometrical clusters, the noise is not very well identified. This suggested us to use also a measure related to the data density.

5.3.1 Density Information in `AntClust`

In order to introduce density information in `AntClust` we proposed to attach to each ant, i, a new parameter, D_i. This parameter contains an estimation of the ant's perception on the density of the region were it is placed. This parameter is set to zero at the beginning and at each meeting between the ant i and a different ant j it is adjusted by $D_i := D_i + \Delta_i$ where

$$\Delta_i = \begin{cases} exp\left(\frac{(1-S(i,j))^2}{2\sigma_i^2}\right) & \text{if } 1 - S(i,j) \leqslant \sigma_i \\ 0 & \text{if } 1 - S(i,j) > \sigma_i \end{cases} \tag{5.1}$$

where σ_i are parameters controlling the influence area of each data. Usually $\sigma_i = \sigma$ for all i. This density is similar to the one used in DENCLUE but instead of computing it by a systematic search of the neighborhood as in DENCLUE it is estimated based on the random meetings of ants.

After the meetings phase, the parameter D_i is divided by the ant's age, A_i, the density estimation being always in $[0, 1)$. The first question concerning D_i is if it really offers different information than M_i^+ does.

A first way of using the values of D_i is in the clusters refining phase: the elements without clusters are assigned to a cluster only if their density value is larger than the density's cuartile. Moreover, only the elements already belonging to clusters that have a density value larger than the density's cuartile are taken into account when we are searching for the most similar element. In this way, a new class appears: that of unclassified data which can be considered noise.

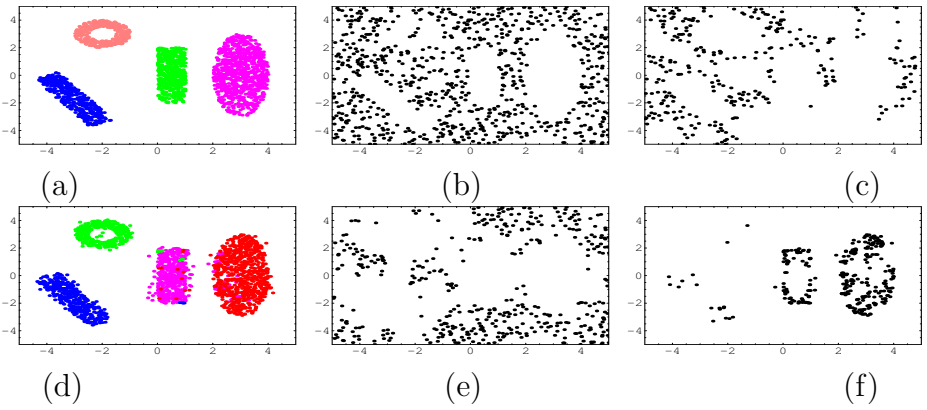


Figure 5.2: Geometrical clusters (a) the original clusters; (b) the noise superposed on the clusters; (c) data in the third category; (d) identified clusters (fourth category); (e) identified noise (first category); (f) data in the second category.

The parameter σ plays an important role in identifying the clusters. We analyzed the influence of this parameter on the behavior of `AntClust` for the data set consisting of four geometric clusters (Figure 5.1 (d)) and a noisy data uniformly distributed on the exterior of the geometric shapes. The algorithm should identify 5 classes, four corresponding to useful data and one to noisy data. In order to evaluate the clustering quality we computed an error measure introduced in [LMV02].

On the other hand, the different roles which D_i and M_i^+ play suggest to use both of them in order to control the separation of useful data from noise. This means to split the data in four categories based on the values of their M^+ and D parameters. The separation is based on some threshold values T_M (threshold for the acceptance degree) and T_D (threshold for the density). The four categories are: (1)*First category.* This contains all data i for which $M_i^+ \leqslant T_M$ and $D_i \leqslant T_D$ and corresponds to data having a high probability to be noisy (see Figure 5.2(e)); (2)*Second category.* This contains all data i for which $M_i^+ \leqslant T_M$ and $D_i > T_D$ and corresponds to data having a high estimation of the density but a low acceptance degree. These data couldn't be classified even if they belong to rather dense regions. Usually they are points at the border of clusters (see Figure 5.2(f)) and are somewhat similar to border points identified in DBSCAN; (3)*Third category.* This contains all data i for which $M_i^+ > T_M$ and $D_i \leqslant T_D$ and corresponds to data having a high acceptance degree in their cluster but a low estimation of the density. Usually these data belong to rather sparse regions but they participated to many meetings (see Figure 5.2(c)). The existence of this category is explained by the different thresholds used in the estimation of M_i^+ (T_i) and in the computation of D_i (σ); (4)*Fourth category.* This contains all data i for which $M_i^+ > T_M$ and $D_i > T_D$ and

corresponds to data which belong to dense regions and have been accepted by their clusters (see Figure 5.2(d)).

The results presented in Figure 5.2 have been obtained by using as thresholds, T_D and T_M, the quartiles of parameters D and M^+ respectively, computed over the entire set of data.

5.4 AntClust Clustering for Medical Data

In [ZZ05a] we analyzed the effectiveness of the `AntClust` clustering algorithm proposed by Labroche et al. [LMV02] applied to medical data characterized by the presence of both numerical and categorical components but also by missing values. Ant-based clustering has been successfully applied to documents categorization, web usage mining but, the applicability to medical data with missing components and subjective categorical values has not been analyzed. In [LMV02], `AntClust` is applied to a set of medical data containing numerical results from 5 laboratory tests in order to identify three groups corresponding to: euthyroidism, hypothyroidism and hyperthyroidism, respectively. However, all the data components are numerical and there are no missing values.

5.4.1 Similarity/dissimilarity measures appropriate for medical data

A key element in a clustering process is the choice of an adequate similarity/dissimilarity measure. When choosing such a measure we must take into account the nature of data to be processed because a measure adequate to numerical data could not be appropriate for categorical data. On the other hand, medical data are frequently characterized by missing attribute values. There are different strategies to approach this problem [JD88]: incomplete data removal, partial distance strategy and nearest prototype strategy.

Medical data frequently contains categorical attributes, expressing levels (degrees) of the presence of a feature (e.g. a symptom). The classical coding of this type of data is: 0-if the symptom is not present and 1,2,3 ... for increasing levels of the presence of the symptom. Sometimes a sequence of consecutive values is not necessarily the most appropriate because the difference between 0 (no symptom) and the first level (lowest level of the symptom presence) should be larger than the difference between two other levels. This is because the

levels of the symptom presence carry some subjective information which should be somehow reflected by the values used in coding them.

We illustrated this situation for a set of data available at UCI Machine Learning Repository - Dermatology database. We studied the influence of the of the threshold learning phase for different calculation formulas and the adequacy of a given coding or of a dissimilarity measure.

5.5 Conclusions

In this chapter we analyzed the ability of a nature inspired clustering `AntClust` to deal with noisy and medical data. `AntClust` algorithm is inspired by ants behavior of generating new nests, accepting ants in nests and reorganizing nests. Section 5.2 contains the description of the algorithm and some parameters setting.

In Section 5.3, the ability of an ant-based clustering algorithm (`AntClust`) to separate noise from data was analyzed. An analysis on the usefulness of both an existing parameter attached to ants (M^+) and that of a new parameter related to the density (D) have been done. The computation of the density parameter and the postprocessing step of separating the data in different categories do not modify significantly the complexity of the algorithm. Moreover, since the parameter σ can be computed based on the similarity thresholds, it is not necessary to tune a new parameter. The preliminary results are encouraging but a lot of things concerning the information carried by the parameters attached to ants still remains unrevealed. The values of thresholds, T_M and T_D used in postprocessing the data in order to identify the noise are mainly determined based on experiments and not on analytical reasons. Some theoretical results concerning the estimations of parameters M, M^+ and D computed during the meetings phase would be highly desirable.

In Section 5.4.1 an analysis of the ability of `AntClust` algorithm to identify natural clusters arising in some medical data is presented. The influence of the numerical coding of categorical data representing degrees of a feature presence is also analyzed and a comparison with the classical K-means algorithm for some medical data is presented. In this case, we observed that as long as the similarity measure and the parameters are well chosen, `AntClust` can be successfully applied in medical data clustering. On the other hand, when level-type attributes are involved, an adequate choice of their coding could improve the clustering result.

Chapter 6

Evolutionary Rules Mining

Data mining is the core step of KKD, that includes the application of several processing methods to facilitate the data mining algorithm to improve and refine the discovered knowledge.

6.1 Mining Rules

A rule can be presented in the following form: *IF "some conditions on the values of predicting attributes are true" THEN "some conditions on the goal attributes are true"*. If there is only one goal attribute and it specifies a class, then we discuss about a *classification rule*, expressing the possibility that data satisfying the antecedent (IF) condition belong to the class specified in the consequent (THEN) part. When the goal attributes do not express a class, then we deal with *prediction rules*, expressing hypotheses on the dependence between the antecedent and consequent parts of the rules. Finally, when the potential sets of antecedent and consequent attributes are not previously established, we investigate general *association rules* expressing co-occurrence of different attribute values. Discovering and selecting rules in data is a search process usually guided by measures quantifying the *accuracy*, *comprehensibility* and *interestingness* of the rules. These measures are usually conflicting, i.e. an accurate rule is not necessarily interesting or easy to read, thus the searching process has to be multicriterial. On the other hand, the data encountered in practice (e. g. medical data studied in [LZZ08]) usually suffer from a significant number of missing values (MVs), that implies different correction techniques that have to be applied to data in order to be evaluated.

6.1.1 Evaluation Measures

The quality of classification, prediction or association rules can be quantified using different statistical measures, each of them capturing specific characteristics of the rules. The rules extracted from data should explain the data, be comprehensible, and contain novel and interesting knowledge, thus the measures of quality are divided in three main classes:

- accuracy measures - quantifying how well the rules explain the data (*Supp*), confidence (*Conf*), accuracy (*Acc*), specificity (*Spec*) and sensitivity (*Sens*);
- comprehensibility measures - quantifying how easily the rules can be interpreted;
- interestingness measures - quantifying the potential to provide new, previously unknown knowledge (Phi-coefficient (Φ), odds ratio (OR) and cosine measure (cos)).

The rules have the following general structure: $IF\ (AT_1, AT_2, \ldots, AT_k)\ THEN\ (CT_1, CT_2, \ldots, CT_l)$, where AT_i denotes an antecedent term while CT_j denotes a consequent term. Each term involves one data attribute and it is a triplet $\langle a, op, value \rangle$ where a is an attribute, op is an operator (equal, different, in, not in, less than, greater than) and $value$ is a possible value or a set of values for the attribute.

6.1.2 Handling Missing Values in Data

When we evaluate a rule with respect to a set of incomplete data we have to decide how the missing values will be handled. The most radical approach to deal with missing values (MVs) is to simply eliminate the observations with any missing variable values (i.e. the incomplete cases) from the analysis. The variants we analyzed in [LZZ08] are based on the idea of dealing with missing values during the rule mining process. The step when we have to take into account the existence of missing values is while checking whether or not a record with MVs matches a given rule.

In the case of incomplete data, when a rule R is evaluated, the missing values can interfere with the evaluation process only when they correspond to attributes involved in the rule. Therefore we have to decide on how the computation should be modified in such a situation. We analyzed two methods.

Method 1. In this variant we try to limit the penalization of rules involving incomplete attributes.

Method 2. In the second variant, we did not ignore the incomplete records but we penalized them by using a non-crisp matching value. Instead of the 0 match value for a missing value, each missing attribute value will be assigned a probability to satisfy the corresponding term of the rule.

This is a variant of estimating the probability that a missing value would satisfy the rule term, based on the simplified assumption that the values of the attributes are uniformly distributed on their range. By using other distribution models for the attributes values, different matching values would be obtained. The main difference between the first variant and the current one is the fact that in the first case $cover(R, S) \in \{1, \ldots, cardS\}$ while in the second one $cover(R, S) \in [1, cardS]$.

6.2 An Evolutionary Rule Mining

In [ZLZ08] we proposed to involve the user in the search process, as a predefined aggregation of quality criteria is difficult to find and, moreover, it has been suggested that users can also change their opinion on the rules' quality during the evaluation process itself [OAT+99]. The approach we proposed is based on a multi-objective evolutionary algorithm (MOEA).

We applied the mining technique on medical data sets from the UCI repository and for a set of obstetrical data collected during one year in a hospital of Obstetrics-Gynecology. Since the aim of our study was to design tools which can help medical specialists in making decisions, we have to take into account the feedback provided by the specialist and to allow him or her to intervene in the rules discovery process.

Encoding.

Each element (chromosome) of the population corresponds to a rule and it consists of a list of components (genes) corresponding to all attributes in the data set. Each component consists of three fields: $\langle$ *presence flag*, *operator*, *value* $\rangle$. An element is a fixed-length list with mixed values (binary, integer, real and interval). The difference between the antecedent and consequent attributes is made only in the evaluation of an element.

Evolutionary Operators. During each generation, a new population is constructed from the current one using some evolutionary operators.

By *crossover*, a new rule is constructed starting with two randomly selected rules from the current population.

The *mutation* has the role of modifying the rules obtained by crossover. For each attribute, mutation is applied with a given probability (e.g. $p_m = 1/n$) and it can affect one of the fields (i.e. presence flag, operator or value) and only one at each mutation step.

Selection and Archive. After a new population is created by crossover and mutation, a selection step (typical to MOEAs) is applied. Our selection strategy is similar to that used in NSGA-II [DK07], meaning that the elements in the joined population (parents and offsprings) are ranked based on the non-domination relationships. For stimulating the diversity of the resulting Pareto front, a crowding distance is used as a second selection criterion: from two elements having the same rank, the one with a larger crowding distance (suggesting that it belongs to a less crowded region) is selected. The crowding distance can be defined in either the objective or the decision variables space. A particular characteristic of our approach is related to the crowding distance between rules.

We analyzed two types of distances, one expressing the structural differences between rules and the other expressing the difference between the data subsets covered by the rules.

After a given number of generations, an archive of nondominated elements is constructed. Not all non-dominated elements from the current population are transferred in the archive, but they are filtered such that both the structural and the cover-based distances between any two elements of the archive are larger than a given threshold (in our analysis we used 0.01).

6.3 User Interaction with the Mining Process

An interactive search allows the user to interfere with the evolutionary process in order to guide it towards interesting regions of the search space. The idea of permitting the user to interfere with the evolutionary process has already been explored in the context of multiobjective evolutionary optimization [DK07] by asking the user to provide a so-called "reference point" in the objectives space.

In the interactive variant, the search process consists of several stages; at each one, the population is evolved for a given number of generations and the archive of the selected non-dominated rules is provided to the user together with all the objective measures computed for the testing dataset (measures not necessarily limited to those used as criteria in the optimization process). In our implementation, we used the following set of measures: support, confidence, accuracy, specificity, sensitivity, comprehensibility, odds ratio, lift, uncovered

negative and relative risk [OAT+99]. Based on these criteria and on a subjective evaluation, the user can decide whether there are uninteresting or incomprehensible rules. Then (s)he can mark these rules and proceed to the next stage of the search. The effect of marking the undesirable rules is twofold: firstly, the population elements corresponding to the marked rules are replaced with randomly initialized elements. That allows the algorithm to explore new parts of the search space, similar with random immigrants technique; secondly, the marked rules are added to a list (L_p) of prohibited rules, creating an archive that stores the parts of the search space that are considered to be uninteresting by the user.

A possible drawback of using a supplementary optimization criterion is that it usually leads to a larger number of nondominated elements. In order to avoid this, the list of uninteresting rules can be used to introduce some constraints, i.e. all rules similar to those in the list are considered to be unfeasible. The rules' feasibility is employed when the domination relationship between two rules is checked: if one rule is feasible and the other one is unfeasible, the first rule dominates the second one, disregarding the values of the quality criteria; if both rules are unfeasible or both rules are feasible, the domination is decided based on the values of the optimization criteria.

6.4 Evolutionary Pruning of Mined Rules

The NNGE (Non-Nested Generalized Exemplars) algorithm is a hybrid instance based learning method which infers data classification rules represented as non-nested and non-overlapping axes-parallel hyperrectangles [Mar95]. After the set $\mathcal{H}$ of hyperrectangles has been generated by the NNGE algorithm it is postprocessed in order to reduce its size and, hopefully, to improve the classification accuracy [ZPNZ11].

6.5 Numerical Experiments

Numerical tests are done on two data sets types: medical data and task scheduling data. In the case of the medical data sets, a multiobjective evolutionary algorithm is analyzed. For task scheduling data sets, we analyzed the behaviour of a post-processing evolutionary selection strategy on a set of rules generated using non-nested generalized exemplars techniques.

6.5.1 Medical Data Sets

The numerical experiments contain results tested for the medical data sets from the UCI repository and for a set of obstetrical data collected during one year in a hospital of Obstetrics-Gynecology. In [ZLZ08] the aim of the experiments was to validate the ability of the evolutionary approach to discover accurate rules, and to analyze the impact of the user's intervention in the searching process. On the other hand in [LZZ08] we analyzed the influence of several methods to handle the missing values when searching for classification rules in medical data.

Test Data

ObGyn dataset represents the data collected during one year (2006) in an Obstetrics-Gynecology hospital. The set of data contains 2686 records corresponding to two classes: the class of pre-term births (370 records, representing 13.77%) and the class of on-term births (2316 records, representing 86.23%). Each record contains 63 attributes corresponding to different characteristics of mothers and new-born children. The overall percentage of missing values is 23% but they are non-uniformly distributed over attributes.

Association Rule Discovery

In order to validate the ability of the implemented multi-objective evolutionary algorithm to extract reliable rules, we firstly tested it in the case of classification problems. The results were obtained for *Pima Indians Diabetes* and *Wisconsin Breast Cancer* (1991) datasets, based on two optimization criteria: accuracy (*Acc*) and uncovered negative (*UN*). The MOEA outcome is comparable to those obtained by applying other rule-based classifiers implemented in WEKA data mining tool): simple rules classifiers (ZeroR, OneR), conjunctive rules classifier (CR), decision table majority classifier (DT), propositional rule learner based on repeated incremental pruning (JRIP), nearest neighbour-like classifier with non-nested generalized exemplars (NNge), partial decision trees (PART).

Impact of User Interaction

Two variants of including the user evaluation in the evolutionary process were implemented and tested: user-criterion and user-constraint. *User-criterion*: For each element, a supple-

Var/ Stage	No. rules	Marked rules	Conf. range	Sens. range	Spec. range	Acc. range	Lift range
I/1	16	–	[0.42, 1]	[0.25, 0.98]	[0.31, 1]	[0.54, 0.93]	[1.24, 2.9]
I/2	6	–	[0.23, 1]	[0.25, 0.55]	[0.88, 1]	[0.81, 0.93]	[2.88, 12.4]
I/3	4	–	[0.97, 1]	[0.28, 0.53]	[0.99, 1]	[0.94, 0.96]	[12.1, 12.4]
II/1	16	10	[0.42, 1]	[0.25, 0.98]	[0.31, 1]	[0.54, 0.93]	[1.24, 2.9]
II/2	36	16	[0.75, 1]	[0.07, 0.98]	[0.97, 1]	[0.92, 0.98]	[10.06, 12.4]
II/3	32	19	[0.21, 1]	[0.21, 0.99]	[0.88, 1]	[0.84, 0.98]	[2.6, 12.4]
III/1	16	10	[0.42, 1]	[0.25, 0.98]	[0.31, 1]	[0.54, 0.93]	[1.24, 2.9]
III/2	15	6	[0.23, 1]	[0.28, 0.98]	[0.83, 1]	[0.81, 0.99]	[2.8, 12.4]
III/3	11	2	[0.75, 1]	[0.27, 0.98]	[0.97, 1]	[0.96, 0.99]	[9.3, 12.4]

Table 6.1: Breast data set: ranges of the quality measures for rules evolved in three MOEA scenarios. Variant I: no user intervention. Variants II (*user-criterion*) and III (*user-constraint*): at each stage, the user marks the rules having the value of confidence, sensitivity, specificity, or accuracy less than 0.75.

mentary optimization criterion is the product between the minimal structural and cover-based distances to all the elements in the list of marked rules. *User-constraint*: An element which is structurally or semantically similar to at least one marked rule is considered unfeasible. The feasibility property is employed when the domination relationship between two rules is checked: one unfeasible rule is always dominated by a feasible one and cannot dominate a feasible one; if both rules are either feasible or unfeasible, they are compared according to the optimization criteria.

The impact of these variants on the number of nondominated rules evolved after three stages is illustrated in Table 6.1, for the *Wisconsin Breast Cancer* dataset (each stage consisted of 100 generations, the final quality measures lying within large ranges). The user intervention consisted in marking all rules with the quality values smaller than a threshold (e.g. confidence, sensitivity, specificity, or accuracy smaller than 0.75). The results in Table 6.1 prove that the user intervention led to rules of higher sensitivity and accuracy. As expected, the Variant II (*user-criterion*) led to a larger number of evolved rules, compared to Variant III (*user-constraint*).

We also used the interactive variants of MOEAs to explore the rules' space when analyzing a set of ordinary obstetrical data (*ObGyn* data set), with the final aim of identifying potential risk patterns for preterm births (i.e. birth before 37 weeks of gestation). The risk patterns

could be expressed either as classification rules (IF "some antecedent conditions are satisfied" THEN class=preterm) or as prediction rules (IF "some antecedent conditions are satisfied" THEN "the gestational age is less than a value").

The strategy we proposed to allow the user to influence the process of rules' discovery is only a first step in developing an interactive system aimed at supporting medical doctors in exploring the data and extracting new, possibly unexpected knowledge. The results we obtained with the obstetrical data were not as relevant as we might have expected due to multiple factors: on the one hand, the particularities of the data (many missing and erroneous values); on the other hand, the limitations of the evolutionary strategy itself. Using numerical values for the continuous attributes in order to avoid a preliminary discretization was appealing, but this led to a very large searching space and to the discovery of rules which were not easy to interpret. Using fuzzy variable instead of crisp ones could improve the quality of the final rules, especially in the case of medical data.

Influence of Missing Values Handling on Classification Rules

Since the UCI data used for test does not contain missing vales we prepared the data for tests by randomly eliminating $mv\%$ values of attributes ($mv \in \{10, 20, 30\}$ and it represented a percent of the number of total number of attributes in the dataset). For cross-validation, the data were randomly split in five folds. The missing values were introduced only in the data used for training, the validation sets being with the original values of attributes.

For handling missing values from data we analyzed three methods: *Method 1* and *Method 2* are described in section 6.1.2 and *Method 3* is based on a simple imputation strategy (the data are pre-processed such that the missing values are replaced with the median of all existing values for the corresponding attribute). First method is the variant that tries to limit the rule penalization and the second one is the variant with non-crisp matching values.

The experiments illustrated that by using different methods for handling the missing values different sets of rules are obtained. On the other hand, the differences in the quality of the obtained rulesets are not statistically significant. Slightly better results were obtained for methods which adjust the computation of the quality measures in order to deal with missing values (Method 1 and Method 2).

We further conducted a similar analysis on the set of obstetrical data set (obGyn). Our aim was to evolve rules corresponding to the pre-term births class. The large number of attributes leads to a very large search space, creating difficulties for the evolutionary algo-

rithm to discover high quality rules. Therefore we selected various subsets of attributes to be involved in the rules. The results presented in Table 6.2 were obtained when using as possible antecedents in the classification rules the information about previous pregnancies, miscarriages and abortions and about the fundal height. This last attribute has 16% of missing values. We used the product between the specificity and sensitivity as unique optimization criterion. The optimization being with a single criterion each run led to only one rule. All obtained rules contain a term corresponding to "fundus uterus" height attribute (e.g. "fundus uterus" height$\leqslant$ 29) even if this attribute has missing values while the other ones do not contain missing values. This can be explained by the fact that the fundal height attribute has a predictive value for the pre-term birth. On the other hand, as Table 6.2 suggests, the results obtained by the three analyzed methods are not significantly different. However the first method leads to slightly better results than the other two.

	acc	spec	sens	UN	lift
Method 1	0.70±0.09	0.73±0.13	0.52±0.17	0.58±0.12	1.92±0.63
Method 2	0.64±0.08	0.64±0.12	0.59±0.18	0.55±0.1	1.57±0.34
Method 3	0.64±0.05	0.66±0.07	0.54±0.11	0.57±0.06	1.54±0.29

Table 6.2: Results for obstetrical dataset

6.5.2 Task Scheduling Data Sets

An alternative to the switching algorithms is a *brute force Best Selection* (BS) strategy in which every existing SA is tested against the existing system configuration, this is time consuming. An alternative could be to apply BS only in constructing a training set of data. The training data contains several platform characteristics of the scheduling scenario (tasks and resources related) together with the best SA (class label) for that specific configuration, found by BS. This data set could be further used to train a classification system. Then, when new configurations occur the classifier generated in the previous step is used to infer the corresponding SA. In [ZF11a, ZF11b] we tested several classification strategies in order to find the one that ensures the largest classification accuracy and to identify which characteristics of the scheduling events influence most the choice of an adequate SA.

Test Data

The training set was synthetically generated using the models described in [Fei10]. Each instance in the training set contains values corresponding to the following attributes: the *time when the schedule was completed*, the *mean task Estimated Execution Time (EET)*; the *mean standard deviation of the EET*; the *mean task Estimated Completion Time (ECT)*; the *mean standard deviation of the ECT*; the *mean task size*; the *mean standard deviation of the task size*; the *total number of tasks*; and the *number of long tasks* used in the experiment. Besides this information for each configuration, the best SA found by BS was added in order to characterize the class. Seven SAs were used for determining the best policy: Max-Min [MAS+99], Min-Min [MAS+99], Sufferage [CLZB00], MinQL [FMC09], MinQL-Plain [FMC09], DMECT [Fr9] and DMECT2 [Fr9].

Tested Classifiers. In the experimental analysis we used several classifiers implemented in the WEKA data mining toolkit (MultiLayer Perceptron (MLP) neural network, Radial Basis Function (RBF) network, Non-Nested Generalized Exemplars classifier (NNGE)), a Fuzzy C-Mean unsupervised classifier and the hybrid classifier that combine the NNGE algorithm with an evolutionary selection of relevant attributes and exemplars (EP-NNGE described in Section 6.4).

Test Results. A first analysis was developed in order to determine the time spent by the strategies for schedulers selection in order to find the most suitable scheduling heuristics. The average runtime of each (un)supervised technique was bellow 2.5s (training step + classification), while the BS strategy in the case of the 7 SAs requires around 6 seconds to complete one schedule event. The high classification percentages as well as the low runtimes make the learning techniques suitable for determining the best SAs without requiring a BS or switching policy.

A second analysis is done regarding the accuracy of the classification techniques. The behaviour of EP-NNGE and EPA-NNGE classifiers is similar for the three data sets even and better than of the others heuristics.

6.6 Conclusions

In this chapter we presented an multiobjective evolutionary approach for rules mining in case of medical data and an evolutionary selection strategy for finding classification rules that ensure the selection of a good scheduling strategy based on scheduling event characteristics.

Chapter 7

Conclusions and Future Work

This thesis is focused on the application of nature inspired heuristics in uncertain environments; three concrete optimization problems (grid scheduling, data clustering, rule discovery) that can be formulated like optimization problems in dynamic and uncertain environments and some test functions for dynamic optimization are discussed.

In the case of the dynamic optimization we analyzed two algorithms: particle swarm optimization and differential evolution. In order to improve the heuristics behavior in the dynamic environments we introduced a simple perturbation mechanism for particle position in the case of PSO heuristic (Section 3.3.2) and random elements for differential evolution (Section 3.3.2).

For the scheduling problem we proposed a simple population based scheduler and analyzed its robustness (Section 4.3) and behavior in the case of batch scheduling in static (Section 4.2.3) and dynamic (Section 4.5.3) environments and online scheduling (Section 4.4). We also done an analysis of the impact of different variants of the initialization (Section 4.2.1) and perturbation strategy (Section 4.2.3) in case of population based scheduler.

In the case of clustering problem we added a density parameter (Section 5.3.1) to `AntClust` heuristic in order to improve its performance when is applied to noisy data and analyzed the heuristic performance in clustering medical data affected by missing values (Section 5.4).

In the case of rules discovery, we proposed the introduction of human evaluator (Section 6.3) besides the evolutionary algorithm that performs the mining step. Another particularity of our approach consists in introducing a crowding distance between rules, that acts as a diversity criteria and the Pareto set is stimulated in order to select elements from less crowded regions that are added into an archive (Section 6.2). We also analyzed the influence of several methods to handle the missing values when searching for rules in data (Section 6.5.1).

Another problem approached, in the case of rules discovery, is the classification of the scheduling events, so when a new scheduling event appears the classifier should be able to determine the scheduling algorithm that most minimize the scheduling costs. The contribution in this case is related to introduction of a hybrid classifier based on non-nested generalized exemplars and an evolutionary selection of attributes and exemplars (Section 6.4).

Some future research directions are presented at the end of each chapter. In the case of grid scheduling problem the future work will address the case of interrelated tasks and the case of using other metrics such as the Total Processing Consumption Cycle which is an alternative to the makespan and it is hardware independent. In the case of clustering problem a more detailed analysis of the behavior of the algorithm for other data should be done and also application on real data would be desirable. In the case of evolutionary rule mining more investigations can be done regarding missing values handling methods and evolutionary rule miner design. Since the scheduling data analyzed using the evolutionary pruning technique are unbalanced, a study regarding the influence of the balancing techniques on the proposed technique is desired.

Since optimization problems in uncertain and dynamic environments are often encountered in real life and nature inspired heuristics are good candidates to resolve such problems, future research will focus on analyzing others optimization problems and different nature inspired heuristics for them.

Bibliography

[BB02] T. M. Blackwell and Peter J. Bentley. Dynamic search with charged swarms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '02, pages 19–26, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[Bra] J. Branke. Moving peaks benchmark. *www.aifb.uni-karlsruhe.de/jbr/MovPeaks/movpeaks/*.

[BSB+01] Tracy D. Braun, Howard Jay Siegel, Noah Beck, Lasislau L. Bölöni, Muthucumara Maheswaran, Albert I. Reuther, James P. Robertson, Mitchell D. Theys, Bin Yao, Debra Hensgen, and Richard F. Freund. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. *J. Parallel Distrib. Comput.*, 61:810–837, June 2001.

[CJSZ08] Louis-Claude Canon, Emmanuel Jeannot, Rizos Sakellariou, and Wei Zheng. Comparative evaluation of the robustness of dag scheduling heuristics. In Sergei Gorlatch, Paraskevi Fragopoulou, and Thierry Priol, editors, *Grid Computing*, pages 73–84. Springer US, 2008. 10.1007/978-0-387-09457-1-7.

[CLZB00] H. Casanova, A. Legrand, D. Zagorodnov, and F. Berman. Heuristics for scheduling parameter sweep applications in grid environments. In *Procs. 9th Heterogeneous Computing Workshop (HCW)*, pages 349–363, Cancun, Mexico, May 2000.

[CX06] J. Carretero and F. Xhafa. Using genetic algorithms for scheduling jobs in large scale grid applications. *Journal of Technological and Economic Development - A Research Journal of Vilnius Gediminas Technical University*, 12:11–17, 2006.

[DK07] Kalyanmoy Deb and Abhishek Kumar. Interactive evolutionary multi-objective optimization and decision-making using reference direction method. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07, pages 781–788, New York, NY, USA, 2007. ACM.

[Fei10] D.G. Feitelson. Workload modeling for computer systems performance evaluation. URL http://www.cs.huji.ac.il/ feit/wlmod/, 2010.

[FMC09] M. Frîncu, G Macariu, and A. Cârstea. Dynamic and adaptive workflow execution platform for symbolic computations. *Pollack Periodica*, 4(1):145–156, 2009.

[Fr9] Marc E. Frîncu. Dynamic scheduling algorithm for heterogeneous environments with regular task input from multiple requests. In *Proceedings of the 4th International Conference on Advances in Grid and Pervasive Computing*, GPC '09, pages 199–210, Berlin, Heidelberg, 2009. Springer-Verlag.

[JB05] Y. Jin and J. Branke. Evolutionary optimization in uncertain environments - a survey. *Evolutionary Computation, IEEE Transactions*, 9(3):303–317, 2005.

[JD88] Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[KE95] J. Kennedy and R.C. Eberhart. Particle swarm optimization. *Proceedings of IEEE International Conference on Neural Networks*, pages 1942– 1948, 1995.

[LA05] Hongbo Liu and Ajith Abraham. Fuzzy adaptive turbulent particle swarm optimization. In *Proceedings of the Fifth International Conference on Hybrid Intelligent Systems*, pages 445–450, Washington, DC, USA, 2005. IEEE Computer Society.

[LMV02] N. Labroche, N. Monmarche, and G. Venturini. A new clustering algorithm based on the chemical recognition system of ants. *Harmelen, F. van (Ed.) Proc. of the 15th European Conference on Artificial Intelligence, Lyon, France*, pages 345–349, 2002.

[LZ**Z**08] Diana Lungeanu, Daniela Zaharie, and **Flavia Zamfirache**. Influence of missing values handling on classification rules evolved from medical data. In *Industrial Conference on Data Mining - Posters and Workshops'08*, pages 86–95, 2008.

[Mar95] B. Martin. Instance-based learning: Nearest neighbour with generalisation. In *Working Paper Series 95/18 Computer Science*, page 90, Hamilton,University of Waikato, 1995.

[MAS+99] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund. Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. *Journal of Parallel and Distributed Computing*, 59:107–131, 1999.

[MM05] R. Mendes and A. Mohais. Dynde: a differential evolution for dynamic optimization problems. *Proc. of CEC2005. Edinburgh, Scotland, September 2-5*, pages 2808–2815, 2005.

[OAT+99] M. Ohsaki, H. Abe, S. Tsumoto, H. Yokoi, and T. Yamaguchi. Discovering interesting prediction rules with a genetic algorithm. *Proc. of of Congress on Evolutionary Computing (CEC 1999)*, pages 1322–1329, 1999.

[PKN10] Andrew J. Page, Thomas M. Keane, and Thomas J. Naughton. Multi-heuristic dynamic task allocation using genetic algorithms in a heterogeneous distributed system. *J. Parallel Distrib. Comput.*, 70:758–766, July 2010.

[RL04] G. Ritchie and J. Levine. A hybrid ant algorithm for scheduling independent jobs in heterogeneous computing environments. *Proceedings the 23rd Workshop of the UK Planning and Scheduling Special Interest Group*, pages 1–7, 2004.

[SP95] R. Storn and K. Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. *Technical Report TR-95-012, ICSI, March*, 1995.

[XA10] Fatos Xhafa and Ajith Abraham. Computational models and heuristic methods for grid scheduling problems. *Future Gener. Comput. Syst.*, 26:608–621, April 2010.

[Xha07] F. Xhafa. A hybrid evolutionary heuristic for job scheduling on computational grids. In Ajith Abraham, Crina Grosan, and Hisao Ishibuchi, editors, *Hybrid Evolutionary Algorithms*, volume 75 of *Studies in Computational Intelligence*, pages 269–311. Springer Berlin / Heidelberg, 2007. 10.1007/978-3-540-73297-6-11.

[**Z**F11a] **Flavia Zamfirache** and Marc Frîncu. Automatic selection of scheduling algorithms based on classification models. *STUDIA*, (2), 2011.

[**Z**F11b] **Flavia Zamfirache** and Marc Frîncu. Automatic selection of scheduling algorithms based on classification models. *Proc. of International Conference on Knowledge Engineering: Principles and Techniques, Cluj-Napoca*, 2011.

[**Z**FZ11] **Flavia Zamfirache**, Marc Frîncu, and Daniela Zaharie. Population-based metaheuristics for tasks scheduling in heterogeneous distributed systems. In *Proceedings of the 7th international conference on Numerical methods and applications*, NMA'10, pages 321–328, Berlin, Heidelberg, 2011. Springer-Verlag.

[ZL**Z**08] Daniela Zaharie, Diana Lungeanu, and **Flavia Zamfirache**. Interactive search of rules in medical data using multiobjective evolutionary algorithms. In *Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation*, GECCO '08, pages 2065–2072, New York, NY, USA, 2008. ACM.

[ZPNZ11] D. Zaharie, L. Perian, V. Negru, and F. Zamfirache. Evolutionary pruning of non-nested generalized exemplars. *Proc. of 6th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI 2011) to be held on May 19-21, 2011 in Timisora, Romania*, pages 57–62, 2011.

[Z**Z**05a] Daniela Zaharie and **Flavia Zamfirache**. Ant-based clustering of medical data. *HCMC 2005, First East European Conference on Health Care Modelling and Computation (eds: F. Gorunescu, E. El-Darzi, M. Gorunescu)*, pages 332–343, 2005.

[Z**Z**05b] Daniela Zaharie and **Flavia Zamfirache**. Dealing with noise in ant-based clustering. In *Congress on Evolutionary Computation'05*, pages 2395–2401, 2005.

[Z**Z**06] Daniela Zaharie and **Flavia Zamfirache**. Diversity enhancing mechanisms for evolutionary optimization in static and dynamic environments. *Proc. of 3rd Romanian-Hungarian Joint Symposium on Applied Computational Intelligence*, pages 460–471, 2006.

[**Z**ZC10a] **Flavia Zamfirache**, Daniela Zaharie, and Ciprian Craciun. Evolutionary task scheduling in static and dynamic environments. *Proc. ICCC-CONTI, Timisoara, Romania*, pages 619–624, 2010.

[**Z**ZC10b] **Flavia Zamfirache**, Daniela Zaharie, and Ciprian Craciun. Nature inspired metaheuristics for task scheduling in static and dynamic computing environments. *Scientific Buletin of Politehnica University of Timisoara, Romsnis, BS-UPT TACCS*, 55(69)(3):133–142, 2010.

[Z**Z**N$^+$07] Daniela Zaharie, **Flavia Zamfirache**, Viorel Negru, Daniel Pop, and Horia Popa. A comparison of quality criteria for unsupervised clustering of documents based on differential evolution. *Proc. of International Conference on Knowledge Engineering: Principles and Techniques*, pages 114–121, 2007.