4

Algorithm Selection: From Meta-Learning to Hyper-Heuristics

Laura Cruz-Reyes[1], Claudia Gómez-Santillán[1],
Joaquín Pérez-Ortega[2], Vanesa Landero[3],
Marcela Quiroz[1] and Alberto Ochoa[4]
[1]*Instituto Tecnológico de Cd. Madero*
[2]*Centro Nacional de Investigación y Desarrollo Tecnológico*
[3]*Universidad Politécnica de Nuevo León*
[4]*Universidad de Ciudad Juárez*
México

1. Introduction

In order for a company to be competitive, an indispensable requirement is the efficient management of its resources. As a result derives a lot of complex optimization problems that need to be solved with high-performance computing tools. In addition, due to the complexity of these problems, it is considered that the most promising approach is the solution with approximate algorithms; highlighting the heuristic optimizers. Within this category are the basic heuristics that are experience-based techniques and the metaheuristic algorithms that are inspired by natural or artificial optimization processes.

A variety of approximate algorithms, which had shown satisfactory performance in optimization problems, had been proposed in the literature. However, there is not an algorithm that performs better for all possible situations, given the amount of available strategies, is necessary to select the one who adapts better to the problem. An important point is to know which strategy is the best for the problem and why it is better.

The chapter begins with the formal definition of the Algorithm Selection Problem (ASP), since its initial formulation. The following section describes examples of "Intelligent Systems" that use a strategy of algorithm selection. After that, we present a review of the literature related to the ASP solution. Section four presents the proposals of our research group for the ASP solution; they are based on machine learning, neural network and hyper-heuristics. Besides, the section presents experimental results in order to conclude about the advantages and disadvantages of each approach. Due to a fully automated solution to ASP is an undecidable problem, Section Five reviews other less rigid approach which combines intelligently different strategies: The Hybrid Systems of Metaheuristics.

2. The Algorithm Selection Problem (ASP)

Many optimization problems can be solved by multiple algorithms, with different performance for different problem characteristics. Although some algorithms are better than others on average, there is not a best algorithm for all the possible instances of a given problem. This phenomenon is most pronounced among algorithms for solving NP-Hard problems, because runtimes for these algorithms are often highly variable from instance to instance of a problem (Leyton-Brown et al., 2003). In fact, it has long been recognized that there is no single algorithm or system that will achieve the best performance in all cases (Wolpert & Macready, 1997). Instead we are likely to attain better results, on average, across many different classes of a problem, if we tailor the selection of an algorithm to the characteristics of the problem instance (Smith-Miles et al., 2009). To address this concern, in the last decades researches has developed technology to automatically choose an appropriate optimization algorithm to solve a given instance of a problem, in order to obtain the best performance.

Recent work has focused on creating algorithm portfolios, which contain a selection of state of the art algorithms. To solve a particular problem with this portfolio, a pre-processing step is run where the suitability of each algorithm in the portfolio for the problem at hand is assessed. This step often involves some kind of machine learning, as the actual performance of each algorithm on the given, unseen problem is unknown (Kotthoff et al., 2011).

The Algorithm Selection Problem (ASP) was first described by John R. Rice in 1976 (Rice, 1976) he defined this problem as: learning a mapping from feature space to algorithm performance space, and acknowledged the importance of selecting the right features to characterize the hardness of problem instances (Smith-Miles & Lopes, 2012). This definition includes tree important characteristics (Rice, 1976):

a. *Problem Space*: The set of all possible instance of the problem. There are a big number of independent characteristics that describe the different instances which are important for the algorithm selection and performance. Some of these characteristics and their influences on algorithm performance are usually unknown.
b. *Algorithm Space*: The set of all possible algorithms that can be used to solve the problem. The dimension of this set could be unimaginable, and the influence of the algorithm characteristics is uncertain.
c. *Performance Measure*: The criteria used to measure the performance of a particular algorithm for a particular problem and see how difficult to solve (hard) is the instance. There is considerable uncertainly in the use and interpretation of these measures (e. g. some prefer fast execution, others effectiveness, others simplicity).

Rice proposed a basic model for this problem, which seeks to predict which algorithm from a subset of the algorithm space is likely to perform best based on measurable features of a collection of the problem space: Given a problem subset of the problem space P, a subset of the algorithm space A, a mapping from P to A and the performance space Y. The Algorithm Selection Problem can be formally defined as: for a particular problem instance $p \in P$, find the selection mapping $S(p)$ into the algorithm space A, such that the selected algorithm $a \in A$ maximizes the performance measure $\|y\|$ for $y(a,p) \in Y$. This basic abstract model is illustrated in Figure 1 (Rice, 1976; Smith-Miles & Lopes, 2012).

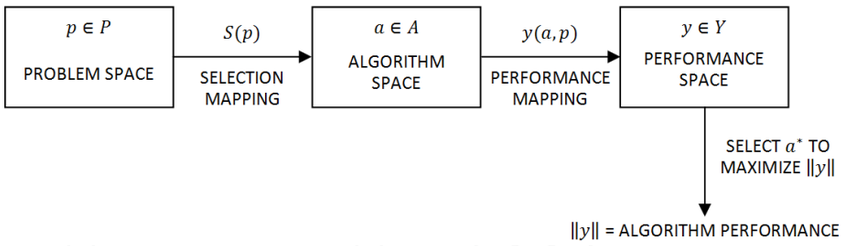


Fig. 1. The Algorithm Selection Problem (ASP)

The Figure 2 shows the dimensions of ASP and allows see a higher level of abstraction scope. There are three dimensions: 1) in the x-axis expresses a set of algorithms of solution $\{s, t, w, y, z\}$, 2) z-axis shows a set of instances of the problem $\{a, b, c, d\}$, and a new instance e to solve, 3) in the y-axis the set of values of the results of applying the algorithms to each of the instances is represented by vertical lines. As shown in figure, to solve the instance a and b the algorithms have different performances, it is noteworthy that no algorithm is superior to others in solving all instances. Moreover, as shown in figure the algorithm s has a different performance by solving each of the instances. Finally the problem to be solved is to select for the new instance e the algorithm that will solve better.

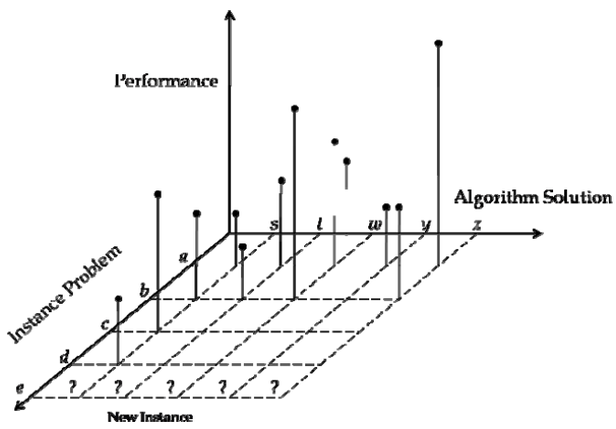


Fig. 2. Dimensions of algorithm selection problem

As we can see in the definition of the Algorithm Selection Problem there are three principal aspects that must be tackled in order to solve the problem:

a. The *selection of the set features of the problem* that might be indicative of the performance of the algorithms.
b. The *selection of the set of algorithms* that together allow to solve the largest number of instances of the problem with the highest performance.
c. The *selection of an efficient mapping mechanism* that permits to select the best algorithm to maximize the performance measure.

Some studies have been focused in construct a suitable set of features that adequately measure the relative difficulty of the instances of the problem (Smith-Miles et al., 2009; Messelis et al., 2009; Madani et al., 2009; Quiroz, 2009; Smith-Miles & Lopes, 2012). Generally there are two main approaches used to characterize the instances: the first is to

identify problem dependent features based on domain knowledge of what makes a particular instance challenging or easy to solve; the second is a more general set of features derived from landscape analysis (Schiavinotto & Stützle, 2007; Czogalla & Fink, 2009). To define the set of features that describe the characteristics of the instances is a difficult task that requires expert domain knowledge of the problem. The indices of characterization should be carefully chosen, so as to permit a correct discrimination of the difficulty of the instances to explain the algorithms performance. There is little that will be learned via a knowledge discovery process if the features selected to characterize the instances do not have any differentiation power (Smith-Miles et al., 2009).

On the other hand, portfolio creation and algorithm selection has received a lot of attention in areas that deal with solving computationally hard problems (Leyton-Brown et al., 2003; O'Mahony et al., 2008). The current state of the art is such that often there are many algorithms and systems for solving the same kind of problem; each with its own performance on a particular problem. Machine learning is an established method of addressing ASP (Lobjois & Lemâitre, 1998; Fink, 1998). Given the performance of each algorithm on a set of training problems, we try to predict the performance on unseen problems (Kotthoff et al., 2011). There have been many studies in the area of algorithm performance prediction, which is strongly related to algorithm selection in the sense that supervised learning or regression models are used to predict the performance ranking of a set of algorithms, given a set of features of the instances (Smith-Miles & Lopes, 2012).

In the selection of the efficient mapping mechanism a challenging research goal is to design a run-time system that can repeatedly execute a program, learning over time to make decisions that maximize the performance measure. Since the right decisions may depend on the problem size and parameters, the machine characteristics and load, the data distribution, and other uncertain factors, this can be quite challenging. Some works treats algorithms in a black-box manner: each time a single algorithm is selected and applied to the given instance then a regression analysis or machine learning techniques are used to build a predictive model of the performance of the algorithms given the features of the instances (Lobjois & Lemâitre, 1998; Fink, 1998; Leyton-Brown et al., 2003; Ali & Smith, 2006). Other works focus on dynamic selection of algorithm components while the instance is being solved. In that sense, each instance is solved by a mixture of algorithms formed dynamically at run-time (Lagoudakis & Littman, 2000; Samulowitz & Memisevic, 2007; Streeter et al., 2007). The use of efficient mapping mechanism in intelligent systems is described in the next section.

3. Applications of algorithm selection to real world and theorists problems

The principles applied to ASP can be used in a wide range of applications in the real world and theoretical. Generally an application that solves a real problem is an extended version of parameters and constraints in another application that solves a theoretical problem. The nature of the algorithm selection problem is dynamic because it must incorporate new knowledge periodically, in order to preserve the efficacy of selection strategies. This section describes some applications to real-world complex problems, such as knowledge discovery and data mining, bioinformatics and Web services. It also describes some applications to solve complex theoretical problems; some examples are NP-hard problems, also called combinatorial optimization problems.

3.1 Bioinformatics

In (Nascimento et al., 2009) the authors investigate the performance of clustering algorithms on gene expression data, by extracting rules that relate the characteristics of the data sets of gene expression to the performance achieved by the algorithms. This represents a first attempt to solve the problem of choosing the best cluster algorithm with independence of gene expression data. In general, the choice of algorithms is basically driven by the familiarity of biological experts to the algorithm, rather than the characteristics of the algorithms themselves and of the data. In particular, the bioinformatics community has not reached consensus on which method should be preferably used. This work is directly derived from the Meta-Learning framework, originally proposed to support algorithm selection for classification and regression problems. However, Meta-Learning has been extended to other domains of application, e.g. to select algorithms for time series forecasting, to support the design of planning systems, to analyze the performance of meta-heuristics for optimization problems. Meta-Learning can be defined by considering four aspects: (a) the problem space, P, (b) the meta-feature space, F, (c) the algorithm space, A and (d) a performance metric, Y. As final remark, authors demonstrated that the rule-based ensemble classifier presented the most accuracy rates in predicting the best clustering algorithms for gene expression data sets. Besides, the set of extracted rules for the selection of clustering algorithms, using an inductive decision tree algorithm, gave some interesting guidelines for choosing the right method.

3.2 WEB services

In recent years, many studies have focused on developing feasible mechanisms to select appropriate services from service systems in order to improve performance and efficiency. However, these traditional methods do not provide effective guidance to users and, with regard to ubiquitous computing, the services need to be context-aware. In consequence, the work achieved by (Cai et al., 2009) proposed a novel service selection algorithm based on Artificial Neural Network (ANN) for ubiquitous computing environment. This method can exactly choose a most appropriate service from many service providers, due to the earlier information of the cooperation between the devices. Among the elements that exist in the definition of a service, Z represents the evaluation value of respective service providers' service quality, and its value is calculated with a function that involves the time and the conditions of current context environment, e.g. user context, computing context, physical context, with a division into static and dynamical information.

Among the advantages of using ANN to solve the service selection problem, is that, the method can easily adapt the evaluation process to the varying context information, and hence, it can provide effective guidance so that lots of invalid selecting processes can be avoided. The neural network selected was Back Propagation (BP) because is the most commonly used; however, this algorithm was improved with a three-term approach: learning rate, momentum factor and proportional factor. The efficiency of such algorithm was obtained because adding the proportional factor enhanced the convergence speed and stability. In conclusion, the authors claim, that this novel service selection outperforms the traditional service selection scheme.

3.3 Learning systems

In (Bradzil et al., 2003) is described a meta-learning method to support selection of candidate learning algorithms. Bradzil et al. use the Instance-Based Learning (IBL) approach because IBL has the advantage that the system is extensible; once a new experimental result becomes available, it can be easily integrated into the existing results without the need to reinitiate complex re-learning. In this work a k-Nearest Neighbor (k-NN) algorithm to identify the datasets that are most similar to the one is used. The distance between datasets is assessed using a relatively small set of data characteristics, which was selected to represent properties that affect algorithm performance; it is used to generate a recommendation to the user in the form of a ranking. The prediction, is constructed by aggregating performance information for the given candidate algorithms on the selected datasets. They use a ranking method based on the relative performance between pairs of algorithms. This work shown how can be exploited meta-learning to pre-select and recommend one or more classification algorithms to the user. They claimed that choosing adequate methods in a multistrategy learning system might significantly improve its overall performance. Also it was shown that meta-learning with k-NN improves the quality of rankings methods in general.

3.4 Knowledge discovery and data mining

In (Hilario & Kaousis, 2000) is addressed the model selection problem in knowledge discovery systems, defined as the problem of selecting the most appropriate learning model or algorithm for a given application task. In this work they propose framework for characterizing learning algorithms for classification as well as their underlying models, using learning algorithm profiles. These profiles consist of metalevel feature-value vectors, which describe learning algorithms from the point of view of their representation and functionality, efficiency, resilience, and practicality. Values for these features are assigned on the basis of author specifications, expert consensus or previous empirical studies. Authors review past evaluations of the better known learning algorithms and suggest an experimental strategy for building algorithm profiles on more quantitative grounds. The scope of this paper is limited to learning algorithms for classification tasks, but it can be applied to learning models for other tasks such as regression or association.

In (Kalousis & Theoharis, 1999) is presented an Intelligent Assistant called NOEMON, which by inducing helpful suggestion from background information can reduce the effort in classifier selection task. For each registered classifier, NOEMON measures its performance in order to collect datasets for constituting a morphologic space. For suggest the most appropriate classifier, NOEMON decides on the basis of morphological similarity between the new dataset and the existing collection. Rules are induced from those measurements and accommodated in a knowledge database. Finally, the suggestions on the most appropriate classifier for a dataset are based on those rules. The purpose of NOEMON is to supply the expert with suggestions based on its knowledge on the performance of the models and algorithms for related problems. This knowledge is being accumulated in a knowledge base end is updated as new problems as are being processed.

3.5 Scheduling problem

In (Kadioglu et al., 2011) the main idea is taken from an algorithm selector called Boolean Satisfiability (SAT) based on nearest neighbor classifier. On one hand, authors presented two extensions to it; one of them is based on the concept of distance-based weighting, where they assign larger weights to instances that are closer to the test instance. The second extension, is based on clustering-based adaptive neighborhood size, where authors adapt the size of the neighborhood based on the properties of the given test instance. These two extensions show moderate but consistent performance improvements to the algorithm selection using Nearest-Neighbor Classification (Malitsky et al., 2011). On the other hand, authors developed a new hybrid portfolio that combines algorithm selection and algorithm scheduling, in static and dynamic ways. For static schedules the problem can be formulated as an integer program, more precisely, as a resource constrained set covering problem, where the goal is to select a number of solver-runtime pairs that together "cover" (i.e., solve) as many training instances as possible. Regarding dynamic schedules, the column generation approach works fast enough when yielding potentially sub-optimal but usually high quality solutions. This allows us to embed the idea of dynamic schedules in the previously developed nearest-neighbor approach, which selects optimal neighborhood sizes by random sub-sampling validation. With SAT as the testbed, experimentation demonstrated that author's approach can handle highly diverse benchmarks, in particular a mix of random, crafted, and industrial SAT instances, even when deliberately removed entire families of instances from the training set. As a conclusion, authors presented a heuristic method for computing solver schedules efficiently, which O'Mahony (O'Mahony et al., 2008) identified as an open problem. In addition, they showed that a completely new way of solver scheduling consisting of a combination of static schedules and solver selection is able to achieve significantly better results than plain algorithm selection.

3.6 Traveling salesman problem

In (Kanda et al., 2011), the work is focused in the selection of optimization algorithms for solving TSP instances; this paper proposes a meta-learning approach to recommend optimization algorithms for new TSP instances. Each instance is described by meta-features of the TSP that influences the efficiency of the optimization algorithms. When more than one algorithm reaches the best solution, the multi-label classification problem is addressed applying three steps: 1) decomposition of multi-label instances into several single-label instances, 2) elimination of multi-label instances, and 3) binary representation, in order to transform multi-label instances into several binary classification problems. Features were represented as a graph. The success of this meta-learning approach depended on the correct identification of the meta-features that best relate the main aspects of a problem to the performances of the used algorithms. Finally the authors claimed that it is necessary to define and expand the set of metafeatures, which are important to characterize datasets in order to improve the performance of the selection models.

3.7 Satisfiability problem

In (Xu et al., 2009) is described an algorithm for constructing per-instance algorithm portfolios for SAT. It has been widely observed that there is no single "dominant" SAT solver; instead, different solvers perform best on different instances. SATzilla is an

automated approach for constructing per-instance algorithm portfolios for SAT that use so-called empirical hardness models to choose among their constituent solvers. This approach takes as input a distribution of problem instances and a set of component solvers, and constructs a portfolio optimizing a given objective function (such as mean runtime, percent of instances solved, or score in a competition). The algorithm selection approach is based on the idea of building an approximate runtime predictor, which can be seen as a heuristic approximation to a perfect oracle. Specifically, they use machine learning techniques to build an empirical hardness model, a computationally inexpensive predictor of an algorithm's runtime on a given problem instance based on features of the instance and the algorithm's past performance. By modeling several algorithms and, at runtime, choosing the algorithm predicted to have the best performance; empirical hardness models can serve as the basis for an algorithm portfolio that solves the algorithm selection problem automatically.

3.8 Vehicle routing problem

In (Ruiz-Vanoye et al., 2008) the main contribution of this paper is to propose statistical complexity indicators applied to the Vehicle Routing Problem with Time Windows (VRPTW) instances in order that it allows to select appropriately the algorithm that better solves a VRPTW instance. In order to verify the proposed indicators, they used the discriminant analysis contained in SPSS software, such as a machine learning method to find the relation between the characteristics of the problem and the performance of algorithms (Perez et al., 2004), as well as the execution of 3 variants of the genetic algorithms and the random search algorithm. The results obtained in this work showed a good percentage of prediction taking into account that this based on statistical techniques and not on data-mining techniques. By means of the experimentation, authors conclude that it is possible to create indicators applied to VRPTW that help appropriately to predict the algorithm that better solves the instances of the VRPTW.

4. Related work on automatic algorithm selection

In this section some examples of related works of the reviewed literature are classified by Methods or methodologies utilized for establishing the relation between the problems and algorithms, and solve the algorithm selection problem. 2.1. Solution Environments, where the algorithm selection problem is boarded, are described in section 2.2.

4.1 Simple statistical tests

The most common method to compare experimentally algorithms consists in the complementary use of a set of simple well-known statistical tests: The Sign, Wilcoxon and Friedman tests, among others. The tests are based on the determination of the differences in the average performance, which is observed experimentally: if the differences among the algorithms are significant statistically, the algorithm with the best results is considered as superior (Lawler 1985). Reeves comments that a heuristic with good averaged performance, but with high dispersion, has a very high risk to show a poor or low performance in many instances (Reeves 1993). He suggests as alternative to formulate for each algorithm, a utility function adjusted to a gamma distribution, whose parameters permit to compare the heuristics on a range of risk value.

4.2 Regression analysis

Gent and Walsh make an empirical study of the GSAT algorithm, it is an approximation algorithm for SAT, and they apply regression analysis to model the growth of the cost of obtaining the solution with the problem size (Gent 1997).

In (Cruz 1999), Pérez and Cruz present a statistical method to build algorithm performance models, using polynomial functions, which relate the performance with the problem size. This method first generates a representative sample of the algorithms performance, and then the performance functions are determined by regression analysis, which finally are incorporated in an algorithm selection mechanism. The polynomial functions are used to predict the best algorithm that satisfies the user requirements.

The performance of local search algorithms Novelty and SAPS for solving instances of the SAT problem were analyzed by (Hutter 2006). The authors used linear regression with linear and quadratic basis functions to build prediction models. Firstly, they built a prediction model, using problem features and algorithm performance, to predict the algorithm run time. Secondly, they build another prediction model, using problem features, algorithm parameter settings and algorithm performance. This model is used to automatically adjust the algorithm's parameters on a per instance basis in order to optimize its performance.

4.3 Functions of probability distribution

Frost finds that the performance of the algorithms to solve CSP instances can be approximated by two standard families of functions of continuous probability distribution (Frost 1997). The resoluble instances can be modeled by the Weibull distribution and the instances that are not resoluble by the lognormal distribution. He utilizes four parameters to generate instances: number of constraints, number of prohibited value pairs per constraint, the probability of a constraint existing between any pair of variables, the probability each constraint is statistically independent of the others, and the probability that a value in the domain of one variable in a constraint will be incompatible with a value in the domain of the other variable in the constraint.

Hoos and Stuzle present a similar work to Frost. They find that the performance of algorithms that solve the SAT instances can be characterized by an exponential distribution (Hoos 2000). The execution time distribution is determined by the execution of k times of an algorithm over a set of instances of the same family, using a high time as stop criteria and storing for each successful run the execution time required to find the solution. The empirical distribution of the execution time is the accumulated distribution associated with these observations, and it allows projecting the execution time t (given by the user) to the probability of finding a solution in this time. A family is a set of instances with the same value of the parameters that are considered critical for the performance.

An algorithm portfolio architecture was proposed in (Silverthorn 2010). This architecture employs three core components: a portfolio of algorithms; a generative model, which is fit to data on those algorithms past performance, then used to predict their future performance; and a policy for action selection, which repeatedly chooses algorithms based on those predictions. Portfolio operation begins with offline training, in which a) training tasks are

drawn from the task distribution, b) each solver is run many times on each training task, and c) a model is fit to the outcomes observed in training. In the test phase that follows, repeatedly, (1) a test task is drawn from the same task distribution, (2) the model predicts the likely outcomes of each solver, (3) the portfolio selects and runs a solver for some duration, (4) the run's outcome conditions later predictions, and (5) the process continues from (2) until a time limit expires.

The models of solver behavior are two latent class models: a multinomial mixture that captures the basic correlations between solvers, runs, and problem instances, and a mixture of Dirichlet compound multinomial distributions that also captures the tendency of solver outcomes to recur. Each model was embedded in a portfolio of diverse SAT solvers and evaluated on competition benchmarks. Both models support effective problem solving, and the DCM-based portfolio is competitive with the most prominent modern portfolio method for SAT (Xu 2009).

4.4 Functions of heuristic rules

Rice introduced the poly-algorithm concept (Rice 1968) in the context of parallel numeric software. He proposes the use of functions that can select, from a set of algorithms, the best to solve a given situation. After the Rice work, other researchers have formulated different functions that are presented in (Li 1997, Brewer 1995). The majority of the proposed functions are simple heuristic rules about structural features of the parameters of the instance that is being solved, or about the computational environment. The definition of the rules requires of the human experience.

The objective of the proposed methodology in (Beck 2004) is to find the best solution to a new instance, when a total limit time T is given. Firstly, the selection strategies for a set of algorithms A were formulated and denominated as prediction rules, these are: Selection is based on the cost of the best solution found by each algorithm; Selection is based on the change in the cost of the best solutions found at 10 second intervals; Selection is based on the extrapolation of the current cost and slope to a predicted cost at T.

These rules are applied for the training dataset and the optimal sampling time t* (required time to select the algorithm with the less solution error) is identified for each of them. After, when a new instance is given, each prediction rule is utilized to find the algorithm with the best found solution in a time tp = |A| x t*, and it is executed in the remaining time *tr* = T - tp. One of the advantages is that the methodology can be applied to different problems and algorithms. Nevertheless, the new dataset must have similarity with the training dataset.

4.5 Machine learning

The algorithm selection problem is focused by Lagoudakis and Littam in (Lagoudakis 2000) as a minimization problem of execution total time, which is solved with a Reinforced Learning algorithm (RL). Two classical problems were focused: selecting and ordering. A function that predicts the best algorithm for a new instance using its problem size is determined by means of training. The learned function permits to combine several recursive algorithms to improve its performance: the actual problem is divided in subproblems in

each recursive step, and the most adequate algorithm in size is used for each of them. This work is extended to backtracking algorithms to SAT problem in (Lagoudakis 2001).

A system (PHYTHIA-II) to select the most appropriated software to solve a scientific problem is proposed in (Houstis 2002). The user introduces the problem features (operators of the equation, its domain, values of the variables, etc.) and time requirements and allowed error. The principal components of PHYTHIA-II are the statistical analysis, pattern extraction module and inference engine. The first consists in ranking the algorithms performance data by means of Friedman rank sums (Hollander 1973). The second utilizes different machine learning methods to extract performance patterns and represent them with decision and logic rules. The third is the process to correspond the features of a new problem with the produced rules; the objective is to predict the best algorithm and the most appropriated parameters to solve the problem.

The METAL research group proposed a method to select the most appropriate classification algorithm for a set of similar instances (Soares 2003). They used a K-nearest neighborhood algorithm to identify the group of instances from a historical registry that exhibit similar features to those of a new instance group. The algorithm performance on instances of historical registry is known and is used to predict the best algorithms for the new instance group. The similarity among instances groups is obtained considering three types of problem features: general, statistical and derived from information theory.

A Bayesian approach is proposed in (Guo, 2004) to construct an algorithm selection system which is applied to the Sorting and Most Probable Explanation (MPE) problems. From a set of training instances, their features and the run time of the best algorithm that solves each instance are utilized to build the Bayesian network. Guo proposed four representative indexes from the Sorting problem features: the size of the input permutation and three presortedness measures. For the MPE problem he utilizes general features of the problem and several statistical indexes of the Bayesian network that represents the problem.

A methodology for instance based selection of solver's policies that solves instances of the SAT problem was proposed by (Nikolic 2009). The policies are heuristics that guide the search process. Different configurations of these policies are solution strategies. The problem structure of all instances was characterized by indices. The problem instances were grouped by the values of these indices, forming instances families. All problem instances were solved by all solution strategies. The best solution strategy for each family is selected. The k-nearest neighbor algorithm selects the solution strategy for a new input instance. The results of the performance of the algorithm ARGOSmart, that performs the proposed methodology, were superior to ARGOSAT algorithm.

5. Approaches to building algorithm selectors

In this chapter we solve ASP with two approaches: meta-learning and hyper-heuristics. The meta-learning approach is oriented to learning about classification using machine learning methods; three methods are explored to solve an optimization problem: Discriminant Analysis (Pérez, 2004), C4.5 and the Self-Organising Neural Network. The hyper-heuristic approach is oriented to automatically produce an adequate combination of available low-level heuristics in order to effectively solve a given instance (Burke et al., 2010); a hyper-

heuristic strategy is incorporated in an ant colony algorithm to select the heuristic that best adjust one of its control parameter.

5.1 Selection of metaheuristics using meta-learning

In this section a methodology based on Meta-Learning is presented for characterizing algorithm performance from past experience data. The characterization is used to select the best algorithm for a new instance of a given problem. The phases of the methodology are described and exemplified with the well known one-dimensional Bin-Packing problem.

5.1.1 Algorithms for the solution of the Bin Packing Problem

The Bin Packing Problem (BPP) is an NP-hard combinatorial optimization problem, in which the objective is to determine the smallest number of bins to pack a set of objects. For obtaining suboptimal solutions of BPP, with less computational effort, we used deterministic and non-deterministic algorithms. The algorithm performance is evaluated with the optimal deviation percentage and the processing time (Quiroz, 2009).

The deterministic algorithms always follow the same path to arrive at the same solution. The First Fit Decreasing (FFD) algorithm places the items in the first bin that can hold them. The Best Fit Decreasing (BFD) places the items in the best-filled bin that can hold them. The Match to First Fit (MFF) algorithm is a variation of FFD, wich uses complementary bins for holding temporarily items. The Match to Best Fit (MBF) algorithm is a variation of BFD and, like MFF uses complementary bins. The Modified Best Fit Decreasing (MBFD) partially pack the bins in order to find a "good fit" item combination.

The Non-Deterministic Algorithms do not obtain the same solution in different executions, but in many cases they are faster than deterministic algorithms. The Ant Colony Optimization (ACO) algorithm builds a solution with each ant: it starts with an empty bin; next, each new bin is filled with "selected items" until no remaining item fits in it; finally, a "selected item" is chosen stochastically using mainly a pheromone trail (Ducatelle, 2001). In the Threshold Accepting (TA) algorithm, a new feasible solution is accepted if the difference with the previous solution is within a threshold temperature; the value of the temperature is decreased each time until a thermal equilibrium is reached (Pérez, 2002).

5.1.2 Methodology

The methodology proposed for performance characterization and its application to algorithm selection consists of three consecutive phases: Initial Training, Prediction and Training with Feedback. Figure 3 depicts these phases.

In the *Initial Training Phase*, two internal processes build a past experience database: the Problem Characterization Process obtains statistical indices to measure the computational complexity of a problem instance and, the Algorithm characterization Process solves instances with the available algorithms to obtain performance indices. The Training Process finally builds a knowledge base using the Problem and Algorithms Database. This knowledge is represented through a learning model, which relates the algorithms performance and the problem characteristics. In the *Prediction Phase*, The relationship learned is used to predict the best algorithm for a new given instance. In the *Training with*

Feedback phase, the new solved instances are incorporated into the characterization process for increasing the selection quality. The relationship learned in the knowledge base is improved with a new set of solved instances and is used again in the prediction phase.

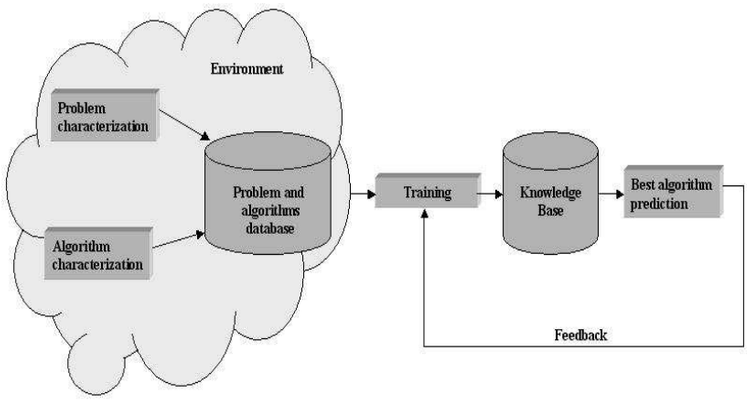


Fig. 3. Phases of the algorithm selection methodology

Initial training phase

The steps of this phase are shown in Figure 4 In step 1 (Characteristics Modeling) indices are derived for measuring the influence of problem characteristics on algorithm performance (see Equations 1 to 5). In step 2 (Statistical Sampling) a set of representative instances are generated with stratified sampling and a sample size derived from survey sampling. In step 3 (Characteristics Measurement) the parameter values of each instance are transformed into indices. In step 4 (Instances Solution) instances are solved using a set of heuristic algorithms. In Step 5 (Clustering) groups are integrated in such a way that they are constituted by instances with similar characteristics, and for which an algorithm outperformed the others. Finally, in step 6 (Classification) the identified grouping is learned into formal classifiers.

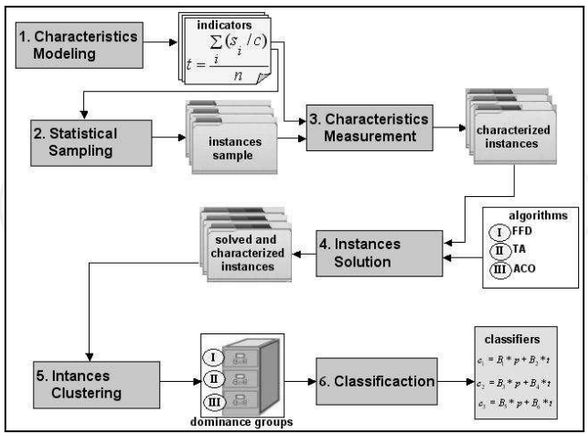


Fig. 4. Steps of the initial training phase

We propose five indices to characterize the instances of BPP:

Instance size p expresses a relationship between instance size and the maximum size solved, where, *n* is the number of items, *maxn* is the maximum size solved

$$p = \frac{n}{maxn} \tag{1}$$

a. *Constrained capacity t* expresses a relationship between the average item size and the bin size. The size of item *i* is s_i and the bin size is *c*.

$$t = \frac{\sum_i (s_i / c)}{n} \qquad 1 \le i \le n \tag{2}$$

b. *Item dispersion d* expresses the dispersion degree of the item size values.

$$d = \sigma\,(t) \tag{3}$$

c. *Number of factors f* expresses the proportion of items whose sizes are factors of the bin capacity.

$$f = \frac{\sum_i factor(c, s_i)}{n} \qquad 1 \le i \le n \tag{4}$$

d. *Bin usage b* expresses the proportion of the total size that can fit in a bin of capacity *c*.

$$b = \begin{cases} 1 & \text{if} \quad c \ge \sum_i s_i \\ \dfrac{c}{\sum_i s_i} & \text{otherwise} \end{cases} \qquad 1 \le i \le n \tag{5}$$

Prediction phase

The steps of this phase are shown in Figure 5. For a new instance, step 7 (Characteristics Measurement) calculates its characteristic values using indices. Step 8 uses the learned classifiers to determine, from the characteristics of the new instance, which group it belongs to. The algorithm associated to this group is the expected best algorithm for the instance.

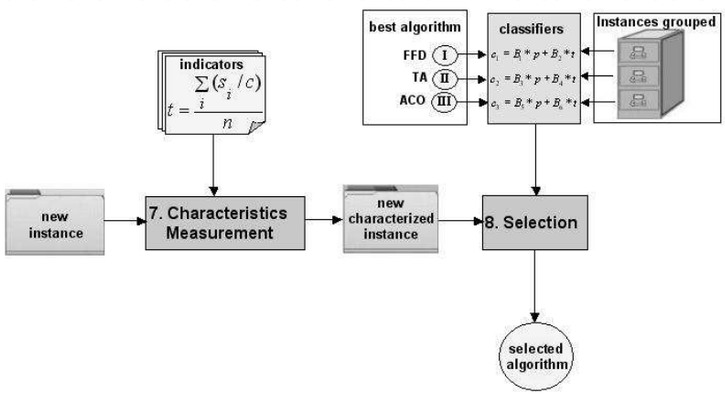


Fig. 5. Steps of the prediction phase

Training and FeedBack phase

The steps of this phase are shown in Figure 6. The objective is to feedback the system in order to maintain it in a continuous training. For each new solved and characterized instance, step 9 (Instance Solution) obtains the real best algorithm. Afterwards, step 10 (Patterns Verification) compares the result, if the prediction is wrong and the average accuracy is beyond an specified threshold, then the classifiers are rebuilt using the old and new dataset; otherwise the new instance is stored and the process ends.

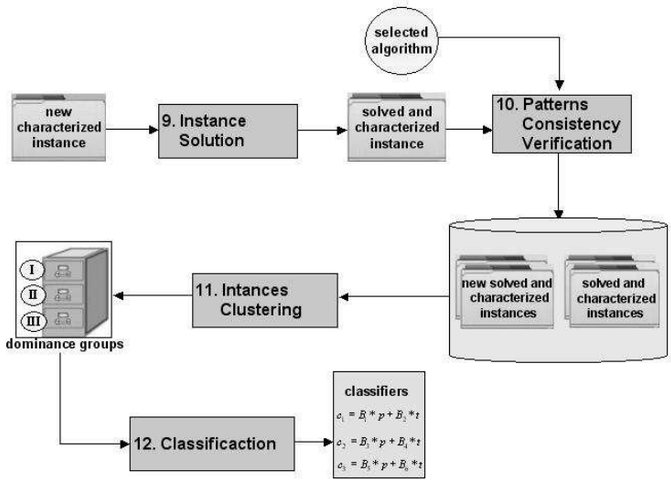


Fig. 6. Steps of the training with feedback phase

5.1.3 Experimentation

For test purposes 2,430 random instances of the Bin-Packing problem were generated, characterized and solved using the seven heuristic algorithms described in Section 5.1.1. Table 1 shows a small instance set, which were selected from the sample.

Instance	Problem characteristic indices					Real best algorithms
	p	*b*	*t*	*f*	*d*	
E1i10.txt	0.078	0.427	0.029	0.000	0.003	FFD,TA
E50i10.txt	0.556	0.003	0.679	0.048	0.199	BFD,ACO
E147i10.txt	0.900	0.002	0.530	0.000	0.033	TA

Table 1. Example of random intances with their characteristics and the best algorithms

The K-means clustering method was used to create similar instance groups. Four groups were obtained; each group was associated with a similar instances set and an algorithm with the best performance for it. Three algorithms had poor performance and were outperformed by the other four algorithms. The Discriminant Analysis (DA) and C4.5 classification methods were used to build the algorithm selector. We use the machine learning methods available in SPSS version 11.5 and Weka 3.4.2, respectively. Afterwards, for validating the system, 1,369 standard instances were collected [Ross 2002]. In the selection of the best algorithm for all standard instances, the experimental results showed an accuracy of 76% with DA and 81% with C4.5. This accuracy was compared with a random selection from the

seven algorithms: 14.2%. For the instances of the remaining percentage (100-76%), the selected algorithms generate a solution close to the optimal.

The selection system with feedback was implemented using a neural network, particularly the Self-Organizing Map (SOM) of Kohonen available in Matlab 7.0. The best results were obtained with only two problem characteristic indices (p,t) in a multi-network. The accuracy increased from 78.8% in 100 epochs up to 100% in 20,000 epochs. These percentages correspond to the network with initial-training and training-with-feedback, respectively. The SOM was gradually feedback with all the available instances. Using all indices (p,b,t,f,d) the SOM only reached 76.6% even with feedback.

5.2 Selection of heuristics in a hyper-heuristic framework

A hyper-heuristic is an automated methodology for selecting heuristics to solve hard computational search problems (Burke et al., 2009; Burke et al., 2010; Duarte et al., 2007). Its methodology is form by a high-level algorithm that, given a particular problem instance and a number of low-level heuristics or metaheuristic, can select and apply an appropriate low-level heuristic or metaheuristic at each decision step. These procedures on their way to work raise the generality at which search strategy can operate. General scheme for design a hyper-heuristic is shown in Figure 7**.**

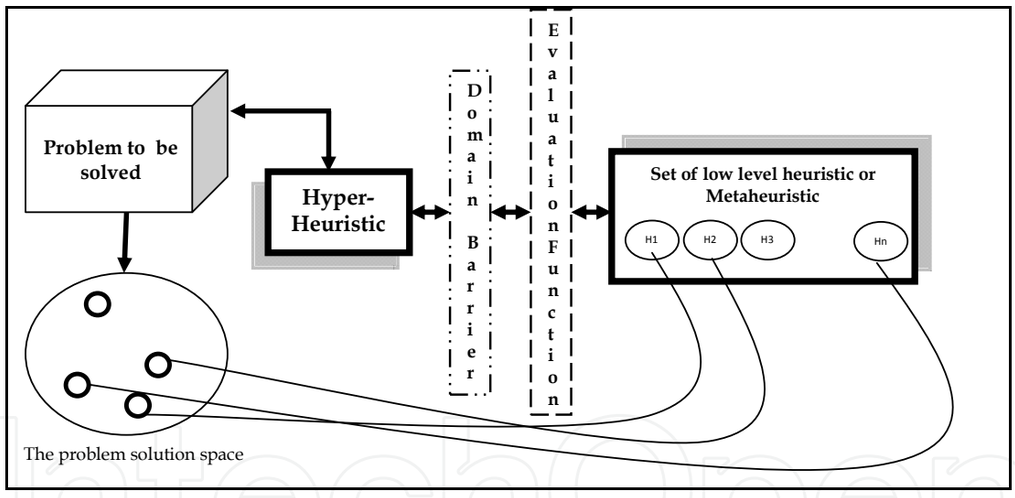


Fig. 7. Hyper-heuristic Elements

The first low-level algorithms build a solution incrementally; starting with an empty solution with the goal is to intelligently select the next construction heuristics or metaheuristic to gradually build a complete solution (Garrido, & Castro, 2009).

5.2.1 Representative examples

SQRP is the problem of locating information in a network based on a query formed by keywords. The goal of SQRP is to determine the shortest paths from a node that issues a query to nodes that can appropriately answer it (by providing the requested information). Each query traverses the network, moving from the initiating node to a neighboring node and then to a neighbor of a neighbor and so forth, until it locates the requested resource or

gives up in its absence. Due to its complexity (Michlmayr, 2007) solutions proposed to SQRP typically limit to special cases.

Hyper-Heuristic_AdaNAS (HH_AdaNAS) is an adaptive metaheuristic algorithm, which resolves SQRP (Hernandez, 2010). This algorithm was created from AdaNAS (Gómez et al., 2010). The *high-level algorithm* is formed by HH_AdaNAS, which use as solution algorithm AdaNAS, that is inspired by an ant colony and the set of *low-level heuristics* are included in the algorithm called HH_TTL. The goal of hyperheuristic HH_TTL is to define by itself in real time, the most adequate values for time to live (TTL) parameter during the execution of the algorithm. The main difference between AdaNAS and HH_AdaNAS are: when applying the modification of the TTL and the calculation of the amount of TTL to be allocated. In the Figure 8 we show HH_AdaNAS is form by AdaNAS + HH_TTL.

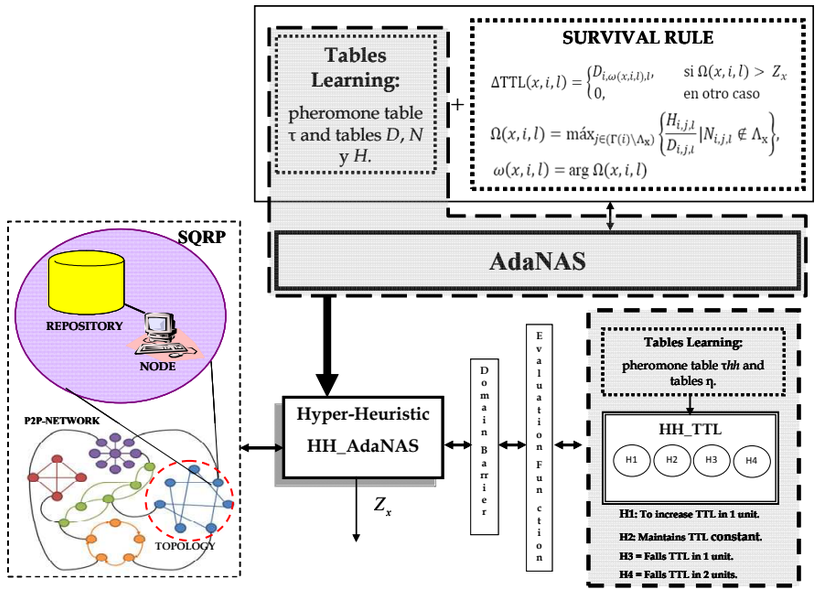


Fig. 8. HH_AdaNAS is form by AdaNAS + HH_TTL.

Data structures of HH_AdaNAS

HH_AdaNAS inherited some data structures of AdaNAS, as the pheromone table τ and the tables *H*, *D* and *N*. Besides the data structures of the high level metaheuristics, are the structures that help to select the low-level heuristic these are the pheromone table τ*hh* and the table hiperheuristic visibility states η. All the tables stored heuristic information or gained experience in the past. The relationship of these structures is shown in Figure 9.

When HH_AdaNAS searches for the next node, in the routing process of the query, is based on the pheromone table τ and tables *D*, *N* y *H*; these tables are intended to give information on distant *D*, *H* is a table that records the successes of past queries and number of documents *N* which are the closest nodes that can satisfy the query. In the same way, when HH_TTL chooses the following low level heuristic, through data structures τ*hh* and η. The memory is composed of two data structures that store information of prior consultations. The first of these memories is the pheromone table τ*hh* which has three dimensions, and the other memory structure is the table hiper-heuristic visibility states η, which allows the hiper-

heuristic know in what state is SQRP. Is to say, if is necessary to add more TTL, because the amount of resources found are few and decreases the lifetime.

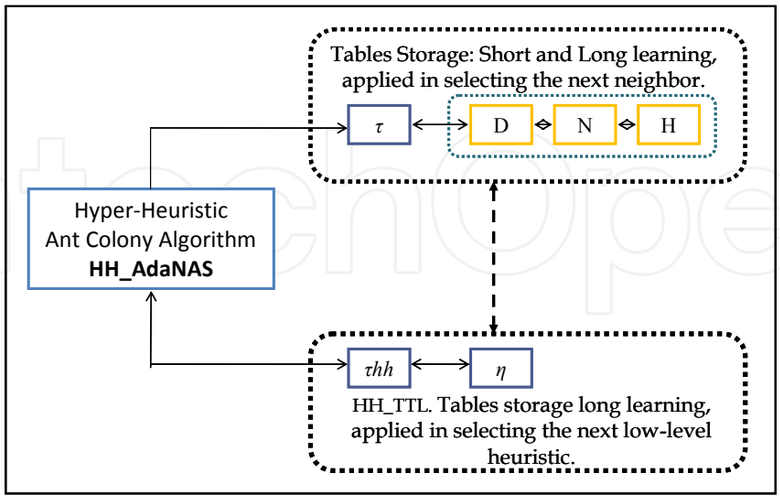


Fig. 9. Storage structures of HH_AdaNAS.

The pheromone table τ is divided into n two-dimensional tables, one corresponding to each node i of the network. These tables contain only entries for a node fixed i, therefore, its dimensions are at most $|L| \times |\Gamma(i)|$, where L is the dictionary, which defines the keywords allowed for consultation and $\Gamma(i)$ is the set of neighboring nodes of i. Each in turn contains a two-dimensional table $|m| \times |h|$, where m is the states visibility set of the problem and h is the available heuristics set. The pheromone table is also called learning structure long.

The visibility state table η expresses the weight of the relation between SQRP-states and TTL-heuristics and was inspired by the deterministic survival rule designed by Rivera (Rivera G. 2009). Table η is formed by the combination of $|m| \times |h|$, where a visibility state m_i is identified mainly by α, which depends on the node selected by AdaNAS to route the query SQRP. The variable α in Equation 6 contributes to ensure that the node selected by HH_AdaNAS, in the future, not decreases the performance of the algorithm. A TTL-heuristic is intelligently selected according with the past performance given by its pheromone value, and its visibility value, given by an expert. The Figure 10 shows the visibility state table used in this work.

	h_1	h_2	h_3	h_4
m_1	1	0.75	0.5	0.25
m_2	0.75	1	0.5	0.5
m_3	0.5	0.5	1	0.75
m_4	0.25	0.5	0.75	1

Fig. 10. Visibility state table

$$\alpha=(H_{i,j,l} / D_{i,j,l}) / Z_x \quad (6)$$

Where $H_{i,j,l}$ indicates the number of documents consistent with the query l, $D_{i,j,l}$ indicates the length of the route to obtain the documents, i represented the current node and j is the node chosen, and Z_x is a measure of current performance. In this work the visibility states are: m_1 = (α > 1)&(TTL < D)&(TTL != 1), m_2 = (α > 1)&(TTL < D)&(TTL = 1), m_3 = (H = 0) ||((α > 1)&(TTL ≥ D))|| ((α ≤ 1)&(TTL = 1)) and m_4 = (α ≤ 1)&(TTL > 1). All the visibility states are calculated to identify which heuristic will be applied to TTL.

5.2.2 Experimentation

The experimental environment used during experiments, and the results obtained are presented in this section. **Software:** Microsoft Windows 7 Home Premium; Java programming language, Java Platform, JDK 1.6; and integrated development, Eclipse 3.4. **Hardware:** Computer equipment with processor Intel (R) Core (TM) i5 CPU M430 2.27 GHz and RAM memory of 4 GB. **Instances:** It has 90 different SQRP instances; each of them consists of three files that represent the topology, queries and repositories. The description of the features can be found in (Cruz et al. 2008).

The average performance was studied by computing three performance measures of each 100 queries: **Average hops**, defined as the average amount of links traveled by a Forward Ant until its death that is, reaching either the maximum amount of results required or running out of TTL. **Average hits**, defined as the average number of resources found by each Forward Ant until its death, and **Average efficiency**, defined as the average of resources found per traversed edge (hits/hops). The initial Configuration of HH_AdaNAS is shown in Table 2. The parameter values were based on values suggested of the literature as (Dorigo & Stützle, 2004; Michlmayr, 2007; Aguirre, 2008 and Rivera, 2009).

In this section we show experimentally that HH_AdaNAS algorithm outperforms the AdaNAS algorithm. Also HH_AdaNAS outperforms NAS (Aguirre, 2008), SemAnt (Michlmayr, 2007) and random walk algorithms (Cruz et al., 2008), this was reported in (Gómez et al., 2010), so HH_AdaNAS algorithm is positioned as the best of them.

Parameter	Description	Value
τ_0	Pheromone table initialization	0.009
D_0	Distance table initialization	999
ρ	Local pheromone evaporation factor	0.35
β_1	Intensification of local measurements (degree and distance)	2.0
β_2	Intensification of pheromone trail	1.0
q	Relative importance between exploration and Exploitation	0.65
W_h	Relative importance of the hits and hops in the increment rule	0.5
W_{deg}	Degree's influence in the selection the next node	2.0
W_{dist}	Distance's influence in the selection the next node	1.0
TTL_{inic}	Initial Time To Live of the Forward Ants	10

Table 2. Shows the assignment of values for each HH_AdaNAS parameter.

In this experiment, we compare the HH_AdaNAS and AdaNAS algorithms. The performance achieved is measured by the rate of found documents and the experiments were conducted under equal conditions, so each algorithm was run 30 times per instance and used the same configuration parameters for the two algorithms, which is described in Table 2.

The Figure 11 shows the average efficiency performed during a set of queries with HH_AdaNAS and AdaNAS algorithms; for the two algorithms the behavior is approximately the same. The algorithm HH_AdaNAS at the beginning the efficiency is around 2.38 hits per hop in the first 100 queries and the algorithm AdaNAS start approximately at 2.37 hits per query also in the top 100 queries. Analyzing at another example of the experiment, after processing the 11 000 queries at the end the efficiency increases around 3.31 hits per hop for the algorithm HH_AdaNAS and the algorithm AdaNAS at 3.21 hits per query. Finally, due to the result we conclude that HH_AdaNAS achieves a final improvement in performance of 28.09%, while AdaNAS reaches an improvement of 26.16%.

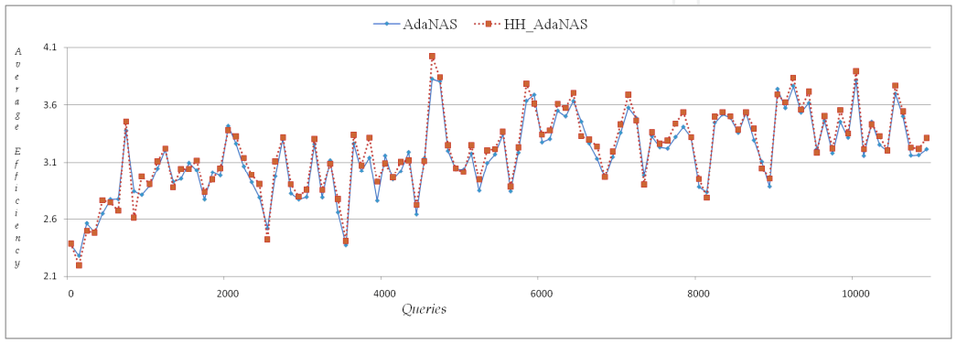


Fig. 11. The average efficiency performed during 11,000 queries with two algorithms.

6. Hybrid systems of metaheuristics: an approximate solution of ASP

The majority of problems related with ASP have a high level of complexity, according to application domains. An alternative solution is the use of Hybrid Systems based on Heuristics and Metaheuristics. Algorithm selection has attracted the attention of some research in hybrid intelligent systems, for which many algorithms and large datasets are available. Hybrid Intelligent Systems seek to take advantage of the synergy between various intelligent techniques in solving real problems (Ludermir et al., 2011).

6.1 Relation of meta-learning and hybridization

Although some algorithms based on Hybrid Systems of Metaheuristics are better than others on average, there is rarely a best algorithm for a given problem according to the complexity and application domain related with the proposal solution. Instead, it is often the case that different algorithms perform well on different problem instances. This condition is most accentuated among algorithms for solving NP-Hard problems, because runtimes of these algorithms are often highly variable from instance to instance.

When algorithms present high runtime variability, one is faced with the problem of deciding which algorithm to use. Rice called this the "algorithm selection problem" (Rice, 1976). The algorithm selection has not received widespread attention. The most common approach to algorithm selection has been to measure the performance of different algorithms on a given instances set with certain distribution, and then select the algorithm with the lowest average runtime.

This "winner-take-all" approach has produced recent and important advances in algorithm design and refinement, but has caused the rejection of many algorithms that has an excellent performance on an specific cases, but result uncompetitive on average. The following two questions emerge from the literature (Leyton-Brown, 2003). How to perform an algorithm selection for a given instance? How to evaluate novel hybrid algorithms?

a. Algorithms with high average running times can be combined to form a hybrid algorithm more robust and with low average running time when the algorithm inputs are sufficiently easy and uncorrelated.
b. New hybrid algorithm design should find more robust solution and focus on problems on which a single algorithm performs poorly.
c. A portfolio of algorithms can also be integrated through the use of hybrid algorithms because the solutions are considering more innovative.

In previous section we use machine learning algorithms to automatically acquire knowledge for algorithm selection, leading to a reduced need for experts and a potential improvement of performance. In general, the algorithm selection problem can be treated via meta-learning approaches. The results of this approach can cause an important impact on hybridization. In order to clarify this point, is important to speculate about how the empirical results of meta-learning can be analyzed from a theoretical perspective with different intentions:

a. Confirm the sense of the selection rules
b. Generate insights into algorithm behavior that can be used to refine the algorithms.

The acquired knowledge is confirmed when the performance of the refined algorithms is evaluated. The knowledge can be used to integrate complementary strategies in a hybrid algorithm.

6.2 Use of hybridization to solve ASP in social domains

The principal advanced in the reduction of Complexity is related with the amalgam of different perspectives established on different techniques which to demonstrate their efficiency in different application domains with good results.

Hybridization of Algorithms is one of the most adequate ways to try to improve and solve different ASP related with the optimization of time. Many applied ASP´s have an impact on social domains specially to solve dynamic and complex models related with human behavior. In (Araiza, 2011) is possible analyze with a Multiagents System the concept of "Social Isolation", featuring this behavior on the time according with interchanges related with a minority and the associated health effects, when this occurs.

In addition, is possible specify the deep and impact of a viral marketing campaign using a Social Model related with Online Social Networking. In (Azpeitia, 2011), an adequate ASP determines the way on the future of this campaign and permits analyze the track of this to understand their best features.

6.3 Future trends on the resolution of ASP using a hybrid system of metaheuristics

We expected that the future trends for solving ASP with hybridization will be based on models that tend to perform activities according to a selection framework and a dynamic

contextual area. The decision of the most appropriate actions requires advanced Artificial Intelligence Technique to satisfy a plethora of application domains in which interaction and conclusive results are needed. This only is possible with Intelligent Systems equipped with high processing speed, knowledge bases and an innovative model for designing experiments, something will happen in this decade.

7. Conclusions

Many real world problems belong to a special class of problems called NP-hard, which means that there are no known efficient algorithms to solve them exactly in the worst case. The specialized literature offers a variety of heuristic algorithms, which have shown satisfactory performance. However, despite the efforts of the scientific community in developing new strategies, to date, there is no an algorithm that is the best for all possible situations. The design of appropriate algorithms to specific conditions is often the best option. In consequence, several approaches have emerged to deal with the algorithm selection problem. We review hyper-heuristics and meta-learning; both related and promising approaches.

Meta-learning, through machine learning methods like clustering and classification, is a well-established approach of selecting algorithms, particularity to solve hard optimization problems. Despite this, comparisons and evaluations of machine learning methods to build algorithm selector is not a common practice. We compared three machine learning techniques for algorithm selection on standard data sets. The experimental results revealed in general, a high performance with respect to a random algorithm selector, but low perform with respect to other classification tasks. We identified that the Self-Organising Neural Network is a promising method for selection; it could reaches 100% of accuracy when feedback was incorporated and the number of problem characteristics was the minimum.

On the other hand, hyper-heuristics offers a general framework to design algorithms that ideally can select and generate heuristics adapted to a particular problem instance. We use this approach to automatically select, among basic-heuristics, the most promising to adjust a parameter control of an Ant Colony Optimization algorithm for routing messages. The adaptive parameter tuning with hyper-heuristics is a recent open research.

In order to get a bigger picture of the algorithm performance we need to know them in depth. However, most of the algorithmic performance studies have focused exclusively on identifying sets of instances of different degrees of difficulty; in reducing the time needed to resolve these cases and reduce the solution errors; in many cases following the strategy "the -winner takes-all". Although these are important goals, most approaches have been quite particular. In that sense, statistical methods and machine learning will be an important element to build performance models for understanding the relationship between the characteristics of optimization problems, the search space that defines the behavior of algorithms that solve, and the final performance achieved by these algorithms. We envision that the knowledge gained, in addition to supporting the growth of the area, will be useful to automate the selection of algorithms and refine algorithms; hiper-heuristics, hybridization, and meta-learning go in the same direction and can complement each other.

8. Acknowledgment

This research was supported in part by CONACYT and DGEST

9. References

Ali, S. & Smith, K. (2006). On learning algorithm selection for classification. *Applied Soft Computing*, Vol. 6, No. 2, (January 2006), pp. 119-38.

Aguirre, M. (2008). *Algoritmo de Búsqueda Semántica para Redes P2P Complejas*. Master's thesis, División de Estudio de Posgrado e Investigación del Instituto Tecnológico de Ciudad Madero, Tamaulipas, México.

Azpeitia, D. (2011). Critical Factors for Success of a Viral Marketing Campaign of Real-Estate Sector at Facebook: The strength of weak learnability. *Proceedings of the HIS Workshop at MICAI*

Beck, J. & Freuder, E. (2004). Simple Rules for Low-Knowledge Algorithm Selection. *Proceedings of the 1st International Conference on Integration of IA and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, Nice, France, April 2004, J. Regin and M. Rueher (Ed.). Springer-Verlag Vol. 3011, pp. 50-64.

Brazdil, P. B., Soares C., & Pinto, D. C. J. (2003). Ranking Learning Algorithms: Using IBL and Meta-Learning on Accuracy and Time Results. *Machine Learning*, Vol. 50, No. 3, pp. 251–277, ISSN: 08856125

Brewer, E. (1995). High-Level Optimization Via Automated Statistical Modeling. *Proceedings of Principles and Practice of Parallel Programming*, Santa Barbara, CA, July 1995, ACM Press, New York, USA, pp. 80-91

Burke, E., Hyde, M., Kendall, G., Ochoa, G., Özcan, E. & Woodward, J. (2009). Exploring hyper-heuristic methodologies with genetic programming. In: *Computational Intelligence*: Collaboration, Fusion and Emergence, Intelligent Systems Reference Library

Burke, K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E. & Woodward, R. (2010). A Classification of Hyper-heuristic Approaches, In: *International Series in Operations Research & Management Science*, Gendreau, M. and Potvin, J.Y. pp.(449). Springer Science+Business Media, ISBN 978-1-4419-1663-1, NY, USA

Cai, H., Hu X., Lü Q., & Cao, Q. (2009). A novel intelligent service selection algorithm and application for ubiquitous web services environment. *Expert Systems with Applications*, Vol. 36, No. 2, Part 1, pp. 2200-2212, ISSN: 09574174

Cruz, L. (1999). *Automatización del Diseño de la Fragmentación Vertical y Ubicación en Bases de Datos Distribuidas usando Métodos Heurísticos y Exactos*. Master's thesis, Instituto Tecnológico y de Estudios Superiores de Monterrey, México.

Cruz, L., Gómez, C., Aguirre, M., Schaeffer, S., Turrubiates, T., Ortega, R. & Fraire,H.(2008). NAS algorithm for semantic query routing systems in complex networks. In: *International Symposium on Distributed Computing and Artificial Intelligence 2008/ Advances in Soft Computing 2009*. Corchado J., Rodríguez S., Llinas J. & Molina J., pp. (284-292), Springer, Berlin /Heidelberg, ISBN 978-3-540-85862-1, DOI 10.1007/978-3-540-85863-8

Czogalla, J. & Fink, A. (2009). Fitness Landscape Analysis for the Resource Constrained Project Scheduling Problem. *Lecture Notes in Computer Science, Learning and Intelligent Optimization*, Vol. 5851, pp. 104-118

Dorigo, M. & Stützle, T. (2004). *Ant Colony Optimization*. MIT Press, Cambridge, MA., ISBN 0-262-04219-3, EUA

Duarte, A., Pantrigo, J., Gallego, M. (2007). *Metaheurísticas,* Ed. Dykinson S.L. España

Ducatelle, F., & Levine, J. (2001). Ant Colony Optimisation for Bin Packing and Cutting Stock Problems. *Proceedings* of the UK Workshop on Computational Intelligence, Edinburgh

Fink, E. (1998). How to solve it automatically: Selection among Problem-Solving methods. *Proceedings of ICAPS 1998*, pp. 128–136

Frost, D.; Rish, I. & Vila, L. (1997). Summarizing CSP hardness with continuous probability distributions. *Proceedings of the 14th National Conference on AI*, American Association for Artificial Intelligence, pp. 327-333

Garrido, P. & Castro C. (2009). Stable solving of cvrps using hyperheuristics. *Proceedings of the 11th Annual conference on Genetic and evolutionary computation (GECCO'09)*, ACM, Montreal, Canada, July 2009

Gent, I.; Macintyre, E.; Prosser, P. & Walsh, T. (1997). The Scaling of Search Cost. In: *Proceedings of the fourteenth national conference on artificial intelligence and ninth conference on Innovative applications of artificial intelligence,* pp. 315-320, AAI Press, Retrieved from: https://www.aaai.org/Papers/AAAI/1997/AAAI97-049.pdf

Gómez, C.G., Cruz, L., Meza, E., Schaeffer, E. & Castilla, G.(2010). A Self-Adaptive Ant Colony System for Semantic Query Routing Problem in P2P Networks. *Computación y Sistemas* Vol. 13, No. 4, pp (433-448), ISSN 1405-5546

Guo, H. & Hsu, W. (2004). A Learning-based Algorithm Selection Meta-reasoner for the Real-time MPE Problem. *Proceedings of the 17th Australian Joint Conference on Artificial Intelligence*, Cairns, Australian, Dec 2004, G. I. Webb and Xinghuo Yu (Ed.), Springer-Verlag Vol. 3339, pp. 307-318

Hernández P. (2010). *Método Adaptativo para el Ajuste de Parámetros de un Algoritmo Evolutivo Hiperheurístico*. Master's thesis, División de Estudio de Posgrado e Investigación del Instituto Tecnológico de Ciudad Madero, Tamaulipas, México

Hilario, M., & Kalousis, A. (2000). Building algorithm profiles for prior model selection in knowledge discovery systems. *International Journal of Engineering Intelligent Systems for Electrical Engineering and Communications*, Vol. 8, No. 2, 2000, pp. 77-88, ISSN: 09691170

Hollander, M. & Wolfe, D. (1973). *Non-parametric Statistical Methods*. John Wiley and Sons. New York, USA

Hoos, H. & Stutzle, T. (2000). Systematic vs. Local Search for SAT. *Journal of Automated Reasoning*, Vol. 24, pp. 421-481

Houstis, E.; Catlin, A. & Rice, J. (2002). PYTHIA-II: A Knowledge/Database System for Managing Performance Data and Recommending Scientific Software, ACM Transactions on Mathematical Software (TOMS) - Special issue in honor of John Rice's 65th birthday, Vol. 26, No. 2, (June 2000)

Hutter, F.; Hamadi, Y.; Hoos, H. & Leyton-Brown, K. (2006). Performance prediction and automated tuning of randomized and parametric algorithms. *Lecture Notes in Computer Science, Principles and Practice of Constraint Programming*, Vol. 4204, pp. 213-228

Kadioglu, S., Malitsky, Y., Sabharwal, A., Samulowitz, H., & Sellmann, M. (2011). Algorithm Selection and Scheduling, *Proceedings of the 17th International Conference on Principles and Practice of Constraint Programming (CP2011)*, Italy, September 2011

Kalousis, A., & Theoharis, T. (1999). NOEMON: Design, implementation and performance results of an intelligent assistant for classifier selection, *Intelligent Data Analysis*, Vol. 3, No. 5, pp. 319-337, ISSN: 1088467X

Kotthoff, L.; Gent, I. & Miguel I. (2011). A Preliminary Evaluation of Machine Learning in Algorithm Selection for Search Problems. In: *AAAI Publications, Fourth International Symposium on Combinatorial Search (SoCS)*, Borrajo, Daniel and Likhachev, Maxim and López, Carlos Linare, pp. 84-91, AAAI Press, Retrieved from: http://www.aaai.org/ocs/index.php/SOCS/SOCS11/paper/view/4006

Lagoudakis, M. & Littman, M. (2000). Algorithm Selection Using Reinforcement Learning. *Proceedings of the Sixteenth International Conference on Machine Learning*. P. Langley (Ed.), AAAI Press, pp. 511–518

Lagoudakis, M. & Littman, M. (2001). Learning to select branching rules in the dpll procedure for satisfiability. *Electronic Notes in Discrete Mathematics*, Vol. 9, (June 2001), pp. 344-359

Lawler, E.; Lenstra, J.; Rinnooy, K. & Schmoys, D. (1985). *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. John Wiley & Sons, New York, USA

Leyton-Brown, K.; Nudelman, E.; Andrew, G.; McFadden, J. & Shoham, Y. (2003). A portfolio approach to algorithm selection. *Proceedings of International joint conference on artificial intelligence*, Vol. 18, pp. 1542-3

Li, J.; Skjellum, A. & Falgout, R. (1997). A Poly-Algorithm for Parallel Dense Matrix Multiplication on Two-Dimensional Process Grid Topologies. *Concurrency, Practice and Experience*, Vol. 9, No. 5, pp. 345-389

Lobjois, L. & Lemâitre, M. (1998). Branch and bound algorithm selection by performance prediction. In: *AAAI '98/IAAI '98 Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, Jack Mostow, Charles Rich, Bruce Buchanan, pp. 353-358, AAAI Press, Retrieved from: http://www.aaai.org/Papers/AAAI/1998/AAAI98-050.pdf

Ludermir, T.B.; Ricardo B. C. Prudêncio, R.B.C; Zanchettin, C. (2011). Feature and algorithm selection with Hybrid Intelligent Techniques. *International Journal Hybrid Intelligent Systems* ,Vol. 8, No. 3, pp. 115-116

Madani, O.; Raghavan, H. & Jones, R. (2009). On the Empirical Complexity of Text Classification Problems. *SRI AI Center Technical Report*

Malitsky, Y., Sabharwal, A., Samulowitz, H., & Sellmann M. (2011). Non-Model-Based Algorithm Portfolios for SAT, *Proceedings of the 14th international conference on Theory and Applications of Satisfiability Testing*, Ann Arbor, June 2011

Messelis, T.; Haspeslagh, S.; Bilgin, B.; De Causmaecker, P. & Vanden Berghe, G. (2009). Towards prediction of algorithm performance in real world optimization problems. *Proceedings of the 21st Benelux Conference on Artificial Intelligence*, BNAIC, Eindhoven, pp. 177-183

Michlmayr, E. (2007). *Ant Algorithms for Self-Organization in Social Networks*. PhD thesis, Women's Postgraduate College for Internet Technologies (WIT), Vienna, Austria

Nascimento, A. C. A., Prudencio, R. B. C., Costa, I. G., & de Souto, M. C. P. (2009). Mining rules for the automatic selection process of clustering methods applied to cancer gene expression data, *Proceedings of 19th International Conference on Artificial Neural Networks (ICANN)*, Cyprus, September 2009

Nikolić, M.; Marić, F. & Janičić, P. (2009). Instance-Based Selection of Policies for SAT Solvers. *Lecture Notes in Computer Science, Theory and Applications of Satisfiability Testing*, Vol. 5584, pp. 326-340

O'Mahony, E., Hebrard, E., Holland, A., Nugent, C., & O'Sullivan, B. (2009). Using Case-based Reasoning in an Algorithm Portfolio for Constraint Solving. (2008).

Proceedings of The 19th Irish Conference on Artificial Intelligence and Cognitive Science, Ireland, August 2008

Pérez, O.J., Pazos, R.A., Frausto, J., Rodríguez, G., Romero, D., Cruz, L. (2004). A Statistical Approach for Algorithm Selection. *Lectures Notes in Computer Science*, Vol. 3059, (May 2004) pp. 417-431, ISSN: 0302-9743

Pérez, J., Pazos, R.A., Vélez, L. Rodríguez, G. (2002). Automatic Generation of Control Parameters for the Threshold Accepting Algorithm. *Lectures Notes in Computer Science*, Vol. 2313, pp. 119-127

Quiroz, M. (2009). *Caracterización de Factores de Desempeño de Algoritmos de Solución de BPP*. Master´s thesis, Instituto Tecnológico de Cd. Madero, Tamaulipas, México

Reeves, C. (1993). *Modern heuristic techniques for combinatorial problems*. John Wiley & Sons, ISBN: 0-470-22079-1, New York, USA

Rice, J. R. (1976). The algorithm selection problem. *Advances in Computers*, Vol. 15, pp. 65-118

Rice, J.R. (1968). On the Construction of Poly-algorithms for Automatic Numerical Analysis. *Interactive System for Experimental Applied Mathematics*, M. Klerer & J. Reinfelds, (Ed.) Academic Press, Burlington, MA, pp. 301-313

Ruiz-Vanoye, J. A., Pérez, J., Pazos, R. A., Zarate, J. A., Díaz-Parra, O., & Zavala-Díaz, J. C. (2009). Statistical Complexity Indicators Applied to the Vehicle Routing Problem with Time Windows for Discriminate Appropriately the Best Algorithm, *Journal of Computer Science and Software Technology*, Vol. 2, No. 2, ISSN: 0974-3898

Samulowitz, H. & Memisevic, R. (2007). Learning to solve QBF. In: *AAAI-07*, pp. 255-260, retrieved from: https://www.aaai.org/Papers/AAAI/2007/AAAI07-039.pdf

Schiavinotto, T. & Stützle, T. (2007). A review of metrics on permutations for search landscape analysis. *Computers & Operations Research*, Vol. 34, No. 10, (October 2007), pp. 3143-3153

Silverthorn, B. & Miikkulainen, R. (2010). Latent Class Models for Algorithm Portfolio Methods. *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*

Smith-Miles, K. & Lopes, L. (2012). Measuring instance difficulty for combinatorial optimization problems. *Computers & Operations Research*, in press (accepted 6/7/11)

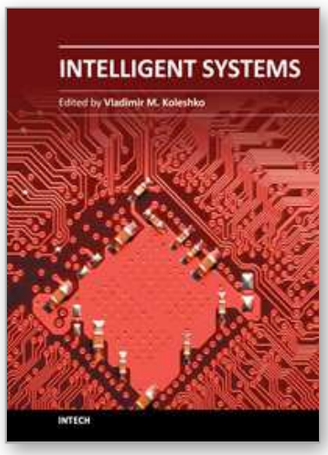
Smith-Miles, K.; James, R.; Giffin, J. & Tu, Y. (2009). Understanding the relationship between scheduling problem structure and heuristic performance using knowledge discovery. In: *Learning and Intelligent Optimization,* LION-3, Vol. 3, Available from: lion.disi.unitn.it/intelligent-optimization/LION3/online_proceedings/35.pdf

Soares, C. & Pinto, J. (2003). Ranking Learning Algorithms: Using IBL and Meta-Learning on Accuracy and Time Results. *Machine Learning*, Vol. 50, No. 3, pp. 251-277

Streeter, M; Golovin, D. & Smith, S. F. (2007). Combining multiple heuristics online. In: *AAAI 2007, Proceedings of the 22nd national conference on Artificial intelligence*, Vol. 22, Anthony Cohn, pp. 1197-1203, AAAI Press, Retrieved from: http://www.aaai.org/Papers/AAAI/2007/AAAI07-190.pdf

Wolpert, D. H. & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, pp. 67–82

Xu, L.; Hutter, F.; Hoos, H. & Leyton-Brown, K. (2009). SATzilla2009: An automatic algorithm portfolio for SAT. In: *Solver description, 2009 SAT Competition*



Intelligent Systems

Edited by Prof. Vladimir M. Koleshko

ISBN 978-953-51-0054-6
Hard cover, 366 pages
Publisher InTech
Published online 02, March, 2012
Published in print edition March, 2012

This book is dedicated to intelligent systems of broad-spectrum application, such as personal and social biosafety or use of intelligent sensory micro-nanosystems such as "e-nose", "e-tongue" and "e-eye". In addition to that, effective acquiring information, knowledge management and improved knowledge transfer in any media, as well as modeling its information content using meta-and hyper heuristics and semantic reasoning all benefit from the systems covered in this book. Intelligent systems can also be applied in education and generating the intelligent distributed eLearning architecture, as well as in a large number of technical fields, such as industrial design, manufacturing and utilization, e.g., in precision agriculture, cartography, electric power distribution systems, intelligent building management systems, drilling operations etc. Furthermore, decision making using fuzzy logic models, computational recognition of comprehension uncertainty and the joint synthesis of goals and means of intelligent behavior biosystems, as well as diagnostic and human support in the healthcare environment have also been made easier.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen