Bayesian Networks to Predict Data Mining Algorithm Behavior in Ubiquitous Computing Environments

Aysegul Cayci, Santiago Eibe, Ernestina Menasalvas, and Yucel Saygin*

Sabanci University, Istanbul, Turkey,
Facultad de Informatica, Universidad Politecnica, Madrid, Spain
aysegulcayci@su.sabanciuniv.edu,
{seibe,emenasalvas}@fi.upm.es,
ysaygin@sabanciuniv.edu

Abstract. The growing demand of data mining services for ubiquitous computing environments necessitates deployment of appropriate mechanisms that make use of circumstantial factors to adapt the data mining behavior. Despite the efforts and results so far for efficient parameter tuning, incorporating dynamically changing context information on the parameter setting decision is lacking in the present work. Thus, Bayesian networks are used to learn, in possible situations the effects of data mining algorithm parameters on the final model obtained. Based on this knowledge, we propose to infer future algorithm configurations appropriate for situations. Instantiation of the approach for association rules is also shown in the paper and the feasibility of the approach is validated by the experimentation.

Keywords: automatic data mining, data mining configuration, ubiquitous data mining.

1 Introduction

Ubiquitous computing, being an immature computing paradigm, brings new challenges to software designers and also to the designers of data mining software. In ubiquitous computing, processing takes place on the restricted resource devices that are embedded or spread in the environment and moreover the context in which the devices are used is subject to change. An important implication of ubiquitous computing is the lack of expert involvement on tuning the software where expert knowledge is most needed due to the scarcity of resources and the variability of the context. Consequently, automatic configuration of software by considering the changing context and constrained resources, is essential.

There is an increasing demand for intelligent applications on ubiquitous devices while data mining methods have been the main way to provide such intelligence. Nevertheless, important challenges need to be addressed on the ubiquitous

* This work has been partially financed by Spanish Ministry of Innovation under project TIN2008-05924.

M. Atzmueller et al. (Eds.): MUSE/MSM 2010, LNAI 6904, pp. 119–141, 2011.

data mining design, which two of them will be focused on in this paper. First of all, circumstantial factors such as the context and the resource limitations of the device should be considered when deciding how to configure the data mining algorithm. Secondly, there is a need to develop methods for the autonomous and adaptable execution of data mining. A number of context-aware and resource-aware data mining approaches have been proposed in the literature to deal with the issues posed due to ubiquity ([11] [12]). The proposed approaches consider the current context and/or resource availability to adjust parameters of data mining process or to determine the configuration setting of data mining in order to make autonomous decisions. However, in these approaches, knowledge from the past experiences is not incorporated into the decision mechanism of the parameter setting. As a result, settings are not adaptable in the sense that they do not improve over time based on past experience. A mechanism that adapts the data mining algorithm's configuration setting decisions according to the experiences learned, is lacking. In order to fulfill this deficiency, we propose to use machine learning techniques for deciding parameter settings in a given situation according to past behavior of the algorithm.

Automatic parameter tuning research area has gained much interest in the recent years. Several studies have been published offering optimization and machine learning techniques to solve the problem. The main idea behind the optimization techniques is to determine the performance criteria to be optimized and the configuration that best satisfies this criteria. Optimization methods proposed for automatic parameter tuning are as follows: racing algorithm by Birattari et al ([4]); iterated local search approach by Hutter, Hoos and Stutzle ([15]); algorithm portfolios paradigm by Gagliolo and Schmidhuber ([10]); experimental design combined with local search by Diaz and Laguna ([9]). Other prominent technique proposed for automatic parameter tuning is based on machine learning classifiers. In general terms, classifiers are used to learn the parameters to set the configuration. Srivastava and Mediratta ([20]) suggest usage of decision trees for automatic tuning of search algorithms. Through classification of previous runs of the algorithm by means of Bayesian network, Pavon, Diaz and Luzon ([18]) have automatized the parameter tuning process.

Interest on automatic parameter tuning originates in alleviating configuration setting of algorithms with a plethora of parameters most of the time but not to provide autonomy. In general, the argument of current work on automatic parameter setting is to find a configuration regardless of the circumstances. In the current proposed methods, neither the state of the device nor the requirements of the current situation are considered. However, in ubiquitous computing environments, state of the device and the current situation are important factors to determine the appropriate parameter settings and to make the device behave autonomously under any circumstance. In our mechanism, we consider the relationship between situations and parameters. Specifically, context in which the device is in when data mining request is received and the availability of the resources are incorporated in the parameter setting decision.

Cao, Gorodetsky and Mitkas ([6]) discuss the contribution of data mining to agent intelligence. They argue that a combination of autonomous agents with data mining supplied knowledge provides adaptability whereas knowledge acquisition with data mining for adaptability relies on past data (past decisions, actions, and so on). Our approach to provide adaptability is similar: we use machine learning approach in order to generate adaptable parameter setting decisions and enhance ubiquitous data mining with autonomy and adaptability. Our mechanism is based on Bayesian networks because Bayesian networks enable (1) the finding of the probabilistic relationships between the circumstances, parameters, and performance criteria, (2) considering several factors rather than a single criteria when determining the setting and (3) adaptability by learning from the past experiences. Pavon, Diaz, Laza and Luzon also proposed Bayesian networks for parameter tuning in [18]. The innovative feature of our mechanism is that we use not only information on parameters but also information on context and resources (what we call circumstances) to discover the appropriate configuration of a data mining algorithm for a given situation. When determining the best configuration, both efficiency of the data mining process and efficacy of the final model are taken into account.

The rest of the paper is organized as follows: Section 2, presents the proposed approach whereas section 3 instantiates it for a case. In Section 4, we explain the experiment that we performed in order to validate of the proposed approach and finally, section 5 is the conclusion.

2 Automatic Configuration for Ubiquitous Data Mining

We present a mechanism to predict the appropriate settings of a data mining algorithm's parameters in a resource-aware and context-aware manner. The mechanism is based on learning from past experiences, that is, learning from the past executions of the algorithm in order to improve the future decisions.

2.1 Analysis of the Problem

Our goal is to configure automatically a data mining algorithm which will run on a ubiquitous device. Since circumstantial factors such as the conditions of the resources and the context in which the device is used are important in a ubiquitous computing environment, availability of the knowledge on the following is useful for determining the algorithm's appropriate configuration:

- the resources that the algorithm needs in order to accomplish its task,
- the algorithm parameters that have an effect on the resource usage or on the data model quality,
- the context features which may have an effect on the efficacy of the data mining model or the efficiency of data mining,
- the features of the mining data set,
- the quality indicators which show the efficacy of the data mining model and efficiency of the data mining.

On the other hand, the problem that we tackle also implies the solution to address an important issue which is to change or improve the configuration setting decisions as the circumstances change. That means that, automatically generated configuration decisions must be adapted to the changing conditions just like a data miner expert who adapts his decisions when the conditions change.

Next, we define the factors for configuration that we derive from the items outlined above, and then present our solution to the problem after briefly introducing the Bayesian networks which we use in our proposed mechanism.

2.2 Problem Definition

When deciding how to set the parameters of an algorithm for a specific run, we determined that in a ubiquitous computing environment, circumstantial factors should be taken into account as well as the required quality. For this reason, we grouped the relevant factors for the configuration as circumstance and quality. Conditions of the device's resources and information on context belong to circumstance factor whereas level of efficiency required for the data mining process and data mining model expected quality level are quality factors. Formal definition of the algorithm configuration and relevant factors are as follows:

Circumstance: Circumstance, denoted by C, is defined by a set of ordered pairs *(f,s)* where f is either a resource or context feature and s is the state of this feature.

Quality: Quality, denoted by Q, is defined by a set of ordered pairs *(q,l)* where q is a quality feature and l is the required level for this quality. Quality features are metrics of efficiency or efficacy of the algorithm.

Parameters: Configuration of the algorithm, denoted by P, is defined by a set of ordered pairs *(p,v)* where p stands for a parameter and v is the value it takes.

In this way, we covered all but the "features of the mining data set" outlined in problem analysis (subsection 2.1). We deliberately disregarded the effect of mining data set features on the configuration decision for the time being because we want to focus on the ubiquitous aspect in this work.

Automatic configuration for ubiquitous data mining: Let Q be the required quality and C be the circumstance sensed or observed, then the configuration of data mining algorithm P is obtained by the mapping:

$$f : C \times Q \rightarrow P$$

We propose to use Bayesian networks to discover configuration of a data mining algorithm (P), aiming to attain the requested quality (Q), for the circumstance (C) observed when a data mining request is issued.

2.3 Bayesian Networks

Bayesian networks which represent the joint probability distributions for a set of domain variables are proved to be useful as a method of reasoning in several research areas. Medical diagnosis([3]), language understanding ([7]), network fault detection([14]) and ecology([2]) are just a few of the diverse number of application areas where Bayesian network modeling is exploited.

A Bayesian network is a directed acyclic graph that shows the conditional dependencies between domain variables and may also be used to illustrate graphically the probabilistic causal relationships among domain variables. The nodes of the network represent the domain variables and an arc between two nodes (parent and child) indicates the existence of dependency among these two nodes. Conditional probabilities of the dependencies among each variable and its parents are also represented along with Bayesian networks. The joint probability of instantiated n variables (i.e. variable x_i has an assigned value) in a Bayesian network is computed by:

$$P(x_1, ..., x_n) = \prod_{i=1}^{n} P(x_i | parents(X_i)) \tag{1}$$

where $parents(X_i)$ denotes the instantiated parents of the node of variable X_i.

Bayesian networks which are convenient for probabilistic inferencing, are commonly used for predicting the values of the subset of variables given the values of observed variables. In depth knowledge on Bayesian networks can be found in [19].

Learning the Bayesian network structure rather than creating the structure by analyzing the dependencies of domain variables, is a field of research which was studied extensively. Algorithms that learn the structure are most useful when there is a need to construct a complex network structure or when domain knowledge does not exist as in a ubiquitous computing environment. A discussion of the literature can be found in [5].

2.4 Behavior Model of the Data Mining Algorithm

In our solution, we propose to create a model that shows under possible circumstances what is the quality obtained against possible configurations of the data mining algorithm's execution. We call this model the *behavior model of the data mining algorithm* since the model represents the algorithm's behavior against different configurations and circumstances. We use machine learning as the main mechanism for creating the behavior model, in particular, we propose to construct a Bayesian network from *execution data* collected during algorithm's previous executions.

Execution Data. Execution data, denoted by E, consists of three types of attributes E=(C^E,Q^E,P^E) where the current circumstance just before the data

mining algorithm runs, is stored in the set of attributes $C^E = \{c_1, c_2, ..., c_n\}$, the quality detected at the end of algorithm's execution in $Q^E = \{q_1, q_2, ..., q_m\}$, and finally, the configuration of the current execution in $P^E = \{p_1, p_2, ..., p_k\}$.

The attribute sets of execution data, C^E,Q^E, and P^E are supersets of the domain sets of every possible C, Q, and P given in problem definition (subsection 2.2). Let C^{dom},Q^{dom}, and P^{dom} be the domain sets of C, Q, and P respectively, then $C^{dom} \subseteq C^E$, $Q^{dom} \subseteq Q^E$, and $P^{dom} \subseteq P^E$. For example, assume $C = \{(f_1, s_1), (f_2, s_2)\}$, then $C^{dom} = \{f_1, f_2\}$ and $f_1, f_2 \in C^E$. This means that, information about all kinds of circumstances and quality requirements that may occur and therefore should be taken into account for the configuration decisions, are collected during algorithm's execution and stored in execution data. Similarly, automatic setting of every parameter value is captured.

Each execution of the data mining algorithm creates an instance in E and thus a history of executions for the data mining algorithm is formed which will be used to construct the behavior model.

Behavior Model. A Bayesian network is learned from the execution data and, afterwards, this Bayesian network representing the behavior model, is used to predict the appropriate configurations for the algorithm. Fig. 1 illustrates Bayesian network construction steps that we propose. K2 algorithm proposed by Cooper and Herskovits([8]) was used when constructing the Bayesian network. Initial step of behavior model generation is to discretize the execution data since K2 assumes database variables to be discrete. K2 learns Bayesian network structure from database of cases (E in our case) by determining the most probable network structure B_s given E:

$$\max_{B_s}[P(B_s|E)] \tag{2}$$

We made use of the open source code of Weka software([13]) to construct the network and updated it to fit our needs. The original algorithm seeks relationships among all the variables. However, execution data has three groups of variables (C^E,Q^E,P^E) and the relationships among the variables within a group such as the relationship among circumstantial variables *location* and *memory available* are not interesting. For this reason, modification of the K2 algorithm is necessary to look for relationships among nodes belonging to different groups of variables. A level (lvl) is assigned to each group based on the possible cause-effect relationship between them. The levels are used to prevent the nodes in the lower level groups to be the parents of the nodes in the upper level groups. Let $V = \{v_{l,k} | l = 1, ..., lvl \text{ and } k = 1, ..., x_l\}$ be the set of nodes of Bayesian network where x_l is the number of nodes in level l and $v_{i,j}, v_{m,n} \in V$. Then, i) nodes $v_{i,j}$ and $v_{m,n}$ can not be related if $i = m$, ii) $v_{i,j}$ can be the immediate parent node of $v_{m,n}$ only if $m = i + 1$.

The Bayesian network that is constructed from past execution data represents the probabilistic relationship between circumstance states, discretized possible

parameter settings, and measured as well as discretized quality indicators. Appropriate setting of an algorithm's parameter is extracted from the Bayesian network as explained in the next section.

2.5 Mechanism to Predict Ubiquitous Data Mining Configuration

Once the behavior model is built, the steps that lead to automatic parameter configuration are as follows (See Fig. 1 for details):

- A data mining model is needed for a specific data set,
- Current circumstance (C) is observed and the quality requirements (Q) are acquired,
- Configuration (P) of the data mining algorithm is determined autonomously by inferencing from the behavior model

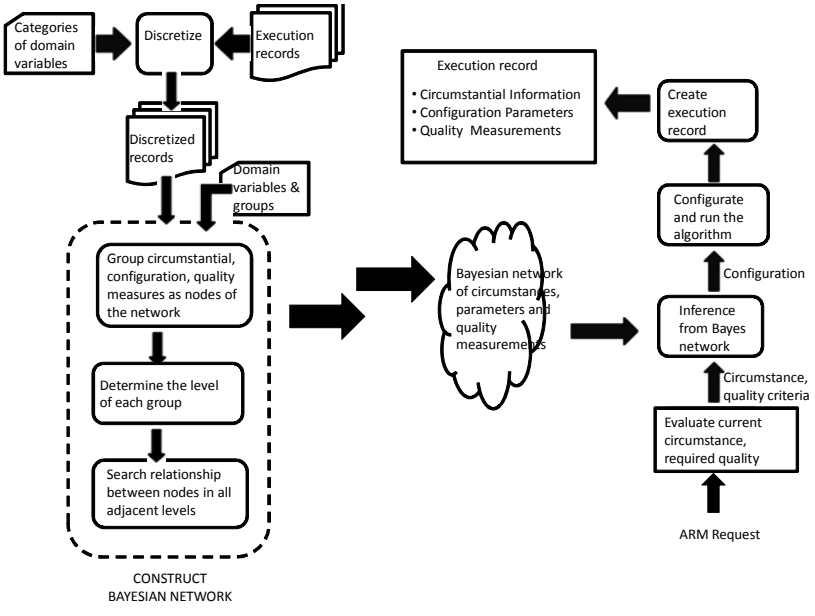

Fig. 1. Bayesian Network construction steps and data mining configuration mechanism

The most likely configuration is inferred from the behavior model by estimating the probabilities of possible parameter settings from previous runs of the algorithm in execution circumstances similar to current in order to obtain quality levels similar to the required. In particular, $p(x|F)$ which is the conditional probability of x (an instantiation of parameter variable) given F (instantiations of circumstance and quality variables), is evaluated. A pseudo code of this calculation is given below.

Pseudo code of parameter setting estimation from the Bayesian network

```
Definitions:
Let C be the set of circumstance sets Ck
     where (fkj,skj) is a feature, state pair in Ck,
            ck is the number of pairs in Ck.
    Qk is the corresponding quality set of Ck
     where (qkj,vkj) is a feature, state pair in Qk,
            nk is the number of pairs in Qk.
    P is the set of parameters sets Px
     where pxy is a parameter setting in Px,
           rx is the number of possible settings for Px.
Pseudo Code:
     for every Ck in C
      let F  be all fkj=skj (forall j <= ck) and qkl=vil (forall l <= nk)
          for every Px in P
              for every pxy in Px (forall y <= rx)
                     calculate Prxy = Probability (Px = pxy | F )
              pxy with highest Pr is the appropriate setting of Px for Ck.
```

3 Autonomous Data Mining for Museum Guide

In order to increase the comprehensibility of our approach, we illustrate its use by instantiation. For this purpose, we describe an example application where ubiquitous devices are used by many people performing the same activity and somehow sharing information on their activity among themselves. In this sense, they form a social network in the ambient intelligence environment that we delineate. We explain how our approach can be employed in this application and finally instantiate our approach with data specific to this example application.

3.1 Motivating Example: A Museum Equipped with Ambient Intelligence

The museum depicted in this example is huge so that it takes a lot of time of the visitors to see all the exhibitions in it and it may even be impossible within the visit time. Museum is conceived as an ambient intelligence environment where visitors are fed information from the environment. The aim is to guide the visitors so that they can visit the pieces (art work) that they would like to see within the available time rather than trying to see all of the exhibitions. For this purpose, an application that we call *museum guide* is loaded to the smartphones of the visitors upon request. *museum guide* directs a visitor to the pieces that he would like in the museum.

As in a common web based book or movie recommender system, the objective of this system is to discover common likes of users. In order to do that, the amount of time each visitor spends in front of a piece is sensed and collected. The amount of time spent in front of a piece is taken as the indication of that

the visitor's liking of that piece. Sensed data of each visitor is transformed to a record that we call *visitor record*. *visitor record* contains the pieces liked by that visitor.

museum guide which runs on the smartphone of a specific visitor while recording the pieces he liked to his smartphone, also downloads other *visitors' records*. Common likes are discovered by association rule mining of all *visitors' records* on the smartphones of the visitors. The purpose is to find the associations such as "visitors who liked Picasso's Three Musicians also liked Matisse's Dance". This museum has special offers for different types of visitors such as tourists, students and elderly. For this reason, visitor profile in one season, month or week of day may be quite different than the other. Since more associations can be found when similar people's likings are mined and the museum has dynamic visitor profile by day, rather than discovering the associations only once from data of one set of visitors, it is preferred to extract the association rules dynamically on visitors' smartphones. Moreover, continuous rotation of artwork also necessitates mining to be performed dynamically.

We focus on determining the configuration of the association rule mining algorithm which must run autonomously since in an ubiquitous computing environment it is assumed that the user can not provide this information. We exploit context and consider the availability of the resources of the visitor's smartphone as the determining factor of the association rule mining parameter settings. The data mining process is referred as *discover artwork associations*. Next, we describe a number of principles of *museum guide* and how it interacts with *discover artwork associations*:

- *museum guide* calls *discover artwork associations* to discover frequent itemsets from the set of *visitor records*. The attribute of each *visitor record* is the list of pieces liked by that visitor.
- *museum guide* downloads fresh data and calls *discover artwork associations* several times during his visit for a visitor in order to incorporate data of newcomers and to try different parameters for better recommendations.
- Association rule mining algorithm Apriori [1] which accepts two parameters: *minimum support* and *minimum confidence* is used to *discover artwork associations*.
- The resulting data mining model generated by Apriori are the association rules demonstrating which pieces that are exhibited in the museum are liked by the same persons. It is out of the scope of this work to speculate on how *museum guide* uses the model to recommend artwork to the visitor or whether the recommendations are ordered by physical location or some other criteria as well as when a new model is needed. On the other hand, we are concerned on the automatic configuration of *discover artwork associations*.
- Past executions of *discover artwork associations* are mined to discover the appropriate configuration that fulfils the required quality under a given circumstance. Initial past execution data is downloaded to the smartphone together with the application and is augmented by data collected during executions of Apriori locally on the visitor's smartphone.

3.2 Circumstantial Factors Effecting Parameter Setting

We argue that in an ubiquitous computing environment, in order to find appropriate settings of the data mining algorithm parameters, context from the environment as well as the conditions of the device's resources need to be utilized. Next, we define the relevant circumstantial factors for determining configuration of *discover artwork associations*.

Context: Context, that we assume have an effect on the required quality of the final model are:
- time left to the museum's closing (*remaining time to close*)
- time left to average visit time since the start of visit time (*remaining time to leave*)
- visitor's past attitude against the recommendations (*feedback*)
- number of visitors in the gallery entered (*no of visitors*)

Resources: Since *discover artwork associations* runs on the smartphones that are restricted resource devices, the resource usage of the data mining process need to be considered when setting the parameters of the data mining process. We assume memory and processor are the resources whose availability are critical for *discover artwork associations*:
- amount of memory available (*memory available*)
- processor idle percentage (*processor idle percent*)

3.3 Heuristics for Parameter Setting

In this subsection, we give example heuristic configuration decisions for *discover artwork associations*. We assume there exist default settings for each *discover artwork associations* parameter: *minimum support* and *minimum confidence* and in the configuration decision whether to increase or decrease the defaults of the relevant parameters depending on the circumstance is indicated. The following are some heuristics which may be used when configuring *discover artwork associations* autonomously where the reasoning explaining the heuristic follows it.

1. if *memory available* is low then increase the *minimum support* (with a higher *minimum support* value it is expected to decrease the size of the frequent itemsets and to optimize memory usage consequently),
2. if *remaining time to close* is small then increase both the *minimum confidence* and *minimum support* (since limited time is left, provide less rules with higher confidence so that the visitor will not miss the pieces that he would like most),
3. if *remaining time to leave* is small given that *memory available* is not low, decrease the *minimum support* (the objective is to make the average visit time longer by providing more pieces to the visitor that he would regret if he would leave without seeing them)

4. if *no of visitors* is high then decrease the *minimum support* but increase the *minimum confidence* (as the visitor may prefer to skip the pieces with a crowded audience in front of them, produce a list of high confidence containing sufficient number of pieces to bypass some of them)
5. if *feedback* is negative then decrease the *minimum support* (if the visitor is not satisfied with the previous recommendations then provide more accuracy)

Circumstance is the motive of each parameter setting but there is also an objective of each recommended setting which is given in parentheses after each heuristic. Objectives can be quantified by making use of the measurements obtained from the operating system of the device as well as the data mining model quality indicators.

Quality Measures: Quality measurements are the means to control whether the heuristic for a setting achieves the objective. The suggested quality measurements are as follows:
- maximum amount of memory used by *discover artwork associations* (*max memory usage*)
- number of association rules in the model (*no of rules discovered*)
- minimum confidence that an association rule in the model may have (*model min conf*)
- minimum support that association rules of the model should have (*model min support*)

Whether the objective of assigning certain value(s) to *discover artwork associations*'s parameter(s) is attained at a specific run of *discover artwork associations* can be assessed by checking the related quality measure(s) after the *discover artwork associations* runs with those settings. Similarly, the objective given is the required quality for a given circumstance.

3.4 Instantiation

In this subsection, automatic configuration setting is instantiated for the well known association rule mining algorithm Apriori. Instantiation is based on the intelligent museum example and consists of appropriate *discover artwork associations* configurations for the heuristic parameter setting decisions given in subsection 3.3 as well as the possible circumstances and quality determined in subsection 3.2 for the intelligent museum example.

Circumstance. It is assumed that context and resource availability values are discretized such that *'low'*, *'high'* and *'moderate'* categories are used for *memory available*, *'few'* and and *'many'* for *no of visitors*, *'not much'* and *'plenty'* are used for *remaining time to close* and *remaining time to leave* and finally, *'positive'* and *'negative'* are for *feedback*. Some possible instantiations of circumstances using discretized values are as follows:

C_1= {(memory available, 'low')}
C_2= {(remaining time to close, 'not much')}

C_3= {(memory available, 'high'),(remaining time to leave, 'not much')}
C_4= {(no of visitors, 'many')}
C_5= {(feedback, 'negative')}

Quality. Similarly, it is also assumed that quality measurement values are discretized such that *'low'*, *'high'* and *'moderate'* categories are used for *max memory usage*, *'few'* and and *'many'* for *no of rules discovered*, *'low'* and *'high'* are used for both *model min conf* and *model min support*. Some possible instantiations of circumstances using discretized values are as follows:
Q_1 ={(max memory usage, 'low')}
Q_2 ={(model min conf, 'high'), (model min support, 'high')}
Q_3 ={(no of rules discovered, 'many')}
Q_4 ={(model min conf, 'high'), (no of rules discovered, 'few')}
Q_5 ={(model min support, 'low')}

Algorithm Configuration. Assuming that the default values of *minimum support* and *minimum confidence* are 0.75 and 0.9 respectively, in each of the parameter setting below either one of them or both are altered to meet the parameter setting decisions of the example heuristics.
P_1 = {(minimum support, 0.9), (minimum confidence, 0.9)}
P_2 = {(minimum support, 0.85), (minimum confidence, 0.98)}
P_3 = {(minimum support, 0.75), (minimum confidence, 0.9)}
P_4 = {(minimum support, 0.5), (minimum confidence, 0.98)}
P_5 = {(minimum support, 0.70), (minimum confidence, 0.9)}

Configuration Decisions. Instantiation of configuration decisions are based on the instantiations of circumstance, quality criteria and algorithm configuration such that for each circumstance C_i given above, Q_i is the corresponding quality aimed for C_i and P_i is the appropriate configuration setting given C_i and Q_i. The following pseudo code generalizes the instantiation of configuration decisions for the heuristics of subsection 3.3:

```
forall i < 6
   if Ci is sensed/gauged and
      Qi is the corresponding required quality
      then Pi is an appropriate configuration.
```

4 Experimental Evaluation

We conducted an empirical study to demonstrate that parameter setting decisions by the proposed mechanism are appropriate in the sense that they are good at delivering the quality requested for the circumstances. To validate the proposed mechanism, we selected Apriori as the data mining algorithm to be configured and created its behavior model in Bayesian network representation to derive configuration decisions. Besides, we employed another approach for

parameter setting, full factorial experiment design and compared the inferences made from the Bayesian network against the results of the full factorial experiment design. The experiment has the following steps:

1. Ubiquitous data mining simulator is developed to generate experiment data.
2. Bayesian network is constructed in order to discover the probabilistic relationships among parameters, circumstances and quality.
3. Multi-level full factorial design is used to find out the parameters that are effective on the quality under a given circumstance.
4. The results of the two methods are compared to assess the proposed mechanism.

Fig. 2 shows the interaction of the experiment steps and the ubiquitous data mining simulator.

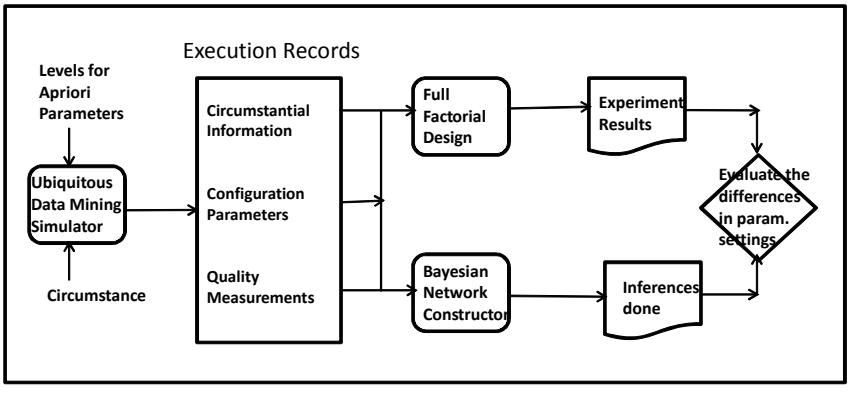

Fig. 2. Experiment phases

4.1 Ubiquitous Data Mining Simulator and Experiment Data

In order to collect data for automatic configuration of a specific data mining algorithm, we have developed a software to simulate the ubiquitous computing environment where data is mined by the same algorithm. Simulator runs the data mining algorithm with various configurations while generating the circumstances given and collects relevant execution data.

Simulator Architecture. Simulator runs Apriori which we selected for the experiments as the sample data mining algorithm to be configured, by calling Weka ([13]) API's. Since our purpose is to simulate the ubiquitous computing environment, Apriori was not executed when the device's resources are in arbitrary state. On the contrary, planned bottlenecks on the device's resources were created by the simulator before starting Apriori. Simulation software was implemented in JAVA language whereas bottleneck creator modules were written in C++ language. JAVA classes and their interactions are given in Figure 3.

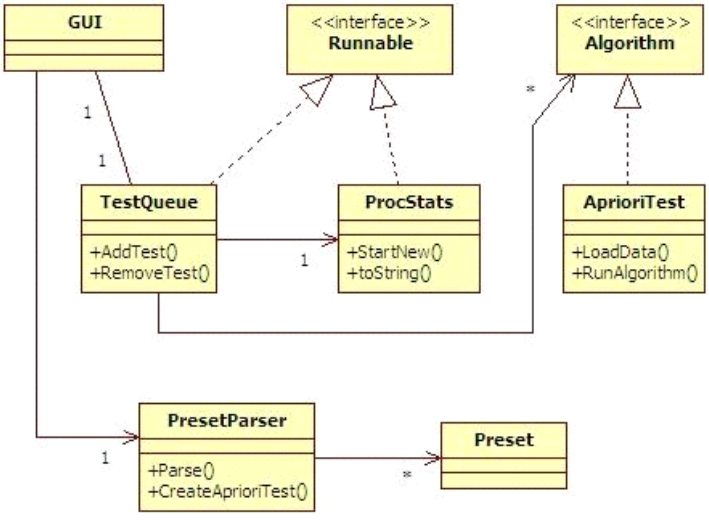

Fig. 3. Class descriptions of Ubiquitous Data Mining simulator

Input of the simulator is the preset file where each line represents a single test case. For each test case the following information are provided: context data associated with this Apriori run, resource bottleneck requests, data set to be mined, and parameter settings to be used for this test case. Resource bottleneck requests indicate the desired percent of memory and/or processor consumption by other workload in the device during execution of this test case.

Output of the simulator is the raw execution data in which there is a record for each execution of Apriori. Circumstance attributes (C^E) of execution data contain gauges showing resources' availability when Apriori was run and also the context which is given in the preset file for this execution. Data mining model as well as the measured, actual resource usage by Apriori are stored in quality attributes (Q^E). Together with parameter attributes (P^E), each execution record has 28 attributes.

Briefly, ubiquitous data mining simulator reads preset file, generates the resource scarcity conditions if the given circumstance requires and runs Apriori with the given parameters. Upon completion, an execution record is created. Ubiquitous data mining simulator (Figure 3) has a graphical interface (*GUI*) to set the name of the preset file and the execution file as well as to start the simulation. *PresetParser* is used to parse the contents of preset file and responsible for invoking bottleneck creators to call some "dummy programs" that will consume the requested amount of related resource. *TestQueue* is typically a queue that contains *Algorithm* instances. *AprioriTest* represents tests of the Apriori algorithm. and implements the interface *Algorithm*, thus its instances can be added to *TestQueue*. *ProcStats* performs the gathering of performance statistics before, after and during the execution of the algorithm tests. Specific system metrics

related to memory or CPU are gathered using specific methods. This class is designed as an independent cohesive unit to measure performance metrics, gather system information and statistics.

Execution Data. Execution data for the experiment was generated using the ubiquitous data mining simulator. The states of the context and the type of resource constraints that we used in forming the circumstances of the test cases are {*home*, *office*} and {*short on memory*, *cpu bottleneck*, *none*} respectively. All the types of resource constraints were simulated for each context state, resulting in six different circumstances. Settings used for each Apriori parameters are given in Table 1. There are different sets of settings for *home* and *office*. In the experiment Apriori was run for all combinations of the determined settings for each of the six circumstance. Therefore, the number of test cases for each circumstance having *home* as context state are 2250(3x6x5x5x5) and *office* as context state is 180(3x3x5x2x2).

Table 1. Levels used for parameters

Context state	Mnemonic	Parameter	Settings
Home	U	upper bound minimum support	0.7, 0.8, 0.9
	M	lower bound minimum support	0.1, 0.2, 0.3, 0.4, 0.5, 0.6
	D	delta	0.01, 0.05, 0.1, 0.15, 0.2
	N	number of association rules	1, 5, 10, 15, 20
	C	minimum confidence	0.5, 0.6, 0.7, 0.8, 0.9
Office	U	upper bound minimum support	0.7, 0.8, 0.9
	M	lower bound minimum support	0.4, 0.5, 0.6
	D	delta	0.01, 0.05, 0.1, 0.15, 0.2
	N	number of association rules	15, 20
	C	minimum confidence	0.8, 0.9

4.2 Parameter Setting by Bayesian Network Inferences

In this step, we applied our mechanism to predict Apriori configurations from the Bayesian network. Bayesian network construction and inferencing from the network are two main tasks of this step.

Execution data generated by ubiquitous data mining simulator were first discretized before constructing the Bayesian network given in Fig. 4. While discretizing, we used equal frequency bins and chose the number of bins that produced the highest number of relationships the among nodes. While constructing the network we made use of the K2 algorithm ([8]) by modifying it to group the nodes and searched causal relationship among these groups of nodes. The nodes in the upper level of the network in Fig. 4 represent the circumstance, middle level nodes represent Apriori parameters, and finally the lowest level nodes are quality measures. The cause and effect relationships between circumstances and parameters present which parameter settings are appropriate under which circumstances, whereas the cause and effect relationships between parameters and quality measures show which parameters are effective on which quality measures.

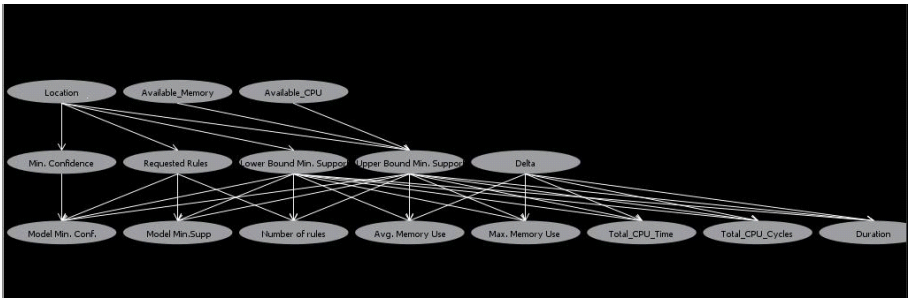

Fig. 4. Bayesian network of Apriori runs

While producing the experiment data for this Bayesian network, we did not determine appropriate parameter settings for circumstances but we ran Apriori for every combination of parameters in each circumstance because our purpose is to find the effect of parameters to quality measurements in the first place. Therefore, at this stage the relationships between circumstances and parameters are not meaningful. We assumed each circumstance node relates to each parameter node in order to include circumstances in the inference mechanism. The relationships between the parameter nodes and quality measure nodes represent the effectiveness of parameters against quality measurements. The Bayesian network in Fig. 4 shows that *minimum confidence* and *requested rules* are related only to efficacy; *delta* to all efficiency measurements as well as *lower and upper bound minimum support*, are related to all.

We determined parameter settings decisions by inferencing from the Bayesian network given in Fig. 4. The pseudocode of the estimation is given in subsection 2.5.

4.3 Multi-level Full-Factorial Experiment Design

In this step, we applied multi-level full factorial experiment design which is one of the Design of Experiment (DoE) methods [17]. Full factorial experiment design is statistically determining the effects of the factors of a process to its response by systematically varying the levels of the factors during testing of the process. In DoE terminology, *response* is the output variable of the process, *factors* are its input variables and *level* is a possible setting for a factor. The process that we want to analyze is the behavior of Apriori, more specifically, to find out which Apriori parameters affect which quality measurements. Therefore, Apriori parameters are the *factors*, their possible settings are the *levels* and resulting quality is the *response*. In full factorial experiment design, data is collected by running the process with all combinations of determined levels of its factors. Hence, we generated execution data similarly by running Apriori for all combinations of settings as explained in subsection 4.1. Moreover, since we ran Apriori by simulating six specific circumstances, we are able to analyze the effects for each circumstance.

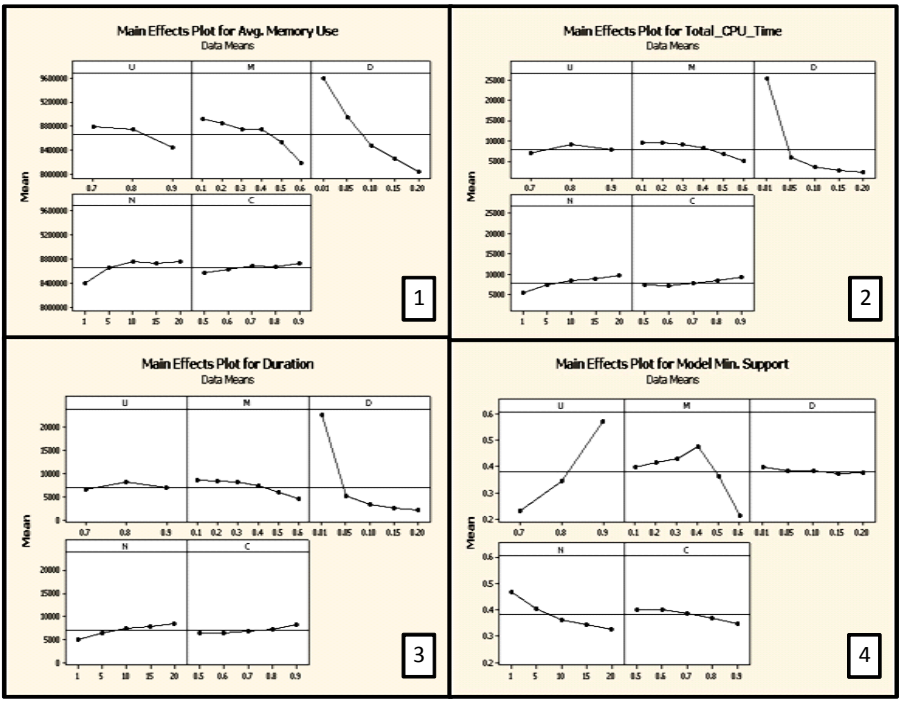

Fig. 5. Main effects plot of 4 quality measurements for *home-short on memory*

We used experiment software Minitab([16]) to estimate the effects and to plot the analysis results. Fig. 5 illustrates the full factorial experiment design results obtained for *home-memory low*. We analyze the results for this circumstance in detail in order to explain the method. In the figure, the means of quality measurements for the utilized levels of parameters are plotted. In quadrants of Fig. 5, plots for *average memory use*, *total CPU time*, *duration* and *minimum support of the model* are given respectively. Each plot (U, M, D, N, C) within a quadrant is for a parameter. The mean of the measured value is plotted for every level we tested for that parameter in the experiment. If the plot is not flat which indicates the means of measured values vary with different value assignments of this parameter, then this parameter is effective on the measured value. We considered the F test values to determine the significance of the effect. While determining the appropriate value of the parameter which is designated as effective on the measured criteria, we have chosen the value that has the smallest mean of response for its factor level combinations. We compare the results of full factorial experiment design against the results of the Bayesian network in the next subsection.

4.4 Comparison of Results

In order to determine the parameter settings of an algorithm, we explained two different approaches, Bayesian networks and full factorial experiment design where the former is a probabilistic approach and the latter a statistical approach. The outcomes of the approaches are summarized as follows:

- Full factorial design provides
  - The list of parameters which are not effective on a quality measure
  - The parameter setting which has the highest/lowest least square mean for a quality measure
- Inference from Bayesian network provides
  - The list of parameters which are not related to a quality measure
  - Most likely parameter setting given the circumstance(s) and the quality measure(s) as evidence

To compare the results, we used two criteria: i) the percentage of alike parameter/quality measure relationships and, ii) the percentage of identical parameter settings, obtained by the two approaches. In Table 2, for each circumstance, we present, (i) and (ii) by grouping quality measures as efficiency related and efficacy related.

Table 2. Comparison of results

Circumstance	Efficiency		Efficacy	
	(i)	(ii)	(i)	(ii)
home-short on memory	73	77	90	100
home-CPU bottleneck	80	89	100	100
home-no constraints	73	77	80	80
office-short on memory	100	67	100	75
office-CPU bottleneck	100	89	90	75
office-no constraints	100	78	90	75

It is possible to say based on the results (Table 2) that in majority of the cases, parameters that are found to be effective on a quality measure under a circumstance in full factorial design, are represented as related to that quality measure under the same circumstance in the Bayesian network. The appropriate parameter settings decided in order to optimize a quality measure in full factorial design is identical in most of the cases to the parameter settings inferred from the Bayesian network given the same quality measure.

4.5 Effects of Mining Data Set Feature Variations on the Behavior Model

In the simulation phase of the experiment (subsection 4.1), we mined always the same data set with Apriori and in this way we eliminated the effects of the data

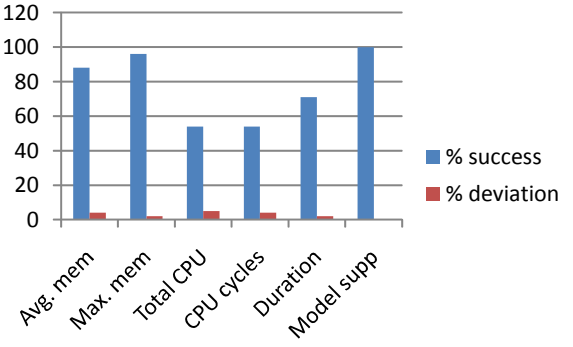

Fig. 6. Percentage of successful recommendations

set feature changes on the behavior model constructed. On the other hand, in a real life situation data set to be mined may grow or shrink either by addition or deletion of instances into the data set or by attribute set changes. Variations on the mining data set features may necessitate refreshing the behavior model that is used to recommend algorithm configurations for mining this data set. Thus, we performed a series of experiments to affirm that mining data set size may have an effect on the parameter setting recommendations and also to speculate on how to detect that the behavior model is decayed. In the experiments, we made use of quality measurement figures collected during the execution of Apriori to assess whether the recommended parameter settings provide the requested quality. Experiments rely on the behavior model that we call *basis behavior model* generated in the same way explained in subsection 4.2 from the execution records collected by running Apriori with input data set (*DSx1*) in a simulated ubiquitous computing environment similar to the one explained in subsection 4.1. Brief explanation of data set size variations effect evaluation experiments are as follows:

– **Verify the recommendations.** In this experiment, Apriori was configured by the recommended settings acquired from the *basis behavior model* and ran with input *DSx1* in the simulated ubiquitous computing environment for every possible recommendation. Afterwards, we determined the appropriateness of each recommendation by comparing the relevant quality measurement value collected during Apriori's execution against the requested quality used when deriving the recommendation from Bayesian network. For example, if an Apriori configuration is recommended to minimize the memory usage of Apriori, we assess the parameter setting objective by comparing the memory usage figures of Apriori's execution with this configuration against the lowest memory usage figures in the behavior model. The percentage of the Apriori executions which achieve the objective of the parameter settings grouped by relevant quality measurement are given in Figure 6. Percentage of deviation from the requested quality is also analyzed for each quality measurement

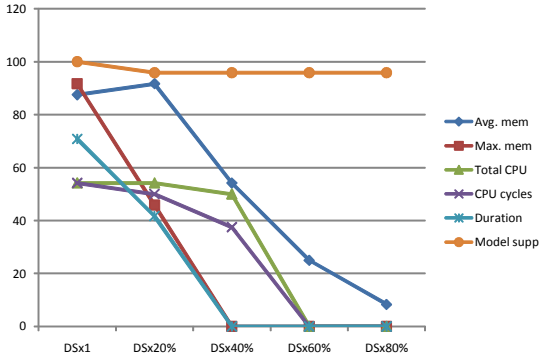

Fig. 7. Behavior model decay

group. The maximum amount of deviation is ten percent of the requested quality whereas the average amount of deviation does not exceed five percent of the requested quality for any of the groups (Figure 6). The results obtained are satisfactory to verify the appropriateness of the recommendations.

– **Demonstrate the data set size effect.** We try to find out in this experiment whether the behavior models extracted from the executions of the same data mining algorithm with same configuration settings but with different data mining data set sizes, are different. For this purpose, we generated another behavior model, *behavior model 10* in a similar way that we generated *basis behavior model* but the size of the data set (*DSx10*) used as input to Apriori in this experiment is ten fold bigger than *DSx1*. After generating the behavior model (*behavior model 10*) for Apriori mining *DSx10*, we compared *behavior model 10* against *basis behavior model* and detected that half of the recommendation decisions are changed. By this way, we have shown that input data set's size of a data mining algorithm may have an impact on certain parameter settings decisions given in order to achieve certain quality objectives.

– **Estimate behavior model decay.** In the final experiment, we gradually increased the size of the mining data set mimicking a possible real life situation in which a data set grows in time. Our purpose is to analyze the deterioration of the recommendations in terms of achieving the quality requested as the data set grows. We iteratively increased the size of the mining data set by twenty percent, run Apriori with all the possible recommended parameter settings extracted from the *basis behavior model* while simulating the relevant circumstance, recorded the requested quality for the recommendation and compare it against the achieved quality. During this process we used the same behavior model (*basis behavior model*) without populating new execution data or refreshing it completely. Figure 7 shows for different

mining data set sizes the percentage of Apriori executions where the objective of the parameter setting is achieved in terms of the quality obtained. Experiment results show that the correctness of the configuration decisions derived from the behavior model in order to obtain the requested qualities of all types except the model minimum support are affected by the data set size change. Furthermore, *basis behavior model* decays needing a refresh before *DSx1* grows by forty percent. This experiment revealed that mining data set size change do not effect every parameter setting decision but if a parameter setting decision do not provide the requested quality, it is possible to detect.

5 Conclusion

Anticipation of the importance of autonomous and adaptable behavior in ubiquitous data mining, led us to research on how to tune its parameters automatically. In order to determine its appropriate configuration, we employed a mechanism to understand the data mining algorithm behavior. Considering the characteristics of ubiquitous computing environments, we stressed making use of circumstantial factors for determining the appropriate parameter values. We also aimed at recommending parameter settings that most likely satisfy the required quality of a circumstance. Thus, we analyzed the effects of parameter settings to quality measures which are related to both efficiency of data mining process and efficacy of the data mining model. We proposed to use Bayesian network for extracting data mining algorithm behavior while taking into account circumstance and the quality.

As the result of our simulation experiment, satisfactory parameter and quality measure relationships to recommend parameter settings, were formed in the Bayesian network. We also validated our proposal by comparing the parameter settings obtained from the Bayesian network against another approach, full factorial experiment design. Experiment on association rule mining shows that proposed method gives parameter settings almost identical to the settings obtained from full factor analysis which is a completely different approach. We also verified the correctness of the configuration recommendations derived from the behavior model by mining data with the recommended settings and obtained high percentage of correct configuration recommendations which produced mining results with required quality.

Experiment that we performed to show the negative effect of mining data set size change on the accuracy of the parameter setting decisions derived from the behavior model, revealed that behavior model may not be generated only once and reused forever but needs to be updated or refreshed with new experiences gained by the execution of data mining algorithm. Hence, health checking the behavior model is an important factor for successful automatic parameter tuning.

In the future, we aim to represent the mining data set characteristics such as the number of attributes and the number of tuples that are relevant for parameter tuning in the behavior model so that a single behavior model can be used for a wider set of configuration requests.

Acknowledgments. Authors would like to thank Can Tunca and Engin Dogusay from Sabanci University who contributed to this study by developing the supporting software.

References

1. Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules. In: Proceedings of the Int. Conf. on Very Large Data Bases (VLDB 1994), pp. 487–499. Morgan Kaufmann, San Francisco (1994)
2. Amstrup, S.C., Marcot, B.G., Douglas, D.C.: A Bayesian Network Modeling Approach to Forecasting the 21st Century Worldwide Status of Polar Bears. Arctic Sea Ice Decline: Observations, Projections, Mechanisms, and Implications. Geophysical Monograph 180, 487–499 (2008); American Geophysical Union, Washington, DC
3. Beinlich, I.A., Suermondt, H.J., Chavez, R.M., Cooper, G.E.: The ALARM Monitoring System: A Case Study with two Probabilistic Inference Techniques for Belief Networks. In: Proceedings of the Second European Conference on Artificial Intelligence in Medicine, London, pp. 247–256 (1989)
4. Birattari, M., Stutzle, T., Paquete, L., Varrentrapp, K.: A Racing Algorithm for Configuring Metaheuristics. In: GECCO 2002 Proceedings of the Genetic and Evolutionary Computation Conf., pp. 11–18. Morgan Kaufmann, San Francisco (2002)
5. Buntine, W.: A Guide to the Literature on Learning Probabilistic Networks from Data. IEEE Trans. on Knowl. and Data Eng. 8, 195–210 (1996)
6. Cao, L., Gorodetsky, V., Mitkas, P.A.: Agent Mining: The Synergy of Agents and Data Mining. IEEE Intelligent Systems 24, 64–72 (2009)
7. Charniak, E., Goldman, R.: A Semantics for Probabilistic Quantifier–Free First–Order Languages with Particular Application to Story Understanding. In: Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, Menlo Park, California, pp. 1074–1079 (1989)
8. Cooper, G.F., Herskovits, E.: A Bayesian Method for Constructing Bayesian Belief Networks from Databases. In: Seventh Conference on Uncertainty in Artificial Intelligence, pp. 86–94. Morgan Kaufmann, San Francisco (1991)
9. Adenso-Diaz, B., Laguna, M.: Fine-Tuning of Algorithms Using Fractional Experimental Designs and Local Search. Oper. Res. 54, 99–114 (2006)
10. Gagliolo, M., Schmidhuber, J.: Learning Dynamic Algorithm Portfolios. Annals of Mathematics and Artificial Intelligence 47, 295–328 (2006)
11. Gaber, M.M., Yu, P.S.: A Framework for Resource-Aware Knowledge Discovery in Data Streams: a Holistic Approach with its Application to Clustering. In: ACM Symposium on Applied Computing, pp. 649–656. ACM, New York (2006)
12. Haghighi, P.D., Zaslavsky, A., Krishnaswamy, S., Gaber, M.M.: Mobile Data Mining for Intelligent Healthcare Support. In: 42nd Hawaii international Conference on System Sciences, pp. 1–10. IEEE Computer Society, Washington, DC (2009)
13. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA Data Mining Software: An Update. SIGKDD Explorations 11 (2009)
14. Hood, A.C., Ji, C.: Proactive Network Fault Detection. In: Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution, INFOCOM, p. 1147. IEEE Computer Society, Washington, DC (1997)

15. Hutter, F., Hoos, H.H., Stutzle, T.: Automatic Algorithm Configuration Based on Local Search. In: 22nd National Conference on Artificial Intelligence, pp. 1152–1157. AAAI Press, Menlo Park (2007)
16. Minitab Inc., http://www.minitab.com/en-US/
17. Montgomery, D.C.: Design and Analysis of Experiments. John Wiley and Sons, Chichester (2006)
18. Pavon, R., Diaz, F., Laza, R., Luzon, V.: Automatic Parameter Tuning with a Bayesian Case-Based Reasoning System. A Case of Study. Expert Syst. Appl. 36, 3407–3420 (2009)
19. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Francisco (1988)
20. Srivastava, B., Mediratta, A.: Domain-Dependent Parameter Selection of Search-Based Algorithms Compatible with User Performance Criteria. In: 20th National Conference on Artificial Intelligence, pp. 1386–1391. AAAI Press, Menlo Park (2005)