Air Quality Management using a multi-Agent System

Elias Kalapanidas and Nikolaos Avouris*

University of Patras,
Electrical and Computer Engineering Department
GR-265 00 Rio Patras, Greece
Email: N.Avouris@ee.upatras.gr

Keywords: multi-agent system, machine learning, air quality prediction, urban air pollution, neural network, decision tree, case-based reasoning, environmental data modeling.

* corresponding author

(to be published in the International Journal of Computer Aided Civil and Infrastructure Engineering, 2001.)

Air Quality Management using a multi-Agent System

Abstract

Management of urban atmospheric pollution necessitates advanced modeling and information processing techniques. The design of the prototype system DNEMO, which is based on a distributed adaptive problem solving approach is the focus of the research reported in this paper. Issues covered in the paper relate to the distributed nature of this environmental problem, handling noise and uncertainty in monitoring data, achieving graceful degradation of performance and system robustness, adaptation of system performance to long-term evolution of the monitored phenomena. The research reported can be applicable to a broad class of Environmental Monitoring applications, since the problems addressed are common to many environmental problems.

1. Introduction

Urban atmospheric pollution is an environmental problem of great importance for many cities of our planet, affecting the quality of life of millions of citizens. Increased traffic emissions in combination with widespread industrial and commercial activities have contributed to aggravation of the problem during the last years. In particular, photochemical pollution involves a series of chemical reactions triggered by solar radiation producing pollutants like ozone (O_3) and nitrogen oxides (NO and NO_2). These substances are known to cause susceptibility to lung infections and asthma when they enter the human respiratory system while long-term exposure to them can weaken the effectiveness of the lung's defense against bacterial infections according to the World Health Organization (WHO, 1987). For this reason dense Monitoring Networks are in operation in many areas, collecting continuously environmental and meteorological data which are fed to Air Quality Management Centers. For instance the Monitoring Stations of the Greater Athens network are shown in figure 1. The collected data need often to be processed at run time, so that air pollution experts can accomplish tasks like short-term prediction of pollution

levels, issuing of alarms to the public, making decisions on emergency counter measures etc.

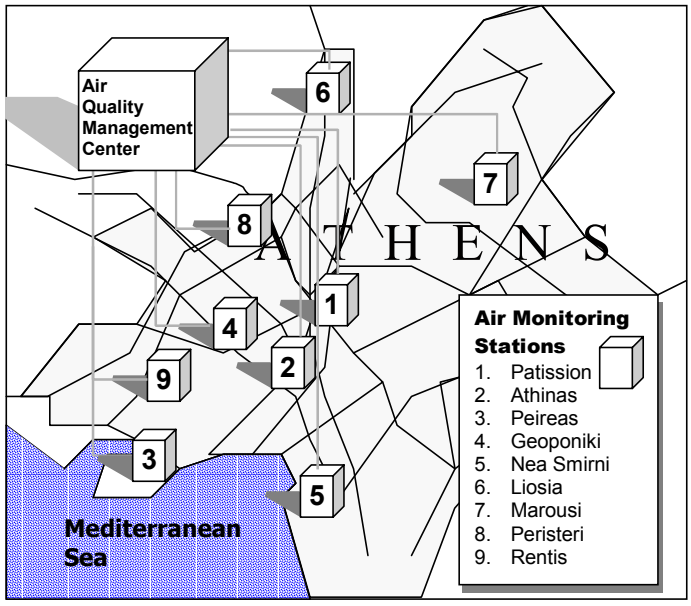

Figure 1. The Athens Air Quality Monitoring Network.
Each Monitoring Station takes measurements of CO, NO, NO_2, O_3, SO_2, smoke, Pb.

A combination of data management and processing modules with environmental modeling and problem solving components are often used in these Air Quality Management centers. Use of Artificial Intelligence (AI)* techniques in this context has been often proven very useful. Expert systems and other knowledge-based techniques have produced promising results, supporting tasks like countering measures selection and alarm levels forecasting (Avouris, Page, 1995). These AI techniques present advantages over more traditional numeric modeling approaches, since the later require heavy computational resources and need as input complex data, often not easily available at run time (van Aalst et al. 1998) . In the ongoing research effort of our group, artificial intelligent prototypes have been developed and experimentally used over the years in this domain, based on case-based reasoning and machine-learning approaches (Lekkas et al., 1994, Kalapanidas, Avouris, 2001, Neagu et al. 2001)..

However for such systems to become operational under real life conditions, special attention must be put in issues relating to: (a) low quality or missing data, a

* See acronyms list in Appendix

common problem in environmental applications and (b) the changing conditions of the environment over long periods of time, which result in failure of typical brittle AI models. Close study of Environmental Monitoring Centers over the years has demonstrated that malfunctioning communication networks or not well calibrated sensory equipment are very often the case, resulting in bad quality data, while the behavior of polluting sources (traffic, industry etc.) change inevitably over the time.

In this context, system adaptability in long-term phenomena changes, graceful degradation of performance under equipment or communication failures and robustness of the overall system are key requirements. In order to achieve these objectives the designers of environmental monitoring systems need to take advantage of some inherent problem characteristics and apply adequate technological solutions.

In environmental monitoring there is a high degree of redundancy in sensory data, due to the multiple monitoring stations. Also many complementary modeling tools can be used in order to increase system robustness. Finally techniques for continuing assessment of system performance need to be developed which would trigger model adjustment if performance degradation should be observed.

A powerful computational model which permits building a distributed intelligent environmental monitoring prototype, which can demonstrate these characteristics, is that of **agent programming** (Shoham, 1997). Interacting intelligent agents can model Monitoring Stations, alternative AI problem solving techniques and adjust the overall system behavior to the evolving situation and the monitored environment.

During the last years such an experimental multi-agent prototype, called Distributed NEMO (DNEMO), has been built for management of urban air pollution in Athens, Greece and is in experimental use in our laboratory under simulated real life conditions. DNEMO was built using the agent programming language LALO (Gauvin, 1995), which is a framework for development of distributed agents with complex mental states. These are made of beliefs, capabilities, commitments and decisions, as described by (Shoham, 1997). These characteristics are useful for providing the DNEMO agents with adapting behavior, resulting in overall system robustness, as described in following sections of this paper. The DNEMO system is made out of a number of autonomous Monitoring *Station Agents*, which are able to seek support in a dynamic way from neighboring stations in case of low quality data. Also a number of existing Machine Learning - based problem solving modules, called

Model Agents, have been integrated in DNEMO, which supply the Station Agents with pollution assessments upon request.

Use of Machine Learning (ML) Algorithms for Air pollution short-term prediction has been subject of research for some time now. For instance there are many experiments using neural networks for short-term ozone or sulfur dioxide prediction (Perantonis et al., 1994, Boznar, Mlakar, 1995, Ruiz-Suarez et al., 1995, Wieland, Wotawa, 1999). An investigation on the application of simple regressors for ozone has also been studied by (Soja,1999). From a chemical point of view, a CBR system for the description and organization of process sequences aiming at reducing the environmental impact is described by (King et al., 1999). A comparative study of different ML techniques on short-term nitrogen dioxide prediction can be found in the work of (Vasilas et al., 2000).

The developed multi-agent is one of the first multi-agent systems in this area that combines multiple problem solving components. The system has been tested under simulated adverse conditions using archived two-year sensory data from the Athens Metropolitan area in Greece. This paper contains a description of the system architecture and scenarios of typical operation, under various conditions. The design methodology and some key design decisions made, which are interesting to designers of AI environmental applications are also described. The research reported here is of interest to a broad community of Environmental Monitoring researchers and practitioners, since the issues tackled by the proposed design are common to a wide class of environmental problems, since multi-agent solutions seem relevant to many environmental monitoring applications.

2. Multi-agent systems in environmental monitoring

During recent years a paradigm shift has been observed in Artificial Intelligence practice and research, in line with a similar move in Computer Science, which moved from an approach which maintained "information processing" as the center of attention to an interaction based computational model. In Artificial Intelligence this was expressed through the Distributed AI (DAI) approach, which studies methods and techniques for building intelligent systems made out of interacting intelligent agents. These agents, in general have capabilities to take

initiative, to reason, to act and to communicate with each-other and their environment (Huhns, Singh, 1998) . There is no general agreement over the precise definition of an intelligent agent, and to the confusion has contributed the fact that the term has been over-used during recent years in many application areas of Computer Science. Despite this, in Artificial Intelligence, the emergence of agent programming languages like KQML, and inter-agent communication protocols, have resulted in establishment of a common understanding and widespread use of multi-agent systems. As a consequence, an expanding number of application areas of intelligent multi-agent systems has been observed during the last years. In spite of the fact that environmental problems take a less prominent position in this list of applications areas, a number of environmental multi-agent applications have been reported in the literature (Avouris, 1995).

There are many reasons that suggest the use of multi-agent architecture for the design of a flexible urban air quality management prototype. In great extent these reasons hold for many other environmental problems and therefore can be the basis of the rationale for following a similar approach in other similar application areas. These are outlined below.

- The problem is spatially distributed. Urban air pollution is usually dispersed in a wide geographical area and the solution (e.g. the prediction of the evolution of the pollution levels) should maintain this spatial dimension

- Many monitoring stations are involved, each one measuring the monitored indices in the vicinity of the station

- Information collected is noisy and imprecise. This is due to instruments faults, communication errors and other local disturbances at the measurement point.

- There is a great degree of redundancy involved. For example the temperature is very unlikely that will change dramatically over a short period of time and the wind direction to be very different, between two points which are a few kilometers apart, unless there are special geographical reasons.

- Various alternative techniques and models can be used. Symbolic knowledge based systems can express the domain experts explicit knowledge, while machine learning techniques can be applied in order to build systems containing the implicit knowledge hiding in vast amounts of available historical data.

For these reasons, The DNEMO prototype has been based on a multi-agent approach. The characteristics of the developed system and in particular the design

decisions made in order to tackle the problems described above, is the focus of this paper. First the enabling technology used is described in the remaining part of this section. In the following Section 3 the integrated system architecture is presented. Subsequently in Section 4, the building components of the system, that is the air quality prediction modules are presented, while in section 5 a general discussion and future directions are included.

2.1 Multi-agent system characteristics

The DNEMO multi-agent system was developed using the LALO agent oriented development environment (Gauvin, 1995). The characteristics of this environment, common to many similar Multi-agent (MA) systems, are presented here. LALO is an object oriented environment that contains an extendable set of classes, from which application agents can be built. Additionally an agent-oriented programming language is contained in the environment. There are many different agent classes. Simple reactive agents for instance cannot have complex mental states. However even these simple agents have a **behavior** and **communication** component, as shown in figure 2.

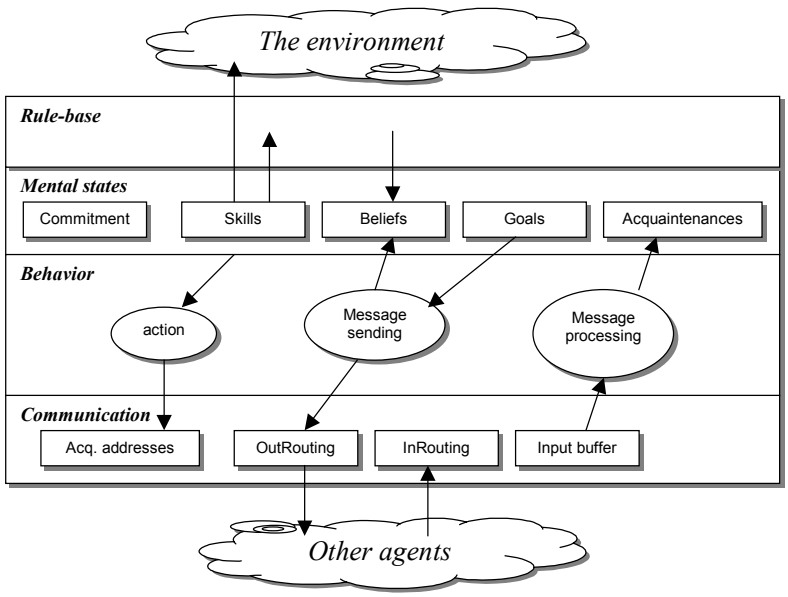

Figure 2. Structure of a DNEMO Agent

The communication component, through threads, permits the asynchronous processing of incoming and outgoing message queues by the agent. The messages sent and received by an agent are based on the KQML protocol (Finin et al., 1997). KQML (Knowledge Query and Manipulation Language) is a high-level agent communication language influenced by *speech-acts theory*, a linguistic analysis of human communication. KQML can be regarded as the de facto standard agent communication language. The *performatives* (commands) of KQML have been defined as a library of communication acts. When sending a KQML message from sender S to receiver R, S either asks R to perform the action described by the performative, or S does perform this action by sending the message. A typical KQML message has the following structure: (performative :sender :receiver :from :to :in-reply-to :reply-with :language :ontology :content). Some typical KQML performatives are: ASK-IF (S wants to know if the :content is in R's knowledge base); ASK-ALL (S wants all of R´s instantiations of the :content that are true of R); STREAM-ALL (multiple-response version of ask-all); ASK-ONE (S wants one of R's instantiations of the :content that is true of R); TELL (the sentence is in S's knowledge base); INSERT (S asks R to add the :content to its knowledge base); ACHIEVE (S wants R to make something true of its physical environment); etc.

The simple reactive LALO agent can process messages of *transport-address* and *achieve* types only. A more complex agent contains also a mental component. This component permits representation of agent *beliefs* of , its *skills (*that is the tasks that the agent can undertake, divided in private tasks and public ones which can be executed after request for other agents), its *goals* and *commitments*. The agent maintains also models of other agents (*acquaintances*), as shown in figure 2. This agent can have a more complex behavior than that of the reactive agent. For instance through the *commitments* structure, the agent can commit itself to a plan, and therefore reject a request for a new job, made by another agent. This agent can respond to messages of type *stream-about*, *ask-if*, *tell* etc. Even more complex agents can be built that contain a control mechanism of their behavior, represented in figure 2 as Agent Rule Base. This can contain rules controlling the processing of the incoming messages, the mental state of the agent and the agent's actions. The LALO agents can be programmed through the LALO programming language, in which the

beliefs the private and public tasks and the other characteristics of the agent can be defined. In figure 3 the definition of an agent is shown according to this scheme. In this figure the *timer.la* component describes the TimeWatcher agent characteristics. This is passed through the pre-processor and subsequently linked with the rest of the agent classes. In the next section the DNEMO architecture, implemented using the LALO environment, is described.

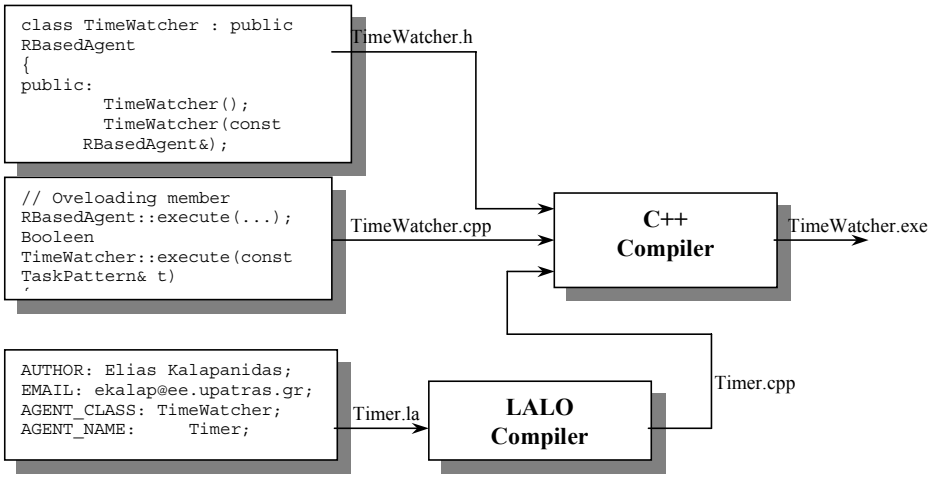

Figure 3. DNEMO Agent development process (extract from development of TimeWatcher Agent).

3. The DNEMO Architecture

According to (Stone, 2000), a taxonomy of multi-agent systems (MAS) relates to the general dimensions of homogeneity-heterogeneity and of communication. The DNEMO system is made of two clusters of agents (*station* and *model* clusters) and a number of auxiliary agents as shown in figure 4. The first cluster is made of homogeneous communicating agents and the second of homogeneous no-communicating ones. However there is communication between the two clusters as shown in figure 4.

The **stations cluster** represents all the spatially distributed monitoring stations. They are assigned with the task of solving the partial problem of predicting the local air pollution levels. These agents store their own sensory data locally, they contain similar planning methods to resolve data integrity issues, and they engage the same strategy to solve their local problem which represents a partial solution to the

overall problem. Each Station Agent maintains a list of other Station and Model Agents capabilities, used for matching in an optimal way a given planned action that cannot be executed locally, with the agent that offers the specified service. Communication between agents belonging to this same cluster is a crucial part of the overall solution building phase.

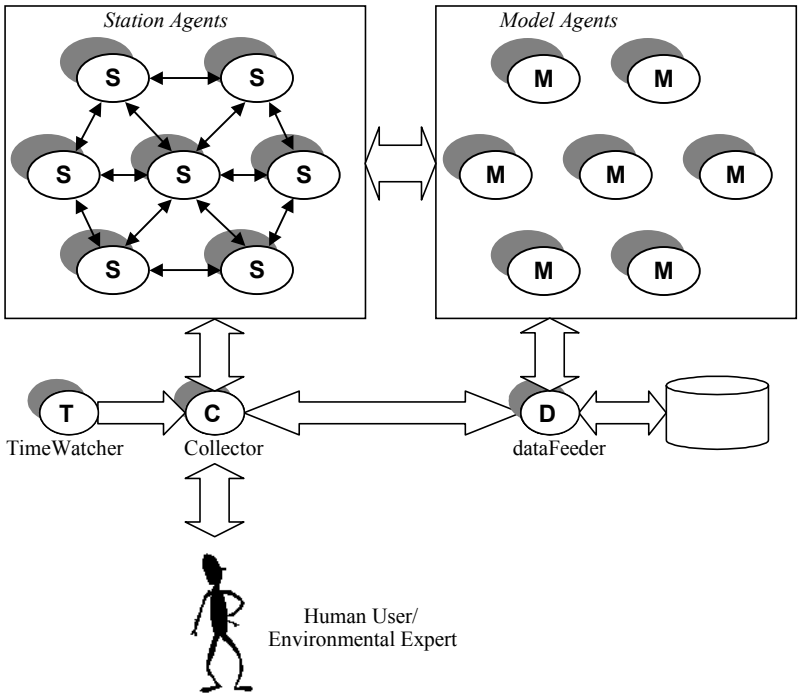

Figure 4. The DNEMO Agents and their inter-relations

The **models cluster** contains agents that perform the same tasks in a different way, without communicating with each other. They have the same goal of constructing a classifier for the given set of input and output parameters, using a Machine Learning (ML) or other AI algorithms. While for the number of station agents there is a one to one mapping to the physical Monitoring Stations, which are dispersed around the Greater Athens area, the Model Agents maintain a many to one relationship with the different ML algorithms that are implemented. A discussion on the performance of some of these agents is included in section 4 of the paper.

Apart from the two clusters of agents, there are three more facilitator agents with special functionality, as shown in figure 4.

(i) An agent named **TimeWatcher** whose duty is to maintain an agenda of scheduled events. This agent for instance informs relevant agents that it is time for prediction of the concentration of a specific air polluting substance.

(ii) The **dataFeeder** agent, that is the source of raw data of DNEMO, acting as a facilitator between DNEMO agents and the database of the Air Quality Management Center.

(iii) The **Collector** agent, which is the user-interface agent that communicates with the end-user. This agent gathers predictions from all station agents and presents them to the human operator.

The primary objective of DNEMO is to make a short term prediction about the pollutants peak concentrations within a defined future time period. The general scheduling of actions of DNEMO during normal operation, shown in figure 5, as sequence of exchanged messages (M1, M2 ..), is presented here:

When the TimeWatcher is triggered by a time event that is activated, it passes this information to the Collector Agent (ie. Messages M11, M1). The Collector constructs a plan of actions, based on the received message. For example if the message indicates that the time to get a *prediction for ozone (O_3) for the next day* has arrived, then the Collector notifies all known Station Agents about the fact by sending the appropriate message M2, and waits for the results.

Every Station Agent S_i from the m available, constructs a dataset made of the 24 previous hourly concentrations for everyone of the 5 measured pollutants in its database. Let this vector be

$$\{[X^{Si}] = \{NO_{2\ t-24}^{Si}, NO_{2\ t-23}^{Si}, ..., NO_{2\ t-1}^{Si}, O_{3\ t-24}^{Si}, ..., SO_{2\ t-24}^{Si}, ..., NO_{t-24}^{Si}, CO_{t-24}^{Si}, ..., CO_{t-1}^{Si}\}$$

Let N = $\{S_1, S_2, \ldots S_i, \ldots, S_n\}$ be the set of the neighboring to S_i stations, according to physical distance. Then S_i requests from every neighbor S_k in N, to provide a prediction on ozone peak concentration for next day for Station S_i from data gathered locally by Station Agent S_k, ie. its correspondent output vector $[Y_{S_k}^{S_i}]$. For each of the neighbors, the Station Agent S_k orders to a Model Agent M the construction and training of a classifier that models the $[Y_{S_k}^{S_i}]$ output based on the input vector $[X^{S_k}]$. After this is accomplished, S_i asks each of these classifiers to provide a prediction on the new case of Y^{Si}.

Subsequently an iterative process called Stations Negotiation Process (SNP) begins, where each station agent negotiates its $Y_{ii'}$ prediction with each of the neighbors' predictions $Y_{ik'}$. SNP is described in the following section.

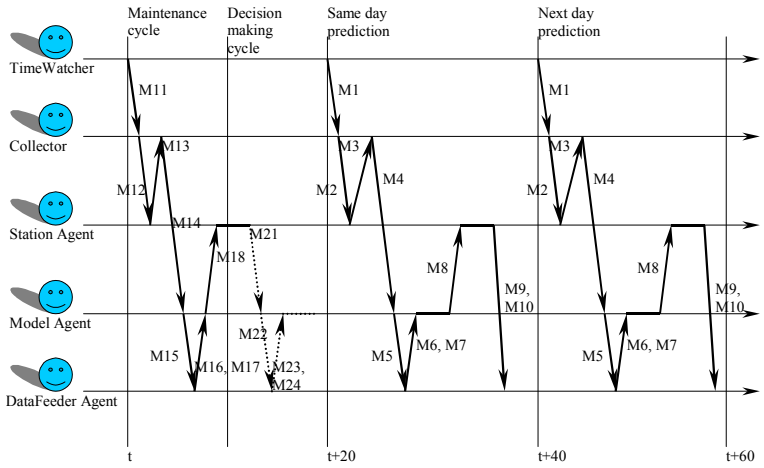

Figure 5 Agents Interaction Diagram

3.1 Agent Robustness

Since one of the main requirements of DNEMO is system robustness, i.e. the system to perform in a satisfactory way under adverse conditions, it is necessary before proceeding further, to provide a more analytical description of the term in the context of this application. The main goal of DNEMO is to supply a reliable solution to the spatially distributed problem of air quality prediction, under various conditions. Following an analysis of the domain of air quality monitoring, two main sources of uncertainty were identified: integrity of monitored data and failure in inter-agent communication.

As for the first problem, well known techniques for data quality have been transferred in the multi-agent context. Two major disturbances have been taken into account: noisy data, and missing values in the data streams.

The inter-agent communication problem on the other hand, necessitates use of a suitable inter-agent communication protocol, which is able to handle message delays and communication deadlocks. The protocol should be able to modify the plans of the agent in order to adapt to the open system characteristics. As discussed in the following, as a result of this feature, when some part of the system fails, graceful degradation of system performance is anticipated.

3.2 Data Integrity

Two main problems are encountered when one processes a stream of sensory data from a measuring station: noisy data and missing values in data. Each time a DNEMO Station Agent receives a measurement from the corresponding monitoring Station, a checking behavior towards the conservation of the data integrity is activated:

1. Check if there is a missing value (that is the measurement to be out of the legal boundaries of the sensor)
2. If it is so, the Station Agent activates the Missing Value Replacement procedure.
3. Check if the value contains noise.
4. If this is true, the agent activates the Noise Elimination procedure.

These procedures are described in more detail in the rest of this section.

Missing Value Replacement

In case of a missing value (m.v.) at the measurement P at time t in the dataflow of Station Agent S_0($P_t^{S_0}$), S_0 notifies all of its neighbors $\{S_1, .., S_n\}$ about the fact. For each agent S_m in N, an input vector containing the differences $dP_{t-k} = P_{t-k} - P_{t-k-1}$ for k ranging from 24 to 1, is calculated:

$$X^{S_m} = \{dP_{t-24}^{S_m}, dP_{t-23}^{S_m}, ..., dP_{t-1}^{S_m}\}$$

The agent searches its *Capabilities KB* for a known Model Agent that can train and evaluate a model. Then it sets a *commitment* for training the model with this vector as input, and assigns the output to $Y^{S_0} = \{dP_t^{S_0}\}$. As soon as the model agent receives this message, a classifier will be built on the specified past data, and then the testing of the new case where the output is the m.v., and is not known, will be performed. The result should be the solution proposal of this station to the sub-problem of the missing value replacement, here denoted as $Y_{S_m}^{S_0}$.

After all station agents in N have induced a candidate replacement for the m.v. , $Y_{S_m}^{S_0}$, the negotiation procedure between S_0 and every other neighbor $\{S_1, .., S_n\}$ is initiated. This is an iterative process that is called Stations Negotiation Process (SNP), with a stop criterion (SC) that of the overall differences not to exceed a pre-defined threshold value ε:

$$\mathrm{SC} = \sum_{m=1}^{n} \frac{|(Y_{S_0}^{S_0} - Y_{S_m}^{S_0})|}{Y_{S_0}^{S_0}} < \varepsilon\,, \quad Y_{S_0}^{S_0} > 0$$

The steps of this iterative procedure are the following:

Compute SC.

While SC > ε AND iteration_counter < MAX_ITERATIONS:

For $S_m = S_1$ to S_n :

$T = Y_{S_0}^{S_0}$

At a value of up to QM_0% station S_0 selects a random new T between T and ($Y_{S_m}^{S_0}$ -T)/2

At a value of up to (1-QM_m)% station S_0 selects a random new T between $Y_{S_m}^{S_0}$ and ($Y_{S_m}^{S_0}$ -T)/2

Compute SC

Accept as m.v. replacement the T candidate as it has been changed thus far.

The term QM_m is a quality measure for station agent S_m, based on the reliability of the station data. It is an average of the number of times that some missing value or noisy data has been accounted in the past, divided by the total number of values in the Station data repository.

After the m.v. procedure has been followed, the assessed replacement value is forwarded to the datafeeder agent for storage in the database.

Noise Detection and Elimination

Noise elimination involves a similar technique to that of missing value replacement, the only difference being that in this case the actual value $P_t^{S_0}$ is known. What is not known is whether this value has been modified by an external source or not. The steps are the following:

Compute $Y_{S_0}^{S_0}$ as previously at the m.v. replacement.

Compute the predicted difference $dP'^{S_0} = Y_{S_0}^{S_0} - P_{t-1}^{S_0}$ Calculate the actual difference $dP^{S_0} = P_t^{S_0} - P_{t-1}^{S_0}$

If $\left|\frac{dP^{S_0} - dP'^{S_0}}{dP^{S_0}}\right| > \zeta$ then initiate SNP for noise elimination, where ζ is a predefined threshold value for noise detection, similar to ε for m.v. replacement.

At the present state of DNEMO, the noisy value is completely re-assessed, following the same scheme as with the missing value replacement.

3.3 Efficient inter-agent communication

As described earlier, the DNEMO architecture makes use of heterogeneous communicating agents. This feature increases the flexibility of the system designer, but has the disadvantage that system reliability depends in this case on the inter-agent effective interaction which necessitates efficient inter-agent communication.

Some simple rules are developed to maintain an efficient message passing mechanism:

- Every KQML performative used in an agent's behavior that can include a *reply-with* field, is applied.
- In every message sending where a service is requested from another agent, a reply timeout threshold is set, after the end of which the agent changes its course of actions.
- When a Station Agent needs a service from a Model Agent, this service is requested from at least two of the model agents that support the specified service. The reply is selected according to timeliness, trust and model complexity criteria.

A representative example from an excerpt of the definition of the station agent type behavior will clarify each of the three previous rules. The following LALO rule engages the KQML performative ACHIEVE and makes use of the reply-with keyword:

IF
EXECUTING:
train_model(rows: ?rs, cols: ?cs, in_cols: ?ics, out_cols: ?ocs, train_file: ?trf, optimization: ?opt, param_min: ?pmin, param_max: ?pmax);
CAN ?model_agent:
AT #now: train_model(rows: ?, cols: ?, in_cols: ?, out_cols: ?, train_file: ?, optimization: ?, param_min: ?, param_max: ?);
THEN
BEGIN_AT ?now: achieve(sender: #myself, receiver: ?model_agent, reply-with: R01, content: AT #now: train_model(rows: ?rs, cols: ?cs, in_cols: ?ics, out_cols: ?ocs, train_file: ?trf, optimization: ?opt, param_min: ?pmin, param_max: ?pmax));

In order to implement a connection timeout, a belief storing the time of the original message is needed, so the right hand of the rule should be modified to:

THEN
BEGIN_AT ?now: achieve(sender: #myself, receiver: ?model_agent, reply-with: R01, content: AT #now: train_model(rows: ?rs, cols: ?cs, in_cols: ?ics, out_cols: ?ocs, train_file: ?trf, optimization: ?opt, param_min: ?pmin, param_max: ?pmax));
BELIEF:
AT ?now: last_message_time(time: ?now, model: ?model_agent);

Two additional rules are described below, one for determining when a reply has been sent, and one for detecting a timeout, (in the example a timeout period of 20 seconds is assumed) :

IF
RECEIVED:
AT ?now: reply(sender: ?model_agent, receiver: #myself, in-reply-to: R01, content: AT #now: model_trained(status: ok));
THEN
BELIEF:
AT ?now: model_trained(status: ok);

IF
BELIEF:
AT ?now: last_message_time(time: ?last_t, model: ?mdl);
?now - ?last_t > 20;
THEN
BELIEF:
AT ?now: timeout(model: ?mdl);

For the case when two station agents S_0 and S_m are negotiating over the predicted value Y, and a similar timeout has occurred, then the S_0 agent should stop the negotiation with S_m and proceed to S_{m+1}:

IF
BELIEF:
AT ?now: timeout(station: ?stat);
THEN
FORGET:
BEFORE ?now: negotiate_with(station: ?stat);
COMMITMENT_TO #myself:
AT ?now: find_next_negotiator(current_station: ?stat);

4. Problem Solving Modules for air quality management

The architecture of the DNEMO multi-agent system, described in the previous section, deals with the problems of uncertainty reduction in environmental monitoring networks. However this mechanism, in order to be effective should incorporate adequate problem solving components. For this purpose we have integrated a number of alternative modules. based on various machine-learning techniques. In particular, our research effort has been concentrated in developing machine learning based AI systems that can solve the problem of NO_2 maximum concentration prediction using real monitoring data. Three approaches have been developed in this process (Kalapanidas, Avouris, 2001):

(a) a Case Based Reasoning (CBR) system using a weighted average for the new case adaptation,

(b) a CART decision tree constructor algorithm with a post pruning stage and

(c) an artificial neural network applying a back propagation training method.

Overall the CBR and CART classifiers achieved similar performance, with the neural network to follow at a close range. All algorithms were calibrated on the training set before they were evaluated on the test set. These three modules were used as problem-solving components, encapsulated in Model Agents in DNEMO. The modules currently included in the developed prototype do not however constrain the architecture to these particular problem-solving components. It is expected that in the future additional modules, based on statistical, symbolic or hybrid (e.g. neurosymbolic) problem solving methods (Hatzilygeroudis, Prentzas, 2000, Neagu et al., 2001), will be integrated in the proposed architecture, as they will become available.

A comparative study of the three developed algorithms is reported in this section which had the objective to determine the performance deviations of the three algorithms. According to (Dietterich, 1996), the Mc Nemar's test (Everitt, 1977) is suggested for situations in which the learning algorithms can be run once, giving the lower overall probability of a Type I error. This test is based on measuring the efficiency difference between two algorithms, and assumes as a null hypothesis the

fact that the two algorithms have the same error rate, that is they perform equivalently on the given test set.

For performing the McNemar's test, one should divide the available sample S into a training set R and a test set T. Two algorithms are trained over the R set producing the classifiers f_A and f_B. Then these two classifiers are tested on T. For each of the examples in T a confusion matrix is constructed as shown in the following table:

n_{00}: Number of examples misclassified by both f_A and f_B	n_{01}: Number of examples misclassified by f_A but not by f_B
n_{10}: Number of examples misclassified by f_B but not by f_A	n_{11}: Number of examples misclassified by neither f_A and f_B

The sum $n_{00} + n_{01} + n_{10} + n_{11} = n$ gives the total number of examples in T. Under the null hypothesis the two algorithms should have the same error rate, which means that $n_{01}=n_{10}$. McNemar's test is based on a χ^2 test for goodness-of-fit, comparing the distribution of counts expected under the null hypothesis to the counts that occur in the test set T. The expected number of counts is:

n_{00}	$(n_{01}+ n_{10})/2$
$(n_{01}+ n_{10})/2$	n_{11}

If the null hypothesis is correct, then the amount of $\frac{(|n_{01}-n_{10}|-1)^2}{n_{01}+n_{10}}$ is greater than 3.84146 at a probability of 0.05, distributed as χ^2 with 1 degree of freedom.

The McNemar's test is done on each of the 3 pairs of classifiers deduced from the three available algorithms. The corresponding confusion matrixes and the test results are presented in the following table:

	n_{00}	n_{01}	n_{10}	n_{11}	McNemar's test
CBR/CART	48	18	9	165	2.37037
CBR/ANN	54	12	20	154	1.53125
CART/ANN	48	9	26	157	7.314286

Table 1 Comparison of 3 alternative air-quality-prediction modules

Clearly from the test results, we cannot reject the null hypothesis, but for the last pair between the CART and the ANN classifiers. We can say in this case that they have not the same error rate, at a 0.05 probability. This is partially true however, because the number of tested classifiers are three, not just two. It is known that as one tests multiple classifiers, the probability of making a type I error increases. That means that among k multiple classifiers there are k number of chances to find some classifier as significantly better than a default classifier. This is known as the multiplicity effect. A solution to this problem is to lower the confidence level used to reject the null hypothesis, making the decision more strict. This is what the Bonferroni adjustment does, by considering a new significance level α*, as shown below:

$$1-(1-\alpha^{*})^{k} <= \alpha$$

where k is the number of tests, α is the original significance level (Salzberg, 1999). For α=0.05 and k=3, we compute the new criterion for our tests, which is lower than 5.5 for α^{*}=0.01695, so the rejection of the null hypothesis for the test between the decision tree classifier (CART) and the neural net classifier (ANN) still holds.
In order to get a graphical insight into the differences of the three classifiers, the chart of figure 6 has been produced.

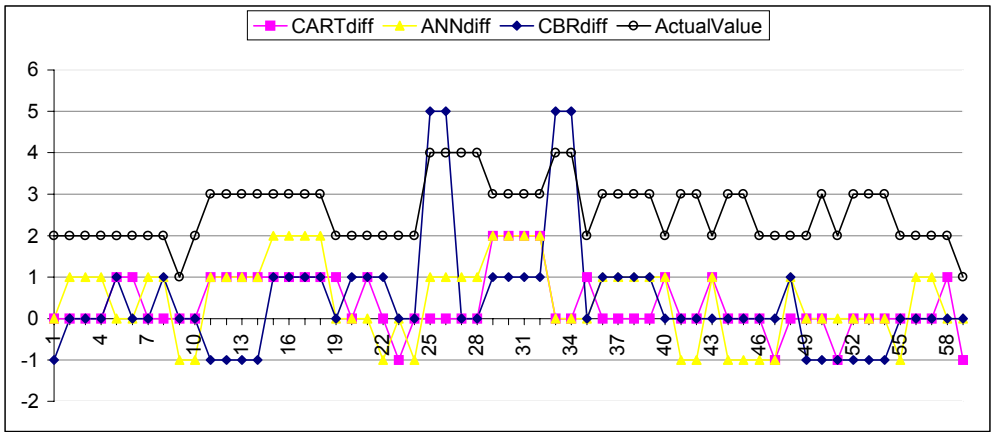

Figure 6 Comparative performance of alternative air quality prediction models

Figure 6 depicts the differences between the actual value and the predicted value from each of the algorithms, concentrating on specific cases where the predicted values are not the same (e.g. CARTdiff = ActualValue – CARTprediction). That

means that all cases where all algorithms predicted the same level are left out. In the chart there are areas where the actual values almost repeat themselves through time. These are interesting patterns demonstrating that in spite of the fact that the modules have overall similar performance, in certain cases their relative performance varies considerably.

The overall performance for the three algorithms favors the CART and CBR algorithms, while ANN follows at a close distance. The confusion charts come to a similar conclusion. CBR presents a better adaptation to the changing levels of actual output, but at some critical moments where pollutant episodes occur, CART is clearly more trustworthy. This study indicates therefore that there is no clear advantage in using only one problem-solving module. Following these observations it was decided to integrate all developed prototypes in the DNEMO system.

5. Discussion on DNEMO Performance

A multi-agent system that permits uncertainty reduction and maintains robust performance under adverse conditions for urban air quality management was described in this paper. The DNEMO system attempts to address in an innovative way common problems for environmental monitoring and problem solving. The approach proposed by this research is that of distributing intelligence. Complex interaction patterns at the level of the agents, combined with the complementary capabilities of the problem solving nodes, result in a powerful system, which is able to maintain a level of performance under adverse conditions. The first results reported are substantial: The system maintains high level of performance, similar to that of previous stand-alone systems even under conditions of simulated faults thanks to the algorithms and the techniques described in this paper. A dedicated simulator is currently under construction in our laboratory, designed to permit monitoring of system performance in a number of simulated conditions encountered in the Air Quality Monitoring Center. In the Center data streams from the spatially distributed data-loggers are gathered. Based on these data, the human experts can evaluate the DNEMO air quality prediction task. Using this simulator a more thorough test of system performance can be undertaken.

Typical scenarios of components failures that DNEMO can handle effectively are described in the following:

1. A Model agent fails while a Station agent S_k waits for a reply
   The reply timeout threshold is expired and the station agent S_k asks its closest neighbor S_l for its solution. If S_l also cannot produce prediction, S_k investigates all of its other neighbors from the closest one to the more distant one.
2. A Station agent S_k fails before or while negotiating with another agent.
   The reply timeout threshold is expired, and the current negotiation ends, passing to the next neighbor.
3. A Station agent S_k fails before submitting its prediction to the Collector
   The Collector asks a model agent to provide a solution, taking as input the vector of the existing predictions and as output the prediction given by the failed agent in the past. After the solution is provided, the missing prediction at S_k is replaced by the classifier output after the application of the data of the unsolved case.
4. All Station agents fail before submitting their prediction to the Collector.
   The remedy here is similar to 3. The difference is that the requested output is not a single prediction but a set of Station predictions. Thus only certain classifiers that comply with multiple output prediction are asked for help (the Neural Network being one of them).
5. The Collector fails. The human inspector can restart the Collector from the same machine or from another machine. All single agent types (including the Collector, Timewatcher and the DataFeeder) make a diagnostic check and try to recover any useful information during their initialization.

A discussion also needs to be done on agent allocation. According to (Lekkas et al., 1995), the decision on task allocation to agents and agent allocation to processors is a crucial one, affecting the efficiency and effectiveness of a multi-agent system. For instance in Figure 7 a proposal is included for DNEMO agent allocation to processors. According to this allocation scheme, no two agents of the same type use the same processing unit. Also, multiple model agents implementing the same algorithm should not share the same machine. Various combinations of this allocation problem and the implication on system performance are subject of further research.

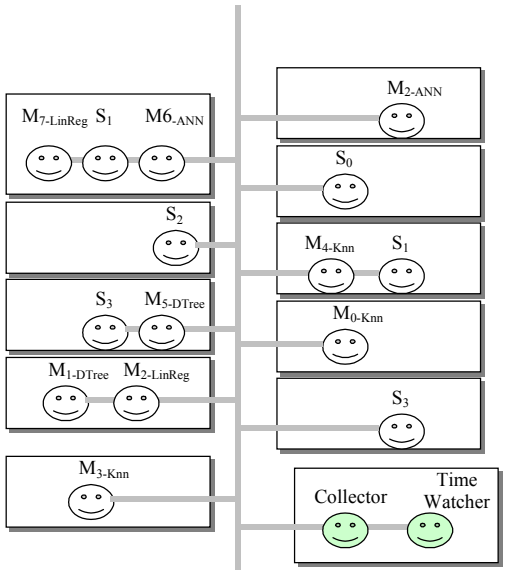

Figure 7 Study on agent allocation to processing units

In conclusion, this paper has proposed an architecture that can be applicable to many similar environmental monitoring problems, in which local solutions should be combined to a global one and many alternative problem-solving approaches can be used. Such environmental applications can be for instance water pollution management, waste site management, emergency management etc. The main advantage of this approach is the openness of the system in which new problem solving and monitoring components can be included while the system is in operation, the robustness in performance and adaptability to failure at run time, as demonstrated in the examples presented here. The proposed approach of combination of an open system architecture and distributed intelligence is suitable to many complex environmental problems and seems to be an appropriate new direction for A.I. and environmental informatics. This new direction, based on recent advances in distributed systems technologies and distributed intelligence theories and systems, we believe that can help us tackle some of the difficult environmental problems of our times.

Acknowledgements

The data used in this study have been supplied by the Athens Air Quality Monitoring Centre (PERPA), the National Observatory of Athens (NOA) and the National Meteorology Service of Greece. Special thanks are due to Dr L. Viras, air pollution expert of PERPA, for his continuing advice and support

during this ongoing research effort and to the four anonymous reviewers for their constructive comments. Financial support by EPPER4 Project - Specifications of the Athens Air Quality Monitoring Centre and GSRT PENED Project is also acknowledged.

References

van Aalst, R., Edwards, L., Pulles, T., De Saeger, E., Tombrou, M., Tunnesen, D., (1998), 'Guidance Report on Preliminary Assessment under EC Air Quality Directives, Technical Report 11, European Environmental Agency, Denmark.

Avouris, N.M, (1995), «Cooperating Knowledge-Based Systems for Environmental Decision Support», Knowledge-Based Systems Journal, vol. 8 (1), pp. 39-54.

Avouris, N.M, Page B. (1995), Environmental Informatics, Methodology and Applications of Environmental Information Processing , Kluwer Academic Publishers, Dordrecht, NL.

Boznar, M.; Mlakar, P. (1995), Neural networks - a new mathematical tool for air pollution modelling. In Air Pollution Theory and Simulation International Conference on Air Pollution - Proceedings v1., pp. 259-26, Billerica, MA, USA.. Computational Mechanics Publ.

Dietterich, T. (1996), Statistical tests for comparing supervised classification learning algorithms. Technical Report, Oregon State University, Corvallis, OR.

Everitt, B.S. (1977), The analysis of contingency tables. Chapman and Hall, London.

Finin T., Labrou Y., Mayfield J., (1997), KQML as an Agent Communication Language, in Bradshaw, J.M. (ed), Software Agents, MIT Press, Cambrisge MA.

Gauvin, D. (1995), "Un environnement de programmation oriente agent" Proc. 3eme journees francophones sur l'intelligence artificielle distribuee et les systemes multiagents, St-Baldoph, Savoie, France, 15-17 mars.

Huhns, M.N. Singh M.P. (1998), Readings in Agents, Morgan Kaufmann Publishers, San Francisco, CA.

Hatzilygeroudis I. and Prentzas J., (2000), Neurules: Improving the Performance of Symbolic Rules, Int. J. on Artificial Intelligence Tools (IJAIT), 9(1), pp. 113-130.

Kalapanidas E. and Avouris N., (2001), Short-term air quality prediction using a case-based classifier, Journal of Environmental Modelling and Software, vol.16/3, pp. 263-272.

King. J.M.P., Banares-Alcantara R., Manan Z.A. (1999), Minimising environmental impact using CBR: an azeotropic distillation case study, Environmental Modelling & Software, 14, pp. 359-366.

Lekkas G.P., Avouris N.M., Papakonstantinou G., (1995), "Development of distributed problem solving systems for dynamic environments", IEEE Transactions on Systems Man and Cybernetics, vol.25, 3, pp. 400-414.

Lekkas G.P., Avouris N.M., Viras L.G., (1994), «Case-Based Reasoning in Environmental Monitoring Applications», Applied Artificial Intelligence, vol. 8, No 3, pp. 359-376.

Neagu C.D., Avouris, N., Kalapanidas, E., Palade, V., (2001), Neurosymbolic Integration in a Knowledge-based System for Air Quality Prediction, Applied Intelligence (forthcoming).

Perantonis S.J., Vassilas N., Amanatidis G.T., Varoufakis S.J. and Bartzis J.G., (1994), Neural Network Techniques for SO_2 Episode Prediction, In S.-E. Grying and M. M . Milan (Eds.) Proceedings of the 20th Int. Tech. Meeting on Air Pollution Modelling and its Applications (Valencia, Spain, Nov. 1993), pp. 305-313. New York: Plenum Publishing Corp.

Ruiz-Suarez, J. C.; Mayora-Ibara, O. A.; Torres-Jimenez, J.; Ruiz-Suarez, L. G. (1995). Short-term ozone forecasting by artificial neural networks. In Advances in Engineering Software v23 n3 1995, p.143-149.

Salzberg, S.L. (1999), On comparing classifiers: A critique on current research and methods. In Data Mining and Knowledge Discovery 1, 1-12, Kluwer Academic Publishers, Boston.

Shoham Y, (1997), An overview of Agent-oriented programming, in Bradshaw, J.M. (ed), Software Agents, pp. 271-290, MIT Press, Cambrisge MA.
Soja G., Soja A-M, (1999), Ozone indices based on simple meteorological parameters: potential and limitations of regression and neural models, Atmospheric Environment, vol 33, pp. 4299-4307.
Stone P., Veloso M., (2000), Multiagent Systems: A Survey from a Machine Learning Perspective, Autonomous Robotics, vol.8, no 3.
Vasilas N., Kalapanidas E., Avouris N., Perantonis S., (2000), Intelligent Techniques for Spatio-Temporal Data Analysis in Environmental Applications, ACAI Proceedings, Springer-Verlag.
WHO (1987), Air quality guidelines for Europe, World Health Organization, WHO Publ., Copenhagen.
Wieland D. and Wotawa F., (1999), Local Maximum Ozone Concentration Prediction Using Neural Networks. In AAAI-99 Workshop on Environmental Decision Support Systems and Artificial Intelligence (W7), pp. 55-62.

APPENIDX – Acronyms used

AI = Artificial Intelligence

ANN = Artificial Neural Network

CART = Classification and Regression Tree

CBR = Case-based reasoning

DNEMO = Distributed NEMO the presented MAS architecture

KQML = Knowledge Query and Manipulation Language

MAS = Multi-agent system

ML = Machine Learning

SNP = Stations negotiation process