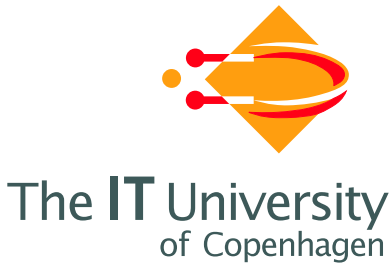

Meeting Ecologists Requirements with Adaptive Data Acquisition

Marcus Chang
Philippe Bonnet

IT University Technical Report Series TR-2009-TR-2009-122

ISSN 1600–6100 **November 2009**

ISSN 1600–6100

ISBN 9788779492042

Copies may be obtained by contacting:

IT University of Copenhagen
Rued Langgaards Vej 7
DK-2300 Copenhagen S
Denmark

Telephone: +45 72 18 50 00
Telefax: +45 72 18 50 01
Web www.itu.dk

Abstract

Ecologists instrument ecosystems with in-situ sensing to collect measurements. Sensor networks promise to improve on existing data acquisition systems by interconnecting stand-alone measurement systems into virtual instruments. Such ecological sensor networks, however, will only fulfill their potential if they meet the scientists requirements. In an ideal world, an ecologist expresses requirements in terms of a target dataset, which the sensor network then actually collects and stores. In fact, failures occur and interesting events happen making uniform, systematic ecosystem sampling neither possible nor desirable. Today, these anomalous situations are handled as exceptions treated by technicians that receive an alert at deployment time. In this paper, we detail how ecological sensor networks can adapt to anomalies and maximize the utility of the collected datasets. More specifically, we present the design of a controller that continuously maintains its state based on the data obtained from the sensor network (as well as external systems), and configures motes with parameters that satisfy a constraint optimization problem derived from the current state. We describe our implementation, discuss its scalability, and discuss its performance in the context of a case study.

I.2.8Computing MethodologiesArtificial Intelligence[Problem Solving, Control Methods, and Search]

C.2.4Computer Systems OrganizationComputer-Communication Networks[Distributed Systems]

Design

Autonomous System, Constraint Optimization Problem, Planning, Scientific Data, Wireless Sensor Networks

1 Introduction

For years, ecologists have deployed in-situ sensing infrastructures to observe and monitor the biotic and abiotic factors in a given ecosystem. They primarily rely on fixed data loggers to collect and store data from a wide variety of sensors. They have been promised that low power wireless sensor networks would be able to provide them with sampling at unprecedented scale and resolution [10]. However, the MEMS revolution has not yet delivered a radical change of the optical, biological and chemical sensors that are pervasive in ecological monitoring, and scientists cannot afford high density deployment of the current generation of sensors, which are still bulky, energy hungry and expensive. Still, low power wireless networks can have a tremendous impact on ecological monitoring by transforming stand-alone devices into a networked system that is monitored and controlled to meet the scientists requirements. In this paper, we study how ecological sensor networks can be steered to improve the utility of the collected datasets.

Ecologists rely on in-situ sensing to collect datasets of the form $(t, x, y, z) \rightarrow (v_1, v_2, ..., v_n)$, where the independent variables represent time (t) and space (x, y, x), and the dependent variables correspond to the modalities of the sensors deployed. These raw measurements are the foundation of the scientific

workflow. They are tagged with metadata, and transformed into derived data products via calibration, verification, or extrapolation processes. The derived data products are then used for modeling purposes. The derivation processes and the models are applied in the lab, as a post-processing phase, based on the primary data collected in the field. If an offline verification process exposes a sensor failure then the collected data is useless. If a model gives evidence of interesting events, then the collected data might not be dense enough (in space, time or modality) to allow a deep analysis of the phenomenon. In this paper, we propose to move portions of the existing offline scientific processes online, within the ecological sensor networks in order to improve the quality of the collected data. Specifically, we propose that anomalous situations should be recognized and handled online, while data is collected, so that the sensor network can adapt and maintain high utility.
Consider a scientist that monitors a lake. She is interested in measuring conductivity and temperature at five different depths, at a sampling rate of one measurement per hour, for a month. This is her initial requirement based on the dataset she wishes to collect. However, if we go further and consider potential anomalous situations, we obtain a much more complete picture:

- *Failures*: She can tolerate that measurements are taken up to once every six hours; however below that threshold, measurements are useless. Also, she requires that both conductivity and temperature are measured together; if either measurement is missing the other is useless. She indicates a valid range for conductivity and temperature measurements; measurements outside these ranges should be considered errors. Conductivity errors might be compensated by either repeating a measurement within a few seconds, and if that fails resetting the conductivity sensor. Temperature errors might be compensated by looking up the temperature at an adjacent depth. In addition, a sensor should not be considered damaged if its measurements drift in time; regular manual samples are taken periodically to compensate for such errors.

- *Interesting events*: The scientist indicates that she is interested in thermoclines (rapid changes in the temperature within a limited depth range) - so if possible, measurements should be taken at additional depths if a thermocline is detected (given a simple temperature variation threshold for detecting thermoclines). Also, in case of a storm (signaled by the RSS feed of a close-by weather station), the sampling rate should be increased to twelve measurement per hour for the two depths that are closest to the surface of the lake. The scientists notes, however, that if energy is limited the baseline measurements should have priority.

The core of the problem is that ecological data acquisition has been based on the premise of systematic ecosystem sampling: it is assumed that the ecosystem is sampled with given modalities at predefined intervals in time and space. This is neither possible (because of failures), nor desirable (because interesting events might not be captured by the baseline settings). In contrast, we propose that

ecological sensor networks should rely on adaptive ecosystem sampling, where the procedure for selecting samples may depend on the values of observed variables [26]. More specifically, we propose an ecological sensor network controller that checks the measurements it collects and adapts how the next measurements should be obtained (in time, space and modality) to maximize their utility for the scientists [25]. Now, the questions are: (1) How can scientists represent the utility of measurements? (2) How can such a controller operate to maintain high utility at reasonable cost in a changing environment? We address these questions in this paper. Our contribution is the following:

1. We capture the scientist requirements in terms of data collection modes. For each data collection mode, the scientist describes a range of acceptable parameters. Utility is represented as a ranked preference of these data collection modes.

2. We describe a controller that continuously maintains its state based on the data obtained from the sensor network (as well as external sources), and configures motes with parameters that satisfy a constraint optimization problem derived from the current state.

3. We adopt a three-tier architecture, developed for autonomous systems, in the context of a sensor network controller.

4. We detail the implementation of the ADAE system based on this design, and discuss how it scales.

5. We describe a case study based on an actual deployment for lake monitoring.

2 Related Work

In this section, we look back on the evolution of data acquisition based on sensor networks, we discuss the previous use of adaptive sampling in sensor networks, and we review existing work on autonomous systems controllers.

2.1 Data Acquisition with Sensor Networks

Cougar [3] and TinyDB [19] introduced a distributed query processing paradigm for sensor network data acquisition. The goal was to ensure a flexible tasking of motes via a relational query interface. The assumption was that (a) the relational model was appropriate to capture sensor data, (b) that users would submit queries to task motes, and (c) that in-network processing was necessary in the context of a sensor network. The relational query interface is not a good abstraction for ecologists that do not wish to query the sensors but aim at systematically collecting primary data sets for their scientific processes (see [20] for a thorough discussion of the limitations of these approaches).

BBQ [16] and MauveDB [8] introduced the notion of *model-based querying* as an abstraction for data acquisition. The system maintains a statistical model of the data, and instead of blindly collecting time series, only collects the data that are needed to improve the precision of the model. For instance, correlations across modality are leveraged to reduce the cost of data collection as expensive measurements are replaced by cheaper ones. Users obtain probabilistic, approximate answers to their queries. Such approaches are not relevant for ecologists since they are the ones discovering new models and thus need primary datasets as a foundation for their scientific processes.
PRESTO [18] further develops the idea of model-based querying. The PRESTO gateway constructs a seasonal ARIMA model of the time series collected at a given sensor. In order to maintain these models, PRESTO combines the pull approach from MauveDB (data is collect as needed to improve precision), with a push approach, where each sensors use the model parameters defined by the gateway to predict future values and sends data to gateway in case an anomaly is detected (i.e., there is a significant difference between a predication and the actual measurement). The gateway refines the sensor model as it receives new measurements to reflect changes in the sensed data. We share with PRESTO this focus on anomaly detection and on adaptation to a changing environment. PRESTO returns approximate answers that match the confidence interval specified by users. The rationale behind the design of PRESTO is to improve energy efficiency, not to maximize utility for users.
Lance [27] introduced utility-based controllers in the context of sensor network data acquisition. This system focuses on the collection of high-bandwith signals, where not all the data acquired by the motes can be transmitted to the base station. Lance controls bandwidth usage by splitting the data acquired at each mote into a sequence of data packets, and making sure that only the most relevant data packets are transferred back to the base station. The selection is performed by the base station based on summaries sent by motes and on a trade-off between cost and utility provided by the user. We share with Lance a focus on optimizing the utility of the collected data. The fundamental difference is that in Lance the optimization problem concerns the deletion of data collected in a predefined way, while in ADAE the optimization problem concerns the selection of the data collection parameters (e.g., sampling rate, sensor placement, modality). Those two problems are orthogonal. In terms of architecture, Lance focuses on flexible policy modules, while ADAE relies on the three-tier architecture – both aspects are complementary. In the rest of the paper, we assume that all data collected at the motes can be transmitted to the gateway.

2.2 Adaptive Sampling in Sensor Networks

In statistics, adaptive sampling designs are those in which the selection procedure may depend sequentially on observed values of the variable of interest [26]. In the context of sensor networks, adaptive sampling has been introduced to (a) maintain high resolution while covering large regions of space using mobile sen-

sors, e.g., light sampling with Networked Infomechanical Systems (NIMS) [4], or to (b) reduce approximation errors with additional samples taken by mobile sensors, e.g., weather forecasting with autonomous UAVs [7]. Compared to these approaches, we do not seek to improve resolution with a reduced number of sensors, but to maintain utility of measurements in a changing environment. Our challenge is to take a decision on when, where or how to sample whenever the environment changes, rather than gradually improve the resolution of a given model.

2.3 Autonomous Systems

Autonomous systems constitute a popular research topic in the areas of AI and robotics. The most interesting developments have been achieved in the area of autonomous controllers, with contributions ranging from the seminal work on Deep Space 1 [9] to the Mars Rover [1]. An architecture for autonomous systems has emerged [2] based on the following three tier architecture: the bottom tier is the functional layer that is the interface with sensors and actuators, the middle tier executes the planned actions and check their effects, and the top tier implements the planning and scheduling functionalities. As we discuss in Section 0.4.3, we adopt a similar architecture for the ADAE system. Note that NASA has now made publicly available the platforms they developed for their autonomous systems, e.g., Apex [12] or Europa [11]. We did not use these systems because they did not support the type of solver we envisaged for our planner, and because implementation constraints did not allow us to deploy these systems on our target gateway[1]

3 The Ecologists Requirements

We aim at designing a system that autonomously adapts data acquisition to meet the ecologists requirements. In this Section, we detail those requirements. Note that our goal is not to define a rigid template for software engineering purpose, but to put a stick in the ground regarding the scientists expectation of an ecological sensor network.

3.1 Data Collection Modes

Ecologists rely on in-situ sensing to collect primary datasets. In the case of manual sampling, they define a protocol that ensure the relevance, quality and consistency of the collected data. In case of automatic sampling, they have to express requirements to the monitoring system. These requirements are based on the description of the target datasets, $(t, x, y, z) \rightarrow (v_1, v_2, ..., v_n)$, described in the Introduction, where the time domain defines the sampling rate as well as

[1] Apex relies on multi-threaded Lisp, which was not available on the Linux-based platform we used for our deployment.

the lifetime of the deployment, the space domain defines sensor placement, and the dependent variables define sensor modalities and accuracy.
The traditional requirement is that given a dataset description, all data must be stored, i.e., the whole dataset must be collected [21]. The problem with this requirement is twofold. First, it defines an ideal goal. In case of failure, the monitoring system will not be able to deliver the target data set. A consequence is that system designers tend to assume that yield (what percentage of the target dataset is actually collected) is an appropriate metric for system performance. For ecologists however, the relevance of a dataset is not proportional to its yield. In our experience, they identify portions of the collected data set that they can use for modeling purposes, and portions that are useless - typically because the dataset is locally too sparse (in time, space or modality). Second, the requirement of uniform, systematic dataset collection does not account for interesting events. Such events are arguably the most interesting elements of a dataset. Their analysis might require denser sampling in time, space or modality for a limited period of time.
For example, consider the soil monitoring project, "Life Under Your Feet" [21]. This sensor network consists of more than hundred TelosB motes, each equipped with two temperature and moisture sensor probes. These four probes are dug into the soil at specific depths and sampled every ten minutes. This is fast enough to monitor moisture evaporation but not precipitation, which changes within seconds and not minutes. Obviously, continuously sampling at a rate capable of capturing rain events would significantly strain the power supply. However, using external sources such as local weather forecasts and only increase the sampling rate when the chance of a rain event is significant would be far less expensive power wise.
To overcome these limitations, our goal is to (a) capture an envelope of datasets relevant for the ecologists in the context of a given deployment, and (b) a means of representing the scientists preferences within that envelope.
We propose to capture the ecologists requirements as a ranked list of *data collection modes* (e.g., baseline, degraded, failure, event detection). Some of the modes are exclusive (e.g., baseline and degraded), while others can be active simultaneously (e.g., baseline and failure or event detection). For each collection mode, the ecologists defines:

1. A description of the conditions that must be satisfied to activate or deactivate these modes. A condition is specified using a rule (e.g., humidity inside a mote is greater than 50%), a model (e.g., Echo State Network for anomaly detection with a training set specified by the scientist [6]), or a timing constraint (e.g., within five minutes or for five hours).

2. A target dataset, i.e., its time component (lifetime and sampling rate), its space component (sensor location), and its dependent variables (modality and accuracy). Note that, for data collection modes associated to failures, the target data set specifies relevant redundancy in time, space or modality.

3. A sparseness threshold for each modality, i.e., the number (or distribution) of usable measurements per chunk of time and space.

The ranking of the collection modes defines an ordinal utility function. Despite our insistence, none of the scientists we are collaborating with could find a non-trivial cardinal utility in the context of their activity. In addition to these data collection modes, the ecologists defines a target *lifetime* for data collection.
For example, we derive from [21], the following requirements for "Life Under Your Feet":

- We define the following **data collection modes**: *baseline*, *precipitation*, and *fault* modes (which should be defined with the ecologist).
- The *baseline* is always present, while the **condition** for the *precipitation* event is when the weather forecast predicts rain and the *condition* for the *fault* are humidity in the mote greater than a given threshold, out of bounds measurement.
- The **target dataset** for the *baseline* is the four probes sampled every 10 minutes, for the *precipitation* it is every minute instead. For the fault modes, we would need to identify redundancy in time, space or modality.
- We set the **sparseness threshold** to six set of samples every hour in each data collection mode.

We derived this form of requirements from our collaboration with ecologist. When asked about their requirements all scientists initially defined a single ideal target dataset. When faced with the fact that failures might occur, they came up with a form of sparseness threshold, and the definition of one or several degraded modes. They expressed interesting events characterized by simple conditions (external events or simple thresholds on the sensed data).

3.2 The Case for Autonomous Data Acquisition

The solution promoted in commercial data acquisition systems to tackle failures and anomalous situations consists in involving human supervision. Let us go back to the lake monitoring example from the Introduction. A buoy is deployed equipped with a data logger that stores the data it collects at a predefined sampling rate from the CTD sensors (conductivity, temperature, depth) deployed at five different depths. The data logger is equipped with long-range wireless communication and it acts as a server for telemetry and tele-command, possibly alerting a technician in case of problems and accepting commands and configuration operations. This design, which is the state-of-the-art in ecological data acquisition, is however flawed in several respects:

1. Contingency planning is weak. In case the data logger detects an anomalous situation, it raises an alert and it is up to the technician to handle it. This is a best effort approach, where response to anomalies is unspecified

and variable. In our experience, the resources available for monitoring purposes do not allow 24/7 supervision. Because, long-range communication and technician supervision are expensive, the data logger is programmed to send alerts in limited cases. The system is not configured to compensate for errors or to react to interesting events.

2. No graceful degradation. When energy supplies are low, data acquisition continues at the predefined sampling rate at the risk of thrashing. More generally, the assumption is that the system has a single regime, and that human intervention is needed to keep this regime operational in case of failure.

3. The system is stand-alone. Co-located data loggers are not interconnected, thus possibly missing opportunities for increased redundancy, and detection of interesting events.

Our goal with this work is to limit human intervention to the initial requirements, and let an autonomous data acquisition system handle anomalies with a controller that continuously adapts to changes in the environment, and tasks motes to maximize the utility of the measurements[2]. We detail the design and implementation of such a controller in the next Section.

4 The ADAE System

ADAE is an autonomous gateway-based controller that tasks motes to keep on maximizing utility in a changing environment. Before we describe its design, architecture and implementation, we address the following question: Which actions can ADAE take in order to control the sensor network?

4.1 Sensor Network Model

We model an ecological sensor network as a cluster of motes connected to a gateway. We adopt a classical two-tier model [18, 27], where motes are slaves, tasked by the gateway-based controller to sample, store and transmit data. We do not consider any form of in-network aggregation or storage (beyond local computation or storage on the mote that produces data). We further assume a best effort delivery between mote and gateway (e.g., CTP [13]) that allows the gateway to collect routing statistics. Finally, we assume that each mote is appropriately duty cycled (based on the sampling rate and offload rate), and that it is accessible (using a form of low-power listening [24]).
We also assume that the sample, store and transmit tasks are accomplished by a program deployed on all motes, and that this program can be configured with parameters to modify the sampling or transmission policy. We make this

[2] Obviously, when possible and affordable, human intervention should be used to maintain the optimal regime. Our point, here, is that the system should maintain high utility and degrade gracefully when the optimal regime is no longer sustainable.

assumption because it allows for a straightforward integration of legacy systems (including the current generation of commercial motes). Leveraging rich mote APIs or mote reprogramming (via tasklet distribution [14] or full image reprogramming [15]) is an issue for future work.
We introduce virtual sensors to abstract the details of the actual motes[3]. Each virtual sensor represents a modality of a given mote (we describe virtual sensors in more detail below). Virtual sensors export a single API function, that defines the space of possible controller actions (note that such actions must be mapped to the API exported by the actual motes):

- $configure(VS, SR, TR)$ to configure the sampling rate (SR) and transmission rate (TR) on a given virtual sensor (VS).

4.2 Controller Design

The key questions that we need to address are: (a) What is an appropriate abstraction of the sensor network?, (b) How to represent user requirements, i.e., data collection modes and utility?, (c) How to define cost?, and (d) How to plan a sequence of actions given the controller state and the user requirements.

4.2.1 Virtual Sensors

Conceptually, the following relations can be used to organize the state variables representing a sensor network:

```
VirtualSensor(VS_id, Modality, Mote_id,
       X, Y, Z, SampleTime, SampleEnergy,
       TransmitTime, TransmitEnergy)
VirtualSensorState(VS_id,
       SamplingRate, TransmitRate)
Mote(Mote_id, BatteryCapacity, VB_id,
       Forward, Overhear)
Topology(ParentMote_id, ChildMote_id)
VirtualBattery(VB_id, LifeTime, Percentage,
       NbInstallments, Credit, MaxBurnRate)
VirtualBatteryState(VB_id, Balance,
       InstallmentsDone, MaxOverDraft)
```

Each virtual sensor is identified by a VS_{id} and represents the $Modality$ of a physical sensor attached to mote $Mote_{id}$, at a fixed location X, Y, Z[4]. A calibration phase defines the time and energy it takes to make a measurement with the given modality ($SampleTime$, $SampleEnergy$)[5], as well as the time and energy required to transmit one measurement ($TransmitEnergy$, $TransmitTime$). Note that such normalized time and energy attributes correspond to the notion of

[3] Our notion of virtual sensor is inspired by Franklin et al. [20]

[4] For static sensors, (X, Y, Z) are given at deployment time, while for mobile sensor a valid range and possibly constraints are given for these variables.

[5] Note that we rely on constraints to indicate the dependencies that may exist between modalities on a same physical sensor.

platform vector introduced in [17]. The *VirtualSensor* relation is configured at deployment time and remains unchanged thereafter.
Virtual sensors are configured with two data acquisition parameters: the sampling rate ($SamplingRate$), and the transmission rate ($TransmissionRate$)[6].
For each mote, we store the capacity of the battery it contains ($BatteryCapacity$) as well as the reference of a virtual battery VB_{id}, which is the abstraction [5] that we rely on to reason about energy allocation. A calibration phase allows to define for each mote, the cost of forwarding or overhearing a measurement.
We use a simple representation of the collection routing tree using the topology relation. Note that our assumption here is that the topology observed at a given time is a good predictor of the topology in the subsequent epoch. Obviously, we do not capture network dynamics with this model, but this is not our goal. Our goal is to represent topology and transmission costs to estimate the cost of a data acquisition plan (see the discussion of our cost model below).
Each virtual battery is characterized by the $Percentage$ of the total capacity associated to sampling, forwarding or overhearing (with a given mote). Virtual batteries are further specified with a $LifeTime$ requirement (given by the user), a total number of installments ($NbInstallments$), a $Credit$ rate that allows to specify an energy allocation policy ($Credit$ is a real number in the interval [0, 1], e.g., 0 corresponds to a conservative policy that only grant energy installments when there is an energy surplus, while a positive credit rate corresponds to a policy that allows energy deficit up to $Credit$), and a maximum allowed energy burn rate $MaxBurnRate$. The virtual battery state relation is used to maintain the actual energy $Balance$, the number of $Installments$ already received and the maximum overdraft allowed ($MaxOverDraft$ which is a negative number).

4.2.2 Physical Limitations

The controller maintains a set of constraints over virtual sensors that reflect the actual limitations of the physical system. Those constraints concern the range of possible values for the location parameters (X, Y, Z), the sampling rate ($SamplingRate$), and the transmission rate ($TransmissionRate$). In addition, a physical mote is represented as several virtual sensors, one for each modality. We capture the serial or parallel constraints that exist between co-located virtual sensors, in terms of location and in terms of timing of the measurements. In order to represent the timing constraints, we do not need the whole power of the event calculus [25], we just need to represent constraints on serial or parallel executions. We thus introduce variables $t_{i,j}$ that represents the time required before VS_j can take a measurement after VS_i and use integer constraints to represent these timing constraints. To sum up, we represent the physical limitations of the sensor network using two types of constraints: (1) domain restrictions, and (2) integer constraints based on virtual sensors variables.

[6]Note that we force transmissions to be triggered by a time-based condition (the transmission rate) instead of a more general form of condition (e.g., transmit when mote storage is half full) and thus sacrifice flexibility for predictability.

4.2.3 Data Collection Modes

To represent data collection modes, the controller maintains:

- A set of predicates $\mathcal{P}$ to represent the conditions that activate and deactivate the given data collection modes. The controller implements the rule-based, model-based or time-based methods specified by the users to evaluate these predicates.
- Specific constraints imposed by the sparseness threshold for the given data collection modes. These constraints are expressed as restrictions of the state variables domains $\mathcal{D}$ (e.g., $X \in [1..100]$).

4.2.4 Utility Model

We base our controller on the principle of maximum expected utility [25]. Each action taken by the controller configures motes in conformance with one of the data collection modes described by the ecologist. We associate a cardinal utility to each action, based on the ranked preference among the resulting data collection modes. This utility function is a simple scoring function with uniform spacing (for N modes, the scoring function is such that the top ranked mode gets a score of N, and the bottom ranked mode gets a score of 1). Using the binomial distribution, we model the probability of success for a configuration as the probability of collecting a number of samples higher than the sparseness threshold: $1 - \frac{1}{\left(\frac{SamplingRate}{SparsenessThreshold}\right)}$ (where both $SamplingRate$ and $SparsenessThreshold$ are defined in numbers of samples for a given epoch Δ). The sparseness threshold might be defined for several modalities (i.e., several virtual sensors) within a given data collection mode, so we select the lower probability and multiply it by the rank to obtain the expected utility for that data collection mode.

4.2.5 Cost Model

The controller associates a cost to each virtual sensor configuration, based on the energy used to sample, transmit and overhear measurements. We adopt a variation of the cost model introduced in Lance [27], and represent the cost δ_j of a virtual sensor configuration VS_i per virtual battery VB_j. The source mote to which VS_i is associated incurs a sampling and transmit cost, while motes on the communication path incur a forwarding cost, and motes one hop away from the communication path incur a overhearing cost. For a given period of time T,

- Sampling and transmission cost on the source virtual sensor is estimated as $(SampleEnergy_i + TransmitEnergy_i) * (SamplingRate_i * T)$, i.e., the product of the energy cost of obtaining a measurement with the number of measurements in the period.
- Forwarding cost is estimated as $Forwarding_j * (SamplingRate_i * T)$.

- Overhearing cost is estimated as $Overhear_j * (SamplingRate_i * T)$. It is associated to the transmission virtual batteries of all physical motes in the neighborhood of the forwarding motes.

We introduce integer constraints derived from the virtual battery energy allocation model: for a given time period T, (1) the balance is greater than the maximum overdraft ($Balance - \delta_j > MaxOverDraft$), and (2) the amount of energy spent by VS_i is bound by the maximum burn rate ($\delta_j \leq MaxBurnRate_i * T$).

4.2.6 Planning Problem

Now, the question is: How does the controller pick appropriate actions given its current state? Because the controller operates in a changing environment, it needs to proceed online, i.e., select some actions at one point in time and evaluate their impact regularly, possibly selecting new actions to react to a change in the environment. We call epoch, noted Δ, the period of time after which a given action is reevaluated (note that our cost model and utility function are defined for limited time frames). A default epoch size is given as a system parameter. Note that an epoch is shorter than the default, in case a data collection mode predicate requires it (e.g., the actions following a failure might be valid/relevant only for a short period of time). We impose a constraint that the period corresponding to the transmit rate is lower than (or equal to) the epoch Δ.
For each epoch, virtual sensors have a fixed configuration (i.e., fixed location, fixed SR and TR). The actions generated, for a given epoch, are thus a collection of at most one API call per virtual sensor. The planning problem is thus reduced to a constraint optimization problem, where the controller must find values of the state variables that satisfy all the constraints, maximize expected utility and minimize cost: ($\mathcal{V}$, $\mathcal{R}$, $\mathcal{C}$, $\mathcal{U}$), where $\mathcal{V}$ represent variables (i.e., all the attributes from the virtual sensor relations), $\mathcal{R}$ are the restrictions on these variables (either given by the system model, the cost model or the user requirements), $\mathcal{C}$ are the constraints (i.e., physical limitations, or virtual battery constraints) and $\mathcal{U}$ is the expected utility. The size of the search space grows exponentially with the number of virtual sensors $\mathcal{O}(sr\_range \cdot 2^N)$, where N is the number of virtual sensors and sr_range is the average size of the $SamplingRate$ domains

4.3 System Architecture

Our controller needs to address three sub-problems:

1. How to update the controller state?
2. How to generate the appropriate constraint optimization problems when appropriate?
3. How to solve the given constraint optimization problems?

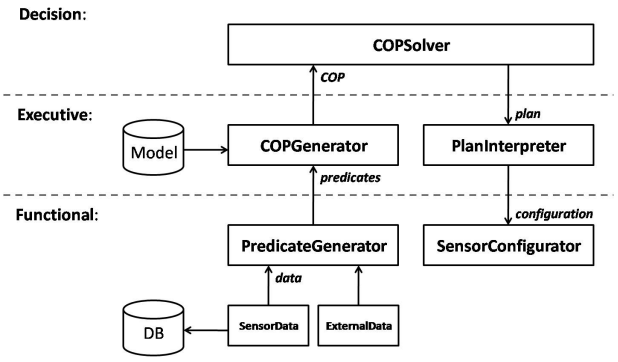

Figure 1: Architectural overview of ADAE.

In order to tackle these problems, we structure our controller using the classical three-layer architecture developed for AI planning [2]:

- Functional Layer, which provides abstractions for the motes, the sensor tasks, the storage subsystem, and the detection modules. Its interface is generic, but its implementation is deployment-specific.
- Executive Layer, which checks the collected data, call the decision layer if a new plan is needed, and transmits the plans from the decision layer to the functional layer. Both its interface and implementation are generic.
- Decision Layer, which produces a new plan based on the data it gets from the executive layer. The decision layer is composed of a generic solver and of a deployment-specific model.

The flow of information in the individual components in this three-layer architecture is illustrated in Figure 1. Data generated from the sensor network (both measurements and network status) is collected by **SensorData**. This data is stored in a local database, and passed along to the upper layers of the controller. We use virtual sensors to present a uniform abstraction to the upper layers of the controller. One issue, though, is to map the data received from actual motes into data associated to virtual sensors. This mapping is straightforward for stationary sensors since there is a direct one-to-one mapping between virtual sensors and the modality of a mote at a given location. Mobile sensorsOn the other hand, have a one-to-many mapping, were each distinct location of a mobile sensor corresponds to a different virtual sensor. The data associated to virtual sensors is then passed on to **PredicateGenerator**. Information from external sources, such as weather forecasts and time and date specific events are collected by **ExternalData**. This data is passed on to **PredicateGenerator**. In **PredicateGenerator**, detection and mapping algorithms are used to transform the time series, network status, and external data into predicates. In terms of architecture, one or several detection modules are attached to each virtual

sensor. For example, the range of each measurement value can be checked and if some are found to be out-of-bounds the *OutOfBounds* predicate is set to true. The conditions described by the ecologist are also checked at this point with each condition generating its own predicate.
These predicates are passed on to **COPGenerator** where they are used to represent the current state of the system. The role of this component is twofolds. First, it maintains the state of the virtual sensor and virtual batteries. Second, it constructs a COP that reflects this state, and incorporates the constraints as well as utility function from the set of data collection modes activated by the predicates that evaluate to true. Note that we generate a single COP for the entire network in order to account for forwarding and overhearing costs.
This COP is then passed on to **COPSolver** which tries to find a sensor configuration that satisfies all the constraints of the COP and at the same optimizes the expected utility and minimize cost. In ADAE, we model our COP using the MiniZinc [22] constraint programming language which allows us to define our COPs at a high level of abstraction. This gives us the flexibility to switch between different engines depending on performance and platform availability. The solving of the COP is accomplished in two-steps. First, the COP formulated in MiniZinc is translated to the FlatZinc language, a lower level constraint programming language. Second, a generic solver with a FlatZinc parser is used to solve the COP. The downside of using a generic language such as MiniZinc is the added overhead from the intermediate step and the missed opportunity to leverage solver specific performance enhancements, i.e., specific API calls.
The resulting plan is passed on to **PlanInterpreter** where a configuration is generated for each mote and using **SensorConfigurator** each mote in the network is reconfigured. Similar to the mapping process in **SensorData**, the configurations for the virtual sensor abstraction are transformed into commands and configurations specific to the physical sensor network. Virtual sensors corresponding to sensors with fixed locations are mapped directly, while for virtual sensors representing mobile sensors, the robot carrying the sensor is instructed to follow a path connecting the virtual sensors. A cache of all the current configurations are kept and motes are only reconfigured if there are any changes. Whenever possible, **SensorConfigurator** requires that motes piggyback their energy status on the data they transmit. Such energy status are identified by **SensorData** and used by **PredicateGenerator** to update the virtual battery state.

5 Evaluation

The key question from a performance point of view is whether our approach based on generating and solving Constraint Optimization Problems is viable, specially in a multi-hop setting with a cluster of 40-50 motes. This is the question we address in this Section. Our implementation of ADAE is publicly available[7] and based on the standard C++ library to ensure portability. We

[7] http://code.google.com/p/adae/

only use the MiniZinc-to-FlatZinc translator provided by [22] and not the corresponding FlatZinc solver. Instead we use the Gecode solver with the FlatZinc interface[8], since it has better performance, supports are wider range of platforms, and allows a more controlled search process. All benchmarks are run on an Intel Core 2 T7600 2.33 GHz processsor.

5.1 Sanity Check

5.1.1 Constraints

Because we are considering Constraint Optimization Problems, we expect the resource constraints (i.e. time and energy) to have a significant dual impact on the search space. On one hand, tight resource constraints will limit the search space by rendering certain states inaccessible, and thus reduce the runtime. On the other hand, loose resource constraints will make even the high utility states accessible, giving the full benefit of the optimization directed search. We thus expect the search space to be largest when the resource constraints are neither restrictive enough to render a significant portion of the state space inaccessible nor loose enough to make the states with the highest utility available.
Of course, this only holds if the cost/benefit relation between time/energy and utility is positive, i.e., states with higher utility requires more resources than states with lower utility. With a negative relation, tight resource constraints would lead to the benefits of both a small state space and a optimization directed search, while a loose constraint would have neither. For the remainder of this Section we choose a positive relation since this seems most applicable, i.e., higher cost yields higher utility.
In Figure 2 we show the runtime for three different COPs, with varying energy constraints. Because the time constraint are modeled completely analog we only consider the energy. The number of motes and available sampling rates are all fixed at one for all three problems.
The energy constraints are set as a fraction of the maximum energy required for the most demanding state, since this depends on the number of virtual sensors. As expected, there is a significant difference in runtime when the constraints are varied. Specifically, there is a difference of three orders of magnitude between the COPs with no energy constraints (100%) and the ones with exactly half available (50%). This confirms our initial analysis that the search space is largest when neither the constraints nor the optimizations can be used to minimize the search space significantly.

5.1.2 Virtual Sensors

We know from Section 0.4.2 that the size of the state space grows exponentially with the number of virtual sensors and linearly with the number of available

[8] Generic Constraint Development Environment. http://www.gecode.com/

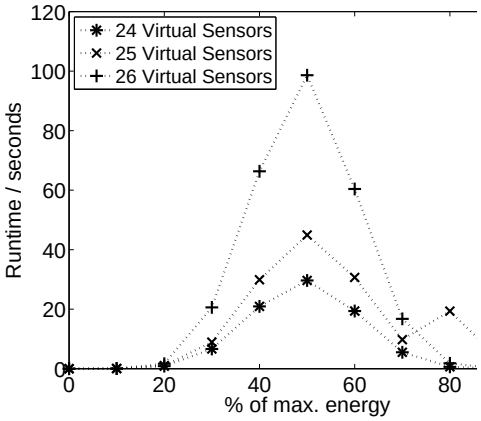

Figure 2: The effect constraining resources has on the search space.

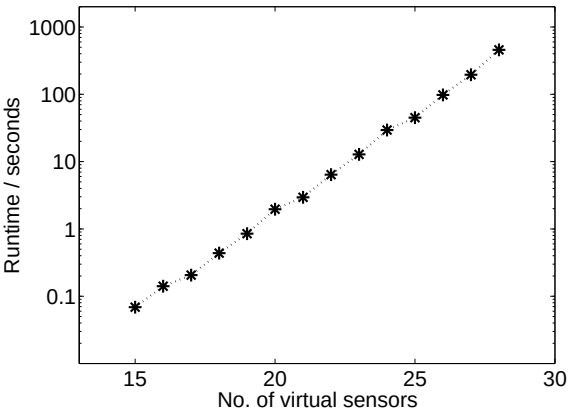

Figure 3: The influence of virtual sensors on the search space. Note the logarithmic scale.

sampling rates. On the other hand, the number of motes only effects the shape of the state space. We thus expect the former to have a significant impact on runtime while the contribution from the latter will mostly be overhead from book keeping.
We explore the scalability with regards to the number of virtual sensors in Figure 3. As before, we keep the number of motes and sampling rates fixed at one. With the new information above, we set the energy constraint to 50% of the highest energy state in order to explore the largest search space. As expected, the runtime grows exponentially with the number of virtual sensors (note the logarithmic scale).

5.1.3 Sampling Rates

Next we explore the impact of the size of the sampling rate domain. Again, we keep the number of motes fixed at one and set the energy constraint to 50% of the highest energy state. We expect from Section 0.4.2 that the runtime grows linearly with the sampling rate domain size, which is also the case as can be seen in Figure 4.

5.1.4 Motes

In Figure 5 we show the run time as a function of increased number of motes. Not surprisingly, increasing the number of motes does not have a significant impact on runtime, with only a small linear addition when adding motes. The reason why motes add a small amount of overhead lies in the way cost and utility are calculated in the model: there is an intermediate calculation step for each mote.

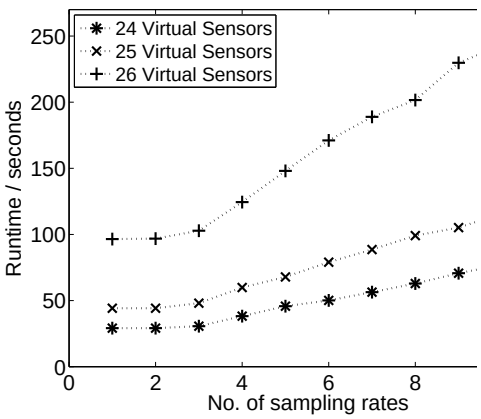

Figure 4: The influence of the number of sampling rates on the search space.

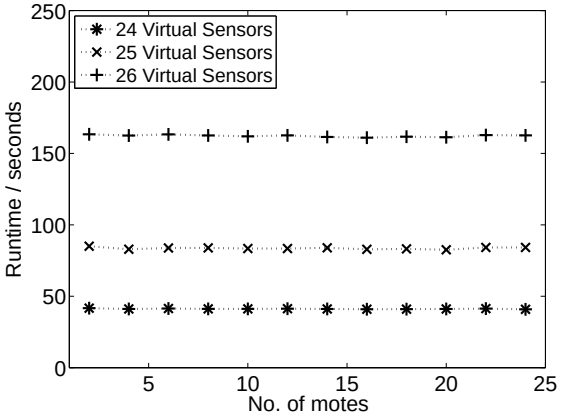

Figure 5: The influence of motes on the search space.

5.2 Constraining Runtime

In the previous experiments, the runtimes we measured have all been for exhaustive searches. However, with an exponential state space we have no guarantees that the search space will be traversed in a timely manner.

Although our goal is the optimal solution, any assignment that satisfies our COP will of course also satisfy the ecologist's requirements. Hence, any solution will be tolerable although one with higher utility is obviously preferred.

Thus, we explore the quality of the intermediate solutions (if any) that the solver discovers during each search when subject to a hard upper bound on the runtime. We turn our attention back to the problems from Section 0.5.1 in order to compare both the constrained runtime with the exhaustive runtime and the intermediate utility with that of the optimal one. However, even with the lowest runtime we could enforce on the search, 15 ms, the solver was able to find a solution with a utility identical to the optimal one. Even when we increased the number of virtual sensors to 40, we still obtained the optimal solution as one of the first solutions where an exhaustive search would have taken more than 20 days.

The reason for this surprising result is that although the problems we seek to solve have an exponentially large search space, the solutions themselves are rather simple; and the reason why the solver requires an exhaustive search to find the optimal solution is because we use a generic one that can only be given general search directions and not a problem specific solver with detailed knowledge of the system's dynamics.

In our case, the general search directions we use are to first establish the optimal sampling rate, by searching in ascending order, and next determine which virtual sensors to include, starting with them all. This strategy favors the localization of *a* solution involving as many sensors as possible, which quite often is exactly

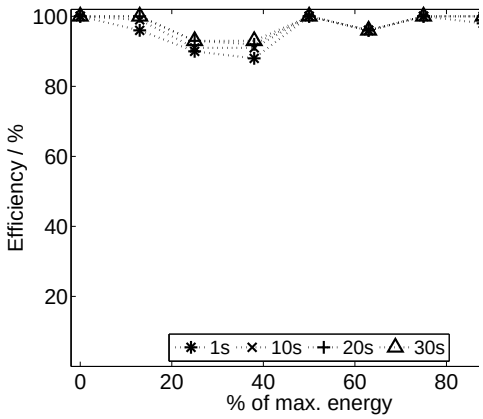

Figure 6: Restricting runtime for the "Life Under Your Feet" COP with 30 motes and 120 virtual sensors.

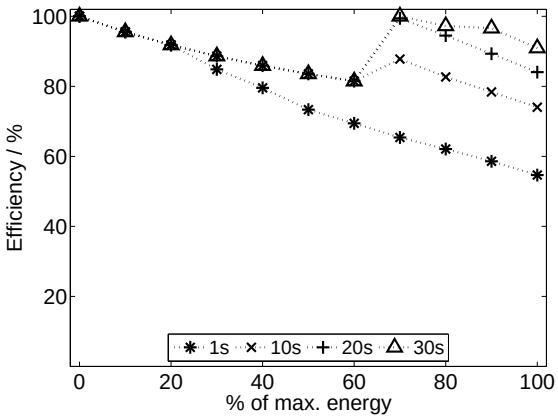

Figure 7: Restricting runtime for the "Life Under Your Feet" COP with 100 motes and 400 virtual sensors.

the lowest tolerable operation mode. We leave the creation of a constraint programming solver tailored for environmental models open as future work.

Instead we turn our attention to a more complex and realistic COP, and consider the soil monitoring sensor network described in Section 0.3.1. Besides the increased virtual sensors needed, this COP also considers each mote's communication cost caused by multi-hop routing. For this evaluation, we choose a simple 4-hop binary tree topology, with the excess motes evenly spread among the leaves.

First we consider a system of 120 virtual sensors, spread evenly among 30 motes, each containing two datasets and two possible sampling rates. We plot the relation between energy constraint and optimal solution for four cut-off runtimes in Figure 6. The energy is varied between the lowest to highest state and the efficiency is measured as the percentage of the optimal solution. Surprisingly, the efficiency is above 88 % for even the shortest runtime of 1 s while almost half of all the solutions found are the optimal one.

We then increase the state space by considering 400 virtual sensors attached to 100 motes. The results can be seen in Figure 7. Overall the increased state space decreases the efficiency for all cut-off times and not surprisingly the 1 s cut-off suffers the most. Looking closer, although the state space has increased by a factor of 2^{70} the 20-30 s cut-off times are still able to achieve 80-100 % efficiency.

This result shows that for environmental monitoring where changes happen on the order of minutes, our controller is efficient enough to instrument the sensor network in a timely manner. Especially, since *any* solution that satisfies the COP also satisfies the needs of the ecologist, regardless of the achieved utility.

5.3 Discussion

In the previous sections we explored the scalability of our model by measuring the runtime over a broad range of parameters. We discovered that even with a generic solver and an exponential search space, the first solutions found are quite often close to the optimal solutions. This last result is encouraging since it suggests that we can plan for even large networks by enforcing an upper bound on the search and still be confident that the result will be close to optimal.
At the same time, it enables several contingency strategies. For instance, if no solution can be found within the given time limit, another search can be initiated but with a higher time limit, or relaxation techniques can be used to simplify the problem by removing datasets from the model one at the time, stopping with the baseline dataset. These simpler problems will thus have a higher chance of success.

6 Case Study: Lake Monitoring

In order to illustrate the usefulness of ADAE we present a case study of a buoy equipped with a mobile water monitor [23]. The virtual sensor abstraction completely shields from the controller that the physical sensors are in fact mobile and not stationary. Unlike typical wireless sensor networks built from low power motes with inexpensive sensors attached, this water monitoring system consists of a $20,000 high-quality data logger which has been network enabled. Being able to control legacy systems is interesting because these are instruments the ecologists know they can trust.
This system consists of a single buoy equipped with a water monitor capable of measuring conductivity, temperature, dissolved oxygen, pH, and fluorescence. These properties are important in estimating the primary production and respiration in the lake ecosystem.
The water monitor is attached to the buoy with a 10 m long cable. An electric motor is used to adjust the vertical location of the water monitor thus enabling measurements at different depths. The buoy is powered by solar panel and is equipped with battery for night time operation. A Real-Time Control Unit (RTCU) instruments the motor and water monitor. Collected data is directly transmitted from the buoy to a back-end database over the Internet through a GSM modem. This connection is also used to transmit new configurations to the buoy, which is used by the RTCU to control the depth and sampling rate of the water monitor. In case of network outage, the RTCU reverts to being a data logger and stores all measurements locally until network service has been restored, at which point the data is offloaded.
This system has been deployed continuously for an entire season. Twice an hour the water monitor is lowered to ten predefined depths and at each depth all five sensors are sampled.

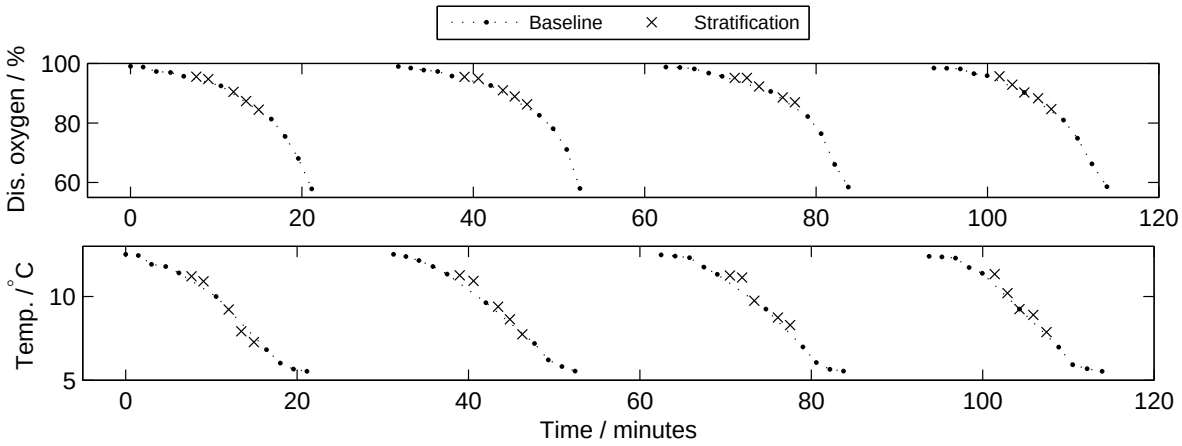

Figure 8: Four vertical profiles of temperature (buttom) and dissolved oxygen (top) measurements. Measurements obtained with the *baseline* dataset has been overlaid with measurements from the *stratification* dataset.

6.1 Problem Statement

Because of surface heating in the summer and the lake's dynamics, two distinct temperature regions, characterized by a sharp boundary, can be formed at the top and bottom. This stratification is interesting for the ecologist because the biological activity is particularly high at this boundary.
Understanding this layering with measurements clustered around the boundary would be of significant scientific value. However, since the formation of this stratification and its location is neither predictable nor static over time it is not possible to specify the exact measurement depths *a priori*. Hence, the previous season's measurements have all been done at ten fixed depth, spread out evenly down to 9 m.
As it turns out, the buoy's power supply has been over dimensioned with a solar panel and battery capacity exceeding the maximum power consumption of the system, even when running continuously. On the other hand, because the water monitor has to physically move, the system can at most measure 15 samples every half-an-hour, due to the actual movement and the following stabilization of the water.
Thus, in this case study the purpose of our adaptive data acquisition is not to conserve energy or meet lifetime requirements, but rather to increase the quality of collected data by adapting the sampling strategy.
In other words, the problems we seek to solve is (1) to detect and track the location of this temperature boundary and (2) instruct the buoy to collect extra samples in this region, besides the ten fixed samples.
Using the formalism presented in Section 0.3.1 we state the ecologist's requirements as follows:

- We define three **data collection modes**: *baseline*, *stratification*, and *correction*. We did not define failure modes in this first deployment. We will leverage some of the lessons we learnt to define failure modes in the next deployment (based on anomalous patterns indicating a problem with a probe as suggested in the Introduction).

- The *baseline* is always present, while the **condition** for the *stratification* event is the formation of a temperature gradient greater than $0.5°C/m$. Finally, the **condition** for the *correction* is when the *sparseness threshold* is reached.

- The **target dataset** for the *baseline* is ten samples at fixed depth. For the *stratification*, it is five samples clustered around the highest temperature gradient. The *correction* is to re-collect missing samples.

- Only the *baseline* is given a **sparseness threshold** of ten samples every half-an-hour.

6.2 COP Modeling

Given the requirements and the physical constraints of the system, we create three datasets. The *baseline* contains five virtual sensors (one for each modality) fixed at each of the ten depths, while the *stratification* is constructed from two parameters: the epicenter and the range of the stratification. Computing the stratification **condition** is straightforward, and the highest value is used as the center and all values exceeding the $0.5°C/m$ threshold constitute the range. Last, the *correction* is constructed from the missing samples.
The reason behind the 30 minutes sparseness threshold is that physical changes in the lake happens on this timescale. Thus, combining a baseline dataset with its associated stratification dataset in one sequence is not an option because the *stratification* set will by definition be shifted with the transmit rate. In other words, the two datasets are mutually exclusive and must be applied sequentially. We constraint the transmit rate and set the *epoch* to 20 min. for the *baseline* and 10 min. for the *stratification*.
The ordering of the **collection modes** are then: *baseline*, *correction*, and *stratification*.
Because energy is not an issue in this case we do not define a constraint for this resource. However, there is still the temporal constraint on the system, where the water monitor's round trip time combined with the sampling time must not exceed the time frame dictated by the sampling rate. Because the water monitor and control unit operates in parallel we do not consider the transmission time since this is significantly smaller than the measurement time.
The water monitor requires 90 seconds to settle at each new location and sample all the sensors. We assign this time cost to each virtual sensor but add in the model that virtual sensors at the same depth can be sampled in parallel. In the application specific part of the model we define the systems temporal cost function as the number of virtual sensors times the cost of each individual sensor

combined with the round-trip-time of the water monitor. We model the round-trip-time as twice the distance of the deepest placed virtual sensor divided by the speed of the electric motor.

6.3 Results

We implemented the model above in ADAE and used it to control the deployed buoy remotely through the back-end server. We used a low-power ARM based single board computer to serve as our controller.
In Figure 8 we show four series of temperature (bottom) and dissolved oxygen (top) measurements. The *baseline* dataset has been overlaid with the *stratification* dataset in order to illustrate the benefit of adaptive sampling. In each series, the measurements to the left are closer to the surface than the measurements to the right.
For the temperature, we see a sharp boundary in the 7-11$°C$ region where only one baseline measurement is present in each series and the rest of the measurements are clustered either above or below this region. When we look at the stratification measurements we see that there indeed are five in each series and that they are filling the gap left by the baseline measurements. For the dissolved oxygen we see a similar trend, with the extra measurements filling out the gaps left by the baseline measurements. Interestingly, the *stratification* measurements do not lie on a straight line between the *baseline* measurements, meaning knowledge has been gained by adding the extra measurements.

7 Conclusion

Sensor networks promise to radically improve the data acquisition systems that ecologists can deploy for in-situ instrumentation. There is however a risk of mismatch between the assumption by ecologists that the sensor network delivers exactly the time series that has been specified, and on the assumption by computer scientists that the goal is to collect as much data as possible (using yield as a performance metric). We argue that it is necessary to take failures and interesting events into account when specifying the ecologists requirements. We proposed data collection modes as a means to represent an envelope of target datasets (e.g., higher sampling rate, or higher accuracy, or different combination of modalities for a given period).
Based on the insight that uniform and systematic ecosystem sampling is neither possible, nor desirable, we proposed ADAE, a utility-based controller, that adaptively configure motes in a changing environment. ADAE is based on the assumption that motes export a simple configuration API in order to easily interface with legacy systems. We described a three-tier architecture to organize the complexity of communicating with motes, representing the sensor network state and the user requirements, generating constraint optimization problems (COP) to determine the configuration parameters, and solving these COP. We showed that a COP solver scales to realistic multihop sensor networks, and

we illustrated the benefits of ADAE in a lake monitoring monitoring system deployed this summer.
We are in the process of preparing new deployments. These are needed to explore the limitations of data collection modes, as well as ADAE. In particular, we are investigating how to handle a very dynamic environment, and how to handle failures in a long-term autonomous deployment.

References

[1] M. Ai-Chang, J. Bresina, and L. e. a. Charest. MAPGEN: mixed-initiative planning and scheduling for the Mars Exploration Rover mission. *Intelligent Systems, IEEE*, 2004.

[2] R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand. An Architecture for Autonomy. *International Journal of Robotics Research*, 1998.

[3] P. Bonnet, P.Seshadri, and J. Gehrke. Towards Sensor Database Systems. In *Mobile Data Management*, 2001.

[4] P. Borgstrom, A. Singh, B. Jordan, G. Sukhatme, M. Batalin, and W. Kaiser. Energy based path planning for a novel cabled robotic system. In *IEEE IROS*, 2008.

[5] Q. Cao, D. Fesehaye, N. Pham, Y. Sarwar, and T. Abdelzaher. Virtual Battery: An Energy Reserve Abstraction for Embedded Sensor Networks. In *IEEE RTSS*, 2008.

[6] M. Chang, A. Terzis, and P. Bonnet. Mote-Based Online Anomaly Detection Using Echo State Networks. In *DCOSS*, 2009.

[7] H.-L. Choi and J. P. How. A Multi-UAV Targeting Algorithm for Ensemble Forecast Improvement. *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2007.

[8] A. Deshpande and S. Madden. MauveDB: supporting model-based user views in database systems. In *ACM SIGMOD*, 2006.

[9] R. Doyle, D. Bernard, E. Riedel, N. Rouquette, J. Wyatt, M. Lowry, and P. Nayak. Autonomy and Software Technology on NASA's Deep Space One. *IEEE Intelligent Systems*, 1999.

[10] D. Estrin, L. Girod, G. Pottie, and M. Srivastava. Instrumenting the World with Wireless Sensor Networks. In *International Conference on Acoustics, Speech, and Signal Processing*.

[11] J. Frank and A. Jónsson. Constraint-Based Attribute and Interval Planning. *Constraints*, 2003.

[12] M. Freed and R. Remington. Managing decision resources in plan execution. In *IJCAI'97: Proceedings of the 15th international joint conference on Artifical intelligence*, 1997.

[13] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection Tree Protocol. In *ACM SenSys*, 2009.

[14] O. Gnawali, B. Greenstein, K.-Y. Jang, A. Joki, J. Paek, M. Vieira, D. Estrin, R. Govindan, and E. Kohler. The TENET Architecture for Tiered Sensor Networks. In *ACM SenSys 2006*.

[15] J. W. Hui and D. Culler. The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale. In *ACM SenSys*, 2004.

[16] G. H. K. Lam, H. Va Leong, and S. C. F. Chan. BBQ: group-based querying in a ubiquitous environment. In *ACM SAC '06*, 2006.

[17] M. Leopold, M. Chang, and P. Bonnet. Characterizing Mote Performance: A Vector-Based Methodology. In *EWSN*, 2008.

[18] M. Li, D. Ganesan, and P. Shenoy. PRESTO: feedback-driven data management in sensor networks. *IEEE/ACM Trans. Netw.*, 2009.

[19] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TinyDB: an acquisitional query processing system for sensor networks. 2005.

[20] S. M.Franklin, J.Hellerstein. Thinking Big About Tiny Databases. *Data Engineering Bulletin*, 2007.

[21] R. Musăloiu-E., A. Terzis, K. Szlavecz, A. Szalay, J. Cogan, and J. Gray. Life Under your Feet: A WSN for Soil Ecology. In *EmNets Workshop*, May 2006.

[22] N. Nethercote, P. J. Stuckey, R. Becket, S. Brand, G. J. Duck, and G. Tack. MiniZinc: Towards a Standard CP Modelling Language. In *13th International Conference on Principles and Practice of Constraint Programming*, 2007.

[23] Peter A. Staehr and Rikke M. Closter. Measurement of whole system metabolism using automatic profiling sensors. In *GLEON 6*, 2008.

[24] J. Polastre, J. Hill, and D. Culler. Versatile Low Power Media Access for Wireless Sensor Networks. In *ACM SenSys*, 2004.

[25] S. Russell and P. Norvig. *Artificial Intelligence A Modern Approach, 2nd Ed.* Prentice Hall, 2003.

[26] G. S. S. Thompson. *Adaptive Sampling*. Wiley Interscience, 1996.

[27] G. Werner-Allen, S. Dawson-Haggerty, and M. Welsh. Lance: optimizing high-resolution signal collection in wireless sensor networks. In *ACM SenSys*, 2008.