Meeting Ecologists' Requirements with Adaptive Data Acquisition

Marcus Chang
IT University of Copenhagen
mchang@itu.dk

Philippe Bonnet
IT University of Copenhagen
phbo@itu.dk

Abstract

Ecologists instrument ecosystems to collect time series representing the evolution in time and space of relevant abiotic and biotic factors. 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 maximize the utility of the collected datasets in a changing environment. 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, the system requirements, and the scientist requirements. We describe our implementation, discuss its scalability, and discuss its performance in the context of two case studies.

Categories and Subject Descriptors

C.2.4 [**Computer Systems Organization**]: Computer-Communication Networks—*Distributed Systems*

General Terms

Design

Keywords

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

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is premitted. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SenSys'10, November 3–5, 2010, Zurich, Switzerland.
Copyright 2010 ACM 978-1-4503-0344-6/10/11 ...$10.00

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. 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 significant impact on ecological monitoring by transforming stand-alone devices into a networked system that is monitored and controlled to meet the scientist's 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,z), 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 higher 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).

The solution promoted in commercial data acquisition systems to tackle failures and anomalous situations consists in involving human supervision. Considering the lake monitoring example above, an ecological data acquisition solution would consist of a buoy equipped with CTD sensors (conductivity, temperature, depth) deployed at five different depths, and a data logger sampling the sensors and storing the measurements at a predefined rate. This data logger would also be equipped with long-range wireless communication and act 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.

In contrast, we propose that ecological sensor networks should rely on adaptive ecosystem sampling, where the procedure for selecting samples is autonomously revised based on the values of observed variables [27]. 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 [26].

Our goal with this work is to limit human intervention to the initial requirements, and let an autonomous data acquisition system handle anomalies. Obviously, when possible and affordable, human intervention should be used to maintain the optimal regime[1]. Our point, here, is that the system should maintain high utility and degrade gracefully when the optimal regime is no longer sustainable. 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 ADAE, 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. ADAE is based on a three-tier architecture, originally developed for autonomous systems. We detail the design and implementation of the ADAE system, and discuss how it scales.
3. We describe experimental results based on a simulation based on the Life Under Your Feet dataset [22], as well as an actual deployment for lake monitoring.

[1]This work is part of the MANA project whose goal is to deploy an autonomous sensor-based data acquisition system for lake monitoring in North-East Greenland [5]. This project is funded by the Danish Strategic Research Council.

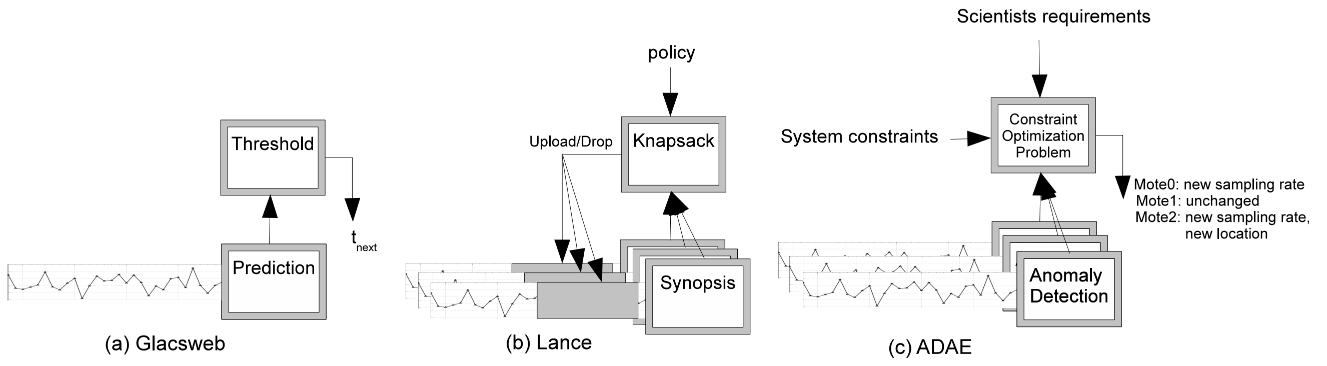

Figure 1. Comparison of the utility-based controllers of Lance, Glacsweb, and ADAE.

2 Related Work

In this section, we position ADAE with respect to existing sensor network-based data acquisition systems, 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 [20] 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 too high-level for ecologists that do not wish to query the sensors but aim at systematically collecting primary datasets for their scientific processes (see [21] 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 [17] 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 collected as needed to improve precision), with a push approach, where each sensor uses 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 [30] 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 based on a policy 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.

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 [27]. In the context of sensor networks, adaptive sampling has mainly been introduced to (a) maintain high resolution while covering large regions of space using mobile sensors, 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.

In Glacsweb [28], adaptive sampling was actually introduced to maintain high utility in a changing environment.

Each mote relies on observed values to forecast future values using a Bayesian linear model, and decides on when to obtain the next sample based on the estimated information gain of these future values. Basically, the next sample is taken at the point in time at which the estimated information gain of the predicted value is higher than a given threshold. While inspired by the same principles, our work differs from Glacsweb on two fundamental aspects. First, ADAE does not attempt to predict future values - observed values are used to detect anomalies, which trigger changes in the controller's model of the environment. Second, ADAEs controller manages how each mote should adapt to changes in the environment and obtain their next measurements in time, space, and modality to satisfy the scientist requirements given the system constraints - as opposed to focusing on when a given mote should take its next sample. Figure 1 summarizes the differences between Lance, Glacsweb and ADAE.

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 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] and 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.[2]

Interestingly, it has been proposed to use an autonomous system to control adaptive sampling experiments in aquatic environments. Stealy et al. discussed this idea in the context of the NIMS-AQ system [29]. The authors introduce A-IDEA, a three-tier architecture with the goal of autonomously sampling an unknown phenomenon - instead of relying on a well-understood model of the phenomenon, the system constructs such a model iteratively with the acquisition of new samples. While the NIMS-AQ system is implemented and deployed to achieve high resolution at low cost over an extended region of space (e.g., over a lake or the cross-section of a river), we are not aware that A-IDEA has actually been designed or implemented. ADAE is a significant step in the direction sketched for A-IDEA.

3 The Ecologists' Requirements

We aim at designing a system that autonomously adapts data acquisition to meet the ecologists' requirements. In this section, we focus on how to represent these requirements. Note that our goal is not to define a rigid template for software engineering purposes, but to put a stick in the ground regarding the scientists expectation of an ecological sensor network.

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 time series typically characterized by a fixed sampling rate, a period of time during measurements should be obtained, the placement of sensors, as well as their 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 [22]. 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.

To overcome these limitations, our goal is to represent (a) an envelope of relevant datasets for the ecologists in the context of a given deployment - as opposed to a single dataset specification, and (b) the scientists preferences within that envelope - i.e., a form of utility function.

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). We present example of data collection modes, summarized in Table 1 and Table 2, in the context of the two case studies discussed in Section 6 and Section 7. For each data collection mode, the ecologists define:

1. A description of the conditions that must be satisfied to activate or deactivate each mode. 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 dataset specifies relevant redundancy in time, space or modality.

3. A sparseness threshold for each dataset, i.e., the number

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

(or distribution) of usable measurements per chunk of time and space. This sparseness threshold ensures that the collected dataset can be used for modeling purposes.

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 define a target *lifetime* for data collection.

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).

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: What 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 [17, 30], 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 [25]).

We also assume that the sample, store, and transmit tasks, at the core of any data acquisition system, 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 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 underlying sensor network.[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). It is up to the underlying sensor network to execute these actions on the motes.

[3]Our notion of virtual sensor is inspired by Franklin et al. [21]

4.2 Principles of Operation

Our controller needs to address two problems:

1. What is the controller state and how to update it? While important for our implementation, the details of the state representation have no significant impact on the system and we will not discuss them here. We refer interested readers to [19].
2. How to pick appropriate actions given the 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 in response 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 might be 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 (stricter constraints, e.g., synchronous requests, might be introduced to enforce latency requirements).

For each epoch, virtual sensors have a fixed configuration (i.e., fixed location, fixed sampling rate). 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 (COP), 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 (e.g., location or sampling rate), $\mathcal{R}$ 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 energy constraints) and $\mathcal{U}$ is the expected utility. The size of the search space grows exponentially with the number of virtual sensors $O(p\_range \cdot 2^N)$, where N is the number of virtual sensors and *p_range* is the average size of the variable domains

4.3 System Architecture

In order to organize the complexity of the controller, we structure its components using the classical three-layer architecture developed for AI planning [2]:

- Functional Layer, which provides abstractions for the underlying sensor network, the sensor tasks, and the storage subsystem. 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.

The flow of information in the individual components in this three-layer architecture is illustrated in Figure 2. The deployment-specific detection modules, target datasets, and

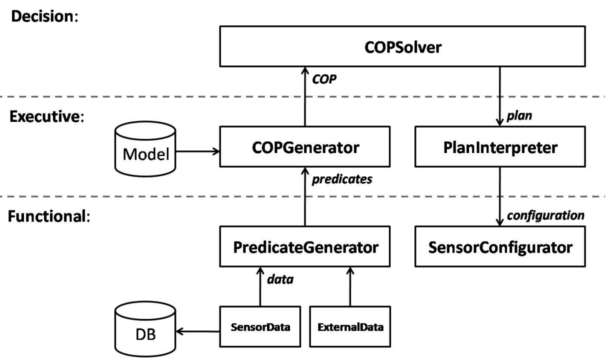

Figure 2. Architectural overview of ADAE.

the model derived from the ecologists requirements are attached to the core ADAE structure through a plugin interface. Automatizing this mapping and thus allowing ecologists to configure and inspect the controller via a high-level interface is a topic for future work.

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 with 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 sensors, on 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 also passed on to **PredicateGenerator**.

In **PredicateGenerator**, anomaly detection 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[4]. 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. 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 sensors. 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 corresponding to the active predicates. Note that we generate a single COP for the entire network in order to account for networking 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 [23] 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.

4.4 Implementation

We implemented the ADAE controller using the standard C++ library in order to ensure portability. The COPs are translated using the MiniZinc-to-FlatZinc 0.9 translator [23] and subsequently solved using the Gecode/FlatZinc[5] 1.3 solver, since it offers a controlled search process and good performances on a wide range of platforms (including the target gateway for our deployment). The C++ code and the MiniZinc models used for the experiments and in the case studies are publicly available [18].

[4]Note that the complexity of a detection module can range from a simple rule to a learning-based classifier. Also, the detection modules might be embedded as a watchdog on the motes as suggested by Chang et al. [6]

[5]Generic Constraint Development Environment. `http://www.gecode.org/`

In the two case studies presented in Section 6 and Section 7, we will show that ADAE is capable of controlling very diverse systems. Such a flexibility results from our decisions to abstract the sensor network as a collection of virtual sensors (at the functional layer), and to provide a framework for defining deployment specific models of the environment, of the system constraints, and of the scientist requirements (at the executive layer).

The complexity of these deployment-specific components is highly variable. For instance, the interface of the buoy control system from Section 7 was a legacy system, designed exclusively for a web-based GUI and was never intended as a programmatic interface. In order to implement a driver suitable to ADAE's sensor network abstraction layer, we had to implement our own API by parsing HTML tagged web pages and generate human readable forms filled with redundant information in order to interact with the buoy. Also, since the original target had been human interaction we experienced several timing issues with the buoy control interface because the system had not been designed to handle rapid sequences of requests from an application, leading to corrupt data and other byzantine behavior. The anomaly detection plugins are also deployment specific. However, because datasets are represented as generic ADAE structures (exposed at the functional layer interface) and target datasets are represented using virtual sensors (also exposed at the functional layer interface), these plugins are reusable across deployments. We refer the reader to [18] for a description of the three layers API and of the ADAE data structures.

Constructing a valid and complete MiniZinc model for a given application is not trivial. Specially, finding the correct relationship between utility, energy, and time requires some trial and error. For example, in an early version of the model for Life Under Your Feet, the optimal solution turned out to be not to sample at all, since the utility gain was not high enough to offset the energy loss inflicted. At this point, a key limitation lies in the current state of the MiniZinc solver, which does not support floats (as exhibited in the rounding error in Figure 10), only supports 20 bit integers, and suffers a significant performance drop for large domains. The result is that model specifications have to be scaled and refactored in order to fit these artificial constraints. These problems are annoying but not fundamental as they will disappear with the expected evolutions of the MiniZinc solver.

5 Evaluation

Before we go further, we need to check two of the assumptions underlying our system design. First, we should check that it is possible to rely on a generic solver for the type of constraints optimization problems ADAE is generating. Second, we should check that we can deal with optimization problems of reasonable sizes, or more conservatively problems of large sizes with hundreds of virtual sensors. The results of the experiments presented in this section were obtained on an Intel Core 2 T7600 2.33 GHz processsor. A MiniZinc code example can be seen in Figure 3.

5.1 State Space vs. Search Space

We claimed in Section 4.2 that the size of the state space grows exponentially with the number of virtual sensors. This is supported by Figure 4, which indeed shows an exponential growth in runtime when the number of virtual sensors in the COP increases.

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 observation 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 an 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 5 we show the runtime for three different COPs, with varying energy constraints (set as a percentage of the maximum energy required for the most resource demanding state). 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.

5.2 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. An aggressively duty cycled gateway, that should be able to operate on batteries for a long period of time, or a short replanning epoch to reduce latency, both limits the time available for COP solving. Although our goal is the optimal solution, any assignment that satisfies our COP should 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.

For the remainder of this evaluation, we choose a simple 4-hop binary routing tree topology, with the motes spread evenly among its leaves. First we consider a system of 120 virtual sensors, spread evenly among 30 motes. 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  var EnergyDomain: M1EnergyUsage =
2    sum([M1SensorCostArray[s]     * bool2int(M1SensorSelectedArray[s]) | s in M1SensorDomain])
3    + sum([M1TransmitCostArray[s] * bool2int(M1SensorSelectedArray[s]) | s in M1SensorDomain])
4    + sum([M2TransmitCostArray[s] * bool2int(M2SensorSelectedArray[s]) | s in M2SensorDomain])
5    ...
6    + sum([MNTransmitCostArray[s] * bool2int(MNSensorSelectedArray[s]) | s in MNSensorDomain])
```

Figure 3. MiniZinc code example showing the energy consumption for mote M1. Line 1 defines the M1Energy variable. Line 2 calculates the sensing cost and Line 3 the transmission cost for the selected sensors. Line 4-6 calculates the forwarding cost induced by the children motes M2 to MN.

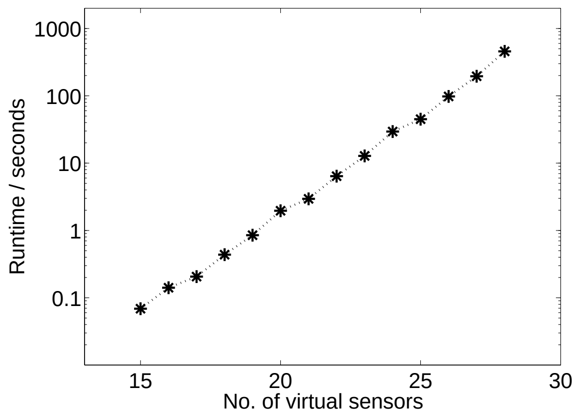

Figure 4. State space grows exponentially with the number of virtual sensors in the COP. Note the logarithmic scale.

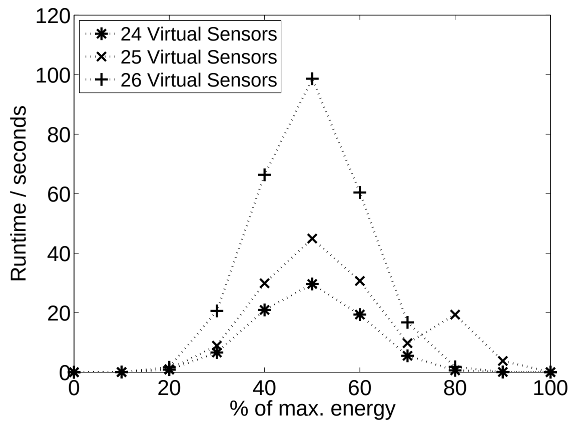

Figure 5. Search space is shaped by the resource constraints.

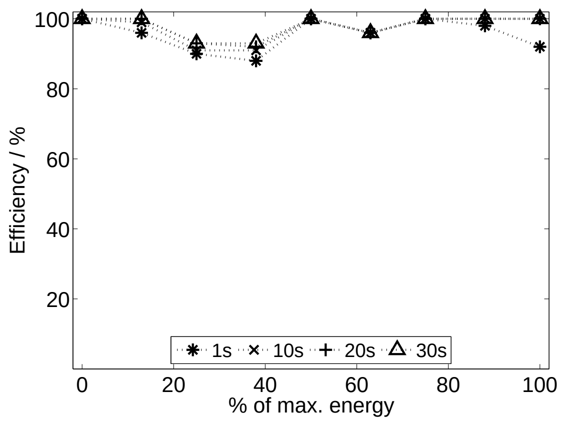

Figure 6. Restricting runtime for the COP with 30 motes and 120 virtual sensors.

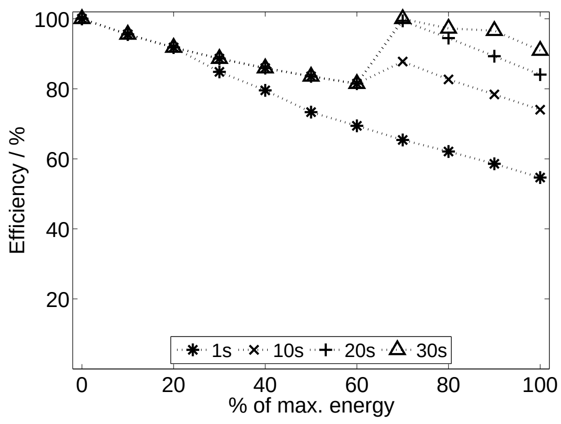

Figure 7. Restricting runtime for the COP with 100 motes and 400 virtual sensors.

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^{280} 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

These experiments were designed to explore the feasibility and the scalability of our approach based on constraint optimization. We found 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: Life Under Your Feet

In this case study we use a simulation based approach to explore the various contingency plans ADAE can use in order to reach lifetime and sparseness requirements in the event of measurement faults and hardware failures. Our focus is to illustrate how ADAE adapts to a changing environment where sensors malfunction and motes loose power. By relying on a simulated sensor network instead of an actual deployment or testbed, we are able to control the injection of faults and hardware failures, and since ADAE, as a sensor network controller, is already dependent on an underlying sensor network this trade-off between actual deployment and experimental control seems reasonable.

We use the Life Under Your Feet (LUYF) [22] soil monitoring sensor network as a template for our simulation. Each mote in this network is measuring both temperature and moisture at two different soil depths. Measurements are sampled, stored, and forwarded to a gateway, which stores the data and collects network statistics. We use real-life data for our simulation, by feeding measurements collected with the LUYF sensor network into ADAE. This is done by creating a simulated sensor network to take the place of the SensorData component described in Section 4.3 (and shown in Figure 2). However, since the measurements we playback were sampled once every hour, whenever we require measurements outside this interval we apply a linear interpolation between known data points.

Since one of the main ideas behind ADAE is to offer a framework in which one can apply a broad variety of detection algorithms through plugins, we simulate sensor faults (noisy measurements, spikes, stuck readings, etc.) as missing data, since this is ultimately the result of any fault detection algorithm.

In order to meet the lifetime requirement, we use the Token Bucket[6] model for power management. The idea is to divide the remaining energy into equal sized allowances to be used in each planning interval. If the allowance is not completely used up, the remaining energy is saved up in each mote's virtual bucket. In each planning interval, ADAE can then dispose of both the energy allowance and the energy bucket. These allowances ensures that energy is available throughout the lifetime of the sensor network, while the buckets allow a more dynamic energy usage with periods of higher activity.

6.1 Problem Statement

The contingency plans ADAE relies on in order to reach lifetime and sparseness requirements, in the event of measurement faults and hardware failures, are based on redundancy:

- **Space**. Spatial redundancy, either in the form of backup motes or motes located close enough to have correlated measurements, can be used to move measurements form one mote to another.
- **Time**. In case the amount of samples (e.g., for statistical reasons) are more important than the actual timing of the samples, temporal redundancy can be exploited by increasing the sampling in order to reach a certain sparseness requirement. Conversely, although higher sampling frequency is often preferred, reducing the sampling frequency to the minimum acceptable can be used to extend the lifetime of the mote.
- **Modality**. Similar to the spatial redundancy, in the presence of backup or highly correlated sensors, measurements can be moved from one modality to the other in order to leverage this redundancy.

These three methods are orthogonal in the sense that ADAE can combine redundancy from different methods to satisfy the ecologist's requirements. Of course, instead of defining a single backup sensor/mote, a group of sensors/motes can be specified as well, either as a union where all backups are activated or as load balancing, where ADAE chooses the optimal candidate (based on available energy and sampling/transmission cost).

For example, in the event of power loss, caused by damage to the mote's power supply or fluctuating power consumption from sensors and actuators, the sparseness and lifetime requirement could be compromised. Spatial redundancy could then be used to move the measuring to a different mote, temporal redundancy by entering a degraded mode with lower sampling frequency, and redundancy in modality by switching to a less expensive, but still correlated sensor.

Another example is missing measurements, ranging from a single dropped sample to a complete failure of a sensor or mote. In the former case, temporarily increasing the sampling rate might be adequate, while the latter will require switching to another sensor (and mote). In the results subsection, we limit our attention to three experiments, chosen to explore the different redundancy methods and show the versatility of ADAE.

6.2 Requirements

Below we present a way to formulate the ecologist's requirements to the LUYF sensor network, which are also summarized in Table 1:

- **Baseline**. By default all four senors should be sampled once per hour. Although this is the preferred operating mode, we do not impose a sparseness requirement on the sampling frequency in order to allow for contingency planning.
- **Degraded**. If the Baseline requirement cannot be fulfilled (due to power shortage), the sampling frequency can be reduced to one sample every six hour.
- **Modality**. If one of the sensors on a mote has faulty measurements, the remaining three sensors should in-

[6] `http://en.wikipedia.org/wiki/Token_bucket`

Name	Priority	Condition	Target	Interval	Replan
Baseline	1	Always	All motes, sensors	1 hour	6 hours
Degraded	2	If not enough power for Baseline	All motes, sensors	6 hours	6 hours
Modality	3	If measurements are missing	Remaining local sensors	30 min.	6 hours
Space	3	If measurements are missing	One of two nearest motes	30 min.	6 hours

Table 1. Data collection modes for Life Under Your Feet simulation.

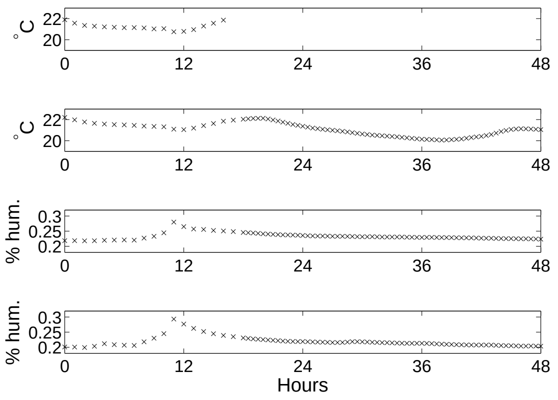

Figure 8. Redundancy in modality. Failure in one sensor causes an increase in sampling frequency on the other three sensors.

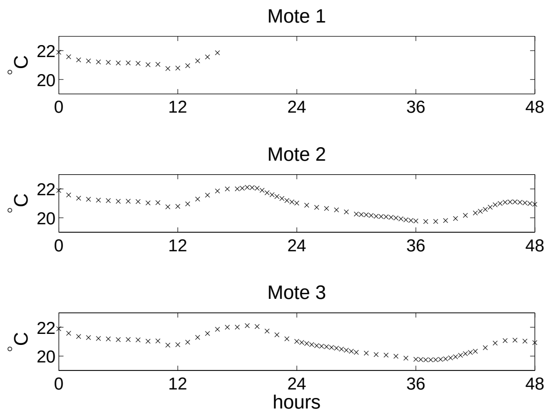

Figure 9. Spatial redundancy. Failure on one mote causes an increase in sampling frequency on the two neighboring motes.

crease their sampling frequency to twice per hour.

- **Space**. If a mote has faulty measurements, either one of the two closest motes, should double their sampling frequency to two samples per hour.

Since the Baseline and the Degraded requirements state the preferred operating mode, they receive the highest and second-highest priority, respectively. The last two contingency requirements, Modality and Space, both receive the same lowest priority.

Based on the Degraded mode requirement, we set the replanning interval to six hours,[7] and for this simple case study we set the lifetime requirement to one month. These data collection modes are then implemented as plugins to the Predicate- and COP-generator modules in ADAE.

6.3 COP Model Generation

The Baseline mode always request all the sensors on all motes to be sampled once every hour. For the Degraded mode, we do not explicitly define a low-on-power region in which the Degraded mode is active. Instead, we use a modified copy of the Baseline mode, where the sampling frequency has been lowered to once every six hours and with the priority set to a lower level. Because of the different priorities, only the Baseline mode will be activated whenever there is enough energy, and when there is not enough energy, the Degraded mode automatically gets highest priority. Both the Modality and Space mode compares timestamps and counts the number of measurements in order to detect missing ones. When these are detected the sampling frequencies on either the local sensors or the sensors on the nearest motes are requested to be increased to two measurements per hour.

[7]In a real-life deployment, the replanning interval should also leave time for the actual planning process and network transmission and latency.

The sensors on each mote are modeled with individual energy cost and sampling frequency, and transmission cost is set to be proportional to the amount of data transmitted. For each mote, the total amount of energy used in each planning interval is constrained by the allocated energy from the bucket model.

Although time constraint is part of the model, i.e., the sampling frequency is constrained by the actual sampling time of each sensor, sampling fixed sensors such as temperature and moisture sensors can be done on the order of milliseconds, which is negligible compared to the desired sampling frequency of every 30 to 60 minutes.

6.4 Results

In the first experiment we look at how a sensor failure is handled by the Modality mode. Figure 8 shows two days worth of measurements from four sensors connected to the same mote. In the first 16 hours, all sensors are sampled once every hour exactly as stated by the Baseline mode. A catastrophic failure then occurs on the top temperature sensor and no more measurements are collected from it. This is detected by ADAE which activates the Modality mode, resulting in the remaining three sensors increasing their sampling rate as stated by the collection mode. Note that in this instance the reaction to the missing measurements happen immediately. However, the reaction time is of course directly related to the replanning interval, with higher replanning intervals leading to slower reactions.

In the second experiment we look at the performance of the Space mode. Similar to the previous example, Figure 9 shows temperature measurements taken over two days, but this time from three different motes. In the first 16 hours we see the same regular sampling from the Baseline mode. However, when the failure occurs and the Space mode is activated we see that: (1) because the Space mode only requires one of the nearest motes to react only Mote 2 increases its sampling frequency. (2) because Mote 2 has now used more energy than Mote 3, in the next planning interval ADAE optimizes the energy usage by balancing the load over to the mote with most energy available. This load balancing results in the alternating sampling frequencies observed. Note that ADAE supports redundancy across motes because the planning layer operates over the whole network, as opposed to Glacsweb where the timing of each measurement is determined on the motes.

In the final experiment we show how the Degraded mode ensures that even in the case of catastrophic energy loss at least a limited amount of measurements can be sampled throughout the rest of the required lifetime. In Figure 10 we show, from top to bottom, the temperature, moisture, battery, and bucket readings for the entire 30 day period dictated by the lifetime requirement. The battery level is chosen to be the exact amount needed in order to fulfill the lifetime requirement, when operating in the Baseline mode, while the bucket is initialized with an extra allowance. On Day 12 we remove 80% of the energy supply, leaving just enough energy for the mote to actually fulfill the lifetime requirement (if the Degraded mode is activated in time that is). The tightened energy constraint forces ADAE to activate the Degraded mode which causes the sampling frequency to drop significantly, as can be seen in the top of the figure.

However, even in this Degraded mode, the diurnal temperature pattern and two rain events are clearly distinguishable in the remaining measurements. Had the mote continued in the Baseline mode, the battery would have been depleted after three days, and the last two rain events would not have been observed. This reinforces our claim that sparseness, rather than yield, should be used as a metric to characterize the collected datasets, since both datasets mentioned would have had the same yield.

Interestingly, on Day 26 the energy allowance increases slightly (because of rounding errors), leading to a small energy buildup in the bucket, which results in small periodic increases in the moisture sampling frequency. Had the energy loss been intermittent, and sufficient power had returned, the Baseline mode would have been activated by ADAE again and operation would have resumed as if the failure had not occurred.

7 Case Study: Lake Monitoring

In the previous case study we showed ADAE's fault correction capabilities. In this one, we will explore how the same adaptability can be used to improve the flexibility of a mobile water monitoring system [24].

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.

The system, designed by Peter Staehr from U.Copenhagen, consists of a single buoy, shown in Figure 11, 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 and an electric motor is used to adjust the vertical location of the water monitor, thus enabling measurements at different depths. Taking the water monitor's physical movement into account, and leaving enough time for the stabilization of the water, the system is capable of measuring 15 different depths every half-an-hour.

The buoy is powered by solar panel and is equipped with battery for night time operation and a Real-Time Control Unit (RTCU) instruments the motor and water monitor. Collected data is directly transmitted from the buoy to a backend 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.

The past two years (except during winther where the system has been brought down for maintenance), the system has been taking water samples twice an hour at ten predefined depths.

7.1 Problem Statement

Because of surface heating during 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*. Because of this, the previous season's measurements have all been done at ten fixed depth, spread out evenly to 9 m depth.

Hence, the problems we seek to solve are (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.

7.2 Requirements

Formally, the ecologist's requirements we seek to fulfill are the following:

- **Baseline**. As the monitoring program's cornerstone, the water column should be measured at ten uniformly distributed depths, twice an hour (i.e., the same sampling strategy previously deployed). These measurements will provide the ecologist with a long-term baseline of the lake's dynamics, and therefore should be given the highest priority. This also means that there is

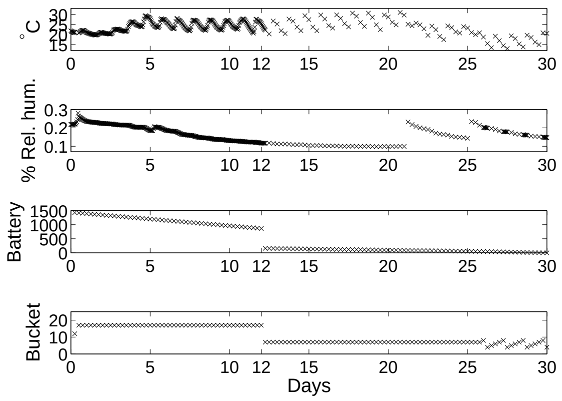

Figure 10. Redundancy in time. On Day 12 a catastrophic failure depletes 80% of the remaining battery. The Degraded mode ensures the lifetime requirement is satisfied.

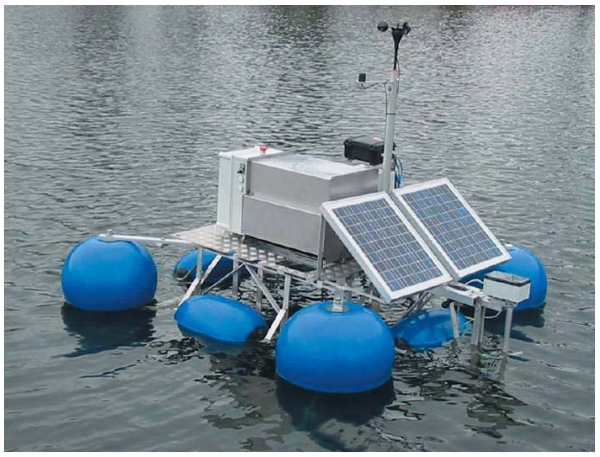

Figure 11. Peter Staehr's buoy deployed in lake.

Name	Priority	Condition	Target	Interval	Replan
Baseline	1	Always	Depths: $\{0.5, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$	30 min.	30 min.
Stratification	2	If $\frac{\Delta t}{\Delta z} > 0.5°C/m$	0-5 normal distributed depths centered around $max : \frac{\Delta t}{\Delta z}$	30 min.	30 min.

Table 2. Data collection modes for lake monitoring.

a sparseness requirement on both the amount of depths measured and on the measuring frequency.

- **Stratification**. If the temperature difference between each measuring depth is above $0.5°C/m$, extra measurements should be taken in this region. Since the exact position of the layering is of most importance to the ecologist, the majority of the extra measurements should be concentrated on the depth with the highest temperature gradient and the remaining ones distributed farther from the center. With the buoy being able to measure at 15 different depth every half-an-hour, the objective is to measure at five additional depths. However, although these extra measurements are significant to the ecologist, they are not as important as the baseline, and are therefore given a lower priority. Individually, measurements closer to the highest temperature gradient receive higher priority than those farther from the center. Furthermore, there is no sparseness requirement attached, since any extra measurements are considered useful.

Although the solar panel and battery on the buoy are both capable of generating and storing enough power for continuous operation, in the event of failure, e.g., one of the battery cells suffer physical damage or the solar panel is covered by debris, energy management becomes an important factor. As seen in the previous case study, this is exactly what ADAE is intended for. Unfortunately, the manufacturer of the buoy was neither able to provide us with any power consumption estimates nor provide us with live energy production and reserve estimates. Hence, the purpose of this deployment is to increase the quality of collected data, by adapting the sampling strategy, rather than to conserve energy and meet lifetime requirements.

We convert the ecologist's two requirements presented above into the data collection modes shown in Table 2. The replanning interval is set to 30 minutes since we have to adapt to the stratification layer as soon as possible.

7.3 COP Model Generation

The data collection modes are implemented as plugins to the Predicate- and COP-generator modules in ADAE, where the Baseline plugin always requests ten measurements at the predefined depths, at the specified interval, and with the stated replanning interval. On the other hand, the Stratification plugin calculates the temperature gradient from the most recently collected dataset, and marks the region satisfying the given condition. Five depths are then chosen by mapping a normal distribution to the marked region with the mean centered around the middle and standard deviation at the border of the region. The five depths are then selected with one at the mean, and one at each 20% and each 40% cumulative quantile of the mean. The third and final plugin uses the requests from the two other plugins as parameters to generate target datasets that are injected into the COP.

The constraints under which this COP must be solved, are defined by the physical limitations of the system. Because the actual performance of the electric motor can vary due to the environment and network congestion can delay the time between the actual sampling and until ADAE can process the data, there is a time window of variable size in which measurements can be performed while still satisfying the frequency sparseness requirement. To model this time constraint we first use a timer to measure the excess time used by the previous iteration, in order to estimate the available time in this iteration.

Second, we model the time constraint of the movable water sensor based on both the number of measurements and the maximum depth considered. At each depth the sensor must wait 90 seconds before sampling can commence in order for the water to stabilize from the movement. With the average speed of the electric motor known, we can estimate the time needed to reach the maximum depth. By combining these time estimations we have a time constraint for the COP model.

This time constraint will mainly impact the Stratification mode, since the remaining time will determine how many extra measurements can be inserted while still fulfilling the frequency sparseness requirement.

We do not enforce a bandwidth constraint, since the size of the generated data is negligible compared to the bandwidth of the GSM modem, thus such a constraint would never be applied. Also, without the relevant power production and consumption figures, as mentioned above, we do not impose energy constraints in the COP model either.

7.4 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.[8]

In Figure 12 we see a measurement series from the buoy showing the temperature at different depths. In the region from four to six meters depth we see a significant temperature gradient, clearly illustrating the need for adaptive sampling. The presence and location of the stratification measurements shows that ADAE is not only capable of detection the stratification layer but also placing the measurements across the entire region as requested. Note that here, adaptive sampling determines both the location, timing and modality of a measurement (and not just the timing as in Glacsweb).

In Figure 13 we see the corresponding dissolved oxygen concentrations taken from the same measurement series. The relation between the stratification layer and dissolved oxygen, and thereby biological activity, is clearly seen by the sharp drop in oxygen concentration at exactly the stratification layer. In fact, with these extra stratification measurements it is possible to estimate a more precise depth of the actual stratification layer and thereby calculate a more precise estimate of the primary production.

8 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 (COPs) to determine the configuration parameters, and solving these COPs. We showed that a COP solver scales to realistic multi-hop sensor networks, and we illustrated the benefits of ADAE in a lake monitoring system deployed last summer.

We are currently 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.

[8]Technologic Systems TS-7260. `http://www.embeddedarm.com/`

9 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] M. Chang and P. Bonnet. Monitoring in a High-Arctic Environment: Some Lessons from MANA. *IEEE Pervasive Computing*, 2010.

[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*, 2001.

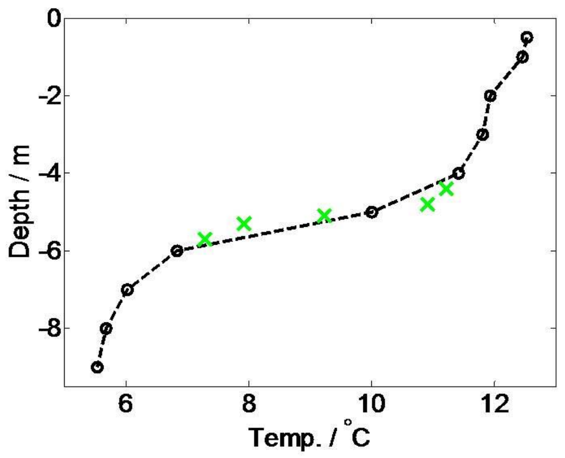

Figure 12. Temperatures measured at different depths. Baseline (o) and Stratification (x).

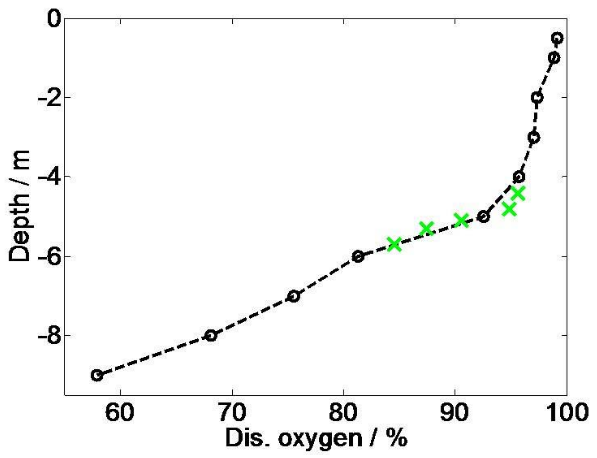

Figure 13. Dissolved oxygen concentrations measured at different depths. Baseline (o) and Stratification (x).

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

[18] M. Chang. Adae - autonomous data acquisition engine. `http://code.google.com/p/adae/`.

[19] M. Chang. *From Automatic to Adaptive Data Acquisition - Towards Scientific Sensornets*. PhD Thesis, University of Copenhagen, 2010.

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

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

[22] 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.

[23] 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.

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

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

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

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

[28] A. U.-B. A. Sensing and M. hop Communication Protocol for Wireless Sensor Networks. Paritosh, P. and Dassh, R.K. and Jennings, N.R. *ACM Transactions on Sensor Networks*, 2010.

[29] M. Stealey, A. Singh, B. J. M.A. Batalin, and W. Kaiser. NIMS-AQ: A novel system for autonomous sensing of aquatic environments. In *IEEE International Conference on Robotics and Automation*, 2008.

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