Proactive Adaptations in Sensor Network Query Processing

Alan B. Stokes, Alvaro A. A. Fernandes, Norman W. Paton

School of Computer Science
University of Manchester
Manchester, UK

\{a.stokes|alvaro|norm\}@cs.man.ac.uk

SSDBM 2014
Basics: Setting the Scene

- You're a researcher on Leach's Storm Petrel's.
- You know of an island where Leach's Storm Petrel's (type of bird) nest for 9 months of every year.
- You're worried about the bird's population levels.
- You decide you want to get some data on the levels of new born petrels.
- You have a very limited budget (£10000).

(http://www.wired.com/wired/archive/11.12/network.html)
Basics: Setting the Scene

Plan A

- Hire out 10 buddies, to go to the island and count new born’s when they emerge from the nests.

Problems

- By disturbing the birds, you’d cause them to not return to the nesting place.
- You can only monitor the birds during the day (12 hours).
- Each buddy wants £50 a day for their time and food. So you can only afford to do this for **20 days**.

http://www.wired.com/wired/archive/11.12/network.html
Plan B

- Buy 49 wireless sensor network motes for £9800 at £200 a mote.
- Buy 98 batteries to power them for ≈ £70.

Benefits

- If motes placed during the 3 months the birds are not there, no disturbance for the birds.
- Can be placed within their barrows to detect heat and therefore a more accurate count.
- You can monitor the birds 24 hours a day.
- You can monitor the birds for much longer (in the range of months or years).

(From [http://www.wired.com/wired/archive/11.12/network.html](http://www.wired.com/wired/archive/11.12/network.html))
Basics: Sensor Networks

WSN
- Static heterogeneous motes
- Radio links
- Data acquired in the network
- Delivered to the gateway

Motes
- Multiple hardware platforms (e.g., micaz, telosB)
- Radio transmitter/receiver
- LEDs
- (Optionally) sensor(s)

Limited Hardware Capabilities
- Finite energy stocks (e.g., batteries).
- Low processing power
- Small memory (e.g., 128kb).
- Short radio range (e.g., 20m - 100m).
Basics: Multihop WSNs

(from http://blogs.salleurl.edu/networking-and-internet-technologies/tag/wns/)
SNQPs

- You don’t want to disturb the birds to replace batteries.
- We want the deployment to last as long as possible.
- You only want readings where the temperature is above a threshold.
- In-network processing can exploit the less energy intensive computation cost in relation to radio transmissions giving an extended lifetime of the network.
- SNQPs generate query evaluation plans (QEPs) that include in-network operators.
- There are a few SNQPs in the literature (e.g., TinyDB [3], AnduIN [2], and SNEE [1]).
A SNEEq! Query

```sql
SELECT RSTREAM b.temp
FROM BARROWS[NOW] b and GROUND[now] g
WHERE b.temp > g.temp
```

% QoS expectations
% acquisition rate: 60 seconds
% delivery time: 1 hour

Problem

- By placement of operators within these QEPs we create non-uniform energy drains on the nodes.
- Once a node fails, the deployment can potentially fail.

Figure 1: An example QEP
Hypothesis, Approach and Aims

**Hypothesis**  By planning adaptations proactively at compile time, we can increase the maximum lifetime of the deployment in relation to the non-uniform energy depletion of QEPs.

**Approach**  By using a TABU to search through sequences of pairs of QEPs and time to switch from it with a genetic search algorithm that generates candidate pairs within the sequences.

**Aims**

1. To avoid node failure.
2. To produce switch times which result in longer deployment lifetime.
Proactive Strategy: Definition

Figure 2: Overall Strategy
Sequence Generator

Description

▶ A TABU search that searches through sequences locating the one with the best estimated lifetime.

Motivation

▶ A TABU search actively avoids searching previously visited areas though a tabu list.

▶ TABU searches work well where an optimised ordering and selection are required.

Figure 3: Sequence Generator Stack
Sequence Generator: A Sequence

Sequence = [(p0, t0), (p1, t1)]

Figure 4: A Graphical Representation of a Sequence
Sequence Generator: 1.1 and 1.2

Generate Neighbourhood

- Generates a set of candidate future states.
- Removes candidates that are tabooed.

Locate Best Solution

- Uses a Fitness function to determine the best candidate (p1)
- Compares ([p0,0],[p1,t1]) to current best sequence ([p0,0]) and returns the best one.

Figure 2: Sequence Generator Stack
Sequence Generator: 1.1 and 1.2

Figure 4: Current Best Sequence

Figure 5: New Current Best Sequence
### Sequence Generator: 1.3

#### TABU List Update Function

<table>
<thead>
<tr>
<th>Pos</th>
<th>Plans Tabued</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>P0, any</td>
</tr>
</tbody>
</table>

**Figure 6:** Tabu list before changes

<table>
<thead>
<tr>
<th>Pos</th>
<th>Plans Tabued</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>P0, any</td>
</tr>
</tbody>
</table>

**Figure 7:** Tabu list when candidate worse

<table>
<thead>
<tr>
<th>Pos</th>
<th>Plans Tabued</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>P0, any</td>
</tr>
<tr>
<td>1</td>
<td>P0, any</td>
</tr>
<tr>
<td></td>
<td>P1, T1</td>
</tr>
</tbody>
</table>

**Figure 8:** Tabu list when candidate better

- ([p0,any]) was added at the beginning of the search.
- ([p1,t1]) was added to the current position.
- ([p1,t1]) was added to the current position.
- ([p0,0],[p1,t1]) added to the next position.
Sequence Generator: 1.4 and 1.5

**Diversity Metric**
- Not executed yet.

**Stopping Criteria**
- Has ran 200 iterations to completion.
- Has not found any improvement and is at the initial point for a period of iterations.
- These were determined empirically.

Figure 2: Sequence Generator Stack
Diversity Metric

- Executed when no improvement foreseen from current position.
- Chooses a random position within the sequence to restart exploration.
- Updates the tabu list to reflect changes.

Figure 2: Sequence Generator Stack
Sequence Generator: 1.3 again

TABU List Update Function

- Entries from future positions are removed.

<table>
<thead>
<tr>
<th>Pos</th>
<th>Plans Tabued</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>P0, any</td>
</tr>
<tr>
<td></td>
<td>P1, T1</td>
</tr>
<tr>
<td>1</td>
<td>P0, any</td>
</tr>
<tr>
<td></td>
<td>P1, any</td>
</tr>
</tbody>
</table>

**Figure 8:** Tabu list before changes

<table>
<thead>
<tr>
<th>Pos</th>
<th>Plans Tabued</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>P0, any</td>
</tr>
<tr>
<td></td>
<td>P1, T1</td>
</tr>
</tbody>
</table>

**Figure 9:** Tabu list after changes
Neighbourhood Generator

Description

- A Genetic search algorithm that generates candidate transitions within the sequences.

Motivation

- A genetic search algorithm uses a population and so is less influenced by starting values.
- A genetic search algorithm will often return a suitable solution in less time than other search methods [4].

Figure 10: Next Candidate Selector Stack
Runtime Module

Description

- Updates the description of the network to reflect the failure of node $n$.
- Executes the compiler that generates a new QEP $q'$ that accounts for $n$.
- Uses $q'$ as a seed to the sequence generation to produce a new sequence that takes account of $n$.

Motivation

- Allows the technique to adapt to node failure events.
Experimental Evaluation

Goals

▶ To gain insights if proactive adaptations could increase the estimated lifetime of a deployment over reacting to node failures.
▶ To study how efficient the decisions made by the proactive strategy are.

Compared Against

▶ A static strategy which has no adaptive behaviour.
▶ A reactive strategy that adapted when a node failed from energy depletion.
Evaluation Constraints

- All valid QEPs must contain all acquisition nodes.
- If an acquisition node fails, no future QEPs can be generated.
Overall Lifetime Measurements

QoS Acquisition rate = 10 seconds, Delivery Time < 600 seconds

Figure 12: Lifetime measurements for static, reactive, and proactive strategies
Figure 13: Lifetimes overlaid with switch times
Related Work

- Most SNQP have little of no adaptive behaviour at runtime, and follow the optimise once, run many paradigm.
- Examples of SNQPs that have no adaptive behaviour are AnduIN[2] and MicroPluse [6].
- The SNQP TinyDB[3] has a reactive ”semantic” routing tree where changes are allowed between parent and child relationships.
Conclusions and Future Work

Conclusions

▶ By planning adaptations that take into account the energy levels on the motes, we can extend the lifetime of the deployment by up to 80% (depending upon the topology, query executed, and QoS expectations).

Future work

▶ We are currently working on a strategy that handles the loss of data through the network from environmental noise and link failure in such a way as to meet QoS expectations more closely than currently possible.
Acknowledgements

Thanks to Ixent Galpin, Christian Y. A. Brenninkmeijer, Alasdair J. G. Gray, and Farhana Jabeen for their work on SNEE and for all their help. Thanks to EPSRC for funding Alan B. Stokes doctoral work.

References

4. Z. Michalewicz and D. B. Fogel. How to solve it.
Generate Initial Population

- A set of heuristics generate a set of routing trees that are used as seeds for the population.
- Extra are generated randomly.
- Switch times range between 0 and 32.

Figure 8: Next Candidate Selector Stack
Neighbourhood Generator: 2.2

**Genome**
- Represents a topology and the switch time.
- $0 \rightarrow$ not active.
- $1 \rightarrow$ active.
- Time is represented by a bit map to one of 32 time stamps.

**Master Genome**
- Xorded with genomes to ensure acquisition nodes and base station are active in each topology.

**Crossover**

![Crossover Diagram]

**Mutation**
- Flips the boolean for nodes.
- Changes the time stamp to a value between 0 and 32.

**Figure 9: CrossOver**
Neighbourhood Generator: 2.3

Check Stopping Condition

- After 50 evolutions.
- No additional candidates have been added for 5 evolutions.
- These were determined empirically.

Figure 8: Next Candidate Selector Stack
Testing Switch Times

**Figure 10:** All possible switch times with estimated lifetimes