Design Document

ID: II

SDGEN: A Synthetic Sensory Data Generator for Bisque

Wenwei Xue Bingsheng He Qiong Luo
Department of Computer Science
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon
Hong Kong, China
{wwxue, saven, luo}@cs.ust.hk
http://www.cs.ust.hk/bisque

This document presents the design of *SDGEN*, a synthetic sensory data generator for Bisque. SDGEN generates a synthetic sensory data set based on an original real-world data set. The major advantages of SDGEN are two-fold: (1) it generates a synthetic data set at an arbitrary spatio-temporal granularity, and (2) the synthetic data set inherits the spatio-temporal trends and correlations of the input real-world data set. The fine spatio-temporal granularity facilitates benchmarking in various setups and the realistic data set makes benchmark results more meaningful.

An analysis of two real-world sensory data sets is first given in Section 1; their limitations bring about the necessity of a synthetic data generator for Bisque. The methodology of synthetic data generation in SDGEN is then presented in Section 2. Section 3 provides a user guide for SDGEN.

1 Design Motivation

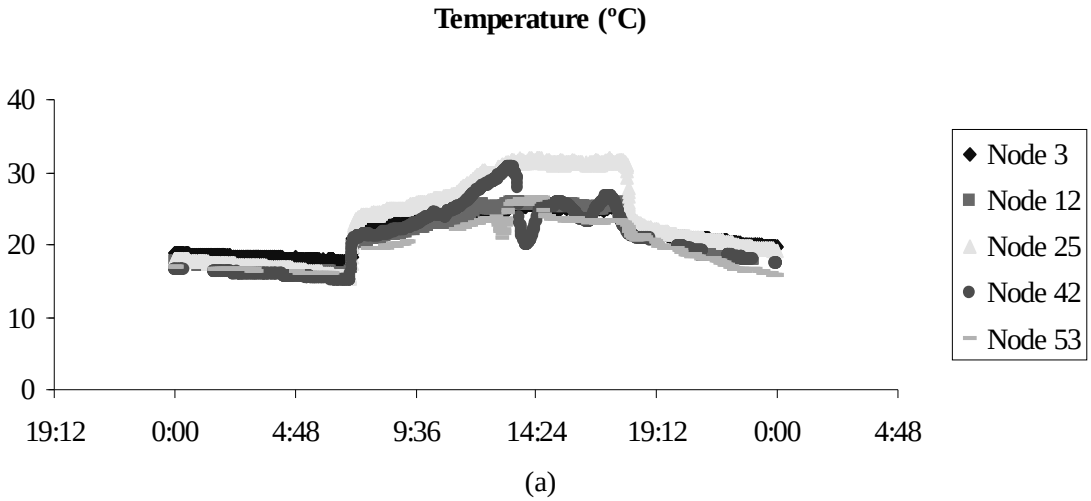
1.1 Analysis of Real-World Data Sets

Different from traditional wireless ad-hoc networks, sensor networks are tightly embedded in and seamlessly interact with the physical world. The sensory data collected from a real-world sensor network deployment reflects the spatio-temporal trends and correlations of the physical phenomena being monitored in the environment. As a result, it is desirable that real-world sensory data collected from field studies can be used for the Bisque database population.

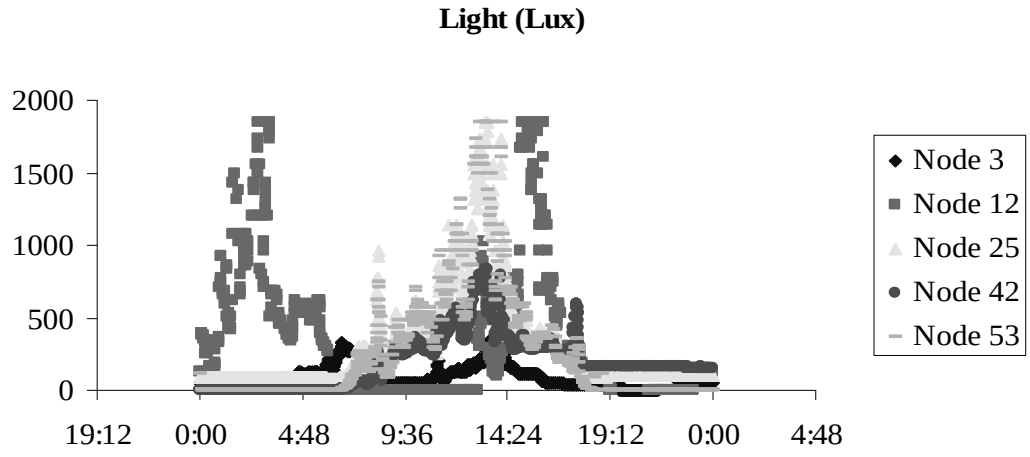
Several real-world data sets are publicly available in the sensor network research community. The Intel Lab data set [4] contains sensory data collected from 54 Crossbow [2] MICA2DOT motes with weather boards deployed in the Intel Berkeley Research lab from Feb 28 to Apr 5 2004. The GDI [3] data set contains habit monitoring data collected from the sensor network deployed on the Great Duck Island in Maine. Some characteristics of these two data sets are given in Table 1. There are two separate subsets of the GDI data set that are from completely different network deployments in the summers of 2002 and 2003, respectively. Only the 2002 data set is investigated in Bisque.

Table 1. Characteristics of the Intel Lab and GDI 2002 data sets

Data Set	#Nodes	Text File Size (MB)	Total #Readings	Sample Period (Seconds)	Attributes in Bisque Database Schema
Intel Lab	54	143	2,313,682	31	nodeid, timestamp, light, temp, humidity, loc_x, loc_y, voltage
GDI 2002	42	255	119,035	71	nodeid, timestamp, light, temp, humidity, pressure, loc_x, loc_y, voltage

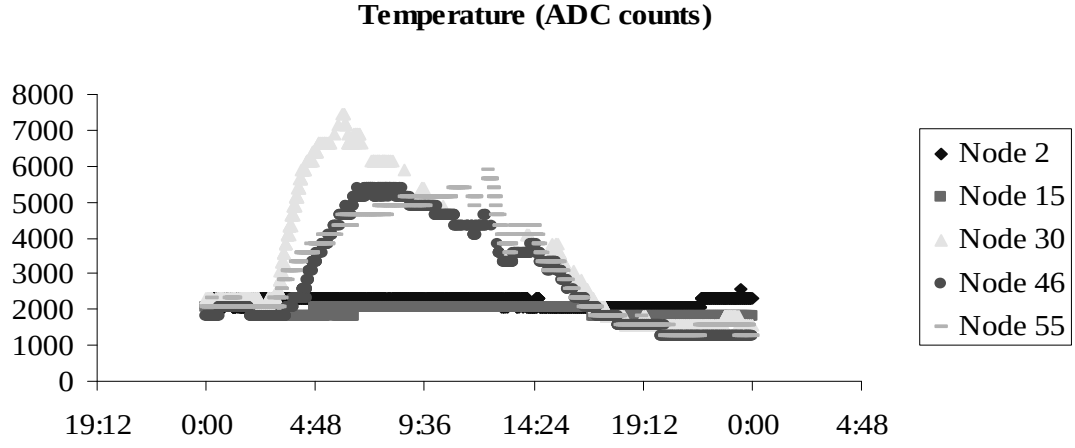


(a)

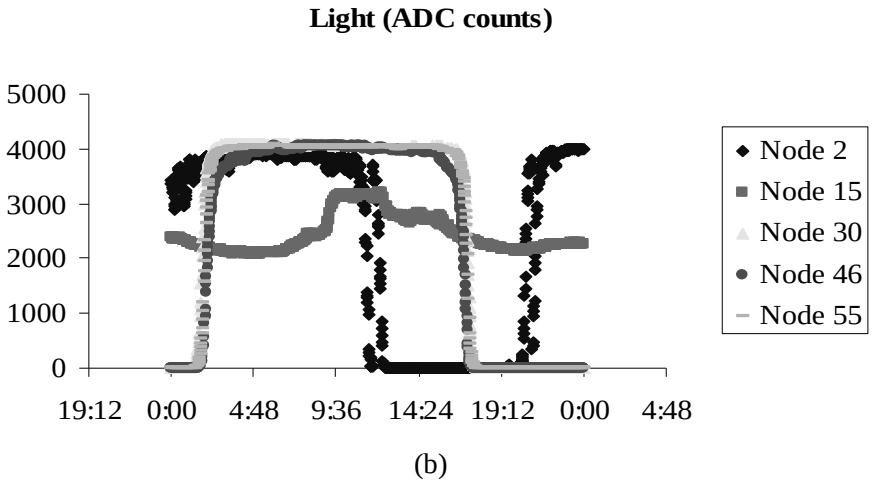


(b)

Figure 1. One-day (a) temperature (b) light readings of five nodes in the Intel Lab data set



(a)



(b)

Figure 2. One-day (a) temperature (b) light readings of five nodes in the GDI 2002 data set

As typical examples, one-day temperature and light readings of a number of nodes in the two data sets are selected to be shown in Figures 1 and 2. The figures illustrate that there exist spatio-temporal trends and correlations in real-world senor readings.

1.2 Limitation of Real-World Data Sets

The preliminary analytical results in Section 1.1 indicate that there are a number of problems in directly using a real-world sensory data set for the Bisque database population:

(1) The spatial granularity of the sensory data in the data set is fixed to a number of nodes in a fixed area. However, benchmarking may require various numbers of nodes and area size.

(2) The temporal granularity of the sensory data is also fixed. Again, benchmarking may require readings at various lengths of sample period.

Considering the limitations of existing real-world data sets for benchmarking, we develop a data generator called SDGEN to generate a synthetic data set with sufficiently fine-grained spatio-temporal granularity given an original real-world data set and two configuration files as the input. The two input configuration files are for the original data set and the target data set respectively. The methodology of synthetic data generation in SDGEN is presented in the next section.

2 Spatio-Temporal Interpolation in SDGEN

Given a set of real-world sensor readings collected from an original network topology, SDGEN applies a simple spatio-temporal interpolation model to this original data set. The model generates a synthetic data set with required sensory data granularity and duration for a target network topology. The network topology is represented in 2-D Cartesian coordinates.

2.1 Spatial Interpolation

Given an original network topology, SDGEN computes the minimum bounding rectangle (*MBR*) that covers all nodes in the original network topology. This MBR is called the *original grid*. When computing the MBR of a network topology, SDGEN slightly enlarges the area of the MBR so that all nodes in the topology lie inside the MBR, not on the boundaries.

Then, SDGEN divides the original MBR into an *m***n* grid with cell width *l*. It is called the *original grid*. Each grid cell contains at most one node in the original network topology. Every node lies inside a cell, not on the cell boundaries. As an example, a 5*5 original grid for a simple real-world sensor network consisting of 9 nodes is shown in Figure 3.

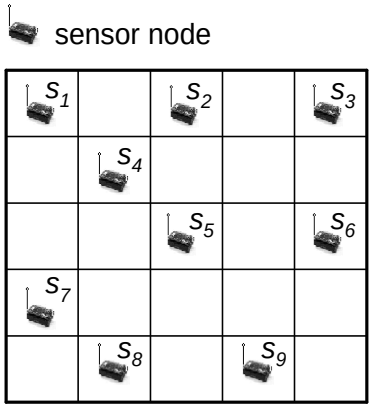


Figure 3. Example original grid

SDGEN assumes that any nodes in one cell have the same reading at one point in time. If a cell (denoted as *S*) contains a node in the original network topology, the reading of the cell is equal to that of the node. If *S* is empty, the reading of *S* is computed as the weighted sum of those of *S*' *k*-nearest neighboring cells that are nonempty (Equation (1)). The value of *k* is an adjustable system parameter of SDGEN with a default value 4.

Assuming the spatial trend of a type of sensor reading is inversely proportional to the distance, we set the weight of a neighbor S_i based on its distance to *S* (denoted as $dist(S, S_i)$) in Equation (2). The distance of two cells is defined as the Euclidian distance between the centroids of the two cells.

$$S.reading = \sum_{i=1}^{k} S_i.reading * w_i \tag{1}$$

$$w_i = \frac{1/dist(S_i, S)}{\sum_{j=1}^{k} 1/dist(S_j, S)} \tag{2}$$

2.2 Temporal Interpolation

SDGEN uses polynomial interpolation [1] to identify the temporal trends of the sensor readings for each node in the original network topology. This converts the discrete sensor readings into a continuous function curve so that readings at any point in time can be computed after the coefficients of the interpolation function are fixed. An example polynomial interpolation curve for a number of readings is shown in Figure 4.

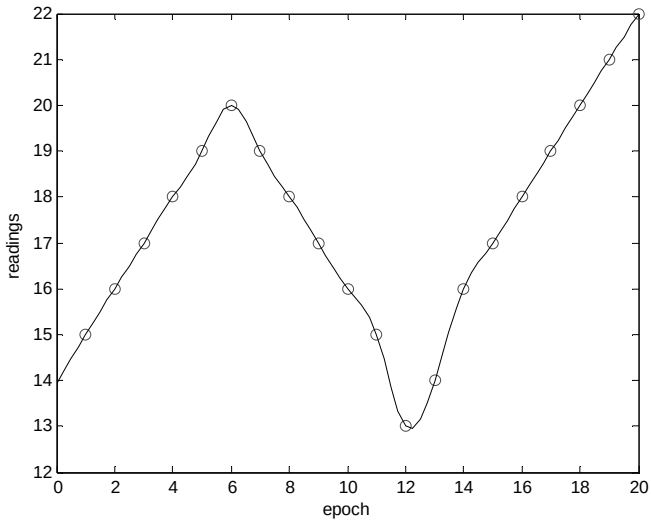


Figure 4. Example polynomial interpolation curve

When applying this temporal model for synthetic data generation, the required time duration of the synthetic data set may be larger than that of the original data set. In this case SDGEN simply repeats readings from the model in rounds. Another problem is that the start timestamp for the sensor readings of each node in the original network topology may be different. To solve this problem, we set the latest start timestamp among all nodes as the start timestamp and perform temporal interpolation on sensory data produced later than this start timestamp.

2.3 Topology Mapping

With the spatio-temporal model, sensor readings at each point in time and at each location within the original MBR are available. In order to apply this model and generate synthetic data for a target network topology, a mapping from the target network topology to the original network topology is needed. SDGEN takes the following steps to establish the mapping:

(1) Compute the MBR for the target network topology, called the *target MBR*.

(2) Scale up or down the target MBR to the same width and height as the original MBR. The locations, i.e., the x-y coordinates, of each node in the target network topology are also scaled proportionally.

(3) Shift the scaled target MBR together with all nodes in it along the x and y axes so that it completely matches the original MBR.

After scaling and shifting, each node in the target network topology falls inside a cell of the original grid. The reading of a node in the target network is set to be that of the cell in the original network. A special case is a node in the target network is mapped to be on a boundary of a cell instead of inside the cell. In this case, the reading of the node is computed as the average of the readings of the cells that share this boundary. An advantage of this topology mapping method is that it preserves the spatial trends and correlations of sensor readings in the original data set.

3 Using SDGEN

This section describes the input and the output of our SDGEN synthetic sensory data generator as well as a quick start guide.

3.1 SDGEN Input

SDGEN requires the following three input files:

(1) An original data set file *data.txt* (plain text file) that contains sensory data collected from a real-world deployment. Each line in the file is a sensor reading tuple from a node at a point in time. The attributes in a tuple are separated by a comma. Figure 5 shows a fragment of an example original data set file.

```
2004-02-28 12:58:46, 3, 20.2040, 36.8871, 50.60, 2.69964
2004-02-28 12:58:47, 1, 19.4298, 39.0763, 60.72, 2.68742
2004-02-28 12:58:47, 4, 19.5964, 37.5737, 172.96, 2.68742
2004-02-28 12:59:16, 2, 19.9884, 37.0933, 45.08, 2.69964
2004-02-38 12:59:16, 8, 19.2534, 38.1897, 57.04, 2.50599
                              ⋮
```

Figure 5. Example original data set file

(2) An original configuration file *original.xml* that describes the original data set. It is an XML text file that contains the following information about the original data set:

- The sample period of the data set.
- The schema of the data set.
- The locations (x-y coordinates) of nodes in the network topology.

Figure 6 shows the input configuration file for the original data set file in Figure 5. The <inputQuery> element contains an SQL-style sensor query that results in the original data set. The order of the attributes in the SELECT clause is the same as that of the attribute values in a line of the data set file. The SAMPLE PERIOD clause gives the sample period (in seconds) of the data set. The schema of the data set specified in this element must satisfy two requirements: (1) it is a

subset of the Bisque database schema, and (2) it includes the *timestamp* and *nodeid* attributes. The data type of each attribute included in the data set schema is known in the Bisque database schema so that it does not need to be specified in the configuration file.

```
<dataSetConfig>
    <inputQuery>
    SELECT timestamp, nodeid, temp, humidity, light, voltage
    FROM sensors
    SAMPLE PERIOD 31
    </inputQuery>
    <nodeLocs>
    <node id="1" x="21.5" y="23" />
    <node id="2" x="24.5" y="20" />
    <node id="3" x="19.5" y="19" />
    <node id="4" x="22.5" y="15" />
        ⋮
    </nodeLocs>
</dataSetConfig>
```

Figure 6. Example original configuration file

Each <node> element under <nodeLocs> specifies the location of one sensor node in the real-world network topology that generates the original data set. The node locations are given in 2-D Cartesian coordinates.

(3) A target configuration file *target.xml* that describes the target data set. Its format is similar to that of the original configuration file and describes the following requirements for the output synthetic data set:

- The sample period and duration of the data set.
- The schema of the data set.
- The network topology of the data set.

Different from the original configuration file, the <nodeLocs> elements of the target synthetic data set can either be specified explicitly, or be specified separately in a network topology file generated by the Bisque network topology generator (see Bisque design document I for the design of this tool). Two corresponding examples are shown in Figure 7.

There are also three requirements for the schema of the synthetic data set specified in the <outputQuery> element: (1) it is a subset of the schema of the original data set specified in the original configuration file, (2) it contains the *timestamp* and *nodeid* attributes, and (3) the LIFETIME clause of the query specifies the duration of the generated synthetic data set.

```
<dataSetConfig>
      <outputQuery>
      SELECT timestamp, nodeid, temp, light
      FROM sensors
      SAMPLE PERIOD 10
      LIFETIME 1000
      </outputQuery>
      <nodeLocs>
      <node id="1" x="10" y="10" />
      <node id="2" x="20" y="10" />
      <node id="3" x="30" y="10" />
      <node id="4" x="10" y="20" />
      <node id="5" x="10" y="30" />
            ⋮
      </nodeLocs>
</dataSetConfig>
```

```
<dataSetConfig>
      <outputQuery>
      SELECT timestamp, nodeid, temp, light
      FROM sensors
      SAMPLE PERIOD 10
      LIFETIME 1000
      </outputQuery>
      <nodeLocs>
      <topology conf=”topology.xrun”/>
      </nodeLocs>
</dataSetConfig>
```

Figure 7. Example target configuration files

3.2 SDGEN Output

The output of SDGEN is a set of data text file *sensori.dat* ($1 \le i \le N$) with a format similar to that of the original data set file in Figure 5. Here N is the total number of nodes in the target network topology. Each file contains a number of synthetic sensor readings for a node.

3.3 SDGEN Quick Start Guide

SDGEN, and one auxiliary tool of it called *SDAnalyzer*, are both developed using Java. The tools are compiled with JDK 1.4.2_03 on Windows XP with Service Pack 2. SDAnalyzer is a preprocessing tool for SDGEN that parses the original data set text file and imports the sensor readings into a table in an MS Access database.

Table 2. SDGEN system requirments

Hardware Requirement	
Processor Speed	P3 133MHz or above
Main Memory (Bytes)	256M or above
Disk Space (Bytes)	512M or above
Software Requirement	
Operating System	Windows 9x/me/2000/XP
Packages Required	• JDK 1.4 or above • MS Access XP • Xerces Java 2 (Version: 2_6_2) (already included in the tool) • ksssci-0.4.0.jar (already included in the tool)

The system requirements for running the current version of SDGEN (Version 1.0) are listed in Table 2. SDGEN uses two packages to handle the XML documents and the interpolation, respectively. The SAX (Simple API for XML) parser provided in the package of Xerces Java 2 (Version 2_6_2) [6] is used to parse the original and target configuration files. The package ksssci-0.4.0.jar [5] is a free java package containing a number of mathematical routines and is used in our temporal polynomial interpolation.

The following are the steps to go through for compilation and execution:

(1) Set the Java environment variable. A batch file named *SetPath.bat* is provided in the SDGEN package to set the *%classpath%* environment variable. The file adds the path of the Xerces and ksssci packages to *%classpath%*.

(2) Compile all java source code in the directory:

*javac *.java*

(3) Use SDAnalyzer to import the original data set text file to an MS Access database.

(4) Prepare the original and target configuration files *original.xml* and *target.xml*.

(5) Generate the synthetic data files using SDGEN with the following command:

java SDGEN -i input.xml -o output.xml

Note that when the data set is large, it may need to increase the heap size of the JVM with the option "*-Xmx<size>*".

java -Xmx<size> SDGEN -i input.xml -o output.xml

4 References

1. Richard L. Burden and J. Douglas Faires. Numerical analysis. 7th Edition. Brooks/Cole, 2001.
2. Crossbow Inc. www.xbow.com
3. Habit Monitoring on Great Duck Island. http://www.greatduckisland.net
4. Intel Lab Data. http://berkeley.intel-research.net/labdata
5. Klassen Software Solutions. http://www.kss.cc/index.shtml
6. Xerces Java 2. http://xml.apache.org/