Fault Tolerant Fusion of Office Sensor Data using Cartesian Genetic Programming

Peter J. Bentley
Braintree Ltd
Department of Computer Science, UCL
London, United Kingdom
p.bentley@cs.ucl.ac.uk

Soo Ling Lim
Braintree Ltd
Department of Computer Science, UCL
London, United Kingdom
s.lim@cs.ucl.ac.uk

***Abstract*—The Smart Grid of the future will enable a cleaner, more efficient and fault tolerant system of power distribution. Sensing power use and predicting demand is an important component in the Smart Grid. In this work, we describe a Cartesian Genetic Programming (CGP) system applied to a smart office. In the building, power usage is directly proportional to the number of people present. CGP is used to perform data fusion on the data collected from smart sensors embedded in the building in order to predict the number of people over a two-month period. This is a challenging task, as the sensors are unreliable, resulting in incomplete data. It is also challenging because in addition to normal staff, the building underwent renovation during the test period, resulting the presence of additional personnel who would not normally be present. Despite these difficult real-world issues, CGP was able to learn human-readable rules that when used in combination, provide a method for data fusion that is tolerant to the observed faults in the sensors.**

Keywords—Smart Grid, sensor fusion, data fusion, Cartesian Genetic Programming (CGP), smart office, fault tolerant, ensemble learning

I. Introduction

The Smart Grid is our response to ever-increasing demand for power. It is a dynamic and diverse system of power supply which responds to demand [1]. This involves a high degree of communication between grid operators, utilities and energy users. For example, in periods of high demand, appliances may be requested to go into power-saving mode and where possible power usage rescheduled for periods of lower demand [2].

For the grid to work there must be a reliable source of information from the end-users about energy demand now and in the future [1]. In the home this can be achieved by the creation of smart devices that are able to monitor power usage, and home hubs that collate and process this information [3]. In a business, however, the problem of monitoring and predicting power demand can be more challenging as it depends on office configuration/layout, the type of business, building usage, facilities/devices installed, people (staff, visitors, cleaners) [4].

In this work, we focus on a real office building in central London used by a computer software company. We investigate the use of a network of smart sensors to gather information about the usage of the building, and whether an evolutionary algorithm can predict occupancy levels by fusing data from multiple sources. As with all hardware-based projects, the sensors were unreliable, resulting in incomplete data [5]. The further goal of this work was to use the evolved solutions to construct a fault-tolerant system that is capable of providing high quality predictions even when some sensors are offline.

The following section of this paper describes related literature. Section III describes the hardware and software used by the system. Section IV describes experiments using CGP to evolve to evolve data fusion rules to predict occupancy levels, with results described in section V and analysis in section VI. Section VII then describes the use of the evolved rules to create a fault-tolerant data fusion system, with VIII and IX providing additional experiments and results. Finally, we conclude in Section X.

II. Background

Researchers have been applying computational intelligence techniques to address sensor related issues such as design and deployment, localisation, security, optimal routing and clustering, scheduling, data aggregation and fusion, and QoS management [6]. Data fusion is a well-known technique that can be useful for the enhancement of data quality [7, 8]. In this section, we review relevant works that use evolutionary algorithms such as GP, GAs, and EP on sensor fusion.

Pinto et al. proposed an approach that allows the implementation of parallel data fusion techniques in IEEE 802.15.4 networks [9]. Through the use of a genetic machine learning algorithm, their approach enables a trade-off between different user-defined metrics.

Chernbumroong et al. investigated the use and contribution of wrist-worn multisensors for activity recognition in elderly people [10]. They found that accelerometers are the most important sensors and heart rate data can be used to boost classification of activities with diverse heart rates. They proposed a genetic algorithm-based fusion weight selection (GAFW) approach which utilises GA to find fusion weights.

Sensors for the same measurement application can differ in terms of cost and accuracy, while fluctuations in environmental conditions can impact both application requirements and available energy. Zowj et al. proposed a method that employs a hierarchy of model ensembles trained by genetic

programming: if model ensembles that poll low-cost sensors exhibit too much prediction uncertainty, they automatically transfer the burden of prediction to other GP-trained model ensembles that poll more expensive and accurate sensors [11].

Yoon and Kim propose an efficient genetic algorithm using a novel normalisation method in order to address the maximum coverage deployment problem in wireless sensor networks [12]. A Monte Carlo method is adopted to design an efficient evaluation function, and its computation time is decreased without loss of solution quality using a method that starts from a small number of random samples and gradually increases the number for subsequent generations.

Kaya et al. proposed a tool condition monitoring system that uses three-axis cutting forces, torque, three-axis accelerometer and acoustic emission sensors [13]. Various time domain and statistical features extracted were used to train support vector machine models in a binary decision tree, which was used to predict the condition of the cutting tool. A genetic algorithm was employed for reducing the dimensionality of the feature set by selecting the features that correlates best with the tool condition.

An alternative method for data fusion is for a mobile agent to fuse the data incrementally as it visits the nodes in a distributed sensor network. Wu et al. proposed a genetic algorithm to compute an approximate solution by suitably employing a two-level encoding scheme and genetic operators tailored to the objective function [14]. Rajagopalan et al. model the mobile agent routing problem as a multi-objective optimization problem, maximizing the total detected signal energy while minimizing the energy consumption and path loss, and successfully applied evolutionary multi-objective algorithms to the problem [15].

A problem of dimensionality can occur with data fusion and it is possible to find an optimal array configuration of reduced dimensionality considering a subset of parameters. Boilot et al. [16] studied the use of pricipal component analysis as a mathemetical transformation and two types of GAs as search methods.

III. Data Fusion using CGP

For this work, a Workplace Management System was developed to monitor an office building and make it smarter. With employees working on different projects on-site and off-site, and with the office laid out across multiple floors, the application was developed to address the following problems:

- Heating or cooling is on when there is no-one in the room
- Lighting is on when there is no-one in the room
- Doors or windows are left open, reducing efficiency of the heating or cooling.
- Computers are left on (unused) overnight and over weekends.
- The availability of rooms for meetings and of people working in the office is unclear, causing wasted journeys by staff.
- It is difficult to report problems with rooms and track whether problems have been fixed or not.
- It is difficult to learn about space usage and optimize space usage, to ensure that no room is over-used or under-used.

A. Hardware (Sensors)

The office is based in central London, UK, and is a self-contained, Grade II listed building[1] of 5 floors and a capacity of about 45 people. Because of its historic status, it is necessary to use wireless sensors; permanently wired sensors would damage the fabric of the structure.

We used the following devices:

- SmartThings[2] Multi Sensor to monitor whether the main office front door is open or closed.
- SmartThings Motion Sensor to monitor movement in a space.
- SmartThings Presence Sensor to know when people arrive or leave the office.
- SmartThings Power Outlet to control lights, electronics and small appliances.
- Aeon Labs MultiSensor 6[3] to monitor humidity, motion, temperature, lights.
- SmartThings mobile app to know when people arrive or leave the office.

Each device can have multiple sensors (see Table I). The SmartThings Hub links to all sensors.

TABLE I. Device and Sensors

Device	Sensors
SmartThings Multi Sensor	Acceleration Battery Contact Temperature ThreeAxis
SmartThings Motion Sensor	Battery Motion Temperature
SmartThings Power Outlet	Power Switch
SmartThings Presence Sensor	Battery Presence
Aeon Labs MultiSensor 6	Acceleration Battery Humidity Illuminance Motion Temperature UltravioletIndex
SmartThings Mobile App	Presence

[1] In UK, a building is listed when it is of special architectural or historic interest considered to be of national importance and there are limitations on the changes that can be made to a building's interior and exterior.

[2] http://www.samsung.com/uk/smartthings/

[3] http://aeotec.com/z-wave-sensor

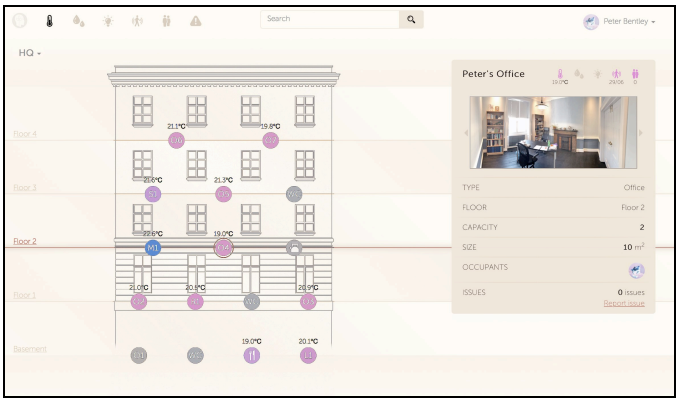

Fig. 1. Screenshot of the Workplace Management System (WMS).

B. *Software*

The Workplace Management System (WMS) is a web application that consists of an admin interface and a user interface. The system interacts with the sensors via the SmartThings REST API[4]. A SmartApp was developed to stream sensor events to the web application. All sensor events are stored in the database of the web application. The SmartApp is developed using Groovy. The web application is developed using the Django web framework[5], Javascript, HTML5, and CSS, and hosted on Amazon Web Services.

Using the WMS, admin users can create offices, add floors and spaces, populate details for each space (e.g., name, image, facilities, space type, capacity, size, location), assign other admins to receive issue reports about a space, update issue status (to do, in progress and completed), and pair the room with the respective sensor(s).

Users of the WMS can have an overview of a specific measure (temperature, humidity, luminance, motion, occupancy, and issues) for all the spaces in the entire office building (Fig. 1), list all the spaces in a floor, and search for specific spaces by name. For each space, users can view sensor details (temperature, humidity, luminance, motion, occupancy), view space details (e.g., name, image, facilities, space type, capacity), report new issues (issues can be about facility, technology, or catering), assign a priority to the issue, view issues and their statuses.

IV. EXPERIMENT 1

For the purposes of this work, the WMS was used to gather data for a specific task: to understand and predict energy usage within the office. Managers of the building report that the level of energy usage is directly proportional to the number of people in the building. However, in such a dynamic environment it is not feasible to count unique people manually (cameras and compulsory presence sensors are seen as an invasion of privacy). Therefore, the task is refined into two parts:

- To use sensor data in order to predict the number of unique members of staff in the building per day.
- To use sensor data in order to predict the number of unique people in the building per day (which in addition to staff may include workmen, surveyors, delivery people, cleaners, and visitors).

Cartesian Genetic Programming [17] (CGP-Library 2.4[6]) modified for this application was used to evolve predictions. It was chosen because it provides a well-established and reliable method for deriving human-readable results, which is important for this real-world application. With an accurate prediction for both parts, it will then be possible to tailor heating/cooling and lighting for the building and have a clear prediction of energy requirements.

A. *Data Cleaning and Feature Extraction*

We studied events for two months, from 1 April 2017 to 31 May 2017 (61 days). For each day, for each device, we extracted sensor data from the database as follows:

- If the device has a humidity sensor, we calculated the min, max, average, count, for all the humidity values for that day.
- If the device has a contact sensor, we calculated the number of times the event value is “open”
- If the device has a motion sensor, we calculated the number of times the event value is “active”
- If the device has a presence sensor, we calculated the number of times the event value is “present”

The result comprised 53 features per day, see Table II. Note that it was observed that during the period there were several sensor failures:

- After the first 26 days, the Kitchen sensor completely failed and stopped producing any data.
- The Ground Floor Office 2 sensor never reported any movement during the entire period (and so this feature was removed altogether).
- For a period of 9 days the Front door contact sensor was offline, producing no data.
- For a period of 18 days the Boardroom sensor incorrectly reported no movement.

Such failures are absolutely typical of current wireless sensor devices, and so the resulting somewhat incomplete data is seen as a realistic – if very challenging – set of features for the experiments. This data was made yet more challenging because in addition to normal staff, the building underwent renovation for 2 weeks of the test period, resulting the presence of additional (very active) personnel who would not normally be present.

[4] http://docs.smartthings.com/en/latest/index.html
[5] https://www.djangoproject.com/
[6] http://www.cgplibrary.co.uk/

TABLE II. SENSORS IN SMART OFFICE

Sensor Location	Measures (Per Day)	Input Number
Basement Lab	Humidity count	i8
	Humidity average	i9
	Humidity min	i10
	Humidity max	i11
	Movement count	i12
Kitchen	Humidity count	i13
	Humidity average	i14
	Humidity min	i15
	Humidity max	i16
	Movement count	i17
Ground Floor Office 1	Humidity count	i18
	Humidity average	i19
	Humidity min	i20
	Humidity max	i21
	Movement count	i22
Ground Floor Office 2	Humidity count	i23
	Humidity average	i24
	Humidity min	i25
	Humidity max	i26
Reception	Humidity count	i27
	Humidity average	i28
	Humidity min	i29
	Humidity max	i30
	Movement count	i31
Boardroom	Humidity count	i32
	Humidity average	i33
	Humidity min	i34
	Humidity max	i35
	Movement count	i36
First Floor Office 1	Movement count	i37
Second Floor Office 1	Humidity count	i38
	Humidity average	i39
	Humidity min	i40
	Humidity max	i41
	Movement count	i42
Second Floor Office 2	Humidity count	i43
	Humidity average	i44
	Humidity min	i45
	Humidity max	i46
	Movement count	i47
Third Floor Office 1	Humidity count	i48
	Humidity average	i49
	Humidity min	i50
	Humidity max	i51
	Movement count	i52
Third Floor Office 2	Humidity count	i53
	Humidity average	i54
	Humidity min	i55
	Humidity max	i56
	Movement count	i57
Front Door	Contact count	i58
Director 1	Presence count	i59
Director 2	Presence count	i60

These 53 features were supplemented with 8 further features indicating the day of the week and whether the day was a holiday, see Table III. (The UK had 3 public holidays in addition to 9 weekends during this period.)

In order to perform supervised learning, the actual number of unique people present at the office on each day was then determined by conducting an extensive survey of every employee and examination of diaries, schedules, events and bookings. This resulted in two sets of labelled data:

TABLE III. CALENDAR INPUTS

Input Name	Input Number
Holiday	i0
Monday	i1
Tuesday	i2
Wednesday	i3
Thursday	i4
Friday	i5
Saturday	i6
Sunday	i7

- Dataset A, comprising 61 features for 61 days, which is labelled with the number of unique members of staff present on each day during the period.
- DataSet B, comprising 61 features for 61 days, which is labelled with the number of unique people present on each day during the period.

B. *Model Checking*

Preliminary experiments were performed using 5x2 cross validation [18] on both DataSet A and DataSet B, creating 5 randomized datasets comprising 31 training and 30 test items for each. These were used for model-checking in order to find optimal number of generations, nodes and optimal function set, to maximize fitness, while minimizing overfitting (i.e., achieve maximum fitness for training and test sets for all 5 datasets). Fitness is the total error (sum of absolute difference between output of evolved solution and correct output for all inputs).

A total of 100 runs for each setting were used, with the result that 200,000 was found to be the optimal number of generations, using 15 nodes, node arity of 2, 61 inputs, 1 output, and a function set comprising the following functions: add, sub, mul, div, abs, sq, sqrt, pow, cube, exp, sin.

C. *Experiment 1.1*

The aim of Experiment 1.1 is to predict the number of unique members of staff in the building per day. Using the settings found above, 1000 runs were performed on a randomized subset of DataSet A comprising 41 training items, and tested on the remaining 20 items.

D. *Experiment 1.2*

The aim of Experiment 1.2 is to predict the number of unique people in the building per day. Using the settings found above, 1000 runs were performed on a randomized subset of DataSet B comprising 41 training items, and tested on the remaining 20 items.

V. RESULTS FOR EXPERIMENT 1

A. *Results for Experiment 1.1*

Table IV (left) shows the top 5 results for Experiment 1.1. In all cases CGP was able to predict the number of unique members of staff with good accuracy – normally error within

1 person. Table V shows the evolved solutions for the experiment.

B. Results for Experiment 1.2

Table IV (right) shows the top 5 results for Experiment 1.2. Again, in all cases CGP was able to predict the number of unique people with good accuracy – normally error within 1 person. Table VI shows the evolved solutions for the experiment.

TABLE IV. AVERAGE ERROR PER DAY FOR TOP 5 SOLUTIONS IN EXPERIMENTS 1.1 AND 1.2

Experiment 1.1 (Mean Error)			Experiment 1.2 (Mean Error)		
Soln	***Training***	***Test***	***Soln***	***Training***	***Test***
294	0.57245	0.783488	179	0.790985	0.8538
789	0.678688	0.830366	580	0.828016	0.95753
501	0.695404	0.851852	859	0.943086	0.969066
94	0.583867	0.858198	488	0.953414	0.976745
89	0.792186	0.865186	417	0.714335	0.989887

TABLE V. TOP 5 EVOLVED RESULTS EXPERIMENT 1.1

Best Solution	Run Number
$\sqrt[4]{\frac{(i47\times i58)+((i18+i37)\times i57)}{e^{i1}}}$	294
$\sqrt{i58+\frac{i13\times i42}{e^{i0/i59}+i1}\text{-}((i3\times i32)\text{-}i60)}$	789
$\sqrt{(i17+(i58-i0/i5)\text{-}(i3+i1)}+i4\times(i59+\sqrt{i42})$	501
$\frac{i17+i58}{\sqrt{i58+\sqrt{i16}}}+\frac{\sin i0}{i5}+i1+i3+i1+i3$	94
$\sqrt{i58+i17}\text{-}i3-\left(\sqrt{i58+i17}*i0\right)+(i4\times i52/i8)$	89

TABLE VI. TOP 5 EVOLVED RESULTS EXPERIMENT 1.2

Best Solution	Run Number
$\sqrt{\sqrt{i22}+i37+i57-i6+i52\text{-}\frac{(i37+i57)\text{-}(i31\times i4)}{i45}}$	179
$\sqrt[4]{((i52+i37+i58)\times(i57+i46)+i57\times i47)\text{-}i49}$	580
$\sqrt[4]{(i4\times i58+i52+i37+i57)\times i46}$	859
$\sqrt{i47+\sqrt{i1+i37}+i57}+\frac{\sqrt{i1+i37}}{i8}+i60$	488
$\sqrt[4]{(i57+i48)\times(i47+i37)}+i59+i4+i1$	417

VI. ANALYSIS

Because the evolved solutions for both experiments are human-readable, it is possible to analyze how CGP uses the input features to produce its output. Fig. 2 shows a tally of input features used for both experiments over all runs. It is clear that many features are rarely used for any solution: typically, those corresponding to Saturday and Sunday (perhaps because *i0* Holiday duplicates their information). Also, humidity features are less commonly used. The most commonly used features are those relating to movement, presence, the front door, and the working days of the week.

While these findings are unsurprising, the results also show a remarkable degree of variation. Fig. 2 shows that some sensors are clearly more useful than others when distinguishing between members of staff (Experiment 1.1) and any person (Experiment 1.2). For example, movement in the Kitchen is a useful way to estimate the number of staff, but not anyone else. Movement in Ground Floor Office 1 (used by the Operations Manager) is useful to estimate the number of people in general, but not the number of staff.

Examining the top five evolved solutions more closely (Tables V and VI) it is possible to see the variation of inputs used by each solution. Tables VII and VIII show the features used as inputs by the top five evolved solutions for each experiment. Each uses a different combination of inputs. In Experiment 1.1 only a single feature is used by all the top five solutions: *i58* the Front Door. Indeed, the simplest solution found initially by CGP is simply the square root of *i58,* i.e., the number of unique staff in the building is very roughly equal to the square root of the number of times the front door opens and closes. This concept is refined, but still visible in all five of the top solutions for Experiment 1.1.

In contrast, for Experiment 1.2, the only common input feature for all five top solutions is *i57* Third Floor Office 2 Movement Count, a room used by support staff, and the front door is no longer an important feature.

The fact that the top five solutions for each experiment tend to use different features to generate the results is an extremely important result, for it enables the creation of an ensemble-based approach that may be tolerant to faults.

VII. FAULT-TOLERANT ENSEMBLE-BASED DATA FUSION

Based on the analysis above we created a fault tolerant data fusion system. The system is an ensemble learning approach, making use of the best five (out of 1000) evolved solutions as described above. Based on the availability of sensor data, and the use of the sensors, the best ranked solution is applied to new data for the given day, see Algorithm 1.

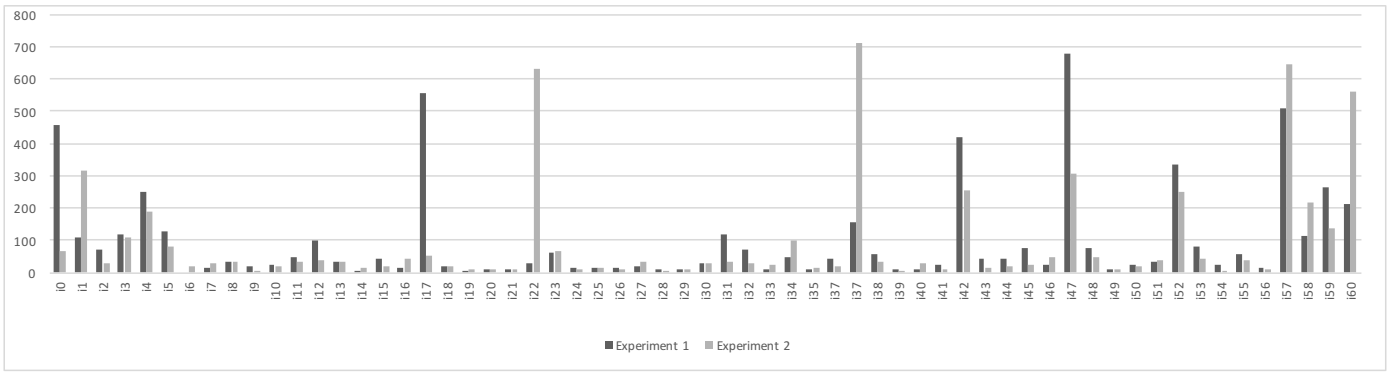

Fig. 2. Usage (y-axis) of different sensors (x-axis) over 1000 evolved solutions for both experiments. In Experiment 1.1 the commonly used inputs (used for more than 1 in 5 solutions) are: i0, i4, i17, i42, i47, i52, i57, i59, i60. In Experiment 1.2, the commonly used inputs (used for more than 1 in 5 solutions) are: i1, i22, i37, i42, i47, i52, i57, i58, i60.

TABLE VII. USE OF INPUTS BY TOP 5 EVOLVED RESULTS FOR EXPERIMENT 1.1

Run Number	i0	i1	i3	i4	i5	i8	i13	i16	i17	i18	i32	i37	i42	i47	i52	i57	i58	i59	i60
294										X		X		X		X	X		
789	X	X	X				X				X		X				X	X	X
501	X	X	X	X	X				X				X				X	X	
94	X	X	X		X			X	X								X		
89	X		X	X		X			X						X		X		

TABLE VIII. USE OF INPUTS BY TOP 5 EVOLVED RESULTS FOR EXPERIMENT 1.2

Run Number	i1	i4	i6	i8	i22	i31	i37	i45	i46	i47	i48	i49	i52	i57	i58	i59	i60
179		X	X		X	X	X	X					X	X			
580							X		X	X		X	X	X	X		
859		X					X		X				X	X	X		
488	X			X			X			X				X			X
417	X	X					X			X	X			X		X	

Algorithm 1: Ensemble-based Data Fusion using previously evolved CGP rules.

1: **For** each day:
2: Get currently available sensor data.
3: Rank evolved solutions into order of number of required inputs that match non-zero sensor data.
4: **If** any top-ranked solutions have equal ranking **then**
5: Choose solution with highest fitness ranking from previous experiments.
6: **Else**
7: Choose top-ranking solution.
8: **End If**
7: Apply solution to data.
8: **End for**

VIII. EXPERIMENT 2

The second experiment investigates this fault tolerant ensemble-based data fusion system (using the 5 best evolved solutions for Experiment 1.1 and 1.2 described previously) by testing on both datasets with further faults added.

A. Experiment 2.1

To test the fault tolerant data fusion system, Dataset A described in the previous experiments was randomly corrupted with 50 sensor readings set to zero, in order to simulate 50 day-long failures spread over all the inputs, in addition to the existing sensor failures. Algorithm 1 was then applied (using the five best evolved results as shown in Table V).

B. Experiment 2.2

To test the fault tolerant data fusion system, Dataset B described in the previous experiments was also randomly corrupted with 50 sensor readings set to zero, in order to simulate 50 day-long failures spread over all the inputs. Algorithm 1 was then applied (using the five best evolved results as shown in Table VI).

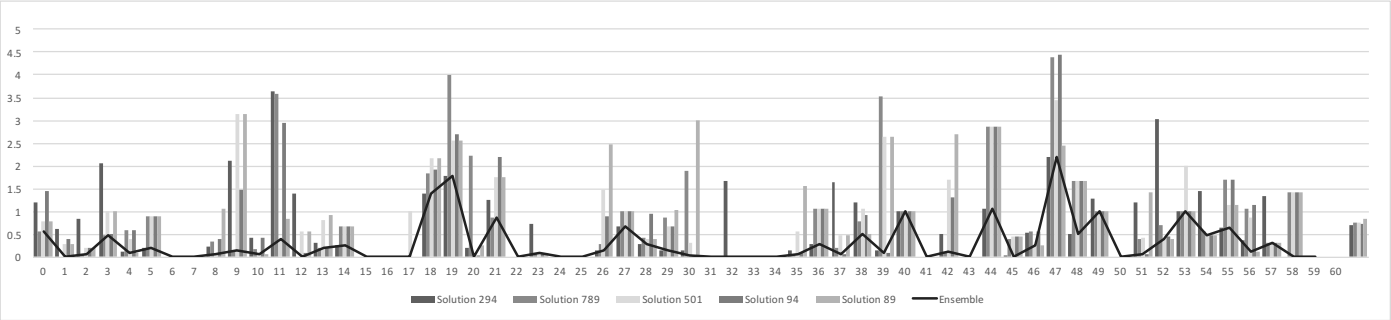

Fig. 3. Ensemble-based Data Fusion using CGP rules for Experiment 2.1. Error (y-axis) of the 5 previously evolved solutions (number of predicted people too few or too many) per day (x-axis) on Dataset A with 50 further random errors added. The line shows the result produced by the ensemble solution.

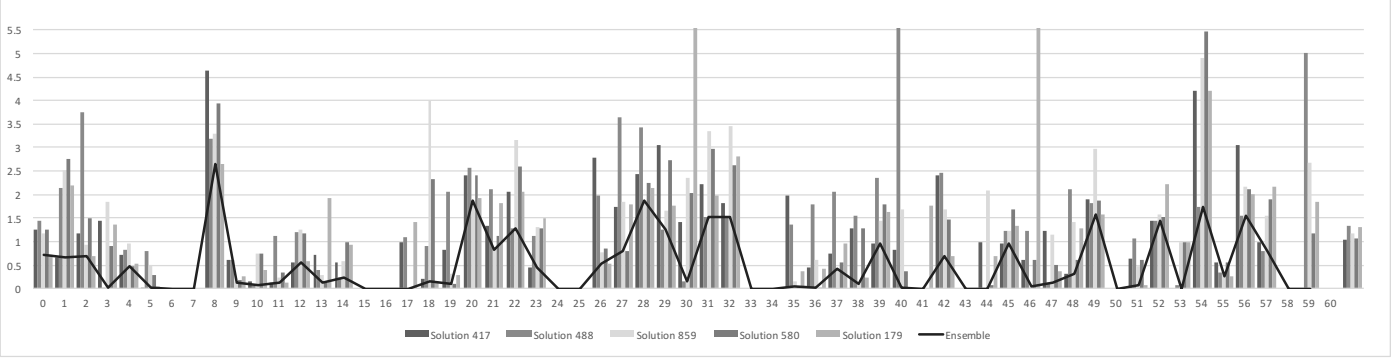

Fig. 4. Ensemble-based Data Fusion using CGP rules for Experiment 2.2. Error (y-axis) of the 5 previously evolved solutions (number of predicted people too few or too many) per day (x-axis) on Dataset B with 50 further random errors added. Results for Solution 488 day 41, and Solution 179 days 31 and 47 are cropped as the true values were very large. The line shows the result produced by the ensemble solution.

IX. Results for Experiment 2

A. Results for Experiment 2.1

Table IX (left) shows the average error for the five solutions for Experiment 2.1 and the result of the fault-tolerant ensemble learning system. Fig. 3 shows the results for each day. Performance is slightly degraded by the additional loss of sensor information, with some solutions at worst predicting 4 members of staff too many or too few in the building. The ensemble result is extremely good, on average only 0.3 wrong (less than half the average error of the best evolved solution alone) and at worst predicting 2 members of staff too many or too few.

B. Results for Experiment 2.2

Table IX (right) shows the average error for the five solutions for Experiment 2.2 and the result of the fault-tolerant ensemble learning system. Fig. 4 shows the results for each day. The additional loss of sensor information significantly affected performance in some cases, with solutions 488 and 179 failing completely for some days, predicting an infinite number of people in the building. Nevertheless, the ensemble result remains extremely good, with the average error less than half the error of the best solution alone, and at worst predicting 2 people too many or too few in the building.

TABLE IX. Average Error per Day for Top 5 Solutions in Experiments 2.1 and 2.2

Experiment 2.1		**Experiment 2.2**	
Soln	***Mean Error***	***Soln***	***Mean Error***
294	0.688888917	179	∞
789	0.766368133	580	1.068208933
501	0.766046983	859	1.183191533
94	0.713926983	488	∞
89	0.836625633	417	1.033114467
Ensemble	0.301135333	Ensemble	0.49976155

X. Conclusions

Smart Grids require good information from energy users. In this work, we applied Cartesian Genetic Programming to the problem of understanding and predicting energy usage within a real London office building, by fusing data gathered from smart sensors. In the building, power usage is directly proportional to the number of people present. CGP was used to perform data fusion on the data collected from smart sensors embedded in the building in order to predict the number of people over a two-month period. This was a challenging task, as the sensors were unreliable, resulting in incomplete data. It

was also challenging because in addition to normal staff, the building underwent renovation for 2 weeks of the test period, resulting the presence of additional personnel who would not normally be present. Despite these difficult real-world issues, CGP was able to learn human-readable rules that provided low error rates (occupancy prediction no more than 2 people wrong) for both staff and for all people who visited the office. One of the key benefits of CGP was that it provided a large variety of solutions, each using a different subset of sensor inputs to produce the same result. This enabled the creation of an ensemble learning fault-tolerant data fusion system which applies the appropriate evolved rules based on which subset of sensors are currently operational. Experiments showed that this fault-tolerant system showed good performance even when 50 further faults were simulated over the 61 days, outperforming all evolved rules alone.

REFERENCES

[1] X. Fang, S. Misra, G. Xue, and D. Yang, "Smart grid—The new and improved power grid: A survey," *IEEE Communications Surveys & Tutorials,* vol. 14, pp. 944-980, 2012.

[2] H. Bae, J. Yoon, Y. Lee, J. Lee, T. Kim, J. Yu, and S. Cho, "User-friendly demand side management for smart grid networks," in *2014 International Conference on Information Networking (ICOIN)*, 2014, pp. 481-485.

[3] T. Hargreaves, M. Nye, and J. Burgess, "Making energy visible: A qualitative field study of how householders interact with feedback from smart energy monitors," *Energy Policy,* vol. 38, pp. 6111-6119, 2010.

[4] J. K. Gruber, M. Prodanovic, and R. Alonso, "Estimation and analysis of building energy demand and supply costs," *Energy Procedia,* vol. 83, pp. 216-225, 2015.

[5] V. C. Gungor, B. Lu, and G. P. Hancke, "Opportunities and challenges of wireless sensor networks in smart grid," *IEEE Transactions on Industrial Electronics,* vol. 57, pp. 3557-3564, 2010.

[6] R. V. Kulkarni, A. Forster, and G. K. Venayagamoorthy, "Computational intelligence in wireless sensor networks: A survey," *IEEE Communications Surveys & Tutorials,* vol. 13, pp. 68-96, 2011.

[7] D. L. Hall and J. Llinas, "An introduction to multisensor data fusion," *Proceedings of the IEEE,* vol. 85, pp. 6-23, 1997.

[8] J. Dong, D. Zhuang, Y. Huang, and J. Fu, "Advances in multi-sensor data fusion: Algorithms and applications," *Sensors,* vol. 9, pp. 7771-7784, 2009.

[9] A. R. Pinto, C. Montez, G. Araújo, F. Vasques, and P. Portugal, "An approach to implement data fusion techniques in wireless sensor networks using genetic machine learning algorithms," *Information fusion,* vol. 15, pp. 90-101, 2014.

[10] S. Chernbumroong, S. Cang, and H. Yu, "Genetic algorithm-based classifiers fusion for multisensor activity recognition of elderly people," *IEEE Journal of Biomedical and Health Informatics,* vol. 19, pp. 282-289, 2015.

[11] A. Yousefi Zowj, J. C. Bongard, and C. Skalka, "A genetic programming approach to cost-sensitive control in resource constrained sensor systems," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, 2015, pp. 1295-1302.

[12] Y. Yoon and Y.-H. Kim, "An efficient genetic algorithm for maximum coverage deployment in wireless sensor networks," *IEEE Transactions on Cybernetics,* vol. 43, pp. 1473-1483, 2013.

[13] B. Kaya, C. Oysu, H. M. Ertunc, and H. Ocak, "A support vector machine-based online tool condition monitoring for milling using sensor fusion and a genetic algorithm," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture,* vol. 226, pp. 1808-1818, 2012.

[14] Q. Wu, N. S. Rao, J. Barhen, S. Iyenger, V. K. Vaishnavi, H. Qi, and K. Chakrabarty, "On computing mobile agent routes for data fusion in distributed sensor networks," *IEEE Transactions on Knowledge and Data Engineering,* vol. 16, pp. 740-753, 2004.

[15] R. Rajagopalan, C. K. Mohan, P. Varshney, and K. Mehrotra, "Multi-objective mobile agent routing in wireless sensor networks," in *The 2005 IEEE Congress on Evolutionary Computation*, 2005, pp. 1730-1737.

[16] P. Boilot, E. Hines, M. Gongora, and R. Folland, "Electronic noses inter-comparison, data fusion and sensor selection in discrimination of standard fruit solutions," *Sensors and Actuators B: Chemical,* vol. 88, pp. 80-88, 2003.

[17] J. F. Miller and P. Thomson, "Cartesian genetic programming," in *European Conference on Genetic Programming*, 2000, pp. 121-132.

[18] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural Computation,* vol. 10, pp. 1895-1923, 1998.