10-2012

Automation Techniques for Intelligent Environments - Prediction of Building Activity Patterns Using a Cyclic Genetic Algorithm

Gary Parker
Connecticut College, parker@conncoll.edu

David T. Alpert
Connecticut College, tatsuroalpert@gmail.com

Recommended Citation
Parker, G.B.; Alpert, D.T., "Automation techniques for intelligent environments," Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on , vol., no., pp.202,207, 14-17 Oct. 2012 doi: 10.1109/ICSMC.2012.6377700

Automation Techniques for Intelligent Environments - Prediction of Building Activity Patterns Using a Cyclic Genetic Algorithm

Keywords
evolutionary computation; smart houses; behavioral modeling; ambient intelligence; genetic algorithm

DOI10.1109/ICSMC.2012.6377700

Automation Techniques for Intelligent Environments

Prediction of Building Activity Patterns Using a Cyclic Genetic Algorithm

Gary B. Parker and David T. Alpert
Department of Computer Science
Connecticut College
New London, CT, USA
parker@conncoll.edu, tatsuroalpert@gmail.com

Abstract— **This work involves learning the use schedule of an academic building in order to intelligently control various aspects of the environment. Motion sensors are used to monitor and record the activity of each of the rooms in the building. After a basic preprocessing of the data, a Cyclic Genetic Algorithm (CGA) is used to pick out the patterns of use of the rooms. The CGA is seen as ideal for such a problem because of its ability to find repetitive cyclic patterns in the data. Our results show that a CGA has the ability to pick out such patterns and construct a schedule of use for a room.**

Keywords: evolutionary computation; smart houses; behavioral modeling; ambient intelligence; genetic algorithm

I. INTRODUCTION

In this digital age, our homes are full of devices designed to make our lives easier, ranging from simple things like an alarm clock, to more advanced items such as the remotely accessible espresso machine in the kitchen and the automated TV recorder in the living room. Smart home technology is designed to link these individual items so that they can be used in combination to increase their usefulness. Current work in the field focuses on the automation of these features and making the home a dynamic object that can better the comfort of its inhabitants.

An important aspect of these intelligent environments is the sensors which are installed in them. There are many sensors currently in our homes that benefit our daily lives. They range from simple items like a motion sensor on a garage light or a thermostat in a heating unit to a more complicated security system with a combination of motion sensors, temperature sensors, window or door sensors, and possibly a camera. As the technology advances, sensors have become more readily available and can be installed in a home more easily for less cost. These sensors have a lot of potential uses and a method to harness their power could be very beneficial.

Although the number of sensors in an average home and their uses in automation of simple tasks is increasing, there is a limit to what these sensors can do individually. Linking these sensors into a sensor network will allow them to work together and will create more possibilities for the automation of more complex tasks. For example the motion sensor of a security system is only used when it is armed and otherwise idle. If the motion sensor were connected to the thermostat in a heater, the motion detected during its idle periods could be used to determine whether a room is not occupied, and based on that information, turn down the heat to save energy.

Yet, simply connecting a bunch of sensors together will not lead to this kind of advanced behavior. They must be connected to a central computer "brain" which collects the data, interprets it, makes decisions, and finally takes actions based on it. The combination of a sensor network and a central "brain" will extend the potential of individual sensors and allow them to be more useful in a living environment.

Finally, in order to automate the control of the environment in a truly effective way, the computer must be equipped with an Artificial Intelligence (AI) system which will make decisions based on the sensor data, allowing the system to run in the background without constant human supervision. An AI system is the most fit for this task because it has the ability to learn patterned behavior. The events in a living environment tend to exhibit patterns, as humans are creatures of habit. Through the generalization of repeated observations, AI systems are able to learn such patterns from abstract data like those collected from the sensor network.

Another important aspect of an AI system is its ability to adapt to various situations as well as changes in the environment. As all living spaces and occupants are unique, it is impossible to design one ideal system which will work in any given setting. Thus its adaptability will allow an AI-based system to adjust to its unique environment and establish control patterns that are adapted to its situation.

Research on such intelligent environments, with various sensors connected up to a computer employing an AI system, has been conducted at various academic institutions around the world. Yet this is still a new area of study and there is a lot to be discovered. AI itself is also a rapidly developing field in computer science, and many new techniques have not yet been applied to this problem domain.

II. RELATED WORKS

In a brief overview of advances in these technologies, Cook and Das [1] describe the use of artificial intelligence to automate homes, which they refer to as smart environments.

They argue for the idea that a sensor network will comprise a necessary backbone to these environments, and will in turn be connected to a centralized intelligence, which makes decisions based on information collected from the sensor network and sends commands to its controlled devices in the home accordingly. They provide several examples of such systems, one of which is based on X-10 technology. X-10 systems control devices using power line communication (PLC) by sending small disturbances along the electrical wiring, a potential advantage over other technologies, since such lines are already present in most all homes. This kind of system allows for easy setup of simple smart home environments. Using the built-in sensor network and artificial intelligence capabilities, such a system can be designed to learn the inhabitants' way of life and adjust the devices in its control to better suit their needs.

In order for a system to understand a person's needs, it must first construct a model of his/her living style and patterns. This is done through the recording and analysis of various forms of data collected by the sensor network. Youngblood et al. [2] describe a system called MavHome (Manage Adaptive Versatile environments), which attempts to model everyday human activities in a home environment, and determine which activity is taking place based on sensor data. This allows the environment to adapt to the specific needs of the occupants. They employ a Hierarchical Hidden Markov Model (HHMM) to learn and predict the activities taking place. This approach is based on the idea that humans are creatures of habit whose actions will show a certain degree of periodicity, thus allowing the HHMM to pick up on the pattern of events leading to a particular activity.

Once a reliable model for an inhabitant is developed, it can be used to make decisions and predictions about the inhabitant's lifestyle. The applications for such a model are numerous, ranging from medical and environmental to recreational. One use, described by Helal et al. [3], involves health monitoring for the elderly. Once a baseline for ordinary behaviour and lifestyle is established, the system can detect deviations and report them to a medical facility, allowing elderly people to live independently and providing peace of mind for their family members. It has been discovered that in some situations, similar systems were able to detect slight deviations which were the symptoms of diseases, such as Alzheimer's in its early stages.

Environmental issues and natural resources are also topics of great interest. In his overview, Hagras [4] discusses the use of computational intelligence (CI) techniques in living environments to reduce energy consumption. He divides CI into the three predominant areas of fuzzy systems, neural networks (NN), and genetic algorithms (GA), and provides various examples where each is utilized.

CI models have also been used to predict energy consumption of larger environments such as office and academic buildings. Bailey and Curtiss [5] describe the use of a NN to adjust heating and air-conditioning to run efficiently and save energy. Other examples focus more on smaller personal living environments. One such system at the University of Colorado as described by Mozer [6] uses a NN to adaptively control environmental parameters such as lighting and room and water temperatures. These systems are designed to reduce energy consumption while maintaining the inhabitant's comfort level.

Another possible application for a behaviour model is to use it as a basis to control different aspects of a living environment in an effort to conserve energy. If an inhabitant's schedule can be determined, the environment can learn to take actions which reduce the waste of energy. If for instance, on a typical week day, the environment determines that the occupant is not home between the hours of 8AM and 6PM, energy consuming appliances such as the water heater or the air conditioning can be turned off until shortly before his/her arrival home.

Before these decisions can take place, the system needs to construct a reliable model which can be used to accurately predict the behaviour of the inhabitant. Thus methods for determining this model or schedule of activities are of great interest.

The goal of the work presented in this paper is to learn a schedule of use for a particular room so that it can be used as a basis for making a prediction as to when the room is going to be in use. If a system can make such a prediction, it can make better decisions on how to control various aspects of the environment. In some cases, say lighting control, knowing the use of a room in advance is not necessary. Simple systems linking a motion sensor to a light switch already exist, and are very effective because lights can be turned on or off almost instantaneously. Advance knowledge of use becomes more important when the aspect being controlled takes time to reach its desired state. Heating control is one such environmental variable. Because heating a room to a desired temperature takes time, the ability to predict the use of a room in advance can be extremely beneficial. Conversely, if it can be predicted how long a person will be occupying a room, the heat can be shut off and the room can begin to cool without causing much discomfort to the occupant.

While this sort of system would be extremely effective in an ideal situation, human beings do not often stick to a precise schedule and tend to act somewhat unpredictably. Thus it is important for a system to include a reactive component, which can make real-time decisions which can override previous decisions. While this work does not include a real-time component, it attempts to develop a system which can create a schedule for a reliable prediction which a system could use to bias its decisions. It will focus on the use of Cyclic Genetic Algorithms to help construct this model for prediction.

III. The Environment and Data

The environment of interest is Strider House, an academic building on the Connecticut College campus used by the Computer Science Department. It houses two computer labs and three office spaces. Strider house is an

ideal domain for modeling as its use is somewhat consistent and predictable, such as weekly scheduled labs and classes. At the same time its use has enough variation from students coming in to do work or other unscheduled uses, to warrant an adaptive system. Each of the rooms was configured to be monitored by a motion sensor (Figure 1) that records any activity taking place. The sensors are strategically placed to capture as much of the motion in the room as possible.

Figure 1. X-10 motion sensor

The motion sensors are part of an X-10 system, as described previously, and communicate with a base station by radio frequency. Data from these motion sensors are recorded on a computer linked to the base station in a format as seen in Table 1. Each sensor event is labeled with a date and time stamp, signal type, and sensor ID. The sensor ID is unique for every sensor and signifies which room the data was received from.

TABLE I. SAMPLE DATA

Date	Time	Signal Type	Sensor ID
11/12/2010	09:30:00:359	RFC	b2
11/12/2010	09:30:49:734	RFC	b1
11/12/2010	09:30:50:000	RFC	b1
11/12/2010	09:30:57:562	RFC	b2

As discussed by Galushka et al. [7], raw data collected from a sensor is often too complex or abstract to perform any sort of data mining. Thus some form of preprocessing is necessary to convert it into a form which can be useful. For this work, the timing and frequency of the sensor events during the hour of interest were used to determine whether a room could be considered "in use" during that hour. This use status was recorded in an easily accessible table.

IV. LEARNING METHOD

The learning method employed in this work is a form of Genetic Algorithm (GA) known as a Cyclic Genetic Algorithm (CGA). The traditional GA, introduced by Holland [8], is a learning algorithm based on the concepts of heredity and the survival of the fittest. It consists of a population of possible solutions to a problem referred to as chromosomes. These chromosomes are often binary strings of fixed length that represent the solution in some way. Each one is assigned "fitness" as to how well it solves the problem. Based on this fitness, they undergo the three genetic operators selection, crossover, and mutation to create a new generation of chromosomes which theoretically are better solutions. Through this process an optimal, or near optimal solution is learned.

The CGA, as discussed by Parker [9], is unique in that it incorporates time as a factor where the chromosome represents a sequential solution that is to be carried out linearly in a given span of time. In addition, the solution, or part of it, can be looped so that a cycle is created. This cycle can represent patterns in the problem that are repeated multiple times. In his work, Parker uses these CGAs to generate an ideal walking gate for hexapod robots. The cyclic portion of the chromosome is ideal for the repetitive nature of this problem. While the robot is walking, the control sequence of the legs must loop the same commands repeatedly. The cyclic portion of the chromosome learns to represent this pattern, which during execution can be repeated the necessary number of times.

In our work, we use the CGA to pick out the cycles in the use patterns of a room over an extended period of time. The type of environment we work with has a tendency to be used with regularity, thus creating a pattern which can be detected. For instance a computer lab may be used for a class on Wednesday afternoon every week, while an office space is used in the morning twice a week on Tuesday and Thursday. This kind of periodicity in the use allows for the CGA to pick a cycle length equal to a week. Although in this particular environment we know that the length of the cycle of use is a week, the CGA has the ability to pick out a cycle of other lengths such as a two or three day cycle. Its ability to pick out the cycle with minimal a priori knowledge is what makes the CGA a powerful tool.

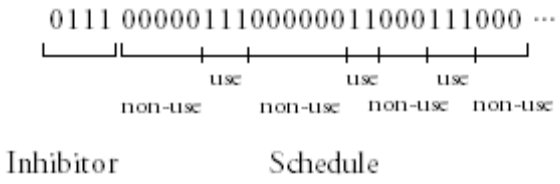

Figure 2. Chromosome representation

A. CGA Chromosomes

Although a CGA can work with both fixed and variable length chromosomes, for the purposes of this work, we only use fixed length chromosomes. Each chromosome (Figure 2) consists of a 4 bit inhibitor (as described in the next section) and 256 bits to represent the schedule. A chromosome of size 256 bits was chosen because it is long enough to represent 10 days worth of use. This allows the

algorithm to pick out cycles of varying lengths ranging from a single day up to ten days. Each bit represents the use of a particular hour in the schedule, where a 1 stands for a period of use and a 0 for a period of non-use. The chromosome in Figure 2 represents a schedule of five inactive hours, followed by three active hours, followed by six inactive hours, and so on and so forth. The down side to this representation is that, representing cycles longer than a week, for instance a bi-weekly schedule, requires a large number of bits and can slow the computation greatly. While other methods which represent the hours of use in a reduced format were attempted, the full representation proved to be the most effective.

B. Inhibitors

The inhibitor portion of the chromosome is used to limit some aspect of its function to prevent it from developing in a particular way that does not suit the problem. In our work, we use the inhibitor to limit the length of the cyclic portion of the chromosome. While the 256 bit chromosome has the ability to represent a schedule just over 10 days long, and will have data for that full length, the algorithm must learn the correct cycle length, which is most often shorter than 10 days. Thus, the value of the inhibitor represents the length of this cycle in days, allowing it to pick shorter cycles. It is a 4 bit binary number, as seen in Figure 2, which is initialized as a random value and is learned over the generations by the algorithm.

When calculating the fitness of a chromosome for selection (as described further in the next section), the value of the inhibitor, in days, is multiplied by 24 to convert it to hours. This value then determines the number of bits from the remaining part of the chromosome that will be used for the cyclic portion. For instance, if the inhibitor has a value of 7 days, the length of the cyclic portion will be 168 bits, which is equal to the number of hours in 7 days. This 168 bit portion is then compared to the training data to evaluate its accuracy. This portion is repeated as many times as necessary to match the length of the data. If the cycle in the data is actually 7 days, it will match and return a high fitness value. If the cycle is different, it will be less and less accurate and return a poor fitness.

We use days as a unit of cycle length based on the assumption that a cycle should not be shorter than a day. Making this limitation also allows the algorithm to pick out a schedule more effectively without getting confused by minor periodicities in the data.

C. Genetic Operators

The probability for selection was calculated based on fitness, which is a function of how well the proposed schedule matches the actual data. The cyclic portion of the chromosome is selected, as discuss in the previous section, and is then lined up with the training use schedule so that each hour is compared and evaluated. The scoring of correct and incorrect hours is determined by the reward scheme shown in Table 2.

TABLE II. FITNESS REWARDS

Actual	Learned	Reward
1	1	x
1	0	-1
0	1	-1
0	0	0

In observing the training data it was clear that the number of active hours is highly disproportionate to the number of inactive hours. This causes a problem because, in a simple scoring scheme, a schedule of always inactive scores very well. To counter this, the reward for a correct prediction was weighted to compensate, as denoted by the x in Table 2. This weighting was the proportion of inactive to active hours in the data, and was adjusted for every training set to match the proportion in that data. For instnace, if the ratio of unused hours to used hours in the data is 10:1, the reward x for a correct use prediction would be 10 points. This scheme, in combination with the -1 reward values for incorrect values, ensured that both the always inactive and always active solutions, on average, would score 0 points.

Once fitness was calculated, the method known as elitism was used, where the highest scoring individual of a population carries directly on to the next, to ensure positive overall fitness growth. All other individuals of the new population are a result of a crossover between two mates stochastically selected where the probability of selection was proportional to the fitness.

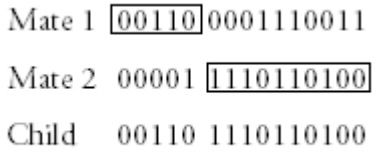

Figure 3. One point crossover

Two types of crossover, single point and universal, were used with equal probability. Single point crossover involves picking a random point along the length of the chromosome and splitting it into two parts, as seen in Figure 3. The resulting chromosome will consist of the first part of one mate and the second part of the other. The order of the mates is selected randomly to prevent biasing towards one of the mates. Universal crossover is a much simpler process where every bit in the chromosome, including the bits in the inhibitor, has an equal chance of being selected for the new chromosome. Finally, the new chromosome that was created as a combination of its two parents undergoes a basic mutation scheme where every bit in the new chromosome has a 0.3% chance of being flipped.

V. TESTS

Training was conducted for 10,000 generations using populations of size 320. The initial population was seeded with randomly generated chromosomes. Training was conducted on both simulated and real data. Each test was repeated five times.

Two different types of simulated data were used. The first, which will be referred to as Type I, was constructed by taking a single week of real data and repeating it 10 times to simulate 10 identical weeks of data. The second, Type II, was constructed by taking a single week of real data and modifying it to simulate a second week with some schedule changes. This modified week was then adjusted further to simulate a progression of change over three weeks. The reasoning behind this is that, if the algorithm can learn to develop a schedule which best fits the three weeks, it can then be used to predict a fourth week. As more data is collected, a system can continuously train the algorithm on the previous three weeks to make predictions for the current week. These three simulated weeks were then repeated three times to create 9 weeks worth of data to train the algorithm. This was done to increase the length of the training set. A longer training set increases the gap between the fitness of good and bad solutions. A poor solution with an incorrect cycle will score worse and worse as it is evaluated against more data.

To test the flexibility of the algorithm on other cycle lengths, similar Type I and Type II data were also constructed to simulate a three day cycle.

Finally, experiments were conducted on three consecutive weeks of real data. The data were formatted in a manner similar to the simulated Type II experiments where the three weeks of data were repeated three times to create a total of nine weeks of data.

VI. RESULTS

The results from the 7-day simulated Type I and Type II experiments show that the CGA is effective in learning a 7-day schedule of use. Figure 4 shows the fitness growth at selected generations averaged over five trials for the 7 day cycle. The fitness values shown are averages over the five trials. The scale for fitness is shown on the y-axis on the right of the figure. The average fitness increases dramatically in the first 100 generations, and continues to grow until about 8000 generations. Note that the x-axis scale on the graph changes after the first 1000 generations. The standard deviation between the five trials is shown in the error bars. The y-axis on the left side of the figure shows the scale for these bars.

In all five trials, the algorithm consistently determined the correct cycle length of 7 days for both the Type I and II tests. The use pattern is matched exactly in the Type I test, showing that the algorithm can learn a perfectly repetitive schedule. This occurred in all five trials by the 8,000th generation, as shown by the standard deviation bars. The deviation for the Type II tests also drops to zero by the 10,000th generation, showing that the algorithm consistently produced the same result in all five trials.

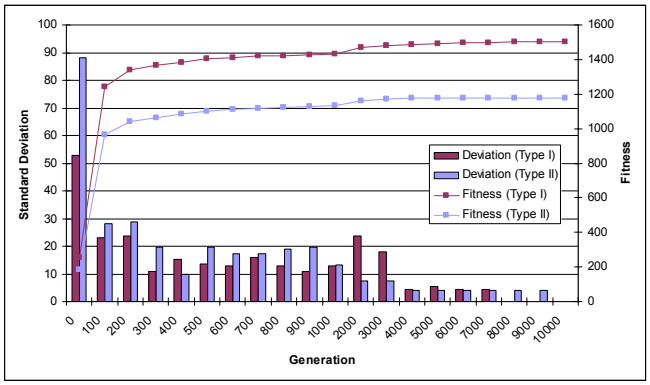

Figure 4. Fitness growth for simulated 7 day cycle

Similar to the test on the 7 day cycle data, the algorithm is able to consistently determine a 3 day schedule for both Type I and Type II tests. These results are shown in Figure 5. Note that the scale for the x-axis scale in this Figure is differs to that of Figure 4. The CGA runs significantly faster on the data for a 3 day cycle and thus learns the pattern in under 1000 generations. The greatest growth in fitness occurs in the first 200 generations, and thus the scale of the figure is adjusted to show this growth in more detail. The values for fitness shown in this graph are again an average of five trials.

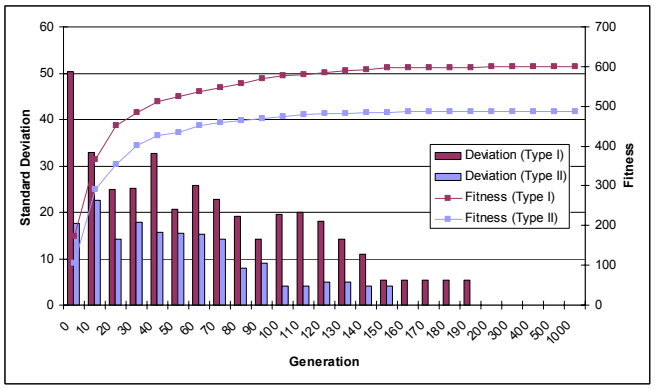

Figure 5. Fitness growth for simulated 3 day cycle

In all five trials, the algorithm consistently determined the correct cycle length of 3 days for both the Type I and II tests. Like the previous tests on the 7 day cycles, the use pattern is matched exactly in the Type I test in all five trials by the 160th generation. The deviation for the Type II tests also drops to zero by the 200th generation.

These positive results on both the 7 day and 3 day schedules shows that the algorithm has the ability to pick out the correct length of the cyclic pattern with no a priori knowledge.

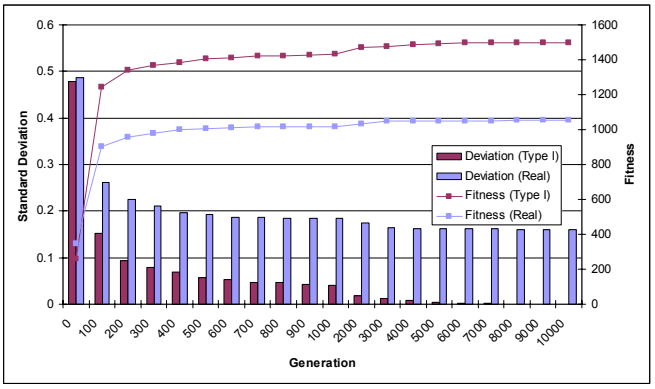

Figure 6. Fitness growth for real data

The experiments on real data show that the CGA can be effective in picking out the ideal schedule. Figure 6 shows the fitness growth across generations for real data averaged over 5 trials. Note again that the x-axis scale on the graph changes after the first 1000 generations. In three out of five trials, the algorithm correctly determined the cycle length of 7 days and minimized the error.

As the data here is three real weeks repeated, similar to Type II simulated data, the Type I data from Figure 4 is also included in Figure 6 as a point of comparison. Type I simulated data is one week of real data repeated and is thus also essentially real data.

The inconsistency in determining the schedule arises from the fact that real data has significantly more variation from week to week, making the perfect schedule somewhat ambiguous. More than one pattern of use could have been in play, which made even the cycle length in question. Nevertheless, the CGA produced seven day cycles that appeared to be reasonable fits for the actual data.

VII. Conclusions and Future Work

Our work employs a Cyclic Genetic Algorithm to pick out patterns in the occupancy of a room, and construct a schedule which can be used to make future predictions of its use. The results show that this algorithm performs extremely well on two different types of simulated data. It also has the ability to pick out cycles of various lengths and thus does not require any a priori knowledge of the length of the patterns in the data.

We also show that the algorithm performs well on data collected from a real environment. Although the accuracy of the determined schedule is highly dependent on the level of variation in the data, the algorithm does effectively discover a schedule which minimizes the error.

Future work will further analyze the variation in the real data in order to adapt this method and design an algorithm which can prioritize recent data over old. We also hope to incorporate this work into a larger control system as a basis for planning as well as real-time decision making with an aim to control variables in a real environment.

References

[1] D. Cook and S. Das, "How smart are our environments? An updated look at the state of the art," *Journal of Pervasive and Mobile Computing*, 3(2), 2007, pp. 53–73.

[2] G. M. Youngblood, D. J. Cook, and L. B. Holder, "Managing adaptive versatile environments," *Journal of Pervasive and Mobile Computing*, 1(4), 2005, pp. 373-403.

[3] S. Helal, B. Winkler, C. Lee, Y. Kaddourah, L. Ran, C. Giraldo, and W. Mann, "Enabling location-aware pervasive computing applications for the elderly," *Proc. 1st IEEE Pervasive Computing Conference*, Fort Worth, TX, 2003, pp. 531–538.

[4] H. Hagras, "Employing computational intelligence to generate more intelligent and energy efficient living spaces," *International Journal of Automation and Computing*, 5(1), 2008, pp. 1–9.

[5] M. Bailey and P. Curtiss, "Neural network modeling and control applications in building mechanical systems," *Proc. International Conference of Chartered Institution of Building Services Engineers and American Society of Heating Refrigeration and Air-conditioning Engineers*, London, England, 2001.

[6] M. C. Mozer, "The neural network house: An environment that adapts to its inhabitants," *Proc. American Association for Artificial Intelligence Spring Symposium on Intelligent Environments*, Menlo Park, CA, 1998, pp. 110–114.

[7] M. Galushka, D. Patterson, and N. Rooney, "Temporal Data Mining for Smart Homes," *Lecture Notes in Computer Science*, 4008, 2006, pp. 85-108.

[8] J. H. Holland, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, 1975.

[9] G. B. Parker, "Generating Arachnid Robot Gaits with Cyclic Genetic Algorithms," *Proc. Third Annual Genetic Programming Conference*, 1998.