Engineering Applications of Artificial Intelligence 82 (2019) 96–101

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Feedback controller design over the internet of things☆

John J. Zakelj, Kevin M. Passino *

Dept. Electrical and Computer Engineering, The Ohio State University, 2015 Neil Ave., Columbus, OH 43210, United States

ARTICLE INFO

Keywords:
Feedback control design
Internet of things
Nongradient optimization
Evolutionary algorithms
Adaptation

ABSTRACT

Suppose you have an inexpensive plant (process) that you have many copies of, and where each requires feedback control to achieve good closed-loop performance. There is then a feedback controller design challenge for each plant. The traditional approach to such a design is to use linear or nonlinear methods for one copy of the plant and implement the designed controller on each copy of the plant. The hope is that the original design is robust so it performs well on all copies. Here, we assume that the plants can be connected on the internet of things and introduce a novel strategy to design, in a distributed fashion, controllers for each plant where the design of the controller for each plant is informed by the success of the designs of all other controllers. We show that a type of robust controller can be designed over the internet of things. Our methods hold promise in practical commercial applications.

1. Introduction

The "internet of things" (IoT) is the interaction between devices to sense the environment, interpret the information, and react to real-world events autonomously with or without human intervention over a network (Vermesan and Friess, 2011). It promotes the idea that an increasing number of devices are being connected to the internet which allows more efficient monitoring and control of these devices (Anzelmo et al., 2011). Furthermore, smart systems, like the smart lights studied here, are some of the building blocks for the IoT (Kortuem and Kawsar, 2010). With mobile TCP/IP communication in hand, mobile devices such as those in control systems of automobiles, aircraft, and trains can be connected to the IoT to improve various aspects of travel (Ernst and Uehara, 2002). Automobiles can be connected over the IoT to receive updates to controllers such as ones for climate control or cruise control for optimal performance over all connected vehicles. Also, home appliances, such as refrigerators, ovens, and washing machines, will be able to communicate with each other to reach optimum performance in each home connected to the IoT.

There is a vast list of possible applications which will benefit from the IoT, and, with the growing number of devices connected, a more robust controller of these devices can be designed through a distributed non-gradient optimization method. The growing number of devices connected to the IoT translates to a growing number of controllers connected to the IoT. It is expected that the non-gradient algorithms provide near optimal design for large sets of controllers, such as those prevalent in the IoT. If this expectation is confirmed, it is hoped that the non-gradient algorithms tested here can be adapted to design many feedback control systems in the IoT more complex than the control system tested in this article.

Distributing control systems over a network allows for them to be monitored, accessed, and changed in real-time. In this article, a distributed non-gradient optimization algorithm is operated over a network to tune a large set of feedback controllers in a relatively large scale laboratory experiment of 40 lighting control systems through a centralized algorithm. A centralized algorithm is required to share and compare information from all controllers to determine the optimal design of the entire set of controllers and communicate the design to the other controllers for verification and improvements. A localized alternative would cause gaps in communicating optimal designs to other controllers. In the centralized algorithm tested in this article, it is shown that a "robust controller" emerges, one that performs well in all of the feedback control systems. By utilizing non-gradient optimization methods, this experimental approach to designing a controller removes the difficulties in modeling of complicated non-linear systems, and, thus, permits minimal system understanding to obtain a good solution with no need to consider manufacturing variances in system devices. Essentially, the information that is exploited in design is from a large set of operating control systems.

Due to the lighting control systems being non-linear and stochastic and including significant delay, an analytical gradient cannot be determined. Therefore, two standard non-gradient optimization methods, the genetic algorithm (GA) and the set-based stochastic (SBS) optimization method, are tested in this experiment. Based on Darwin's natural

☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.engappai.2019.03.018.

* Corresponding author.
E-mail address: passino.1@osu.edu (K.M. Passino).

https://doi.org/10.1016/j.engappai.2019.03.018
Received 25 April 2018; Received in revised form 8 November 2018; Accepted 19 March 2019
Available online xxxx
0952-1976/

selection theory, the GA is a non-gradient stochastic search method which allows multiple simultaneous search points to reach near a global optimum point. The SBS optimization method creates a set of search points centered about the best point of the previous generation to find a solution near the global optimum point (Passino, 2004). Although doubts about performance assurance surfaced in the past, these algorithms have now reached a stage of maturity to solve complex and conflicting problems which used to be considered "deadlocked" (Man et al., 1996). The GA is now being successfully implemented in a variety of engineering applications from the classic inverted pendulum problem to complicated VLSI circuit design problems (Lee and Takagi, 1993; Dasgupta, 1997). Both of these algorithms have been shown to be appropriate methods for feedback control system design, resulting in stable optimal designs (Fleming and Purshouse, 2002).

Smart lighting solutions have been an important topic in energy conservation with the potential of reducing energy costs by 50% in existing buildings (Rubinstein et al., 1993). Smart lighting has been successfully tested to reach desired light values through an illumination balancing algorithm (IBA) while being influenced by cross-illumination effects and ambient light sources (Koroglu and Passino, 2014). Furthermore, sensory nodes communicate illumination levels to the controller through a network to provide real-time changes over a large set of lighting systems. Due to the importance, experimental simplicity, and fast response times, smart light experiments provide an ideal system to test the use of distributed optimization algorithms for controller design over a network.

2. Experiment: Distributed networked light control systems

A proportional–integral (PI) controller sets the LED voltage, $V_{in} = u(t)$, in order to achieve the desired sensor voltage, $V_{out} = y(t)$, in Fig. 1. Unlike the IBA experiments (Koroglu and Passino, 2014), cross-illumination effects and ambient light sources are minimized by barriers between each light source. Each light controller, $j = 1, 2, \ldots, S$, has two gains, K_P^j and K_I^j, which are used to pick $u^j(t)$ via

$$u^j(t) = K_P^j e^j(t) + K_I^j \int_0^t e^j(\tau) d\tau \tag{1}$$

where

$$e^j(t) = r - y^j(t), \tag{2}$$

where e^j is the error input to the controller, r is the reference input, and y^j is the output of the plant. It is practical to choose a constant illumination reference for all controllers as was done in the IBA experiments (Koroglu and Passino, 2014). Also, via extensive experimentation, including system identification, we determined that the system is basically "type 0" so that adding an integrator and gain via PI control would result in zero steady state tracking error, and a fast response after tuning the proportional gain. In our adjustment algorithms below, we keep the PI gains positive so this is the case.

In order to effectively use the distributed optimization algorithm to improve (tune/design) the feedback gains, a large population of control systems are necessary for the best results as this will provide for the evaluation of many alternative designs. Therefore, a control network was created with eight light control systems on each of five computers to achieve $S = 40$ controllers reporting their performance calculations to the central computer as shown in Fig. 2 (it is generally difficult, in a university setting, to get many more control systems implemented simultaneously). Hence, it is the responsibility of each of the 5 computers to implement 5 feedback control systems, one for each light/sensor pair, and also perform control system performance evaluation as the feedback control system operates. Via TCP/IP another computer: (i) gathers performance information for the 5 computers, and (ii) computes the next design iteration. Then, it passes, via TCP/IP the new controller designs to the 5 computers. That is, using Matlab, the central computer runs the optimization algorithm based on the performance of each controller and returns a new set of gains to the population of feedback controllers through real-time Simulink models using transmission control protocol/internet protocol (TCP/IP) communication methods. Other IoT implementation are possible using this overall design approach; the focus here is not on the particular implementation details (as these will change with the introduction of new technologies), but on the experimental approach to control design over a network, something that is novel as a control system design methodology.

3. Closed-loop control system performance evaluation

A population of controllers

$$P(k) = \{\theta^j(k) : j = 1, 2, \ldots, S\} \tag{3}$$

where S is the number of controllers in the population and

$$\theta^j(k) = \left[K_P^j, K_I^j\right]. \tag{4}$$

At $t = 0$, each "step" (run of the control system) initializes the LED voltage at 0 V and sets the feedback control gains per Eq. (4). Each feedback control loop generates a response with a 1 ms sampling time. This sampling time is sufficient for the illumination experiment while allowing the computer to process the values in real-time. The control system runs for only 1 s because any light controller that reaches the desired light level after 1 s is not a practical illumination source. Once step k has completed, each local computer calculates the performance of its closed-loop system. The performance evaluation for each control system is

$$J^j(k) = w_s J_s^j(k) + w_{os} J_{os}^j(k) + w_{st} J_{st}^j(k) + w_{rt} J_{rt}^j(k) \tag{5}$$

which is a combination of steady state Eq. (6), overshoot Eq. (7), rise time Eq. (8), and settling time Eq. (9) values according to

$$J_s^j(k) = \left|\frac{s^j - r}{r}\right|, \qquad s^j = \frac{1}{L}\sum_{t=0.8t_f}^{t_f} y^j(t) \tag{6}$$

$$J_{os}^j(k) = \frac{\max_t\{y^j(t)\} - s^j}{s^j} \tag{7}$$

$$J_{rt}^j(k) = \frac{t_r^j}{t_f} \tag{8}$$

$$J_{st}^j(k) = \frac{t_s^j}{t_f} \tag{9}$$

Here, L is the number of steps in the last 20% of the signal $y^j(t)$, t_r^j is the 10%–90% rise time, t_s^j is the 2% settling time, and t_f is the total test time of 1 s. The performance evaluation parameters were designed to be a percentage of a total. Therefore, the weights of this function $(w_s, w_{os}, w_{st}, w_{rt})$ were each set to 100 to describe the percentage as can be seen later in the results. These weights may be adapted to place more emphasis on specific performance attributes, but this functionality in the performance calculation is not tested here.

4. Non-gradient optimization methods

4.1. Genetic algorithm optimization method

Following a standard genetic algorithm (GA) optimization approach, the performance value for each controller is gathered and used to form a new generation of controllers. Each controller has an opportunity to exchange its PI gain values to the next generation of controllers through a process in which controllers are paired based on

$$p^j(k) = 1 - \frac{J^j(k)}{\sum_{i=1}^{S} J^i(k)}.$$

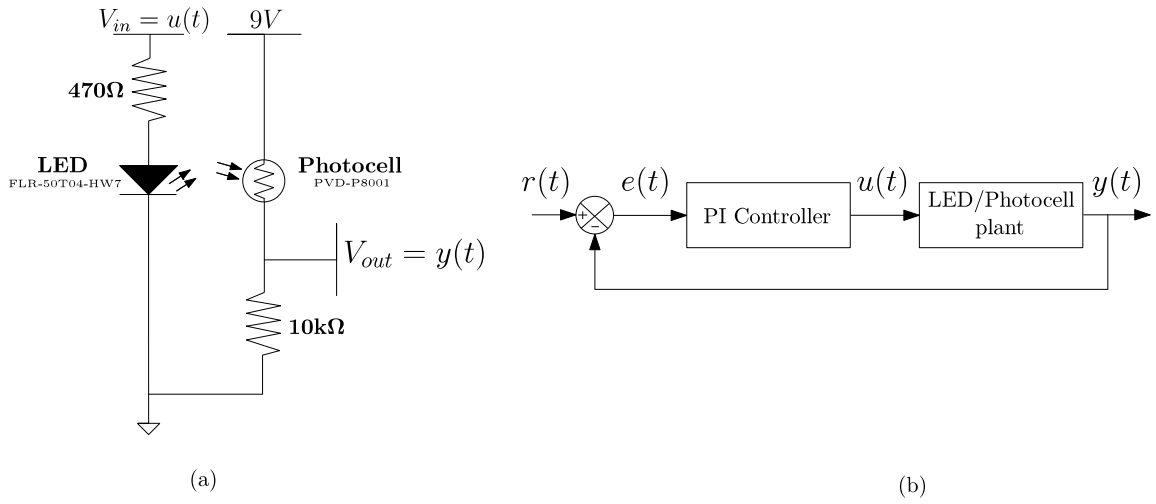

Fig. 1. (a) Light sensor circuit design of feedback control system (right side) and light driver (left side). The voltage V_{in} is applied to the LED by the controller and V_{out} is the voltage from the sensor that is proportional to the light level. (b) Feedback control of light level.

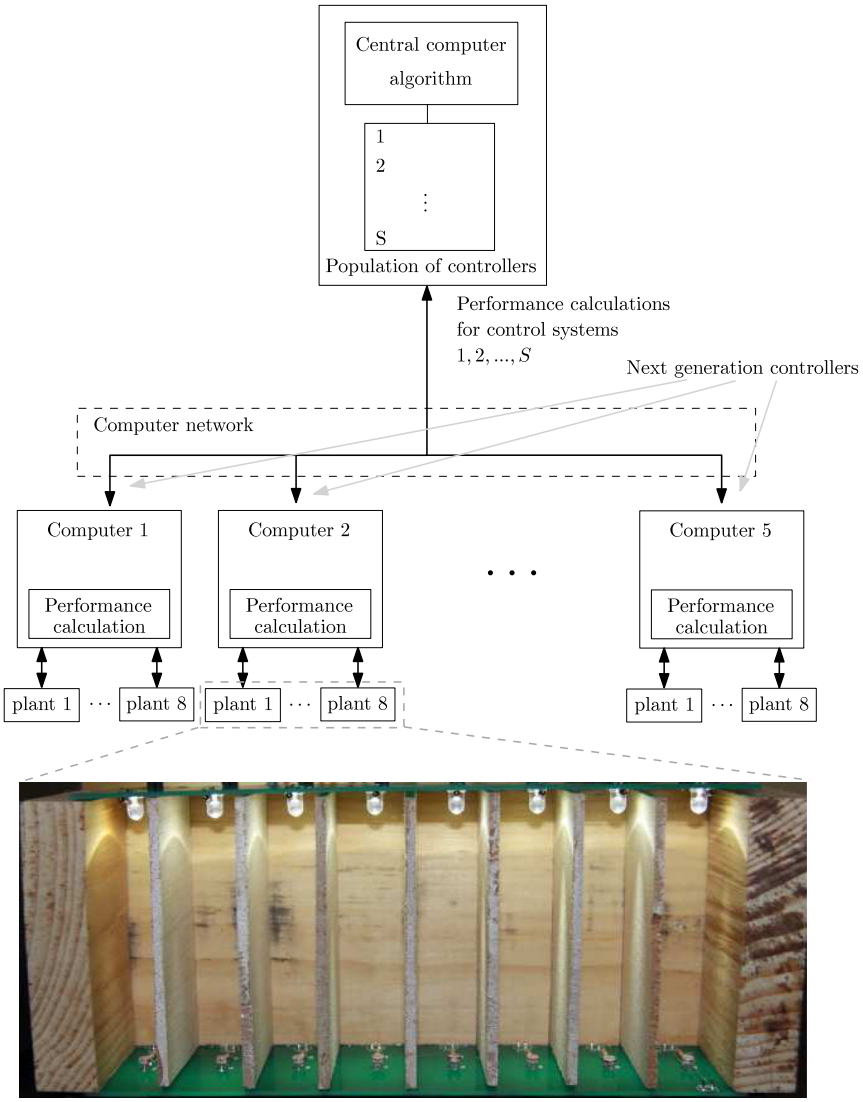

Fig. 2. Distributed feedback control systems over a network. Each set of plants has eight controllerly controlled light systems, which include eight LED–photocell pairs. A single computer's eight physical plants are displayed in the image. The central computer algorithm provides each plant with next generation controllers. The controllers are tested and return their performance calculations to the central algorithm.

Once two controllers have been paired, an exchange of gains may only occur at a crossover probability, p_c. Once a pair of controllers is selected and p_c allows for crossover to occur, the algorithm selects a single gain to be exchanged. Once a gain is selected, a specific digit of a gain is selected to be transferred from one controller to the other. All of the subsequent digits in that gain also transfer to the other controller to complete the process.

The example in Fig. 3 shows the transfer of digits of a gain between two controllers, $\theta^5(k) = [001.045, 010.012]$ and $\theta^9(k) = [005.984, 026.973]$. Notice that each gain always includes six digits with the decimal place between the third and fourth digit. The decimal location cannot be moved, and is therefore not considered in the exchange process. In this example, the crossover occurs at the fourth digit in the first gain to form the first controller of the next generation, $\theta^1(k+1)$. Random pairs of controllers exchange gains S times based on their $p^j(k)$ probabilities to form a new generation of S controllers.

Once the crossover stage is complete, a new generation of controllers, $P(k+1)$, is ready to be tested on the physical plants. However, in order to allow for a more complete search for an optimal solution, a stochastic selection stage is included in the GA optimization method. For some low probability, p_m, a controller's gains are randomly reselected to be tested in the next generation of controllers. This probability remains constant for all generations.

4.2. Set-based stochastic optimization method

The SBS optimization method also uses a population of controllers as described earlier in Eqs. (3) and (4); gathering the performance values of each controller defined in Eqs. (5)–(9) determines the next generation of controllers. However, this non-gradient algorithm selects the best designed controller, j^*, from the previous generation, via

$$j^* = \arg\min_j J^j(k)$$

and creates a "cloud" of new controllers around the best controller by selecting the next generation of controllers via

$$\theta^j(k+1) = \theta^{j^*}(k) + \beta r^j(k)$$

where $r^j(k)$ is a random number from a normal distribution with zero mean and unit variance and β is a scaling factor.

In this way, a cloud of controllers centered about the best controller is created as the next generation. However, since the best controller of the previous generation may not be the best solution for all controllers, a stochastic selection probability is again added to allow the algorithm to search randomly. Given the stochastic selection probability, p_m, a random controller from the previous generation is selected for the next generation by

$$\theta^j(k+1) = \theta^j(k), \qquad j \neq j^*$$

However, instead of using a constant stochastic selection probability as done in the GA optimization method, this method adapts p_m as the performance changes according to

$$p_m(k) = \frac{1}{10}\sum_{j=1}^{S} J^j(k)$$

Here, p_m is defined as 10% of the total performance of the previous generation. Due to the narrow search space of the SBS optimization method governed by β, a varying stochastic selection probability is necessary to move the search away from local minima in early generations but decrease the probability in later generations when the cost is converging on an optimal point for all control systems.

4.3. Time complexity

The time complexity of the approaches are low enough to: (i) be implemented on current technology, and (ii) will scale well for other problems. Note that to compute the controls, first we need to compute the individual PI controller outputs for each of the low-level control systems. To do this, we need to implement the integrator, which in the computer is simply a summing operation. Next, we have to multiply the PI gains by that sum, and the error. These operations, however, are not done for all the feedback loops on one computer, they are done on the individual computers that implement the controllers for each case in a standard distributed computing fashion. Next, the control system performance evaluation must be done to compute $J^j(k)$ for each $k \geq 0$. But, these calculations are simple as they involve standard low-demand operations (sums/differences, multiplies/divides, and a maximum), and most importantly these calculations are done *for each lower-level control system*, so that time-complexity is divided among the low-level control (computer) system implementations. Finally, we have to combine the results and use them in the genetic algorithm or non-gradient optimization method. First, note that we use a standard genetic algorithm; however, we do not iterate it between time steps. We force a time-step to correspond to a genetic algorithm optimization step. In this way, the time complexity is quite low, requiring standard genetic algorithm operations (survival of the fittest, cross-over, and mutation) only once per time step. Second, the case for the set-based stochastic optimization method has an even lower time complexity than the genetic algorithm; all that is need is computation of a maximum, and random number generation. Overall, the time complexity of this approach is low, making it scalable to application to other problems.

5. Experimental results

Since each system has unique circuit devices and construction, variances in individual control gain choices are expected to be found in this real-life system experiment. Therefore, the distributed optimization algorithms were tested on the light control systems repeatedly to represent a Monte Carlo method and to enable statistical analyses of the results.

Here, one set of generations is considered a test, and a set of tests is one experiment. A test number is denoted by n, and the total number of tests is N. Each controller initiates the test with a random pair of initial gain values on the set [0, 100]. Once a test completes the fixed amount of generations, the performance and gains are stored for future analyses. As described in Eq. (10), matrix M_J^n stores the performance values for each controller in each generation. When the experiment is complete, each test matrix is averaged via Eq. (11) to acquire statistical data for each generation. Similarly, the gain matrices are created and averaged to form $\bar{K}_P$ and $\bar{K}_I$ in Eqs. (12)–(15),

$$M_J^n = \left[J^j(1), J^j(2), \ldots, J^j(k)\right] \qquad j = 1, 2, \ldots, S \tag{10}$$

$$\bar{J} = \frac{1}{N}\sum_{n=1}^{N} M_J^n \tag{11}$$

$$M_P^n = \left[K_P^j(1), K_P^j(2), \ldots, K_P^j(k)\right] \qquad j = 1, 2, \ldots, S \tag{12}$$

$$\bar{K}_P = \frac{1}{N}\sum_{n=1}^{N} M_P^n \tag{13}$$

$$M_I^n = \left[K_I^j(1), K_I^j(2), \ldots, K_I^j(k)\right] \qquad j = 1, 2, \ldots, S \tag{14}$$

$$\bar{K}_I = \frac{1}{N}\sum_{n=1}^{N} M_I^n \tag{15}$$

$\theta^5(k)$	0	0	1	0	4	5	0	1	0	0	1	2
				↓	↓	↓						
$\theta^9(k)$	0	0	5	9	8	4	0	2	6	9	7	3
$\theta^1(k+1)$	0	0	5	**0**	**4**	**5**	0	2	6	9	7	3

Fig. 3. GA crossover example. The fourth digit is randomly selected for crossover along with all subsequent digits to produce $\theta^1(k+1)$ for the next generation of controllers.

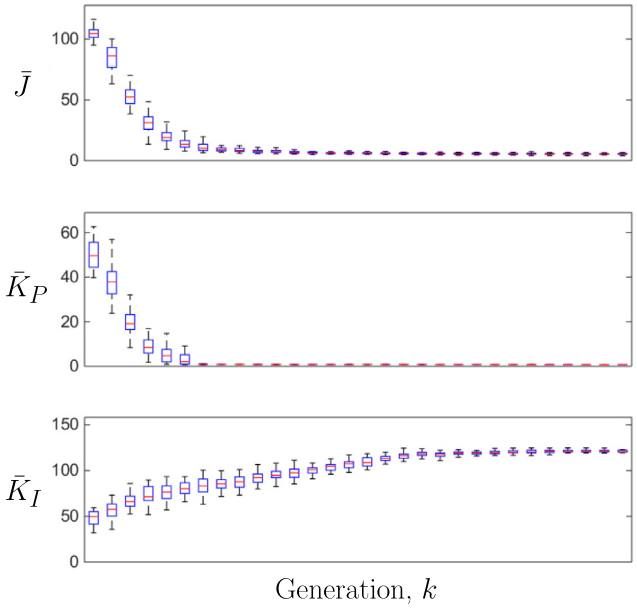

Fig. 4. Performance and gain statistics of the GA with $p_c = 80\%$ and $p_m = 0.5\%$. This box plot contains the interquartile range from the first to the third quartile, the centerline of the box indicating the second quartile (median), the extended line representing the data set without the outliers, and the omission of any outliers for clarity.

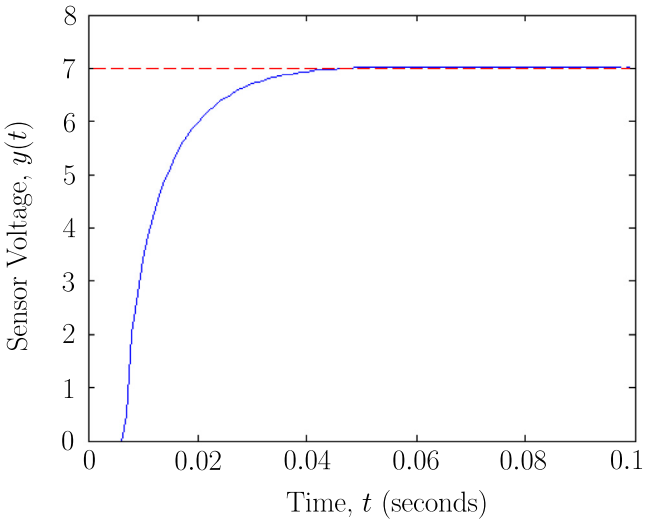

Fig. 5. Response of a single control system with $K_P = 0.767$ and $K_I = 120.062$. Response parameters: $J_{rt} = 0.018$ s, $J_{st} = 0.071$ s, $J_{os} = 4.922 \times 10^{-4}$, $J_{ss} = 4.842 \times 10^{-4}$, resulting in a total $J = 8.998$.

The plots in Fig. 4 show the result of a GA optimization method experiment, which consisted of 25 tests with 30 generations in each test. Fig. 4 shows the average performance matrix, $\bar{J}$, as well as the convergence of the gain matrices, $\bar{K}_P$ and $\bar{K}_I$. The algorithm converges to a high K_I gain and a low K_P gain for each controller. The performance converges to a low cost of less than 10% combined error, which mostly consists of a 18 ms rise time and a 71 ms settling time. Given that the photocell device specifications state a typical total rise time of 55 ms, the final performance of the population is near optimum based on the response of the system of a sample case shown in Fig. 5. Furthermore, the convergence to a thin interquartile range in Fig. 4 shows that the experiment was repeatable over the 25 tests.

Using the same physical plant in Eqs. (1) and (2), population in Eqs. (3) and (4), performance function in Eqs. (5)–(9), and initial conditions as the GA optimization method experiment, a comparison can be made with the SBS optimization method. The plots in Fig. 6 represent the statistics of an experiment with 25 tests and 30 generations. Again, the experiment shows convergence to a low average $\bar{J}$ as well as convergence to a desired set of gains. Also, the thin interquartile range for the performance and gains displays repeatability in all 25 tests. In comparison to Figs. 4 and 6, similar but not identical results are obtained.

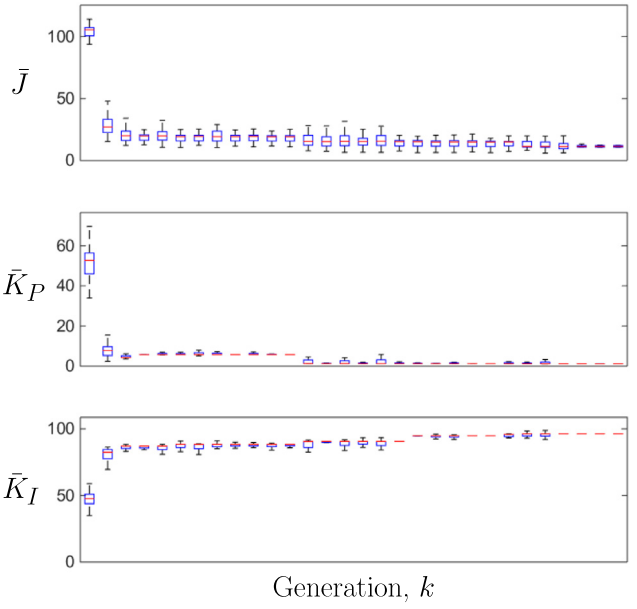

Fig. 6. Performance and gain statistics for the SBS optimization method with $\beta = 0.01$. This box plot contains the interquartile range from the first to the third quartile, the centerline of the box indicating the second quartile (median), the extended line representing the data set without the outliers, and the omission of any outliers for clarity.

6. Conclusion

Utilizing non-gradient optimization methods on light control systems over a distributed network, experiments were implemented on many feedback control systems and have shown convergence and robustness. It is important to understand that this distributed method using non-gradient algorithms does not guarantee robustness or optimal performance for more complex systems. This is a first step in distributed non-gradient control over the IoT. However, in this test of the non-gradient algorithm, both non-gradient optimization methods selected desirable gains for the light control system even while inherent with noise, delays, and manufacturing differences. Moreover, *all* feedback control systems in the network had desirable responses, which shows a type of robustness. Finally, these non-gradient optimization methods allow control engineers to retrieve robust control designs without the need to model the system.

The overall practical significance of this design method is that it sets a robust initial system while also being capable of maintaining robustness as the devices are sent out to the real world through continuous monitoring and adaption over the IoT. The ability to access, monitor, and change dispersed device controller designs in real time separates this design method from other conventional methods which design controllers then disperse the controllers without further design communication. It is hopeful that the distributed non-gradient optimization methods discussed in this paper will be tested further and eventually used to improve the robustness of IoT applications through this centralized non-gradient design method.

References

Anzelmo, E., Bassi, A., Caprio, D., Kranenburg, R., 2011. Internet of things. In: 1st Berlin Symposium on Internet and Society (Version electronica), Consultado el, Vol. 20.

Dasgupta, D., 1997. Evolutionary Algorithms in Engineering Applications. Springer, Berlin.

Ernst, T., Uehara, K., 2002. Connecting automobiles to the internet. In: ITST: International Workshop on ITS Telecommunications, Vol. 3.

Fleming, P., Purshouse, R., 2002. Evolutionary algorithms in control systems engineering: A survey. Control Eng. Pract. 10 (11).

Koroglu, M.T., Passino, K.M., 2014. Illumination balancing algorithm for smart lights. IEEE Trans. Control Syst. Technol. 22 (2), 557–567.

Kortuem, G., Kawsar, F., 2010. Smart objects as the building blocks for the internet of things. IEEE Internet Comput. 14 (1), 30–37.

Lee, M., Takagi, H., 1993. Intergrating design stage of fuzzy systems using genetic algorithms. In: Second IEEE Int. Conf. on Fuzzy Systems, Vol. 1. pp. 612–617.

Man, K.F., Tang, K.S., Kwong, S., 1996. Genetic algorithms: Concepts and applications. IEEE Trans. Ind. Electron. 43 (5), 519–534.

Passino, K.M., 2004. Biomimicry for Optimization, Control, and Automation. Springer, London.

Rubinstein, F., Siminovitch, M., Verderber, R., 1993. Fifty percent energy savings with automatic lighting controls. IEEE Trans. Ind. Appl. 29 (4), 768–773.

Vermesan, O., Friess, P., 2011. Internet of Things: Global Technological and Societal Trends. River Publishers, Aalborg.

John J. Zakelj received a B.S. in Electrical and Computer Engineering from The Ohio State University in May of 2013. He is currently a graduate student in the Electrical and Computer Engineering department at The Ohio State University and is expected to graduate May of 2015 with a Master's degree in Electrical and Computer Engineering.

Kevin M. Passino (S'79, M'90, SM'96, Fellow 2004) received his Ph.D. in Electrical Engineering from the University of Notre Dame in 1989. He is currently a Professor of Electrical and Computer Engineering and the Director of the Humanitarian Engineering Center at The Ohio State University. He has served as the Vice President of Technical Activities of the IEEE Control Systems Society (CSS); was an elected member of the IEEE Control Systems Society Board of Governors; was the Program Chair of the 2001 IEEE Conf. on Decision and Control; and is currently a Distinguished Lecturer for the IEEE Society on Social Implications of Technology. He is a Fellow of the IEEE. He is co-editor (with P.J. Antsaklis) of the book "An Introduction to Intelligent and Autonomous Control," Kluwer Academic Press, 1993; co-author (with S. Yurkovich) of the book "Fuzzy Control," Addison Wesley Longman Pub., 1998; co-author (with K.L. Burgess) of the book "Stability Analysis of Discrete Event Systems," John Wiley and Sons, NY, 1998; co-author (with V. Gazi, M.L. Moore, W. Shackleford, F. Proctor, and J.S. Albus) of the book "The RCS Handbook: Tools for Real Time Control Systems Software Development", John Wiley and Sons, NY, 2001; co-author (with J.T. Spooner, M. Maggiore, R. Ordonez) of the book "Stable Adaptive Control and Estimation for Nonlinear Systems: Neural and Fuzzy Approximator Techniques," John Wiley and Sons, NY, 2002; author of "Biomimicry for Optimization, Control, and Automation," Springer-Verlag, London, UK, 2005; and co-author (with V. Gazi) of the book "Swarm Stability and Optimization", Springer-Verlag, Heidelberg, Germany, 2011; author of "Humanitarian Engineering: Creating Technologies that Help People," Edition 1, Bede Pub., OH, 2014. For more information, see: http://www.ece.osu.edu/~passino/.