Parameter Self-Configuration and Self-Adaptation in Industrial Wireless Sensor-Actuator Networks

Junyang Shi, Mo Sha
Department of Computer Science
State University of New York at Binghamton
{jshi28, msha}@binghamton.edu

***Abstract*—Wireless Sensor-Actuator Network (WSAN) technology is gaining rapid adoption in process industries in recent years. A WSAN typically connects sensors, actuators, and controllers in industrial facilities, such as steel mills, oil refineries, chemical plants, and infrastructures implementing complex monitoring and control processes. IEEE 802.15.4 based WSANs operate at low-power and can be manufactured inexpensively, which make them ideal where battery lifetime and costs are important. Recent studies have shown that the selection of network parameters has a significant effect on the network performance. However, the current practice of parameter selection is largely based on experience and rules of thumb involving a coarse-grained analysis of expected network load and dynamics or measurements during a few field trials, resulting in non-optimal decisions in many cases. In this work, we develop the *Parameter Selection and Adaptation FramEwork (P-SAFE)* that optimally configures the network parameters based on the application Quality of Service (QoS) demand and adapts the configuration at runtime to consistently satisfy the dynamic requirements. We implement P-SAFE and evaluate it on three physical testbeds. Experimental results show our solution can significantly better meet the application QoS demand compared to the state of the art.**

***Index Terms*—Industrial Wireless Sensor-Actuator Networks, Parameter Adaptation, Parameter Selection**

I. Introduction

Wireless Sensor-Actuator Network (WSAN) technology is gaining rapid adoption in process industries in recent years. A WSAN typically connects sensors, actuators, and controllers in industrial facilities, such as steel mills, oil refineries, chemical plants, and infrastructures implementing complex monitoring and control processes. IEEE 802.15.4 based WSANs operate at low-power and can be manufactured inexpensively, which make them ideal where battery lifetime and costs are important. Battery-powered wireless modules easily and inexpensively retrofit existing sensors and actuators in industrial facilities without running cabling for communication and power. The stringent *reliability* and *real-time* requirements of industrial process applications distinguish industrial WSANs from traditional Wireless Sensor Networks (WSNs) designed for best effort services. To meet the stringent requirements, industrial WSAN standards such as WirelessHART [1] made a set of specific design choices. For instance, WirelessHART adopts a centralized network architecture and employs the Time Slotted Channel Hopping (TSCH) technology [2]: Time is divided into time slots, which are long enough for packet transmission and its acknowledgement; All devices in a network are time synchronized and hop channels to exploit frequency diversity; A TSCH schedule determines each device what to do in each time slot: transmit, receive, or sleep. Compared to the Carrier-Sense Multiple Access with Collision Avoidance (CSMA/CA), the Time-Division Multiple Access (TDMA) based TSCH offers time-deterministic packet deliveries, which makes it attractive for real-time communication. With a decade of real-world deployments, industrial standards have demonstrated the feasibility for the TSCH based WSANs to achieve reliable low-power wireless communication in industrial environments. Therefore, the TSCH technology was adopted by the leading industrial WSAN standards (WirelessHART [1] and ISA100 [3]) and the one being standardized by IETF (6TiSCH [4]) and amended into the IEEE 802.15.4 standard in 2012 [5].

Recent studies have shown that the selection of network parameters such as the Packet Reception Ratio (PRR) threshold for link selection, the number of channels used in the network, and the number of transmission attempts for each packet has a significant effect on the performance of industrial WSANs [6], [7]. However, the current practice of parameter selection is largely based on experience and rules of thumb involving a coarse-grained analysis of expected network load and dynamics or measurements during a few field trials. For instance, WirelessHART has specified the use of all available channels after the human network operator manually blacklists noisy ones [1] and the Emerson Process Management, a leading process automation supplier, suggests using a constant value (i.e., 60%) as the PRR threshold to select links for routing [8]. Unfortunately, a recent study shows that these specifications are error prone [6]. For example, using more channels is not always desirable in industrial WSANs, since more channels mean more channel diversity but a large number of channels may reduce route diversity with negative effects on routing and scheduling.

Thanks to some recent work [9]–[12], we are confident we are just seeing the tip of the iceberg in terms of how much performance can be improved through enabling parameter adaptation. However, to fully realize the benefits offered by the parameter adaptation, two fundamental challenges must be overcome: (i) *Conceptual gap*: There exists a large conceptual gap between the high-level application Quality of Service (QoS) requirements and the low-level network parameters. It requires expert knowledge to find the parameters whose performance satisfies given requirements. Although most network

parameters have been studied individually in the context of WSNs, there still exist phenomena that are unknown under an industrial WSAN setting. For example, a recent study shows that the performance of WSANs does not improve monotonically with more channels used because of the tradeoff between channel diversity and route diversity [6]. Owing to the lack of understanding of the underlying functional form of the relationships between high-level requirements and low-level parameters, the selection of suitable parameters becomes challenging. (ii) *Complex QoS demand*: Most industrial process applications today pose multiple (sometimes conflicting) QoS requirements on information exchange to their underlying networks. Learning the QoS demand of process applications that truly reflects their needs is particularly challenging, as multiple requirements must be met and tradeoffs have to be made among conflicting ones. The traditional solutions, which require users to order their QoS requirements or rely on a coarse-grained weighted sum calculation, always result in non-optimal decisions in practice.

To address the above-stated challenges, we develop the *Parameter Selection and Adaptation FramEwork (P-SAFE)* that optimally configures the network parameters based on the application QoS demand and adapts the configuration at runtime to consistently satisfy the dynamic requirements. Specifically, this paper makes the following contributions:

- We design a rigorous modeling method that relates the high-level application QoS requirements to the low-level network parameters;
- We formulate the parameter selection into a multi-objective optimization problem and employs the NSGA-II algorithm to identify the best tradeoff decisions.
- We develop a novel approach that learns the QoS preferences from the control application based on its specified ranges of desirability and uses the linear physical programming technique to identify the single (most attractive) best tradeoff decision.
- We implement P-SAFE and evaluate it on three physical testbeds. Experimental results show our solution can significantly better meet the application QoS demand compared to the state of the art.

The remainder of the paper is organized as follows. Section II reviews the related work. Section III introduces the design of our P-SAFE and Section IV presents our QoS learning approach. Section V evaluates P-SAFE and Section VI concludes the paper.

II. Related Works

Recently, there has been significant research on real-time industrial WSANs spanning transmission scheduling, routing algorithms, and network protocols. For instance, Watteyne et al. presented the implementations of IP-based real-time communication over TSCH [13].Saifullah et al. presented a schedulability analysis under graph routing in WirelessHART Networks [14] and Gunatilaka et al. proposed a channel selection approach [6]. Wu et al. and Shi et al. developed real-time routing protocols [15]–[17]. Lu et al. provided a comprehensive survey of recent advances in this increasingly important class of wireless networks [18]. Yet, a key missing piece in industrial WSANs is a self-adaptation component, which allows WSANs to optimally configure themselves based on specific QoS requirements and adapt the configurations at runtime to consistently satisfy the dynamic requirements in uncertain environments. This paper aims to accomplish this and advance the state of the art of real-time industrial WSANs through creating a new paradigm of parameter adaptation.

The characteristics of 802.15.4 wireless links have been studied extensively in the context of WSNs. For instance, there has been a vast array of research that empirically studied the link quality with different platforms, under varying experimental conditions, assumptions, and scenarios [19]. There also has been extensive research investigating the benefit of multi-channel communication in WSNs and mesh networks. The extensive studies produced valuable guidelines on selecting parameters but also caused a perception that those parameters can be selected manually during the deployment based on experience and rules of thumb involving a coarse-grained analysis of expected network load and dynamics or measurements during a few field trials. As a result, WirelessHART has specified the use of all available channels after the human network operator manually blacklists noisy ones [1] and Emerson Process Management, a leading process automation supplier, has specified to use 60% as a threshold to select links [8]. Unfortunately, a recent study shows that these specifications are error prone [6]. Thanks to some recent works, we are confident we are just seeing the tip of the iceberg in terms of how much performance can be improved through enabling runtime parameter adaptation. Zimmerling et al. developed the pTunes framework that reduces the packet loss when facing network changes through enabling adaptation of radio on and off timings and demonstrated its performance through applying it to X-MAC and LPP protocols [9]. Peng et al. [10] and Wang et al. [11] developed methods to reduce energy consumption by adapting the sleep intervals in duty-cycled MACs. Fu et al. highlighted the challenges of adapting multiple parameters simultaneously because of their joint effect on performance [12]. Although these works have some fundamental limitations, such as only adapting deployment-independent parameters, requiring a precise knowledge of their effect on performance, and optimizing towards a single requirement, they have shed light on the promising opportunity for constructing a self-adaptive network. However, there is hardly any precedent for a rigorous scientific method to model the effect of deployment-dependent parameters and generate a robust set of strategies to support network parameter decisions. This motivates our work to enable the parameter adaptation in WSANs.

III. P-SAFE Design

Figure 1 shows the design of our P-SAFE. After the engineers deploy a WSAN in the field, the **Network Analysis Engine** in P-SAFE guides them to implement the deployed network and target control application in it and also feed in the collected link (PRR) traces. The engine then simulates network

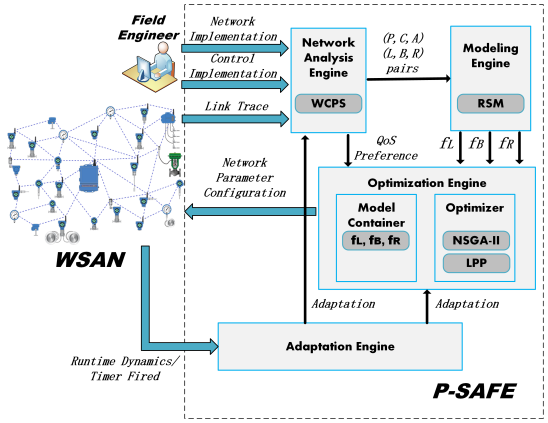


Fig. 1. System overview.

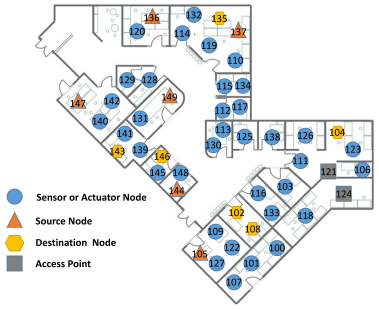


Fig. 2. Network setting on the BU Testbed used in Section III.A (Illustration and Example) and Section V.A.

TABLE I
DATA FLOWS SET ON THE BU TESTBED USED IN SECTION III.A (ILLUSTRATION AND EXAMPLE) AND SECTION V.A.

Flow ID	Source	Destination	Period (ms)	Priority
1	147	146	800	1
2	144	143	800	2
3	105	104	800	3
4	149	102	800	4
5	136	135	1600	5
6	137	108	1600	6

performance under each parameter configuration and forwards the results to the **Modeling Engine**. The Modeling Engine generates the empirical models that relate the parameter configurations to the performance of WSANs. The **Optimization Engine** stores the empirical models and selects the best-suited network parameters based on the QoS preferences learned from the control application. The **Adaptation Engine** allows the Network Manager (a software module specified by WirelessHART to manage the network) and control application to update the network setting, link traces, and QoS preferences at runtime and adapts the parameters accordingly.

A. Network Analysis Engine

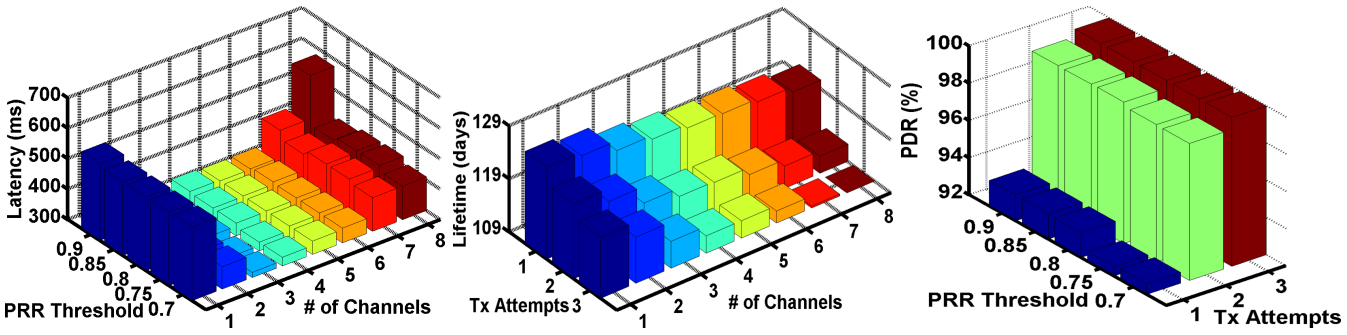
The design goal of our Network Analysis Engine is to analyze the network performance under each parameter configuration. It is impractical to perform test runs on the physical WSAN due to the significant overhead. Fortunately, the state-of-the-art wireless control simulators such as WCPS [20] and Truetime [21] are capable of holistic studies of CPU scheduling, communication, and control algorithms [18]. Our Network Analysis Engine adopts WCPS which employs a federated architecture that integrates Simulink for simulating the physical system dynamics and controllers and TOSSIM [22] for simulating WSANs. Control engineers commonly use Simulink to design and study control systems, while TOSSIM has been widely used in the WSN community to simulate WSANs based on wireless link models that have been validated in diverse real-world environments [23]. WCPS also provides a WirelessHART implementation in TOSSIM.

After the field engineers deploy a physical WSAN, our Network Analysis Engine guides them to (i) implement the deployed WSAN in TOSSIM (e.g., specifying the data sources and destinations, sampling rates, routing and scheduling algorithms), (ii) feed the link traces collected from the deployed WSAN into TOSSIM, and (iii) implement the target control application in Simulink. The Network Analysis Engine then performs simulations under each parameter configuration. Three key network parameters, identified in the recent study [6], including (i) PRR threshold for link selection P, (ii) number of channels used in the network C, and (iii) number of transmission attempts scheduled for each packet A are considered simultaneously in the simulations. Assuming the pool of candidate parameters contains n_P for P, n_C for C, n_A for A, our Network Analysis Engine measures the network performance including (1) end-to-end latency L, (2) battery lifetime B, and (3) end-to-end reliability R, under all $n_P \times n_C \times n_A$ combinations among those three parameters (i.e., P, C, and A). The simulated performance together with its associated parameter configurations are forwarded to the Modeling Engine. We will next use an example to illustrate the process.

Illustration and Example:

In the example, we configure six data flows on the BU Testbed consisting of 50 TelosB motes placed throughout several office areas including student offices, lounge, labs, and conference rooms [24]. Figure 2 shows the deployment and Table I lists the period and priority of each data flow. The packet delivery deadline is equal to the period. The graph routing and priority scheduling are employed in the simulations. We consider $P \in \{0.7, 0.75, 0.8, 0.85, 0.9\}$, $C \in \{1, 2, 3, 4, 5, 6, 7, 8\}$, and $A \in \{1, 2, 3\}$. The performance is simulated under 120 different parameter configurations.

Since it is not feasible to show the data in a four dimension view, we fix a dimension and present three 3-D plots in Figure 3. Figure 3(a) shows the end-to-end latency under different PRR threshold P and number of channels used C combinations. The results confirm the observation reported in the early study that more number of channels used cannot always provide lower latency because of the tradeoff between route diversity and channel diversity [6]. We also observe that the effect of PRR threshold on latency gets stronger when more channels are used in the network. Figure 3(b) shows that the battery lifetime decreases when either number of channels used C or number of attempts per packet A increases. The significance of effects from two parameters is different. Figure 3(c) shows the end-to-end reliability in



(a) Average latency with three attempts per packet. (b) Average battery lifetime with a 85% PRR threshold. (c) Average PDRs with 5 channels used.

Fig. 3. 3D-plot of network performance with different parameter configurations.

term of Packet Delivery Rate (PDR) that increases slightly with PRR threshold P and significantly with number of attempts per packet A. While Figure 3 shows some interesting observations helping us understand the joint effect of those three parameters, it also highlights an important challenge posed by the interplay among them, which will be addressed by our Modeling Engine.

B. Modeling Engine

The design goal of our Modeling Engine is to generate the empirical models that relate the parameter configurations to the performance of WSANs. It is a significant challenge to empirically model the joint effect of the three interplaying parameters without an understanding of the underlying functional form of the relationships being modeled. Modeling the relations theoretically is not efficient since the models can be deployment-dependent (e.g., depending on the particular network topology, setting, and protocols). Therefore, our Modeling Engine takes a black-box modeling approach and adopts the widely used Response Surface Methodology (RSM) [25] to construct the models. RSM is a black-box modeling technique and uses polynomial functions to approximate the model functions between the independent variables (inputs) and the response (outputs) without comprehending the underlying physical meaning between inputs and outputs, thus it provides a tractable and inexpensive approximation of the actual system behavior using polynomial functions.

Our Modeling Engine takes a tuple of performance metrics (L, B, R) and corresponding parameters (P, C, A) to construct three performance functions:

$$\begin{aligned} L &= f_L(P, C, A) + \varepsilon_1 \\ B &= f_B(P, C, A) + \varepsilon_2 \\ R &= f_R(P, C, A) + \varepsilon_3 \end{aligned} \tag{1}$$

where ε_i is a random experimental error assumed to have a zero mean. It is important to note that our Modeling Engine allows a P-SAFE user to replace the default RSM with another modeling technique. As an example, we replace RSM with a Kriging surrogate modeling approach [26] in the following example and show the models constructed by RSM and Kriging, respectively. Kriging is a type of spatial interpolation that uses complex mathematical formulas to estimate values at unknown points based on the values which are already

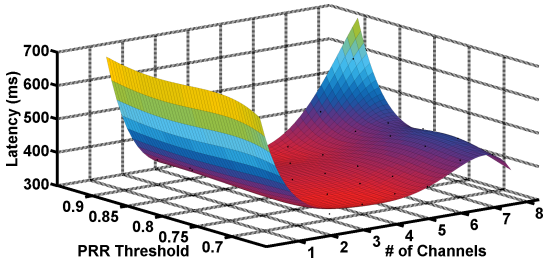


Fig. 4. The surface plot of latency when applying RSM. The number of transmission attempts per packet is set to 3.

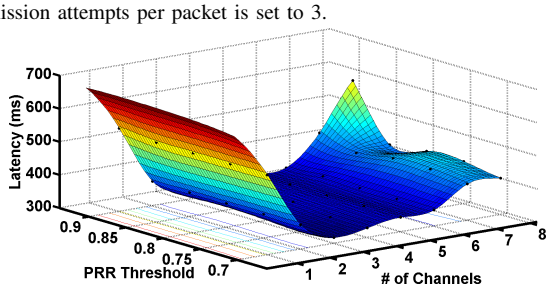


Fig. 5. The surface plot of latency when applying Kriging. The number of transmission attempts per packet is set to 3.

sampled. The estimation of the value is denoted as Z_0 and the observed values are $\{Z_1, ..., Z_N\} = Z^T$, so the estimated value can be expressed as:

$$Z_0 = \sum_{i=1}^{N} w_i Z_i \tag{2}$$

where w_i denotes the influence weight. Kriging uses the minimum variance method to calculate the weights w_i.

Illustration and Example (continued):

We use the implementations provided by Design Expert10 [27] and Matlab SUMO toolbox [28] for RSM and Kriging modeling, respectively. Figure 4 and Figure 5 show the surface plots of latency (Eq. 1) when applying RSM and Kriging on the data trace plotted in Figure 3(a). The generation of Kriging models consumes much more time compared to RSM but introduces no error between the resulting mathematical functions and samples. For example, we run the modeling process on a Dell Linux laptop with the 2.8GHz Intel Core E3-1505M with 40 samples plotted in Figure 3(a). The modeling time when applying RSM and Kriging is 160ms and 4.27s, respectively. The average modeling errors under RSM and Kriging are 3.19% and 0%.

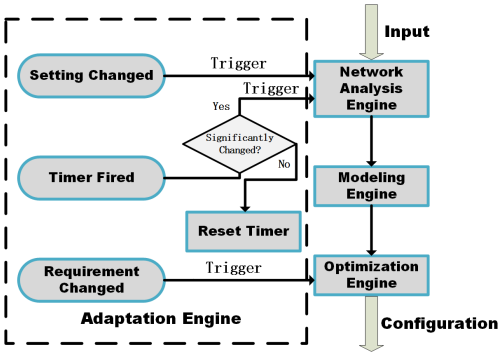


Fig. 6. Adaptation Engine's responses to different changes.

With the open design, a P-SAFE user is free to choose any modeling technique based on the need of target application. Please note that the modeling overfitting may happen, which motivates us to design the Adaptation Engine (see Section III-D) to overcome the modeling inaccuracy at runtime.

C. Optimization Engine

After obtaining the empirical models in Section III-B, the next step is to generate a novel set of decision-making strategies to select the best-suited network parameters based on the given QoS requirements specified by the control application. Specifically, for each given (L, B, R), the parameters can be obtained by solving an optimization problem based on Eq. 1. The challenge is that most industrial process applications today pose multiple (sometimes even conflicting) QoS requirements on information exchange to their underlying networks. The traditional solutions, which require network users (e.g., a control engineer) to order their QoS requirements or rely on a coarse-grained weighted sum calculation, simplify the problem but results in non-optimal decisions in many cases. To address this problem, we develop a novel approach that learns the QoS demand of a given process application that truly reflects its needs and simultaneously applies them to the parameter selection process. The detailed design will be presented in Section IV.

D. Adaptation Engine

The Adaptation Engine allows the Network Manager and control application to update the network setting, link traces, and QoS preferences at runtime and adapts the parameters accordingly. Since changing network parameter introduces significant communication and computation overhead, our engine employs a hybrid approach that combines event-driven and time-driven adaptations and responds differently when facing different changes. Figure 6 shows the actions of our Adaptation Engine in response to different kinds of changes. For example, the engine invokes the Network Analysis Engine and Modeling Engine to remodel the network and then reperform the optimization if the network setting (e.g., data sources and destinations) changes. It skips the modeling process and reruns the optimization if the application requirements change but the network setting stays the same. The modeling and optimization processes are invoked to examine the network by a timerif no event-driven adaptation is triggered during a long period. If the new optimized parameter configuration is not significantly better than the current one (i.e., smaller than a threshold), it is retained; else the network switches to a new configuration.

IV. QoS Learning Approach

As discussed in Section III-C, most industrial process applications today pose multiple (sometimes even conflicting) QoS requirements on information exchange to their underlying networks. Learning the QoS demand of process applications that truly reflects their needs is particularly challenging, as multiple requirements must be met and tradeoffs have to be made among conflicting ones. The traditional weighted sum approach merging multiple QoS requirements into a single objective function suffers from three significant limitations when applied in industrial WSANs:

- It is difficult to specify good weights that truly reflect the QoS preferences of a process application;
- A change in the QoS preferences of a process application does not readily translate into a change in specified weights;
- When a process application has non-linear QoS requirements, if the Pareto frontier [29] (i.e., the collection of non-dominated or best tradeoff solutions) is non-convex and/or disjointed, it can fail to obtain the best tradeoff solutions, and even higher order weighted combinations (i.e., $\sum w_i f_i^n$, where n is an even number such as 2, 4, ...) could face difficulty in leading to the Pareto frontier [30].

The difficulty is pervasive in the context of conflicting QoS requirements, where blindly optimizing a weighted aggregate of the multiple QoS requirements provides limited to no information regarding the tradeoffs that exist among them. For example, is it beneficial to improve the reliability by A% in exchange of degradations of B% latency and C% energy consumption?

To avoid using the rules of thumb QoS orders or weights, we formulate our optimization problem into a multi-objective optimization problem and applies a widely used evolutionary algorithm to identify the best tradeoff solutions. We then develop an approach using the physical programming technique[1] to identify the most attractive tradeoff decision. We also provide the entire set of best tradeoff solutions to the P-SAFE users in case they want to see all available choices and tradeoffs.

A. Problem Formulation and Optimization

The objective of selecting the best parameter configuration is to (i) minimize the end-to-end latency L, (ii) maximize the

[1]The physical programming technique was developed in the area of multidisciplinary design optimization to address engineering design problems such as aircraft and automobile design. It provides a powerful methodical approach to obtain the most attractive best tradeoff decision from the set of best tradeoff solutions [31], [32].

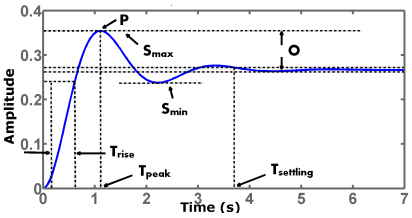


Fig. 7. Control performance metrics marked in a step response example.

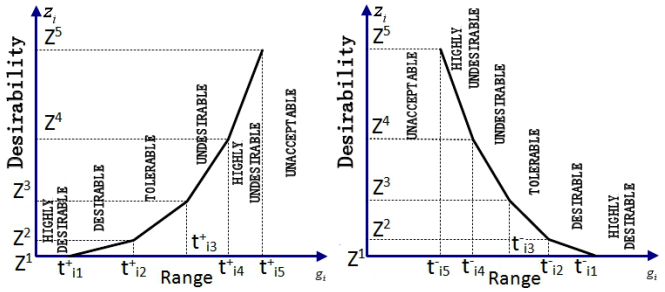
lifetime B, and (iii) maximize the network reliability R. Thus the problem can be formulated as

$$\begin{aligned} &min/max: f_L(P,C,A), f_B(P,C,A), f_R(P,C,A) \\ &subject\ to: P \in [P_{min}, P_{max}] \\ &\qquad\qquad C \in [C_{min}, C_{max}] \\ &\qquad\qquad A \in [A_{min}, A_{max}] \end{aligned} \tag{3}$$

where $[P_{min}, P_{max}]$, $[C_{min}, C_{max}]$, and $[A_{min}, A_{max}]$ denote the feasible ranges of the PRR threshold P, the number of channels used C, and the number of attempts for each packet A, and $f_L(P,C,A), f_B(P,C,A), f_R(P,C,A)$ represent the vector of objectives that should be minimized or maximized subject to a number of bounds. We adopt the NSGA-II algorithm [33], one of the most widely used multi-objective evolutionary algorithms, to solve the problem. Since Eq. 3 defines three different objectives, there does not exist a single best solution which simultaneously optimizes all objectives. NSGA-II gives a large number of best tradeoff solutions lying on or near the Pareto frontier, which can serve as the parameter selection candidates. The next step therefore is to obtain the single (most attractive) best tradeoff decision based on the needs of target process application.

B. LPPA: A Linear Physical Programming based Approach

Instead of requesting the network users to specify the weights among different QoS requirements (with which they are less familiar), our Linear Physical Programming [34] based Approach (LPPA) allows the users to specify meaningful ranges of desirability on the control performance metrics (with which they are familiar). For example, Figure 7 illustrates seven example control metrics: overshoot O, settling time $T_{settling}$, rise time T_{rise}, peak time T_{peak}, peak P, setting max S_{max}, and setting min S_{min}. For the decreasing preference metrics (e.g., O), we use g_i to denote the ith generic criterion value, so the range of desirability can be defined as: (i) *Highly Desirable Range* ($g_i \leq t_{i1}^+$): an acceptable range over which the improvement that results from further reduction of the criterion is desired but is of minimal additional value; (ii) *Desirable Range* ($t_{i1}^+ \leq g_i \leq t_{i2}^+$): an acceptable range that is desirable; (iii) *Tolerable Range* ($t_{i2}^+ \leq g_i \leq t_{i3}^+$): an acceptable range that is tolerable; (iv) *Undesirable Range* ($t_{i3}^+ \leq g_i \leq t_{i4}^+$): an acceptable range that is undesirable; (v) *Highly Undesirable Range* ($t_{i4}^+ \leq g_i \leq t_{i5}^+$): an acceptable range that is highly undesirable; and (vi) *Unacceptable Range* ($t_{i5}^+ \leq g_i$): the range of values that is not acceptable (can be perceived as a hard constraint). Similar ranges of desirability,



(a) Class-1S. (b) Class-2S.

Fig. 8. Different class functions ith objective.

t_{ij}^-, can be defined for the increasing preference metrics (e.g., T_{rise}). Therefore, we define two different class-functions as follows:

- Class-1S: Smaller-Is-Better, i.e., minimization.
- Class-2S: Larger-Is-Better, i.e., maximization.

It is important to note that unlike weights in the weighted sum method, the parameters t_{ij}^+ and t_{ij}^- defined above are physically meaningful constants that are specified by control applications in light of user-supplied preferences associated with the ith metric (e.g., latency, battery lifetime, or reliability).

Our Network Analysis Engine guides the field engineer to implement the target control application in the simulator (see section III-A). Our LPPA first translates the desirability ranges on the control performance metrics into the desirability ranges on the QoS metrics by performing control simulations.

The ranges of all QoS metrics are then to be exploited by physical programming through an inter-criteria rule called "One Versus Others (OVO)," where a full improvement of g_i across a given range of preference is over a full reduction of all the other criteria across the next better range of preference. This is accomplished through a mapping of the preferences to a transformed class function space. Figure 8 shows the functions for Class-1S and Class-2S. The L, B, and R values are mapped to the desirability values z_i (called z-value). A lower z_i is always more desirable. By using the class functions, our approach converts three different criteria (i.e., L, B, and R) on the horizontal axis to z-value on the vertical axis for comparison. Then the aggregated z is used as a metric for desirability. The mathematical relations are as below:

$$z^s \equiv z_i(t_{is}^+) \equiv z_i(t_{is}^-)\ \forall i; (2 \leq s \leq 5); z^1 \equiv 0 \tag{4}$$

where s denotes a generic junction and i denotes the criterion number, and z^s means the z-value in each junction point on the vertical axis as shown in Figure 8. Eq. 4 guarantees that different metrics are treated equally when they are in the same desirability region.

The increasing value of z_i for ith criterion between adjacent junction points can be calculated by:

$$\bar{z}^s \equiv z^s - z^{s-1}; (2 \leq s \leq 5); z^1 \equiv 0 \tag{5}$$

showing that different criteria increase uniformly across the same desirability region.

Following the OVO rule, we apply the relationship:

$$\bar{z}^s = \beta(n_c - 1)\bar{z}^{s-1}; (3 \leq s \leq 5); n_c > 1; \beta > 1 \tag{6}$$

where n_c denotes the number of criteria which equals to 3 based on our design. β is used as a convexity parameter. And $\bar{z}^2$ should be initialized manually with a small positive number. However, it cannot guarantee the convexity of the class function only based on Eq. 6. The following functions should be satisfied in order to meet with convexity property:

$$\bar{t}_{is}^{+} = t_{is}^{+} - t_{i(s-1)}^{+}; \bar{t}_{is}^{-} = t_{is}^{-} - t_{i(s-1)}^{-}; (2 \leq s \leq 5) \quad (7)$$

where $\bar{t}_{is}^{+}$ and $\bar{t}_{is}^{-}$ denote the sth range of the ith criterion on the horizontal axis. So the magnitude of the slopes of each line (w_{is}^{+} and w_{is}^{-}) takes the following form:

$$w_{is}^{+} = \bar{z}^s / \bar{t}_{is}^{+}; w_{is}^{-} = \bar{z}^s / \bar{t}_{is}^{-}; (2 \leq s \leq 5) \quad (8)$$

Based on Eq. 8, the difference between the slope of each line $\bar{w}_{is}^{+}$ and $\bar{w}_{is}^{-}$ is:

$$\bar{w}_{is}^{+} = w_{is}^{+} - w_{i(s-1)}^{+}; \bar{w}_{is}^{-} = w_{is}^{-} - w_{i(s-1)}^{-}; (2 \leq s \leq 5) \quad (9)$$

The convexity requirement can be achieved by the relationship:

$$\bar{w}_{min} = \min_{i,s} \{\bar{w}_{is}^{+}, \bar{w}_{is}^{-}\} > 0; (2 \leq s \leq 5) \quad (10)$$

indicating that the slope of lines should increase monotonically. An iteration of increasing β by a step of 1 is needed until it satisfies Eq. 10 to meet with convexity.

We use the deviation value (d_{is}^{-},d_{is}^{+}) to calculate the aggregated z and the final decision making among the best tradeoff solutions is selected by calculating the below expression:

$$\begin{aligned} &\min_{d_{is}^{-}, d_{is}^{+}} \sum_{i=1}^{n_c} \sum_{s=2}^{5} (\bar{w}_{is}^{-} d_{is}^{-} + \bar{w}_{is}^{+} d_{is}^{+}) \\ &subject\ to: \\ &g_i - d_{is}^{+} \leq t_{i(s-1)}^{+}; d_{is}^{+} \geq 0; g_i \leq t_{i5}^{+}\ (i = 1, 2, 3, s = 2, ..., 5) \\ &g_i + d_{is}^{-} \geq t_{i(s-1)}^{-}; d_{is}^{-} \geq 0; g_i \geq t_{i5}^{-}\ (i = 1, 2, 3, s = 2, ..., 5) \end{aligned} \quad (11)$$

All the best tradeoff solutions on the Pareto frontier computed by NSGA-II are fed into Expr. 11. The final best-suited network parameter configuration results in the minimum aggregated z (Expr. 11).

V. Evaluation

To validate the efficiency of P-SAFE in optimally configuring the network parameters and adapting them at runtime to consistently satisfy the application QoS demand, we perform a series of experiments. We first examine the capability of P-SAFE to effectively adapt the parameters to accommodate QoS demand changes and then evaluate P-SAFE's performance under different network settings. We compare our P-SAFE against three baselines: (i) the method specified in *WirelessHART*, (ii) the *CR+CP* approach [6], and (iii) the *optimal* solution using a brute-force method[2] and repeat the experiments on three physical testbeds located in different cities: (1) the *BU* Testbed consisting of 50 TelosB motes deployed on a single floor of a building [24]; (2) the *CPSL* Testbed with 60 motes spanning three floors of a building [35]; and (3) the *Indriya* Testbed, an open access 105-node testbed deployed in a 3-floor building at National University of Singapore [36]. In all experiment, we empirically set β to 15 and $\bar{z}^2$ to 0.1 in P-SAFE to satisfy the convexity and OVO requirements in LPPA and assume that two Lithium Ion AA batteries with a total capacity of 22,100J are used to power each node for battery lifetime calculation.

[2]The brute-force method cannot be used in practice because of its heavy computation overhead. We run it offline and use it only for the comparison purpose.

TABLE II
Parameters selected by P-SAFE.

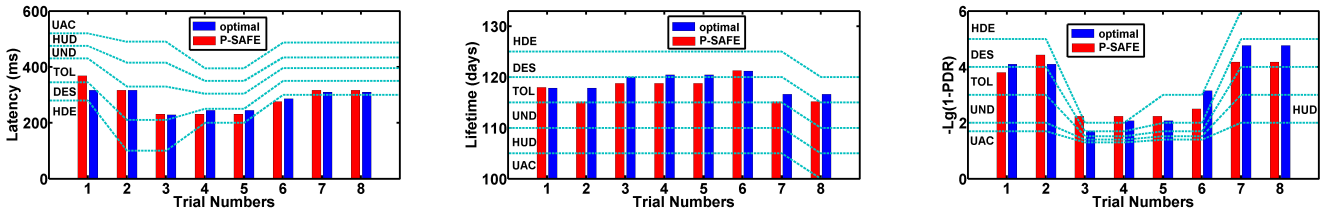
Set #	# of Channels	PRR Threshold	# of Attempts
1	2	89%	3
2	3	73%	3
3	3	71%	2
4	3	71%	2
5	3	71%	2
6	2	70%	2
7	3	72%	3
8	3	72%	3

TABLE III
Parameters selected by optimal.

Set #	# of Channels	PRR Threshold	# of Attempts
1	3	76%	3
2	3	76%	3
3	3	72%	2
4	3	85%	2
5	3	85%	2
6	2	88%	2
7	3	71%	3
8	3	71%	3

A. Adaptation to QoS Demand Changes

To test P-SAFE's capability to consistently satisfy the application QoS demand, we perform a series of controlled experiments where the control application specifies different ranges of desirability on three QoS attributes: latency, battery lifetime, and PDR. We set up six data flows with periods ranging from 800ms to 1600ms and two access points (node 121 and 124) on the BU Testbed. Figure 2 shows the network setting and Table I lists the source, destination, data period, and priority of each data flow. We employ an interacting two-tank control system [37] running on top of the network. The control application uses a timer to issue eight different sets of real-world desirability ranges, provided by our industry partner, to P-SAFE one by one with a 1 hour time interval. Only one QoS attribute is changed at a time. Table II and Table III show the parameters selected by P-SAFE and optimal for each set of desirability ranges. The selections made by P-SAFE and optimal are alike resulting in similar performance, as shown in Figure 9. P-SAFE effectively selects the best tradeoffs among three QoS attributes and always keeps the latency, battery lifetime, and PDR in the tolerable range or better. Please be noted that WirelessHART and CR+CP do not consider the desirability ranges, thus they select the same parameters for all eight sets. WirelessHART decides to use 8 channels, 60% as the PRR threshold, and 3 attempts per packet, while CR+CP selects 4 channels, 85% as the PRR



(a) Latency. (b) Battery lifetime. (c) PDR. $-lg(1-PDR)$ is used for y-axis for clarity.

Fig. 9. Resulting performance under P-SAFE and optimal.

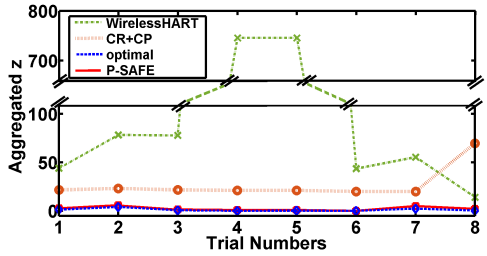


Fig. 10. Aggregated z under different methods.

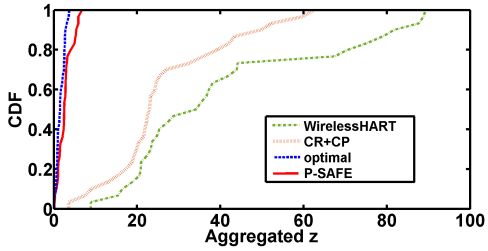


Fig. 11. Aggregated z with 30 different network settings on the BU Testbed.

threshold, and 75% threshold for the backup route, and 3 attempts per packet. Figure 10 shows the comparisons on the aggregated z-value z, the metric indicating how well the performance meets the application QoS demand, among four approaches. P-SAFE significantly outperforms WirelessHART and CR+CP. The maximum aggregated z under P-SAFE is no more than 5.78, very close to what optimal provides (4.22). WirelessHART has the worst performance, with an average aggregated z of 225.55 and a worst-case value of 745.55. This is because WirelessHART uses a predetermined PRR threshold and operates on all available channels. CR+CR reduces the average aggregated z to 26.40 and the worst-case to 69.85, since it considers the effect of PRR threshold and number of channels used on network performance. However, CR+CR fails to make tradeoffs when facing conflicting QoS requirements resulting in substantially higher aggregated z values compared to what P-SAFE and optimal offer.

B. Performance under Different Network Settings

To explore the consistency of P-SAFE's performance, we run the experiments under different network settings on three testbeds. We create thirty different network settings by varying the sources, destinations, data periods, and priorities of data flows on the BU Testbed. Figure 11 plots the Cumulative Distribution Function (CDF) of aggregated z under WirelessHART, CR+CP, P-SAFE, and optimal, respectively. As Figure 11 showed, the performance of P-SAFE is very close to the one under optimal. The worst-case (maximum) values are 6.78 and 3.69 under P-SAFE and optimal, while the maximum values under WirelessHART and CR+CP are 89.32 and 62.41. P-SAFE achieves an average aggregated z of 2.76, representing 14.1X and 8.9X lower compared to WirelessHART and CR+CP, respectively.

Fig. 12. The deployment of CPSL Testbed. Blue circles denote sensors and actuators and red squares represent two access points.

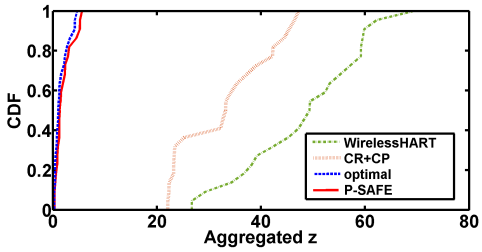


Fig. 13. Aggregated z with 30 different network settings on the CPSL Testbed.

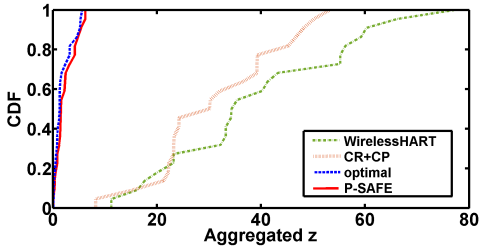


Fig. 14. Aggregated z with 30 different network settings on the Indriya Testbed.

We also repeat the experiments on the other two testbeds. On each testbed, we perform the experiments under 30 different network settings. Figure 12 shows the deployment of 60 TelosB motes on the CPSL Testbed spanning three floors of a building. Figure 13 plots the CDF of aggregated z under different approaches. We observe similar performance. The maximum aggregated z values are 5.61 and 4.67 under P-SAFE and optimal, while the worst-case values under WirelessHART and CR+CP are 69.28 and 47.23, respectively. P-SAFE achieves an average aggregated z of 2.08, representing

22.5X and 15.0X lower compared to WirelessHART and CR+CP, respectively. Figure 14 plots the CDF of aggregated z under different approaches. The maximum z under P-SAFE is 6.24, while the one under optimal is 5.62. WirelessHART and CR+CP have substantially higher aggregated z, with an average aggregated z of 41.13 under WirelessHART and 32.73 under CR+CP. The consistent results collected from various network settings under all three testbeds show that P-SAFE consistently better meets the application QoS demand, benefiting from our modeling method, multi-objective optimization, and QoS learning approach discussed in Section III.

VI. Conclusions

Recent studies have shown that the selection of network parameters has a significant effect on industrial WSANs' performance. However, the current practice of parameter selection is based on experience and rules of thumb involving a coarse-grained analysis of expected network load and dynamics or measurements during a few field trials, resulting in non-optimal decisions in many cases. This paper presents the P-SAFE, a framework that optimally configures the network parameters based on the application QoS demand and adapts the configuration at runtime to consistently satisfy the dynamic requirements. P-SAFE has been evaluated on three physical testbeds. Experimental results show our P-SAFE can significantly better meet the application QoS demand compared to the state of the art.

Acknowledgment

This work was supported by the NSF through grant CRII-1657275 (NeTS).

References

[1] WirelessHART. [Online]. Available: https://fieldcommgroup.org/technologies/hart

[2] IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH). [Online]. Available: https://tools.ietf.org/html/rfc7554

[3] ISA 100. [Online]. Available: https://isa100wci.org/

[4] 6TiSCH: IPv6 over the TSCH mode of IEEE 802.15.4e. [Online]. Available: https://datatracker.ietf.org/wg/6tisch/documents/

[5] http://www.ieee802.org/15/pub/TG4e.html. [Online]. Available: https://isa100wci.org/

[6] D. Gunatilaka, M. Sha, and C. Lu, "Impacts of Channel Selection on Industrial Wireless Sensor-Actuator Networks," in *IEEE Conference on Computer Communications (INFOCOM)*, 2017.

[7] D. Gunatilaka and C. Lu, "Conservative Channel Reuse in Real-Time Industrial Wireless Sensor-Actuator Networks," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2018.

[8] System Engineering Guidelines IEC 62591 WirelessHART by Emerson Process Management. [Online]. Available: http://www2.emersonprocess.com/siteadmincenter/PM%20Central%20Web%20Documents/EMR%5fWirelessHART%5fSysEngGuide.pdf

[9] M. Zimmerling, F. Ferrari, L. Mottolay, T. Voigty, and L. Thiele, "pTunes: Runtime Parameter Adaptation for Low-power MAC Protocols," in *ACM/IEEE International Symposium on Information Processing in Sensor Networks (IPSN)*, 2012.

[10] Y. Peng, Z. Li, D. Qiao, and W. Zhang, "I2C: A Holistic Approach to Prolong the Sensor Network Lifetime," in *IEEE Conference on Computer Communications (INFOCOM)*, 2013.

[11] J. Wang, Z. Cao, X. Mao, and Y. Liu, "Sleep in the Dins: Insomnia Therapy for Duty-cycled Sensor Networks," in *IEEE Conference on Computer Communications (INFOCOM)*, 2014.

[12] S. Fu, Y. Zhang, Y. Jiang, C. Hu, C.-Y. Shih, and P. J. Marron, "Experimental Study for Multi-layer Parameter Configuration of WSN Links," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2015.

[13] T. Watteyne, V. Handziski, X. Vilajosana, S. Duquennoy, O. Hahm, E. Baccelli, and A. Wolisz, "Industrial Wireless IP-Based CyberPhysical Systems," in *Proceedings of the IEEE, Special Issue on Industrial Cyber Physical Systems*, vol. 104, no. 5, 2016.

[14] A. Saifullah, D. Gunatilaka, P. Tiwari, M. Sha, C. Lu, B. Li, C. Wu, and Y. Chen, "Schedulability Analysis under Graph Routing in WirelessHART Networks," in *IEEE Real-Time Systems Symposium (RTSS)*, 2015.

[15] C. Wu, D. Gunatilaka, A. Saifullah, M. Sha, P. B. Tiwari, C. Lu, and Y. Chen, "Maximizing Network Lifetime of WirelessHART Networks under Graph Routing," in *IEEE International Conference on Internet-of-Things Design and Implementation (IoTDI)*, 2016.

[16] C. Wu, D. Gunatilaka, M. Sha, and C. Lu, "Real-Time Wireless Routing for Industrial Internet of Things," in *IEEE International Conference on Internet-of-Things Design and Implementation (IoTDI)*, 2018.

[17] J. Shi, M. Sha, and Z. Yang, "DiGS: Distributed Graph Routing and Scheduling for Industrial Wireless Sensor-Actuator Networks," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2018.

[18] C. Lu, A. Saifullah, B. Li, M. Sha, H. Gonzalez, D. Gunatilaka, C. Wu, L. Nie, and Y. Chen, "Real-Time Wireless Sensor-Actuator Networks for Industrial Cyber-Physical Systems," in *Proceedings of the IEEE, Special Issue on Industrial Cyber Physical Systems*, vol. 104, no. 5, 2016.

[19] N. Baccour, A. Kouba, L. Mottola, M. A. Ziga, H. Youssef, C. A. Boano, and M. Alves, "Radio Link Quality Estimation in Wireless Sensor Networks: a Survey," in *ACM Transactions on Sensor Networks*, vol. 8, no. 4, 2011.

[20] WCPS Simulator. [Online]. Available: http://wsn.cse.wustl.edu/index.php/WCPS:_Wireless_Cyber-Physical_Simulator

[21] TrueTime: Simulation of Networked and Embedded Control Systems. [Online]. Available: http://www.control.lth.se/truetime/

[22] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications," in *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2003.

[23] H. Lee, A. Cerpa, and P. Levis, "Improving Wireless Simulation through Noise Modeling," in *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2007.

[24] BU testbed at Binghamton University. [Online]. Available: http://www.cs.binghamton.edu/%7Emsha/testbed

[25] M. A. Bezerra, R. E. Santelli, E. P. Oliveira, L. S. Villar, and L. A. Escaleira, "Response Surface Methodology (RSM) as a Tool for Optimization in Analytical Chemistry," *Talanta*, vol. 76, no. 5, pp. 965 – 977, 2008.

[26] T. W. Simpson, J. Korte, T. Mauery, and F. Mistree, "Kriging Models for Global Approximation in Simulation-Based Multidisciplinary Design Optimization," in *AIAA Journal*, vol. 39, no. 12, 2001.

[27] Design-Expert 10. [Online]. Available: https://www.statease.com/soft-ftp

[28] D. Gorissen, I. Couckuyt, P. Demeester, T. Dhaene, and K. Crombecq, "A surrogate modeling and adaptive sampling toolbox for computer based design," *J. Mach. Learn. Res.*, vol. 11, pp. 2051–2055, Aug. 2010.

[29] K. Deb, "Multi-objective Optimization," in *Search Methodologies*, E. K. Burke and G. Kendall, Eds. Springer, 2014, ch. 15, pp. 403–449.

[30] I. Das and J. E. Dennis, "A Closer Look at Drawbacks of Minimizing Weighted Sums of Objectives for Pareto Set Generation in Multicriteria Optimization Problems," in *Structural Optimization*, vol. 14, no. 1, 1997.

[31] A. Messac, "Physical Programming: Effective Optimization for Computational Design," in *AIAA Journal*, vol. 34, no. 1, 1996.

[32] ——, "Multiobjective Decision-Making Using Physical Programming," in *Decision Making in Engineering Designs*, K. E. Lewis, W. Chen, and L. C. Schmidt, Eds. ASME, 2006, ch. 15, pp. 155–172.

[33] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," in *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, 2002.

[34] C. McAllister, T. Simpson, K. Hacker, K. Lewis, and A. Messac, "Integrating Linear Physical Programming within Collaborative Optimization for Multiobjective Multidisciplinary Design Optimization," *Structural and Multidisciplinary Optimization*, vol. 29, no. 3, pp. 178–189, 2005.

[35] CPSL Testbed at Washington University in St. Louis. [Online]. Available: http://cps.cse.wustl.edu/index.php/Testbed

[36] INDRIYA: A Wireless Sensor Network Testbed. [Online]. Available: https://indriya.comp.nus.edu.sg/motelab/html/index.php

[37] M. Changela1 and A. Kumar, "Designing a Controller for Two Tank Interacting System," in *International Journal of Science and Research*, vol. 4, no. 5, 2015.