Optimizing Warehouse Forklift Dispatching Using a Sensor Network and Stochastic Learning *

***Abstract*—The authors report on a successful deployment of an inexpensive mobile wireless sensor network in a commercial warehouse served by a fleet of forklifts. The aim is to improve forklift dispatching and reduce the costs associated with the delays of loading/unloading delivery trucks. To that end, an integrated system including both hardware and software is constructed. First, the forklifts are instrumented with sensor nodes that collect an array of information, including the forklifts' physical location, usage time, bumping/collision history, and battery status. The hardware's capability is enhanced with a theoretically sound hypothesis testing technique to capture the rather elusive location information, and the collection of the data is done in an efficient event-driven manner. The information acquired combined with inventory information is fed into a sophisticated actor-critic type stochastic learning method to generate dispatching recommendations. Because noise is inevitable in such wireless sensor networks, the performance of the algorithm is investigated under different noise levels. In combining wireless sensing with state-of-the-art decision theory, this work extends beyond the standard use of wireless sensor networks as monitoring devices.**

I. Introduction

The past decade has seen a large body of work aimed at collecting information using sensor networks, with a firm belief that this information will bring dramatic benefits (see [1–3] for comprehensive surveys). Some earlier works were motivated by military applications, with examples ranging from large-scale acoustic ocean surveillance systems to small networks of unattended ground sensors for target detection [4, 5]. Sensor networks can be used to improve infrastructure security too, for instance, detecting biological, chemical, and nuclear attacks [6, 7]. More examples of security applications come from home and office security systems, where monitoring means not only to prevent intrusion, but also to detect the occurrence of fire or carbon monoxide leakage [8]. Another large area of sensor network applications is environmental protection. These include tracking the movement of animals [9–13], environmental monitoring [14–16], pollutant detection [7], forest fire detection [17], and geophysical research [11]. Agriculture may also benefit from sensor networks. For example, [18] studied the potential use of sensor network in vineyard management; see also [19] on precision agriculture. Sensor networks have also been proven applicable in health care, including body motion analysis [20, 21], tracking doctors and monitoring patients inside hospitals [22–24], and physiological monitoring for daily human activities [25–28]. Sensor networks have also found applications in industrial environments. For example, [29] reported the monitoring of structural health through networked vibration sensors, and [30] discussed the wireless communication issues in the industrial environment rather thoroughly.

Despite this rich body of work, the applications explored focus primarily on monitoring rather than control. For example, [1] envisioned the use of sensor networks for warehouse management but did not go beyond inventory monitoring. Yet, for sensor networks to realize their full potential, more research is needed to combine the information they can provide with sophisticated control and decision making methodologies. This is the central aim of this paper which focuses on warehouse management. The same objective has been pursued in a few other areas, even though *management* has been mentioned as a target application in many works (see [18, 23]). In this work, the authors report on an on-going collaboration with a forklift manufacturer (Raymond Corp.) and a commercial warehouse in studying the opportunities of deploying a forklift sensor network system in a large warehouse for groceries.

This paper shows how deploying a sensor network in a warehouse served by a fleet of forklifts is feasible for a moderate cost, and can remarkably decrease warehouse operation costs. The authors elaborate on the choice of deployed sensors, the key information collected to support decision making, and on how the acquired information is integrated tightly with state-of-the-art decision theoretic techniques. In addition to the hardware, the key components of the system include an efficient information collection system, a localization algorithm developed by the co-authors, and a powerful approximate dynamic programming method, all integrated into a unified framework to optimize the dispatching of the forklift fleet. It is worth noting that this work goes beyond the standard use of sensor networks and shifts the paradigm from *monitoring* to *control* and *optimization*, where the information collected by the sensor network not only can help the warehouse managers make intuitive decisions, but can also drive sophisticated computer-aided decisions.

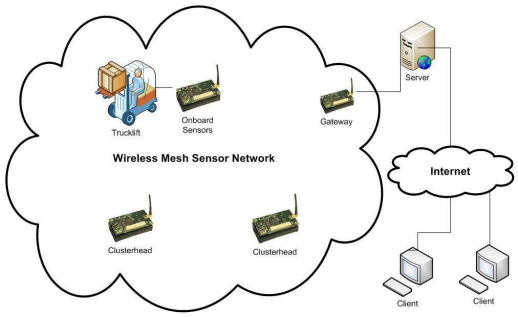

Fig. 1. System architecture.

In what follows, Section II describes concretely the choice of the deployed sensors, the information collected, and the data acquisition approach. Section III discusses system deployment issues. Section IV focuses on utilizing the collected information

in a stochastic learning framework to optimize forklift dispatching. A simulation model is constructed based on an on-site visit and discussions with the forklift manufacturer and warehouse managers. The results strongly encourage the adoption of this integrated system. Conclusions are drawn in Section V.

II. Data Acquisition System

The Data acquisition system consists of three key components (see Figure 1): (1) sensors and motes installed on the forklifts, (2) database and web server, and (3) users over the Internet. The sensors deployed are capable of collecting an array of real-time information from the forklifts, including ID number, usage time, history of collisions, battery level, and location. The data is transmitted via motes that form a dynamic mesh network to a base station connected to a PC. Then, the web server and its PHP scripts running on the PC respond to user queries remotely and interactively. The PC also runs control and optimization software needed for forklift dispatching and allows users to access the data locally or remotely through the Internet.

A. Sensing Forklift Usage Time

The usage time is of interest to dispatching because it determines when maintenance is required. It also gives the manager insight on when and how much the forklifts are utilized. The usage time of a forklift includes the *running time* and the *lifting time*. The running time is measured with the help of a membrane button installed under the pedal of the forklift. An MDA300 I/O board is used as the interface between the mote and the membrane button. To move the forklift, the driver has to step on the pedal which promptly triggers the membrane button to send an interruptible digital signal. This event is captured by the MicaZ mote through the MDA300 board. A timer on the mote will start counting until the driver releases the pedal, and then the mote will transmit a message containing the counted running time of the forklift to the server. The server then updates the usage time for the truck and plots the usage chart according to users' requests. Note that the memory and processing power of the motes are put into good use in implementing this event-driven approach.

The lifting time is measured by a pair of infrared sensors (a transmitter and a receiver) mounted on the rack of the forklift. The transmitter is mounted on the lifting part of the truck. Upon receiving the infrared beam, the receiver generates an interruptible digital signal to the mote. When the truck is not lifting, the transmitter and the receiver point to each other. The connection breaks when lifting is under way, in which case the mote starts counting like it does for the membrane button, and then it sends updates to the server in a similar fashion.

B. Detecting Forklift Collisions

It is not uncommon that a forklift collides with warehouse walls, shelves, or other forklifts. Frequent collisions may cause damages that are not apparent immediately, but accumulate to result in great losses if precautionary actions are not taken. For collision detection, the authors use a 2-axis accelerometer on a sensor board MTS310 to sample the movement of the forklift at a high frequency. The high frequency is necessary because collision signals are in the form of sharp impulses. One axis of the sensor is used to detect forward-backward bumping, and the other axis is used to detect the bumping from the sides. The threshold is set at $\pm$ 1G locally at the mote, so the mote signals a bumping event to the server when the sample reading exceeds this threshold. The server then displays the data both numerically and graphically via the web interface.

C. Monitoring Battery Level

When the battery level of a forklift is below a certain threshold, the forklift can not be operated until the battery is recharged, hence the battery level is important for the dispatching decisions to be discussed in Section IV. In order to measure the battery level of the forklift, two cells of the forklift battery were connected to the power supply port of the MDA300 I/O board of the mote. The mote includes the voltage readings in every packet it transmits to the server, thus no additional traffic is caused. The server can then use this information to calculate the "battery remaining" indicator as

$$\text{Battery Remaining} = \frac{\text{Voltage} - \text{Voltage}_{\text{Empty}}}{\text{Voltage}_{\text{Full}} - \text{Voltage}_{\text{Empty}}}.$$

Note first that the authors assume the forklifts' battery cells are load balanced, meaning that different cells have almost the same voltage readings. Second, a new battery has different output voltages compared to an old battery, hence, calibration for each battery is necessary. Specifically, the age of each battery is stored in the server, and the value of $\text{Voltage}_{\text{Full}}$ is reduced for older batteries according to the operator's experience.

D. Determining Forklift Location

The authors also implemented a localization engine based on received signal strength (RSSI). The underlying algorithm was developed in earlier work by the authors' group [31]. A brief overview of the algorithm is provided below.

The robust localization engine adopted compares the RSSI measurements with location profiles generated using prior measurements. In this localization engine, the site is divided into N regions, and M distinct positions are selected for clusterhead placement (a clusterhead is a stationary wireless node). The system seeks to identify the region where a forklift resides based on the RSSI recorded by the clusterheads when the mote on the forklift is transmitting (other RF characteristics may also be used, however, RSSI is the most commonly available measurement). During a *profiling phase*, measurements from a dense network deployed in the site are collected and a pdf family for each clusterhead-region pair is constructed. Let these pdf families be $P_{Y^{(k,p)}|\boldsymbol{\rho}_i}(y)$, where $Y^{(k,p)}$ denotes the random variable corresponding to observations $y^{(k,p)}$ at clusterhead B_k when the transmitting power is p and the transmitting sensor resides in some location i. $\boldsymbol{\rho}_i \in \Omega_i$ is a vector in some space Ω_i parameterizing the pdf family. In fact, k and p should also be in the subscript of the parameter $\boldsymbol{\rho}$, but they are omitted for notation simplicity.

Naturally, the profile of each location comes from signal samples taken at various spots around the location. Different values of $\boldsymbol{\rho}$ represent different scenarios affecting the RSSI. By taking all these scenarios into consideration, robustness is achieved. Without going into details, the use of $\boldsymbol{\rho}$ to construct the pdf family can be understood as a mechanism of weighing these samples. But different from most profile based techniques, the "weighing mechanism" here, which the authors named *linear interpolation of probability distributions*, has a sound theoretical basis and is also experimentally validated. Note that the linear pdf interpolation of two or more pdfs is not a simple weighted sum. Instead, it has the nice property of interpolating both the mean and the variance according to the specified weights, and furthermore interpolates the shapes of the pdfs in a reasonable sense. As a sanity check, note that the linear interpolation of two Gaussian pdfs will remain Gaussian, and this would not be true with simple weighted sum.

After profiling, the system is ready to make localization decisions. A mobile node in need of localization would broadcast a series of messages. The clusterheads measure the RSSI and send observations to the server where the *generalized likelihood* is calculated for each location hypothesis using the associated pdf family. The region with the maximum generalized likelihood is then selected as the localization result. To be precise, the index of the decision location is

$$\arg\max_{i} \sum_{p,n,k} \log \sup_{\boldsymbol{\rho}_i \in \Omega_i} P_{Y^{(k,p)}|\boldsymbol{\rho}_i}(y_n^{(k,p)}), \qquad (1)$$

where the subscript of y indicates that clusterheads make n i.i.d observations for each detection. Note that the feature of each location, against which measurements are compared, is a family of pdfs rather than a single pdf. This provides robustness with respect to the exact position of the sensor within region i.

A performance guarantee of this localization engine is calculated analytically and further optimized by clusterhead placement. Formally, the performance measure used is the error exponent, defined as

$$\lim \sup_{n\to\infty} \frac{1}{n} \log \mathrm{P}_{\mathrm{err}}(n),$$

where n is the number of the observations made, and $\mathrm{P}_{\mathrm{err}}(n)$ is the corresponding error probability.

To summarize the advantage of the localization engine, first, it is measurement-based, i.e., no a priori signal propagation model is required [32]. Moreover, this engine goes beyond the Gaussian assumption of the signal strength distribution and records the RSSI profiles of locations as general distribution functions. Earlier work of the co-authors [33] has shown that the general probabilistic model captures important information regarding the location compared to the Gaussian model. In addition, this engine is accompanied by an analytical performance guarantee and performance optimization through deployment.

III. Data Acquisition System Deployment

The system is deployed in a commercial warehouse located in Lawrence, Massachusetts. The warehouse is about 30,000 square feet and divided into 3 areas. An overview of the system is shown in Figure 2.

Five forklifts are instrumented with sensors and on-board motes. There are different models of forklifts manufactured by Raymond Corp., yet the method is general enough to be adapted to any forklift model without a need to interact with the forklifts' electronics.

Stationary nodes are installed in Areas 1 and 2 to relay the messages from the forklifts to the server. A total of 16 stationary nodes acting as clusterheads are placed along the walls of the warehouse to implement the localization system in Area 3. All the stationary nodes, the clusterheads, and the motes installed on the forklifts are Iris motes from Crossbow Technology (the Iris motes have a larger transmission range than the alternative MicaZ motes). In order to achieve a long battery life for the stationary nodes, the authors used 12V, 18Ah lead acid batteries and built DC-DC conversion circuits to power them. The expected battery life is about 30 days.

To enhance the robustness of the localization system, the authors profiled the RSSI distributions for two distinct frequencies (2420MHz and 2450MHz) and two distinct power levels (3dBm and -4.2dBm). Area 3 of the warehouse was partitioned into 28 regions as shown in Figure 3. The size of each region is 375 square feet. Two regions are considered *immediate neighbors* if their borders overlap and *secondary neighbors* if they have a common neighbor. For example, the immediate neighbors and secondary neighbors of Region 18 are those marked in dark gray and light gray, respectively.

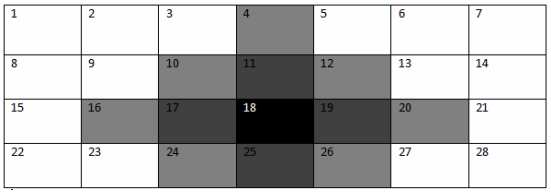

Fig. 3. Layout of regions in Area 3.

Two TinyOS networks were deployed to serve the system. A TinyOS-1 dynamic network deployed in Areas 1 and 2 was used to collect information on the usage time, battery status, and bumping events. This mesh network enables a dynamic ad-hoc multi-hop routing which connects the sensor network in a tree topology and uses a shortest-path-first algorithm with a single destination node (the gateway) and active two-way link estimation. A TinyOS-2 static network was used to gather information for localization purpose. Although there are multi-hop routing features in TinyOS-2, the authors decided to implement a static routing due to the stability of the network topology. These two networks were configured to operate at different frequencies in order to avoid interference. Two gateways were set up for collecting data from these networks. Each gateway was implemented by a MIB520 interface board and an Iris mote. The gateways were each connected to a server via the Internet.

To monitor the forklifts in real time, small enough delay and processing time at each sensor node is desirable. Further, one

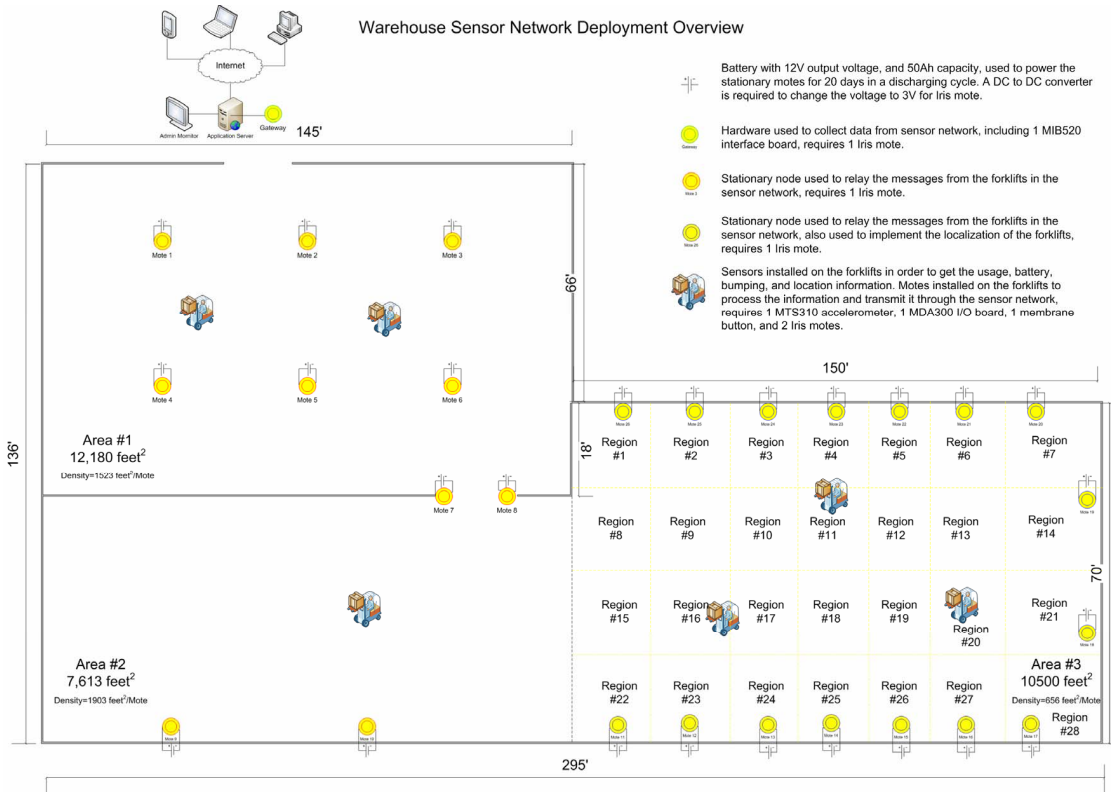

Fig. 2. Warehouse sensor network deployment overview.

wants to ensure that event notifications are received at the base station, so that the warehouse manager can be take immediate action in case of emergency. The sensor network platform used provides enough bandwidth and a reliable transmission protocol to meet such a requirement. The typical transmission range of an Iris mote is 150 feet and very few (only 1-2) relays are needed for the packets to reach the gateways. The event driven model of TinyOS ensures that the events are handled once they occur. The IEEE 802.15.4 compliant – ZigBee radio – is capable of sending packets at a rate of 250Kbps. The size of each packet is about 20 bytes, and there are a maximum of 2 hops from any sensor node to the base station, hence, the transmission delay for each event packet is less than 2ms. The standard TinyOS reliable packet transmission protocol is used and it utilizes a "handshake" between the gateway and the node. This ensures that each event packet is received at the server. The dynamic mesh network is established, and each sensor node always has an alternative routing path to enhance robustness. In addition, the user at the server side can also monitor the connection of the motes on the forklifts as well as the battery status of the stationary nodes and clusterheads. Since the nodes on the forklifts are powered by the forklifts' own batteries and these batteries are recharged periodically, the energy consumption at the nodes due to multiple transmissions of event packets is not an issue.

IV. Forklift Dispatching

The dispatching of the forklifts utilizes all the information provided by the sensor network as described above — location, bumping history, usage time, and battery level, where the latter three are lumped into what can be called the *health* of the forklift. This health indicator is predictive of whether the forklift will require maintenance in the coming hours. The dispatching problem also involves the stocking information of the warehouse, which is already available at a database accessible by the network.

A. System Model

Multiple items are supplied and demanded dynamically in the warehouse. The items come in and out of the warehouse through a central depot. After coming in, they are first stored at some vertically stacked reserve locations, and then moved to pick-up locations that are on the ground level and are specific for each item.

Three types of tasks are performed in the warehouse:

Task 1: Moving unloaded items from the depot to the reserve locations.

Task 2: Moving items from the reserve locations to fill the corresponding pick-up locations.

Task 3: Moving demanded items from the pick-up locations to the depot.

The forklifts perform Tasks 1 and 2, and Task 3 is done by a different type of vehicle (called pallet-truck). Any item in the

warehouse must go through a complete cycle of tasks in the above order (i.e., Task 1 $\rightarrow$ Task 2 $\rightarrow$ Task 3).

The operation of the pallet trucks is simple, rigid, and with no substantial opportunity for optimization. Only the dispatching of the forklifts is considered here, with the goal of moving items through the cycle as efficiently as possible. Intuitively, the factors that need to be considered include the urgency of the demand for specific items, traffic congestion experienced by the forklifts, and the health of the forklifts. The problem is naturally formulated as an average-cost infinite-horizon Markov Decision Process (MDP). The state of the system includes the level of items in each pick-up location (denoted by p_i, $i = 1, \cdots, N$, where N is the number of items), the level of items associated with each isle waiting in the depot (denoted by d_i, $i = 1, \cdots, I$ where I is the number of isles), the location of each forklift, the most recently assigned task to each forklift, and a forklift health indicator. The p_i's are allowed to take negative values; specifically, $p_i < 0$ indicates that $|p_i|$ items are "backordered" at pick-up location i. The capacities of the pick up locations and of the depot are assumed to be finite, and the level of items in each pick-up location and in the depot takes values from a discrete set.

A transition in the state of the MDP occurs when a forklift F finishes its job. A decision is the next job assignment — what F is going to do next. In addition to Tasks 1 and 2 described above, the forklift may also be commanded to recharge, receive maintenance, or simply idle. The process is driven by random arrival of deliveries and demands, plus random variations of traffic congestion. The latter is specific to each isle of the warehouse. The arrival processes of deliveries and demands are assumed to follow Poisson distributions, whereas the quantities of the deliveries and demands are assumed to follow a discrete uniform distribution. No arrival of deliveries is allowed when the depot is full, and no further demand of an item is accepted when the current level of the item in its pick-up location is at the maximum backordered level. The completion time of a task is assumed to follow an exponential distribution whose mean is proportional to the product of the distance that the forklift travels and the traffic congestion within the isles along the forklift's path. Precisely, the traffic congestion in the ith isle, indicated by K_i is defined as

$$K_i = W_1 X_i + W_2 F_i + W_3 H_i, \tag{2}$$

where F_i is the number of forklifts operating in the ith isle (known since the position of the forklifts is part of the state), X_i is an estimate of the pallet truck traffic, defined as

$$X_i = \sum_{I(j)=i,\ p_j<0} \log(-p_j),$$

and W_1, W_2 and W_3 are some empirical weights. The H_i's are obtained from the data of forklift collision history and are used to estimate the congestion within the isles since frequent forklift collision in an isle indicates a higher congestion within that isle.

The one-step cost g includes the opportunity cost of having shortages in the pickup locations (g_1), the cost associated with the delay of clearing the depot (g_2), and the cost of operating the forklifts (g_3), where

$$g_1 = \sum_{i=1}^{N} c_i \max(p_i, 0),\ g_2 = c_s \sum_{i=1}^{I} d_i,\ g_3 = c_f n_f. \tag{3}$$

In the above, c_i of item i equals the price of the item times the chance of having a sale per unit of time if the pickup location is stocked at or above demand; c_s is the average cost of delay of clearing the depot per pallet per unit of time; c_f denotes the average cost of operating a forklift per unit of time; and n_f denotes the number of forklifts that are being operated. Then,

$$g = (g_1 + g_2 + g_3)\Delta t, \tag{4}$$

where Δt is the time interval until transition into the next state occurs.

B. Overview of the LSTD Actor-Critic Algorithm

Notation: In what follows, boldface letters indicate vectors and matrices.

The forklift dispatching described above is naturally formulated as an average-cost infinite-horizon Markov Decision Process (MDP) problem. Due to a large number of items that have different values, demand patterns and other contextual variables, forklift dispatching is a high-dimensional dynamic decision problem, hence, finding an exact optimal solution is computationally impossible. In fact, it is well-known that many such problems suffer from the so-called curse of dimensionality. To overcome the difficulty, a number of approaches have been proposed, such as *reinforcement learning* [34], *neuro-dynamic programming* [35], and *actor-critic algorithms* [36], all of which fall within the broad class of *approximate dynamic programming* [37].

Actor-critic algorithms are typically used to optimize *randomized stationary policies (RSPs)* using policy gradient estimation. The basic ideas is to construct a randomized policy with a low-dimensional tuning knob — the parameter $\boldsymbol{\theta}$ in what follows — and use stochastic learning techniques to find the best value of $\boldsymbol{\theta}$. Many different versions of actor-critic algorithms have been proposed [38–44]. More recently, a powerful convergent actor-critic method using Least Squares Temporal Difference (LSTD) learning was investigated by the co-authors [45], in which the critic was shown to enjoy the optimal convergence rate [46].

Consider an average-cost MDP with finite state and action spaces. Let k denote time, $\mathbb{X}$ denote the state space, and $\mathbb{U}$ denote the action space. Consider $\mathbf{x} \in \mathbb{X}$ and $u \in \mathbb{U}$. Let $g(\mathbf{x}, u)$ be the one-step cost. The policy candidates are assumed to be a parameterized family of RSPs, $\{\mu_{\boldsymbol{\theta}}(u|\mathbf{x})|\boldsymbol{\theta} \in \mathbb{R}^n\}$. That is, given a state $\mathbf{x}$ and a parameter $\boldsymbol{\theta}$, the policy applies action u with probability $\mu_{\boldsymbol{\theta}}(u|\mathbf{x})$. Let $p(\boldsymbol{\xi}|\mathbf{x}, u)$ denote the state transition model (which is typically not explicitly known). Under suitable ergodicity conditions, $\{\mathbf{X}_k\}$ and $\{\mathbf{X}_k, U_k\}$ are Markov chains with stationary distributions under a fixed policy. The stationary distributions are then denoted by $\pi_{\boldsymbol{\theta}}(\mathbf{x})$ and $\eta_{\boldsymbol{\theta}}(\mathbf{x}, u)$, respectively. The average cost of the process is thus well defined and given by $\bar{\alpha}(\boldsymbol{\theta}) = \sum_{\mathbf{x},u} \eta_{\boldsymbol{\theta}}(\mathbf{x}, u) g(\mathbf{x}, u)$. The authors will not elaborate on the ergodicity conditions,

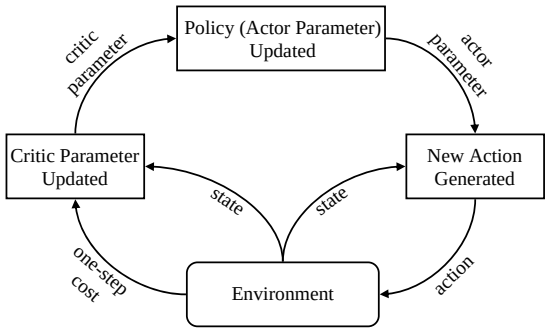

Fig. 4. The actor-critic system.

except to note that in the present case it suffices that the process $\{\mathbf{X}_k\}$ is irreducible and aperiodic given any $\boldsymbol{\theta}$, and for any $\mathbf{x} \in \mathbb{X}$ and $u \in \mathbb{U}$, either $\inf_{\boldsymbol{\theta}} \mu_{\boldsymbol{\theta}}(u|\mathbf{x}) > 0$ or $\mu_{\boldsymbol{\theta}}(u|\mathbf{x}) \equiv 0$, $\forall \boldsymbol{\theta}$ holds (see Theorem 2.6 of [46]).

With no explicit model of the state transition but only a sample path, the actor-critic algorithms typically optimize $\boldsymbol{\theta}$ locally in the following way: first, the critic estimates the policy gradient $\nabla \bar{\alpha}(\boldsymbol{\theta})$ using LSTD learning; then the actor modifies the policy parameter along the gradient direction. The actor and the critic updates take place in the course of a simulation of a single sample path of the process. Let $\mathbf{X}_k$ denote the state of the system at time k. Let $\mathbf{r}_k$ be the parameter vector of the critic at time k, $\boldsymbol{\theta}_k$ be the parameter vector of the actor at time k, and $\mathbf{X}_{k+1}$ be the new state obtained after action U_k is applied when the state is $\mathbf{X}_k$. A new action U_{k+1} is generated according to the RSP corresponding to the actor parameter $\boldsymbol{\theta}_k$. Figure 4 illustrates the actor-critic system. The critic and actor carry out the following updates, where α_k is an estimate of the average cost, and $\mathbf{Z}_k, \mathbf{b}_k$ and $\mathbf{A}_k$ are intermediary variables:

Initialization:

Set $\mathbf{Z}_0, \alpha_0, \mathbf{A}_0, \mathbf{b}_0$ and $\mathbf{r}_0$ to zeros. Let $\boldsymbol{\theta}_0$ take some initial value, potentially corresponding to a heuristic policy.

Critic Update:

$$\begin{aligned} \alpha_{k+1} &= \alpha_k + \gamma_k \left[g(\mathbf{X}_k, U_k) - \alpha_k \right], \\ \mathbf{Z}_{k+1} &= \lambda \mathbf{Z}_k + \boldsymbol{\psi}_{\boldsymbol{\theta}_k}(\mathbf{X}_k, U_k), \\ \mathbf{b}_{k+1} &= \mathbf{b}_k + \gamma_k \left[(g(\mathbf{X}_k, U_k) - \alpha_k)\mathbf{Z}_k - \mathbf{b}_k \right], \\ \mathbf{A}_{k+1} &= \mathbf{A}_k + \gamma_k \left[\mathbf{Z}_k (\boldsymbol{\psi}'_{\theta_k}(\mathbf{X}_{k+1}, U_{k+1}) \right. \\ &\quad \left. - \boldsymbol{\psi}'_{\theta_k}(\mathbf{X}_k, U_k)) - \mathbf{A}_k \right], \end{aligned} \tag{5}$$

where $\lambda \in [0, 1)$, $\gamma_k \stackrel{\triangle}{=} \dfrac{1}{k}$,

$$\mathbf{r}_{k+1} = \begin{cases} -\mathbf{A}_k^{-1}\mathbf{b}_k & : k \geq K \\ \mathbf{0} & : \text{otherwise,} \end{cases} \tag{6}$$

where K is the smallest index such that $\mathbf{A}_K$ is invertible, and

$$\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{x}, u) = \nabla \ln \mu_{\boldsymbol{\theta}}(u|\mathbf{x}),$$

where $\boldsymbol{\psi}_{\boldsymbol{\theta}} = \mathbf{0}$ when $\mathbf{x}, u$ are such that $\mu_{\boldsymbol{\theta}}(u|\mathbf{x}) \equiv 0$ for all $\boldsymbol{\theta}$. It is assumed that $\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{x}, u)$ is bounded and continuously differentiable.

Actor Update:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \beta_k \Gamma(\mathbf{r}_k) \mathbf{r}'_k \boldsymbol{\psi}_{\theta_k}(\mathbf{X}_{k+1}, U_{k+1}) \boldsymbol{\psi}_{\theta_k}(\mathbf{X}_{k+1}, U_{k+1}). \tag{7}$$

In the above, $\{\gamma_k\}$ controls the critic step-size, while $\{\beta_k\}$ and $\Gamma(\mathbf{r})$ control the actor step-size together. An implementation of this algorithm needs to make these choices. The role of $\Gamma(\mathbf{r})$ is mainly to keep the actor updates bounded, and one can for instance use

$$\Gamma(\mathbf{r}) = \begin{cases} \dfrac{D}{||\mathbf{r}||}, & \text{if } ||\mathbf{r}|| > D, \\ 1, & \text{otherwise,} \end{cases} \qquad \text{for some } D > 0. \tag{8}$$

The reader is referred to the work of the co-authors [45] for a detailed discussion of the algorithm and the proof of its convergence. Interested readers may also consult [47].

Last, the computation cost of the two-step update of this algorithm is examined. First, the critic update consists of updating parameters in Equation (5), i.e., α_{k+1}, $\mathbf{Z}_{k+1}$, $\mathbf{b}_{k+1}$, and $\mathbf{A}_{k+1}$ and updating the parameter in Equation (6), i.e., $\mathbf{r}_{k+1}$. The former group only involves linear operations. The latter involves inverting the matrix $\mathbf{A}_k$, but the dimensionality of the matrix $\mathbf{A}_k$, which is the same as the dimensionality of the parameter set $\boldsymbol{\theta}$ of the RSP, is very low (the proposed RSP, discussed next, has 4 parameter scalars, hence $\mathbf{A}_k$ is a 4x4 matrix in the forklift dispatching problem, regardless of the size of the warehouse). Second, the actor update, i.e., updating $\boldsymbol{\theta}_{k+1}$ in Equation (7), also has very low complexity, since it is basically simple multiplication of very low dimensional vectors.

C. LSTD Actor-Critic Algorithm for Optimizing Forklift Dispatching

The LSTD Actor-Critic algorithm was used in a simulation to find the near optimal policy parameter $\boldsymbol{\theta}$, which can then be used in the actual deployment. Note that the algorithm may also be used to learn in real-time, using data from an on-going operation. But practically, that would require a reasonably good starting policy, which means simulation-based learning should probably still be conducted first.

To apply the LSTD Actor-Critic algorithm, the key is to formulate an RSP that is both compact and descriptive. Here, the RSP is constructed with exponential functions and four scalar parameters, $\theta_1, \theta_2, \theta_3$, and θ_4. Respectively, these parameters are used to assign weights to the demand, the delay at the depot, traffic congestion, and forklift health. More precisely, the RSP is a vector $\boldsymbol{\mu}$ of probabilities including $N + I + 2$ elements. Relating to earlier notation, the ith element of $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{x})$ equals $\mu_{\boldsymbol{\theta}}(u_i|\mathbf{x})$, where $\mathbf{x}$ is the state and u_i, $i = 1, \ldots, N + I + 2$ is an element of the action set. Note that at each instance where the RSP makes a decision, we assume only one forklift has finished its previous task and is available for new assignment. This is with little loss of generality because the next instance when another forklift finishes its task can be arbitrarily close (subject only to the discretization of time in simulation). Thus, the action set here is the set of all tasks that can be assigned to a single forklift. The indices of the actions are arranged as follows:

1) the first N elements correspond to Task 2 assignments, where each element maps to one of the N pick-up locations;
2) the next I elements concern Task 1 assignments, where element $N+i$, $i = 1, \ldots, I$, corresponds to moving items from the depot to a reserve location in isle i;
3) element $(N + I + 1)$ corresponds to "going to the battery/maintenance shop"; and, finally,
4) element $(N + I + 2)$ corresponds to "idling".

More precisely, the RSP used by each forklift is given by

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{x}) = \frac{1}{(\sum_{i=1}^{N+I+2} a_i(\mathbf{x}))} \begin{pmatrix} a_1(\mathbf{x}) \\ a_2(\mathbf{x}) \\ \vdots \\ a_{N+I+2}(\mathbf{x}) \end{pmatrix},$$

where $\mathbf{x}$ is a vector whose components include:

- p_i: the level of item i in its pick-up location $(i = 1, \ldots, N)$,
- d_j: the level of items associated with isle j waiting at the depot $(j = 1, \ldots, I)$,
- l_k: the location of forklift k, $\quad (k = 1, \ldots, F)$,
- s_k: the status of forklift k, $\quad (k = 1, \ldots, F)$,
- h_k: health of forklift k, $\quad (k = 1, \ldots, F)$.

Then, the a_i's are constructed as

$$a_i(\mathbf{x}) = \begin{cases} P_i \, e^{\theta_1 c_i (U - p_i) + \theta_3 K_{I(i)}^{-1}}, & 1 \le i \le N \text{ (Task 2)}, \\ D_{i-N} \, e^{\theta_2 d_{i-N} + \theta_3 K_{i-N}^{-1}}, & N+1 \le i \le N+I \text{ (Task 1)}, \\ B \, e^{\theta_4}, & i = N+I+1 \text{ (Maintenance)}, \\ 1, & i = N+I+2 \text{ (Idle)}, \end{cases}$$

where U is the maximum capacity of pick-up locations, and

$$P_i = \begin{cases} 0, & \text{if the pick-up location of the } i\text{th item is full} \\ & \text{or the health of the forklift is "bad"}, \\ 1, & \text{otherwise}, \end{cases}$$

$$D_i = \begin{cases} 0, & \text{if there is no item associated with the } i\text{th} \\ & \text{isle waiting in the depot or the health of} \\ & \text{the forklift is "bad"}, \\ 1, & \text{otherwise}, \end{cases}$$

$$B = \begin{cases} 1, & \text{if the health of the forklift is "bad"}, \\ 0, & \text{otherwise}. \end{cases}$$

Some observations on the structure of the RSP are in order. The probability of filling pick-up location i increases with the value c_i of item i and the backlog $U - p_i$ but it decreases with the congestion level at the corresponding isle $I(j)$. Note that this probability is controlled by parameters θ_1 and θ_3. Congestion, in particular, is an important variable to take into account as multiple forklifts can not safely operate in an isle without significantly slowing down. Similarly, the probability of assigning a forklift to move material from the depot to a reserve location in isle j is increasing with the amount of such material accumulated at the depot (d_j) and decreasing with the congestion in isle j. The latter probability is controlled by parameters θ_2 and θ_3. Finally, the probability of sending the forklift to the maintenance/battery shop is controlled by the parameter θ_4. The task of the actor-critic algorithm is to optimize the policy performance over $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_4)$.

Last, note that the complexity of the problem grows rapidly with the number of isles, pickup locations, and forklifts. Recall that F is the number of forklifts, the status of each forklift has 4 possible values, I is the number of isles, and N is the number of pickup locations. Further, let B the number of forklift health levels, U be the number of stock levels in each pickup location, and Q be the number of stock levels of the depot. Then, the state space of the system has

$$U^N Q^I (N+2)^F 4^F B^F \tag{9}$$

discrete elements.

D. Experimental Results

The authors first obtained the exact optimal policy using standard *Dynamic Programming (DP)* for a small-scale version of this problem, with 2 forklifts, 2 isles, and 4 items. The shortage in a pickup location is assumed to have 2 levels, the stock of the depot is assumed to have 3 levels, and the forklift health is assumed to have 2 levels. That is, $U = 2, Q = 3, B = 2$. The optimal average cost turns out to be 5.44. The authors then used the LSTD actor-critic algorithm to optimize the RSP outlined earlier. Figure 5 shows the progress of the algorithm. Using the actor-critic algorithm, the optimal value of the average cost converges to 5.95 (the actor-critic algorithm took a much shorter time than the DP algorithm and consistently found near-optimal solutions along different sample paths, within 10% of the optimal DP cost). The authors also tracked the changes in the average congestion (Figure 6), average delay at the depot (Figure 7), and average availability of items to fulfill demand (Figure 8). The purpose is to qualitatively understand how the policy obtained trades-off these factors, which is useful in designing a good heuristic policy.

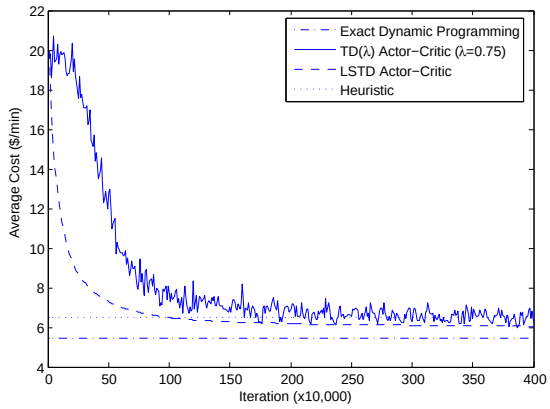

Fig. 5. Results obtained by algorithms for the small-scale problem.

In order to evaluate the efficiency, the authors used it to learn a locally optimal RSP for a larger-scale problem instance involving 10 forklifts, 4 isles and 32 items. In order to have

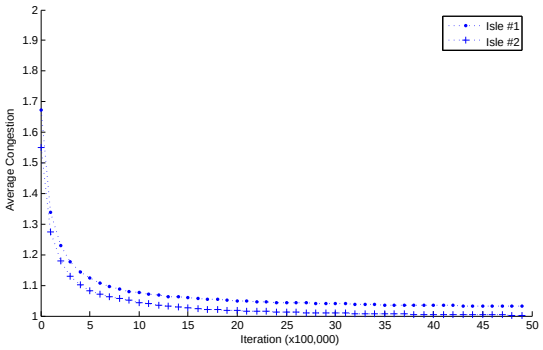

Fig. 6. Congestion data obtained by the proposed actor-critic algorithm for the small-scale problem.

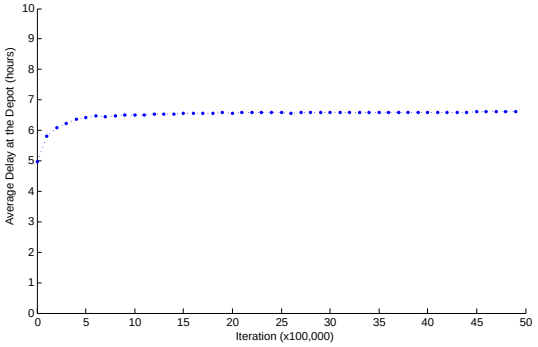

Fig. 7. Delay data obtained by the proposed actor-critic algorithm for the small-scale problem.

a more practical model, the state space of the health of the forklifts was extended from the set {'bad','fine'} to the set $\{1, 2, 3, 4, 5\}$ (the value 5 indicates the best health, and the value 1 is synonymous to "bad" and indicates the worst health). From Equation (9), one may compute that with only 2 forklifts, 2 isles and 4 items (and the values of U, Q, and B mentioned at the beginning of this subsection), the state space contains more than $200,000,000$ discrete elements; the size of the state space for the larger-scale problem is much greater, hence, it cannot be solved to optimality with exact DP in a reasonable amount of time. In order to have a basis of evaluation, the authors also designed a reasonably good heuristic policy. The heuristic was designed based on insights gained from the small-scale problem. The heuristic is believed to be a good one since for the small-scale problem it yields an average cost of 6.52 (within 20% of the optimal cost). It is also consistent with the state-of-the-art policies used by practitioners in the actual warehouse. In particular, the authors found from the DP solution that Task 2 has higher priority than Task 1. As soon as a forklift finishes its task, the heuristic assigns a new task to the forklift according to the following order of priorities. First, if the forklift's health is "bad" (or the corresponding value of the health is 1), then the heuristic policy commands the forklift to go to the battery/maintenance shop. Next, if there are shortages in the pick-up locations, then Task 2 is assigned, with higher priority given to more expensive items. Otherwise, if there are some items waiting in the depot, then Task 1 is assigned. Finally, if none of the above is true, then idling is assigned.

The authors tested the LSTD actor-critic algorithm to see if it is able to learn a better policy than the above heuristic. The result, shown in Figure 9, is encouraging. The average cost under the heuristic policy for the larger problem is 19.20, whereas the LSTD actor-critic converged to a policy that achieved an average cost of 14.66, representing a more than 20% reduction of cost compared to the heuristic policy. The result regarding imperfect information shown in the figure will be explained in Section IV-E.

In particular, the authors noticed that the actor-critic algorithm did a good job avoiding traffic congestion. Table I compares the values of average congestion obtained by the learned policy and by the heuristic policy. In addition, each iteration of the simulation represents the time between two events (when one out of all forklifts finishes its task), which is a varying amount of time that averages 0.1 minutes per iteration. Also, for the large-scale problem, which is computationally intractable with the exact dynamic programming algorithm, it took only about two hours for the actor-critic learning to converge to the solution.

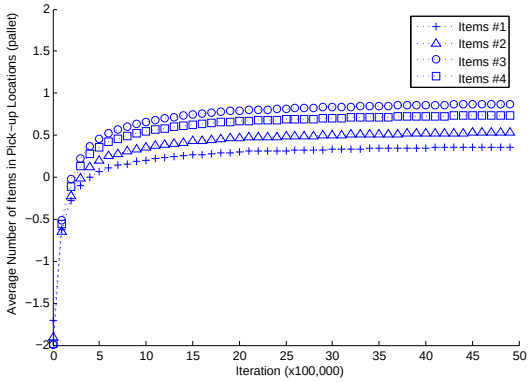

Fig. 8. Pick-up location data obtained by the proposed actor-critic algorithm for the small-scale problem.

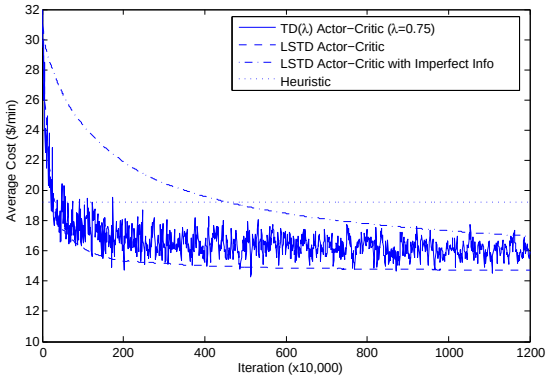

Fig. 9. Results obtained by algorithms for the large-scale problem.

Figure 5 and Figure 9 also compare the performance of the

TABLE I
COMPARISON OF THE AVERAGE CONGESTION LEVELS BETWEEN THE HEURISTIC POLICY AND THE POLICY LEARNED BY THE LSTD ACTOR-CRITIC FOR THE LARGE-SCALE PROBLEM.

	Isle 1	Isle 2	Isle 3	Isle 4
Heuristic Policy	6.47	5.63	4.91	4.03
Learned Policy	3.55	3.39	3.57	3.38

proposed LSTD actor-critic with the traditional TD-based actor-critic developed in [48] (with λ in the TD(λ) critic set to 0.75). Clearly, the LSTD actor-critic shows smoother convergence behavior.

E. Imperfect Information

To tightly integrate the hardware and the algorithm presented in this work, the authors analyzed the effect of imperfect information on the system. In many real-world applications, the state of the system can not be observed perfectly. In the warehouse model, the forklift location observations as well as the battery remaining measurements are subject to noise.

Each forklift has its internal reading of battery level which is displayed on the forklift's control panel and is believed to be very accurate. However, the reading cannot be acquired by the motes due to the design of the forklift. The authors performed experiments and compared the readings on the control panel to the installed sensor's estimates. It was observed that the estimates were within $\pm 5\%$ of the control panel readings.

The localization performance, on the other hand, is evaluated in terms of the frequency that the location estimation coincides with the correct region, a neighbor region, or a secondary neighbor. When the forklift resides in a region in which a clusterhead is placed, the localization engine identifies the correct region 80% of the times and returns the immediate regions the rest of the times. When the forklift resides in a region in which a clusterhead is not placed, the localization engine identifies the correct region 40% of the times, the immediate neighbors 45% of the times, and the secondary neighbors 15% of the times. The results are summarized in Figure 10.

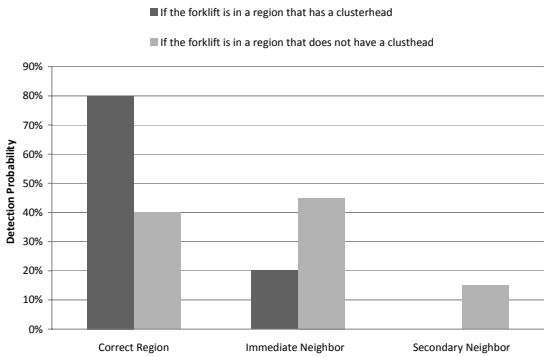

Fig. 10. Error statistics of the localization engine.

The authors used the actual data shown in Figure 10 (representing the performance of the deployed localization engine) and the error statistics in battery reading as the noise model used to assess the effect of imperfect information in the simulation. The results are shown in Figure 9. The average cost obtained by the LSTD actor-critic under imperfect information (presence of the noise) converged to 16.92 which still represents a more than 10% cost reduction compared to the heuristic policy. Although the learning curve is less effective in the presence of noise, the overall trend of the policy improvement is shown to be robust with respect to imperfect information.

Experiments to further study the effect of the noise were then performed. The authors let the noise level vary according to a parameter $\epsilon \quad (0 \leq \epsilon \leq 1)$ which indicates the level of the noise in localization and battery-level measurements, where $\epsilon = 1$ is associated with completely corrupted (random) measurements. To be precise, the simulated measurements are equal to the correct value with probability $1-\epsilon$, otherwise the measurements follow a uniform distribution over all feasible values. Note that this is an aggressive noise model which not only takes into account inaccurate measurements, but also it applies to corrupted data transmission over the wireless channel. Also, the range $0 \leq \epsilon \leq 1$ encompasses the complete spectrum of noise level. The results of stochastic learning are shown in Figure 11. One interesting observation is that, as long as the level of the noise is within $\epsilon \leq 0.24$, the actor-critic solution results in less average cost than that of the heuristic policy. That noise level is comparable to one that is greater than what we are experiencing in our current implementation. Furthermore, by putting forth a methodological demonstration of the utility of the information collected from the sensors, we provide the relevant industry an incentive to invest further in improving the accuracy of the sensing technologies.

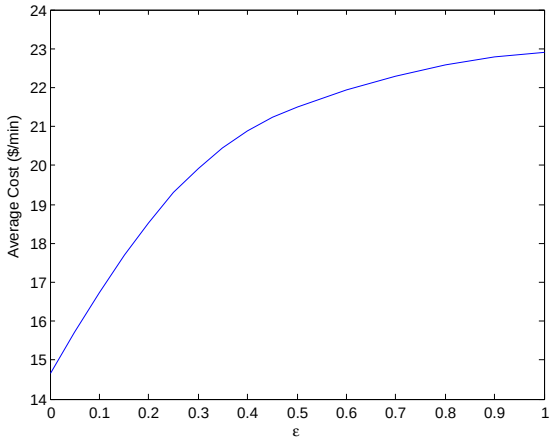

Fig. 11. Results obtained for the case where both location and battery measurements are subject to noise.

V. CONCLUSION

This paper reports on an approach for integrating low-cost, power efficient sensors, probabilistic localization, and stochastic learning in a wireless sensor network-based system applied to warehouse management. Forklifts in the warehouse are instrumented with sensor nodes which collect an array of

information, including data related to the usage of the forklift, its bumping/collision history, its battery status, and its physical location. It is discussed how the information can be collected in an efficient event-driven manner and utilized to optimize forklift dispatching so as to minimize warehouse operational costs. Our solution is quite general and can be easily adapted to any types of trucks and various warehouse models.

REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393 – 422, 2002.

[2] C. Chong and S. Kumar, "Sensor networks: Evolution, opportunities, and challenges," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1247–1256, August 2003.

[3] J. Gehrke and L. Liu, "Sensor-network applications," *IEEE Internet Computing*, vol. 10, no. 2, pp. 16–17, 2006.

[4] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors," *Communications of the ACM*, vol. 43, no. 5, pp. 51–58, 2000.

[5] D. S. Alberts, J. J. Garska, and F. P. Stein, "Network centric warfare: Developing and leveraging information superiority," [Online] http://www.dodccrp.org/NCW/ncw.html, 1999.

[6] R. Hills, "Sensing for danger," Sci. Technol. Rep. [Online] Available: http://www.llnl.gov/str/JulAug01/Hills.html, 2001.

[7] J. Xu, M. P. Johnson, P. S. Fischbeck, M. J. Small, and J. M. VanBriesen, "Robust placement of sensors in dynamic water distribution systems," *European Journal of Operational Research*, vol. 202, no. 3, pp. 707–716, May 2010.

[8] W. B. Heinzelman, A. L. Murphy, H. S. Carvalho, and M. A. Perillo, "Middleware to support sensor network applications," *IEEE Network*, vol. 18, no. 1, pp. 6–14, Jan-Feb 2004.

[9] T. Liu, C. Sadler, P. Zhang, and M. Martonosi, "Implementing software on resource-constrained mobile sensors: Experiences with impala and zebranet," in *Proceedings of Mobisys 2004, the Second International Conference on Mobile Systems, Applications, and Services*, June 2004.

[10] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, Atlanta, GA, September 2002.

[11] C. E. Nishimura and D. M. Conlon, "Iuss dual use: Monitoring whales and earthquakes using sosus," *Mar. Technol. Soc. J.*, vol. 27, no. 4, pp. 13–21, 1994.

[12] H. Wang, J. Elson, L. Girod, D. Estrin, and K. Yao, "Target classification and localization in habitat monitoring," in *Proceedings of the IEEE ICASSP 2003*, Hong Kong, April 2003.

[13] P. Zhang, C. M. Sadler, S. A. Lyon, and M. Martonosi, "Hardware design experiences in zebranet," in *Proceedings of SenSys 2004, the Second ACM Conference on Embedded Networked Sensor Systems*, November 2004.

[14] B. Halweil, "Study finds modern farming is costly," *World Watch*, vol. 14, no. 1, pp. 9–10, 2001.

[15] J. Hart and K. Martinez, "Environmental sensor networks: A revolution in the earth system science?" *Earth-Science Reviews*, vol. 78, no. 3-4, pp. 177–191, October 2006.

[16] D. C. Steere, A. Baptista, D. McNamee, C. Pu, and J. Walpole, "Research challenges in environmental observation and forecasting systems," in *Proceedings of the sixth annual international conference on Mobile computing and networking*, 2000, pp. 292–299.

[17] L. Yu, N. Wang, and X. Meng, "Real-time forest fire detection with wireless sensor networks," in *Proceedings of 2005 International Conference on Wireless Communications, Networking and Mobile Computing*, vol. 2, September 2005, pp. 1214– 1217.

[18] J. Burrell, T. Brooke, and R. Beckwith, "Vineyard computing: Sensor networks in agricultural production," *IEEE Pervasive Computing*, vol. 3, no. 1, pp. 38–45, 2004.

[19] A. Matese, S. D. Gennaro, A. Zaldei, L. Genesio, and F. Vaccari, "A wireless sensor network for precision viticulture: The nav system," *Computers and Electronics in Agriculture*, vol. 69, no. 1, pp. 51–58, November 2009.

[20] Y. Kan and C. Chen, "A wearable inertial sensor node for body motion analysis," *Sensors Journal, IEEE*, vol. PP, no. 99, p. 1, 2011.

[21] H. Ghasemzadeh and R. Jafari, "Physical movement monitoring using body sensor networks: A phonological approach to construct spatial decision trees," *Industrial Informatics, IEEE Transactions on*, vol. 7, no. 1, pp. 66 –77, feb. 2011.

[22] P. Bauer, M. Sichitiu, R. Istepanian, and K. Premaratne, "The mobile patient: wireless distributed sensor networks for patient monitoring and care," in *Proceedings 2000 IEEE EMBS International Conference on Information Technology Applications in Biomedicine*, 2000, pp. 17–21.

[23] D. Malan, T. Fulford-jones, M. Welsh, and S. Moulton, "Codeblue: An ad hoc sensor network infrastructure for emergency medical care," in *In International Workshop on Wearable and Implantable Body Sensor Networks*, 2004.

[24] L. Schwiebert, S. K. S. Gupta, and J. Weinmann, "Research challenges in wireless networks of biomedical sensors," *Mobile Computing and Networking*, pp. 151–165, 2001.

[25] N. Noury, T. Herve, V. Rialle, G. Virone, E. Mercier, G. Morey, A. Moro, and T. Porcheron, "Monitoring behavior in home using a smart fall sensor," in *Proceedings of IEEE-EMBS Special Topic Conference on Microtechnologies in Medicine and Biology*, October 2000, pp. 607–610.

[26] M. Ogawa, T. Tamura, and T. Togawa, "Fully automated biosignal acquisition in daily routine through 1 month," in *Proceedings of the International Conference on IEEE-EMBS*, 1998, pp. 1947–1950.

[27] P. Johnson and D. Andrews, "Remote continuous physiological monitoring in the home," *Journal of Telemed Telecare*, vol. 2, no. 2, pp. 107–113, 1996.

[28] Y.-D. Lee and W.-Y. Chung, "Wireless sensor network based wearable smart shirt for ubiquitous health and activity monitoring," *Sensors and Actuators: B. Chemical*, vol. 140, no. 2, pp. 390–395, July 2009.

[29] M. Whelan, M. Gangone, K. Janoyan, and R. Jha, "Real-time wireless vibration monitoring for operational modal analysis of an integral abutment highway bridge," *Engineering Structures*, vol. 31, no. 10, pp. 2224–2235, October 2009.

[30] A. Willig, "Recent and emerging topics in wireless industrial communications: A selection," *IEEE Transactions on Industrial Informatics*, vol. 4, no. 2, pp. 102–124, May 2008.

[31] I. C. Paschalidis and D. Guo, "Robust and distributed stochastic localization in sensor networks: Theory and experimental results," *ACM Trans. Sensor Networks*, vol. 5, no. 4, pp. 34:1–34:22, 2009.

[32] ——, "Robust and distributed localization in sensor networks," in *Proceedings of the 46th IEEE Conference on Decision and Control*, New Orleans, Louisiana, December 2007, pp. 933–938.

[33] I. C. Paschalidis, K. Li, and D. Guo, *Model-free probabilistic localization of wireless sensor network nodes in indoor environments*, ser. Lecture Notes in Computer Science. Springer, 2009, vol. 5801, pp. 66–78.

[34] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[35] D. P. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*. NH: Athena Scientific, 1996.

[36] A. Barto, R. Sutton, and C. Anderson, "Neuron-like elements that can solve difficult learning control problems," *IEEE Transactions on Systems, Man and Cybernetics*, no. 13, pp. 835–846, 1983.

[37] J. Si, A. Barto, W. Powell, , and D. W. II, *Handbook of Learning and Approximate Dynamic Programming*, IEEE Series on Computational Intelligence, IEEE Press, Piscataway, NJ, 2004.

[38] S. Bhatnagar, R. Sutton, M. Ghavamzadeh, and M. Lee, "Incremental natural actor-critic algorithms," *Neural Information Processing Systems (NIPS)*, 2007.

[39] V. Borkar, "An actor-critic algorithm for constrained markov decision processes," *Systems and Control Letters*, vol. 54, pp. 207–213, 2005.

[40] R. H. Crites and A. G. Barto, "An actor/critic algorithm that is equivalent to Q-learning," *Advances in Neural Information Processing Systems*, vol. 7, pp. 401–408, 1994.

[41] M. Ghavamzadeh and Y. Engel, "Bayesian actor-critic algorithms," in *ACM International Conference Proceeding Series*, vol. 227, 2007, pp. 297–304.

[42] E. Mizutani and S. Dreyfus, "Two stochastic dynamic programming problems by model-free actor-critic recurrent-network learning in non-markovian settings," in *Proceeding of IEEE International Joint Conference on Neural Networks*, vol. 2, 2004, pp. 1079–1084.

[43] C. Niedzwiedz, I. Elhanany, Z. Liu, and S. Livingston, "A consolidated actor-critic model with function approximation for high-dimensional POMDPs," in *AAAI 2008 workshop for Advancement in POMDP Solvers (part of the AAAI 2008 Conference)*, Chicago, 2008.

[44] M. Rosenstein and A. Barto, "Supervised actor-critic reinforcement learning," in *Learning and Approximate Dynamic Programming: Scaling*

Up to the Real World, W. P. J. Si, A. Barto and D. Wunsch, Eds. New York: John Wiley and Sons, Inc., 2004, pp. 359–380.

[45] I. Paschalidis, K. Li, and R. Moazzez Estanjini, "An Actor-Critic Method Using Least Squares Temporal Difference Learning," in *48th IEEE Conference on Decision and Control*, Shanghai, China, December 2009.

[46] V. R. Konda, "Actor-critic algorithms," Ph.D. dissertation, MIT, Cambridge, MA, 2002.

[47] H. Yu and D. Bertsekas, "Convergence results for some temporal difference methods based on least squares," *IEEE Transactions on Automatic Control*, vol. 54, no. 7, July 2009.

[48] V. R. Konda and J. N. Tsitsiklis, "On actor-critic algorithms," *SIAM Journal on Control and Optimization*, vol. 42, no. 4, pp. 1143–1166, 2003.