Distributed Multi-Actuator Control for Workload Balancing in Wireless Sensor and Actuator Networks

Changhua Wu, Girma S. Tewolde, Weihua Sheng, Bin Xu, and Yu Wang

***Abstract*—This technical note presents strategies for systematic ways to control mobile actuators in distributed wireless sensor and actuator networks. The efficiency, responsiveness, and service lifetime of the mobile actuators depend on the allocation of the workload to each individual actuator. Starting from an initial deployment, the proposed distributed multi-actuator control algorithm computes the load in each partition and uses the load imbalance as a virtual force to dynamically move the actuators until a balanced distribution is achieved. The load in each partition for an individual actuator includes both the total travel cost to visit or contact all sensors in the partition and the cost of servicing them. Finding the minimum workload allocation to serve each partition is an NP-hard problem. Therefore, efficient heuristics are proposed to do the partition, plan the path of each actuator and calculate its workload. Simulation results demonstrate the effectiveness of the proposed strategies.**

***Index Terms*—Distributed control, load balancing, path planning, wireless sensor and actuator networks (WSANs).**

I. Introduction

Wireless sensor networks (WSNs) have tremendous prospects due to their capability of obtaining valuable information from locations that are beyond human reach. Current research on WSNs often assumes that the number of sensors in a network is sufficiently large to cover the entire target area and maintain the network connectivity. However, in many applications (such as space exploration), certain types of sensors could be expensive, and hence it is impossible to have thousands of them for deployment. In addition, since the sensors would have relatively weak radios, inter-node separation is common in WSNs. Even if the number of sensors is sufficient and the radio is strong enough, poor deployment could also lead to bad network connectivity which makes data communication or other tasks very hard. Therefore, many recent WSN systems introduce powerful and even mobile actuators to enhance the existing network architectures. Unlike normal sensor nodes (usually small, inexpensive and with limited communication, computation, and energy resources), actuators (e.g., mobile robots and unmanned aerial vehicles) are resource rich devices with more energy, higher communication power, and better processing capabilities. They usually can perform much richer application-specific actions and interact with their environment. Actuators and sensor nodes can communicate among each other via wireless links and work collaboratively to perform automated tasks requiring sensing and actuation capabilities. This kind of WSNs is commonly referred to as *wireless sensor and actuator networks* (WSANs) [1], [2]. WSANs have great potential for a wide range of applications, such as, environmental control, event detection, health-care, home automation, manufacturing, search and rescue, etc. In many of those scenarios, mobile actuators can play important roles in different tasks: area coverage [3], [4], localization and navigation [5], target detection [6], service discovery [7], fault-tolerance [8], and data collection [9]–[13].

Previous research focuses on improvement of performance on a particular task in WSANs. In this technical note, instead we consider a general scenario where multiple actuators are deployed in a large distributed sensor network. These actuators can move freely and offer logistic services (e.g. power supply, sensor calibration, coverage control) and network services (e.g. data collection, hierarchical routing, time synchronization) to large number of sensors. Each actuator travels back and forth among the assigned sensors periodically to ensure the services. In this scenario, the cost of performing the assigned task has great effect on the outcome. Minimizing the cost will result in more efficient utilization of often limited resources and quicker response to urgent events. If the cost to service each sensor is constant and an actuator needs to visit the location of each sensor, then the total cost is dependent on the total traveling distance of the actuator for connecting all sensors. The minimization problem then is essentially a well known traveling salesman problem (TSP) [14]. However, since in many cases an actuator can serve a sensor remotely from a distance via wireless links, the possible service points of the actuator lie within an area instead of a point. This makes the path planning problem more difficult. When there are multiple actuators deployed at the same time, the workload partition for each actuator also becomes a challenge. Unbalanced load partition may lead to one or more actuators running out of their resources first. Therefore, in this technical note, we study an integrated path planning and load balancing problem for multi-actuator control in WSANs.

The goal of our study is to find a work-load partition where each actuator is only responsible for the task in its partition such that (1) the workload of each actuator is minimized and (2) the workloads of all actuators are approximately balanced or equal. To address this complex control problem, we formally define two simplified sub-problems in Section III:

- *Minimum-Load Path Planning for Single Actuator*. Given a fixed partition, find a path formed by a sequence of turning positions, such that (1) the actuator can service every sensor in this partition and make the minimum number of stops; and (2) the path results in the minimum total load or it approximates a minimal path.
- *Load Balancing among Multiple Actuators*. Given an initial set of partitions of all actuators, find the new position of every actuator and its corresponding partition, such that the standard deviation of loads among all actuators is minimized or less than certain threshold.

The two sub-problems are coupled together with the underlying partition method which makes finding the optimal solution very challenging. Several simplified versions of the problems are already known to be NP-hard. Thus, we propose a complete set of heuristics for these problems (Sections IV and V) and verify the performances via simulations (Section VI).

II. Related Work

We first review the current solutions for mobile actuator scheduling problem in WSANs. In [9], multiple mobile nodes (called MULEs)

Manuscript received February 03, 2010; revised February 14, 2011; accepted March 17, 2011. Date of publication August 12, 2011; date of current version October 05, 2011. This work was supported in part by the NSF China Grant 60803124 and 61170212, the US NSF under Grant CNS-0721666, CNS-0915331, and CNS-1050398, and by Tsinghua National Laboratory for Information Science and Technology. Recommended by Associate Editor J. Chen.

C. Wu and G. S. Tewolde are with the Department of Computer Science and the Department of Electrical and Computer Engineering, Kettering University, Flint, MI 48504 USA (e-mail: cwu@kettering.edu; gtewolde@kettering.edu).

W. Sheng is with the School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK, 74078 USA (e-mail: weihua.sheng@okstate.edu).

B. Xu is with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: xubin@tsinghua.edu.cn).

Y. Wang is with the Department of Computer Science, University of North Carolina at Charlotte, Charlotte, NC 28223 USA (e-mail: yu.wang@uncc.edu).

Color versions of one or more of the figures in this technical note are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TAC.2011.2164035

are introduced to pick up data from sensors when in close range and drop the data to access points. The MULEs take random walks in the field. However, the randomness of the paths of MULEs does not optimize or guarantee any performance. In [10], the routes of mobile nodes are carefully designed such that the traffic demand is met and the data delivery delay is minimized. A similar work in [11] studies the speed control of a single mobile actuator for data collection to minimize data delivery latency. All these methods focus on minimizing the data delivery delay instead of the workload of actuators. Recently, Xing *et al.* [12] study how to find a tour of the mobile actuator no longer than a budget and a set of routing trees that are rooted on the tour and connect all sensors, such that the total length of the trees is minimized. In their solution, a set of rendezvous points are picked based on an approximated Steiner tree, and then a TSP solver is used to schedule the actuator to visit all rendezvous nodes. All rendezvous nodes are sensor nodes and only a single actuator is considered. Ma and Yang [13] also study how to plan the paths of mobile actuators such that the length of data gathering tour is minimized. They assume that the possible polling points of actuators are finite, thus they can use a simple greedy algorithm to find an approximated tour. In our study, we consider infinite candidate polling points which makes our problem much harder. For multiple actuators, they assume that each actuator can only travel for certain length and the number of actuators is not fixed. Their solution uses a spanning tree to partition the task for multiple actuators. However, such partition could lead to uneven distribution of workload among actuators. Nesamony *et al.* [15] study how to find a shortest path for a single actuator to visit all sensors (represented as disjoint circles). They link their problem with a NP-hard problem, the *traveling salesman problem with neighborhoods* (TSPN) [16]. A heuristic is proposed in [15], which starts with randomly selected turning points inside the circles with certain visiting order, then iteratively changes to better set of turning points to shorten the path, until the path length stabilizes. They show that their method is more efficient than the brute force method via simulations. However, their method cannot handle the intersection of sensor transmission regions. In our problem, the visiting area could be any shape via intersection of circles. For a complete survey on mobile actuator scheduling in WSANs, refer to [2, Ch.6].

III. Problem Statement

Different from previous work, we aim to study the integrated path planning and load balancing problem among multiple actuators in a WSAN. These actuators service a large number of sensor nodes for a wide range of tasks such as data collection, sensor inspection, battery charging, etc. Hereafter, we use the data collection as an example to define our problem.

We assume that a set of n static wireless sensors, denoted by $S = \{s_1, \cdots, s_n\}$, are distributed in a two-dimensional region $\mathcal{R}$. Each sensor s_i has an omni-directional antenna so that it can talk to other sensors or actuators within a disk region centered at s_i with radius r_i. Hereafter, we call r_i the *transmission range* of s_i. Via wireless links, an actuator does not have to go to the exact location of s_i to perform the data collection task. Instead, the actuator can collect the data if it is within s_i's transmission range or within the transmission range of a sensor who is connected with s_i via a multi-hop relay. To collect data from a sensor s_i, an *on-spot cost* c_i is spent by the actuator during the contact with s_i. Again c_i could vary among different sensors, depending on the sensing data they collect. If the multi-hop relay is used for data aggregation among sensors, the communication cost of such relay can be taken into account in the on-spot cost of these sensors.

A set of k mobile actuators are deployed in the sensor network ($k \ll n$), denoted by $A = \{a_1, \cdots, a_k\}$. The actuator nodes can move freely in two-dimensional space and have the position information of all sensors. We assume that each actuator node has a transmission range large enough to communicate with other actuators via wireless links (i.e., the network formed by actuators is connected). In the beginning of each round, each actuator a_i is at an initial position p_i. By using any partition method, we can divide the region $\mathcal{R}$ into k partitions $R_1, \cdots, R_k$. Each actuator a_i will then be responsible only for the sensors in its partition R_i (sensors around p_i) and periodically perform data collection task for all sensors in R_i. We applied a simple partition method, Voronoi diagram [17], where all points in the partition R_i are closer to the position p_i of a_i than to positions of all other actuators. For each actuator, its workload depends not only on the number of sensors it serves but also on the distance it travels. Therefore, the total load L_i of actuator a_i contains two parts: the *traveling cost* which a_i spends on the road and the total *on-spot cost* which a_i spends at each service point to carry out certain task. We can safely assume that the *traveling cost* T_i is linear to the travel distance of a_i. The total on-spot cost C_i is the summation of all on-spot cost of sensors whose data are collected by the actuator. Therefore, $L_i = T_i + C_i = T_i + \sum_{s_j \in R_i} c_j$.

The multi-actuator control problem we study in this technical note aims to find a work-load partition such that (1) the workload of each actuator is minimized and (2) the workloads of all actuators are approximately balanced. The first goal can improve the response time of the actuator to service requests by sensor nodes in its partition and minimize the cost of each individual actuator, while the second goal tends to extend the life time of the distributed sensor and actuator network system by minimizing the over-utilization of some of the actuators and under-utilization of others. To address this complex control problem, we formally define the following two sub-problems and propose simple but efficient solutions for them in the next two sections.

Sub-problem I: *Minimum-Load Path Planning for Single Actuator*. Given a fixed partition R_i of actuator a_i, a_i's initial position p_i, and all sensors belonging to R_i, denoted by S_i, find a path P_i formed by a sequence of turning positions $U = \{u_1, \cdots, u_h\}$ where the actuator pauses and collects data from sensors, such that (1) the actuator can communicate with every sensor in S_i during the trip and make the minimum number h of stops;[1] and (2) the path $P_i = p_i u_1 \cdots u_h$ which the actuator will travel to collect data has the minimum total load L_i.

Sub-problem II: *Load Balancing among Multiple Actuators*. Given an initial set of partitions $R_1, \cdots, R_k$ of all actuators (and the loads of each actuator $L_1, \cdots, L_k$ which can be calculated by the solution from sub-problem one), find new position p_i of every actuator and corresponding partition, such that the loads L_i of all actuators under the new partitions are balanced. Particularly, we aim to minimize the standard deviation of the loads among all actuators.

Notice that these two problems are coupled together with the underlying partition method. The result from Sub-problem I provides the load estimate in each partition and affects the load balancing procedure, while Sub-problem II takes the load estimation in each round and changes the positions of actuators which affects the inputs of Sub-problem I. Thus, the joint problem of actuator control is a very complex and challenging task. If we define the actuator control problem as a single optimization problem to minimize the maximum work load of actuators, it will be very hard to solve it. Actually, just a special case of the Sub-problem I, where every sensor has an arbitrarily small

[1]Notice that the minimum number of stops does not necessarily lead to the minimum load. However, by adding this additional requirement, the optimization problem is much easier to be solved. We leave the optimization problem without this requirement as one of our future works.

transmission range, is an NP-hard problem, TSP. Therefore, in this technical note, we only focus on efficient heuristic solutions for both sub-problems.

IV. Minimum-Load Path Planning for Single Actuator

In this section, we describe the algorithm for computing the minimum load of an actuator a_i for servicing the sensors $S_i = \{s_1, \cdots, s_m\}$ inside its partition R_i. If we draw a circle around each sensor $s_j \in S_i$ using its transmission range r_j, then an actuator has to get into the circle of a sensor in order to collect data from it directly. If the disks of multiple sensors overlap, then the actuator can simply go to the overlapping area and service all sensors involved. Let $Z = \{z_1, \cdots, z_l\}$ denote the areas formed by the circles and their intersections. If an area z_j intersects the circle defined by sensor s_q, we say that sensor s_q can be covered by area z_j.

There are two optimization problems in the minimum-load path planning of the actuator. The first optimization problem is how to select the minimum number of areas from Z for the actuator to visit to guarantee the coverage of all sensors within this partition. This problem is actually the minimum set cover problem which aims to find the minimum number of subsets to cover the whole space. The minimum set cover problem is an NP-hard problem [14]. However, there exist many heuristics for it. We adopted a classical greedy heuristic (see Algorithm 1) for this problem. Fig. 1 illustrates an example with 9 sensors. If some sensors can communicate with each other, then it will suffice for the actuator to visit only one of these sensors to pick up data. For this situation, these sensors' transmission ranges can be merged into a union area as a single area in the input of Algorithm 1. For example the area z_3' in Fig. 1 is a union area of s_3 and s_6, since they are inside the transmission range of each other. Notice that by asking the sensors to increase their transmission ranges, the connectivity of the sensor network can increase, which will lead to less areas the actuator needs to visit. This is a trade-off between the communication cost plus power consumption at sensors and the power consumption at the actuator.

Algorithm 1 Greedy algorithm to select the minimum number of areas for covering sensors within a single partition R_i

Input: A set of sensors $S_i = \{s_1, \cdots, s_m\}$ within R_i and its corresponding set of areas $Z = \{z_1, \cdots, z_l\}$.

Output: A subset of target areas $Z' = \{z_1', \cdots, z_h'\}$ for the actuator a_i to visit.

1: Initially, set sensors *uncovered*, the uncovered counter $\alpha = m$, $Z' = \emptyset$. Let the potential coverage β_j of each area z_j be the number of circles intersecting with this area.

2: **while** $\alpha! = 0$ **do**

3: Select area z_j with the largest potential coverage β_j (using the minimum ID of nodes in the area to break a tie) and add it into the selected subset Z';

4: Mark all sensors covered by z_j *covered* and let $\alpha = \alpha - \beta_j$;

5: Update β_t for all adjacent areas z_t to the number of intersected circles from *uncovered* sensors.

6: **end while**

After selecting the areas for the actuator to visit, we need to decide the exact turning points where the actuator shall visit. Since the positions of these turning points can affect the total length of the path that it needs to visit, we have to consider the position selection problem jointly with the path planning problem. In our approach, we try to find a position in each selected area so that the total cost of the path traveled by the actuator is minimum. This problem is actually related to the TSPN [16]. Several approximation algorithms exist for TSPN, however most of them are very complex and not practical. Our algorithm (Algorithm 2) is an iterative algorithm in which at each step we add a new turning point inside one of the unvisited areas such that the cost added to the actuator path is minimized. Remember that serving each sensor s_q also incurs certain on-spot cost c_q. Therefore, for each target area z_j', we define an aggregated on-spot cost $c_j' = \sum_{s_q \text{ is covered by } z_j'} c_q$. We consider the aggregated on-spot costs when we calculate the total cost of a path. We have h target areas needed to be visited (output from Algorithm 1). Our algorithm terminates after h rounds, since in each round it adds a new turning point in the path that covers an unvisited area. Fig. 1 also shows the path generated by Algorithm 2.

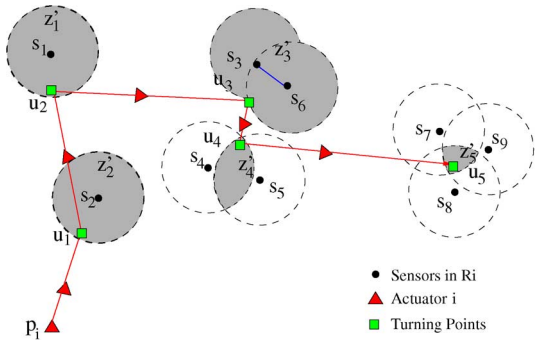

Fig. 1. Example: 13 areas are defined as Z by the 9 transmission ranges of s_i and their intersections. Grey areas are the target areas selected by Algorithm 1. Path P_i (red line) is generated by Algorithm 2.

Algorithm 2 Path planning for actuator a_i within partition R_i

Input: A set of target areas $Z' = \{z_1', \cdots, z_h'\}$ and their aggregated on-spot costs $\{c_1', \cdots, c_h'\}$, the initial position p_i of actuator a_i.

Output: A path $P_i = p_i u_1 \cdots u_h$ which the actuator a_i shall visit and total load L_i of a_i during its visiting.

1: Initially, set all areas z_j' *unvisited* and the unvisited counter $\alpha = h$. Let the initial path $P_i = p_i p_i$ with $\text{total cost} = 0$.

2: **while** $\alpha! = 0$ **do**

3: For each edge on $u_t u_{t+1}$ in path P_i and every unvisited area z_j', we draw an ellipse which uses u_t and u_{t+1} as its foci and is tangent to z_j'. Let u_j be the tangent point. See Fig. 2 for illustration. If z_j is selected to visit between u_t and u_{t+1}, the cost added to the path P_i will be $\|u_t u_j\| + \|u_j u_{t+1}\| - \|u_t u_{t+1}\| + c_j'$. When $u_{t+1} = p_i$, the cost is $\|u_t u_j\| + c_j'$, since it is the last hop in P_i.

4: Select the unvisited area which adds the least cost to path P_i among all edges in P_i. For example, in Fig. 2, z_p', hence u_p, is a better choice than u_j. Mark z_p' visited, and insert u_p between u_t and u_{t+1} in P_i. Thus the number of edges in the path increases by one. $\alpha = \alpha - 1$.

5: **end while**

6: **return** the final path P_i (removing p_i in the end) and its total cost as the total load L_i of a_i

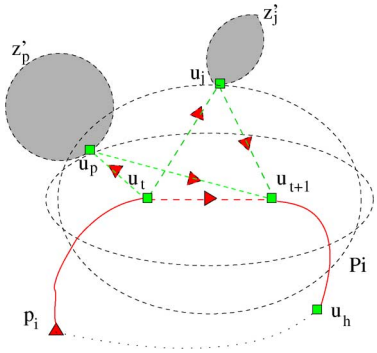

Fig. 2. For each edge on $u_t u_{t+1}$ in path P_i and every unvisited area z'_j (or z'_p), we draw the ellipse which uses u_t and u_{t+1} as its foci and is tangent to z'_j (or z'_p). Our algorithm selects the unvisited area which adds the least total cost to path P_i. In this example, z'_p is a better choice than z_j.

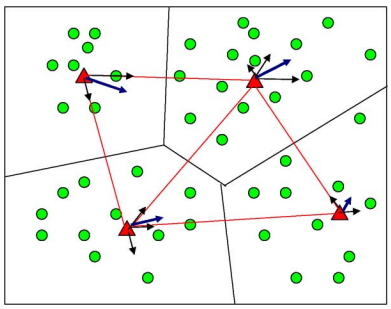

Fig. 3. Voronoi diagram for area partition and the virtual forces caused by load imbalances between neighboring partitions.

V. Load Balancing Among Multiple Actuators

Algorithm 1 and Algorithm 2 present strategies for minimum-load path planning of a single actuator to cover all sensors in its partition. The next step is to address the load balancing problem among multiple actuators in the whole network. The basic goal of the multi-actuator control algorithm is to partition the total area of distributed sensors into a given number k of regions for each actuator to take care of so that we achieve a more balanced load distribution among all actuators. The proposed control method uses classical Voronoi diagram as the partition method to divide the network into k partitions and uses heuristic algorithms in the previous section to estimate each actuator's workload. The basic idea of the control algorithm is quite simple, as illustrated in Fig. 3. Starting with an initial partition, the algorithm employs a virtual force method to iteratively compute and adjust the partitions to obtain a more balanced load distribution. If a net non-zero virtual force acts on an actuator, it will cause it to move in the direction of the force so as to modify the load distribution for a better balance. After the actuators move in response to the applied forces the working area is repartitioned and the new load distribution is computed. This process is repeated iteratively until a balanced load distribution is achieved among all the actuators (i.e., the standard deviation σ_L of the load distribution is smaller than a threshold value σ_{th}). The detailed algorithm is given in Algorithm 3.

Algorithm 3 Distributed multi-actuator control algorithm for workload balancing

Input: Initial location p_i of actuator a_i, and its partition r_i.

Output: A sequence of new location p_i of actuator a_i.

1: Calculate the service load L_i of a_i using Algorithm 1 and Algorithm 2 for serving S_i in R_i.

2: Exchange service load with adjacent actuators $a_{i_1}, a_{i_2}, \cdots, a_{i_{w_i}}$, where w_i is the number of neighboring actuators of a_i.

3: **while** the standard deviation of load distribution is larger than the threshold, i.e. $\sigma_L > \sigma_{th}$ **do**

4: Calculate the combined virtual force f_i on a_i $f_i =: \sum_{j=a_1}^{a_{w_i}} \gamma(L_j - L_i) \cdot \overrightarrow{x}_{ij}$. Here γ is a scale factor and $\overrightarrow{x}_{ij}$ is the unit vector from a_i to a_j.

5: If $f_i \geq f_{th}$ then change the velocity of a_i: $\nu_i = \nu_i + (f_i - \lambda\nu_i)/m \cdot \triangle t$. Here f_{th} is a small threshold value, the term $\lambda\nu_i$ introduces some viscous force so that the deployment can be quickly stabilized, m is a constant for the mass of the actuator node, and $\triangle t$ is the time step size per iteration.

6: Using ν_i update the new location of a_i as follows: $p_i = p_i + \nu_i \cdot \triangle t$.

7: Exchange new location information with neighboring actuators and calculate the *new* Voronoi diagram and update R_i and S_i.

8: Calculate the service load L_i of a_i using Algorithm 1 and Algorithm 2, and exchange the service load with adjacent actuators.

9: **end while**

VI. Experimental Study

A. Simulations of Path Planning Algorithms

We first carried out several simulation experiments to evaluate the proposed path planning methods (Algorithm 1 and Algorithm 2) which only consider the problem within one partition and with a single actuator. In the simulation, all sensors have the same range r_i and on-spot cost c_i. For simplification, the turning point u_j of each selected area z'_j is chosen to be the center of z'_j. However, this simplification does not undermine the virtue of the proposed approach. In the simulation, we compared the total load L_i by the proposed approach with two traditional methods: simple greedy method and near-optimal solution from TSP. In the simple greedy method, the actuator greedily selects the nearest sensor to visit at each step and all turning points are positions of sensors. This method has been used in [13]. The near optimal solution from TSP is obtained by a genetic algorithm [18] which aims to visit all sensors using a minimum cost path.

We conducted simulation experiments to compare the three methods with different settings. Fig. 4 shows the experimental results with a fixed transmission range at 8 and different numbers of sensors (from 5 to 29). We can see that the total load increases almost proportionally to the number of sensors. Among the three methods, our proposed approach can always achieve the smallest total load. Fig. 5 shows total loads found by the three methods with a fixed number (20) of random sensors and various transmission ranges (from 0 to 20). It shows that for the simple greedy method and the genetic TSP method, the total load does not vary much with regard to the transmission range, which is caused by the fact that they always go to the locations of sensors instead of using the overlapping transmission areas where data collection can be performed via wireless links. The total load found by the proposed approach, however, steadily decreases with increasing transmission range. This demonstrates that when the transmission range is large, there is high probability of overlap, therefore traveling only to the overlapping regions will save large amount of time and energy cost. Notice that when the transmission range is small (<5), the genetic TSP method can achieve better performance than our approach. However, it is more complex and computationally expensive.

TABLE I
SIMULATION RESULTS OF THE MULTI-ACTUATOR CONTROL ALGORITHM FOR LOAD BALANCING

Test no.	Service area	k	n	Initial load distribution		Final load distribution	
				Load in each partition	st. d.	Load in each partition	st. d.
1	40×40	4	40	29.9 37.0 74.2 53.4	19.7	57.6 58.4 60.1 56.5	1.5
2	40×40	4	40	49.0 33.7 75.5 17.6	24.6	56.9 50.5 53.8 55.3	2.7
3	40×40	4	40	29.7 42.0 70.3 75.7	22.2	61.5 66.4 64.8 62.8	2.2
4	40×40	5	40	25.0 52.5 14.2 22.0 45.0	16.2	46.5 46.7 42.6 48.9 46.5	2.2
5	40×40	5	40	69.2 56.9 18.1 88.7 36.1	27.6	55.0 51.4 48.3 50.8 51.0	2.4
6	40×40	5	40	33.1 21.2 107.8 38.0 26.8	35.5	56.2 51.1 46.8 52.8 50.5	3.4
7	50×50	5	50	78.0 36.2 126.4 48.0 31.4	39.3	87.1 86.1 82.6 80.5 94.4	5.3
8	50×50	5	50	101.1 79.8 47.9 60.1 56.1	21.4	84.2 82.2 76.5 80.8 76.5	3.5
9	80×80	5	100	190.0 80.0 95.2 67.1 182.2	58.6	155.4 139.0 146.4 144.2 138.4	6.9
10	100×100	5	100	242.4 36.0 117.6 78.1 294.3	110.1	189.1 178.1 176.9 170.6 183.5	7.0
11	100×100	5	200	140.8 175.8 256.3 99.2 453.0	139.9	275.9 261.3 268.7 265.3 259.0	6.7
12	100×100	5	200	147.4 161.7 373.8 310.3 177.9	101.5	271.9 276.1 282.4 277.3 287.6	6.0
13	200×200	10	500	307.1 314.4 556.9 736.1 208.5 212.0 395.5 452.2 325.9 227.6	168.8	443.4 415.1 452.9 462.8 426.8 454.3 410.5 401.8 418.4 415.6	21.4

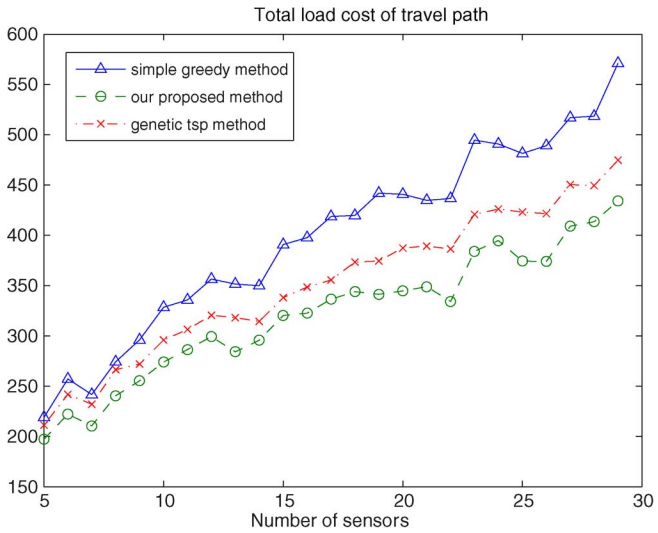

Fig. 4. Load cost L_i of the actuator with regard to the number of sensors.

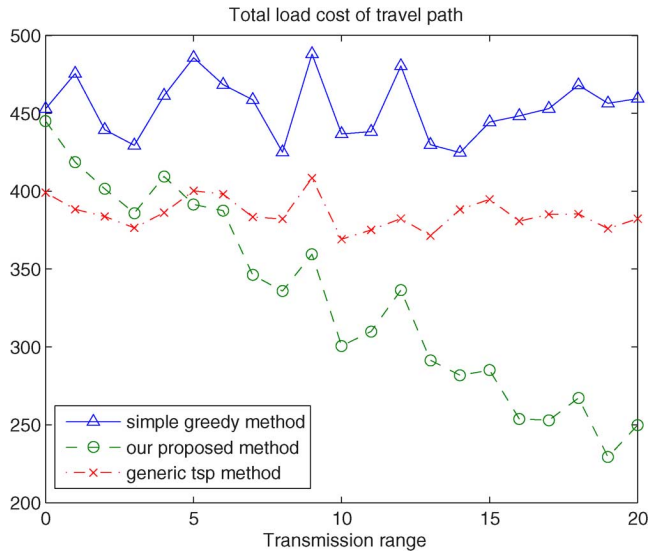

Fig. 5. Load cost L_i of the actuator with regard to the transmission range.

B. Simulations of Overall Actuator Control Algorithm

To evaluate the performance of our multi-actuator control algorithms with the aim of approximately balancing the loads in the different partitions, we further performed a series of simulation experiments with different configurations (various sizes of working area and numbers of sensors). In the simulation runs of Algorithm 3 the initial velocities of all the actuators are set to 0, the viscous friction constant λ and the scale factor γ set to 0.5, and the actuators' mass set to 1.0. We used a combination of termination conditions for Algorithm 3. First, there is a set maximum number of iterations of 1000. But if there is no appreciable change in the load distributions within 100 consecutive iterations the simulation is stopped. The algorithm also monitors the displacement of the actuators and terminates if the maximum displacement of all the actuators falls below a minimum threshold value of 0.001. Since we assume that the actuators are fully connected, the synchronization costs among them are ignored in simulations.

Table I summarizes the list of the test configurations considered in the simulation experiments and the initial & final load distribution and the standard deviations of the corresponding load partitions. In each test scenario the locations of the sensors and the initial locations of mobile actuators are generated randomly at the start of the simulation. As can be observed from Table I, the initial random deployments of actuators results in load distributions with large differences in the service load among the different partitions. This load imbalance is expressed quantitatively by the standard deviations of the loads carried by the different actuators. As the actuator control algorithm iteratively proceeds the load imbalance gradually reduces. The algorithm continues running until a termination condition, specified by either a minimum standard deviation or a maximum number of iterations without appreciable improvements, is satisfied. The standard deviation of the final load distribution indicates how closely the loads between the different partitions have been balanced. The results clearly show that our actuator control algorithm significantly improves the balancing of the load distribution among multiple actuators.

VII. CONCLUSION

In several real world WSAN application scenarios, where access to the deployment area is very limited to humans, automatic service load partitioning techniques are essential to guarantee longer service lifetime, efficient operation and improved response time. This technical note presented complete control strategies for load balancing in distributed sensor networks equipped with mobile actuators to provide logistic and other services. Future work will closely study the trade-offs between minimizing load imbalances and minimizing total service load

in the entire network. Since a minimum load imbalance does not necessarily lead to a minimum total service load, this optimization goal requires some compromise between the two minimization criteria.

REFERENCES

[1] I. F. Akyildiz and I. H. Kasimoglu, "Wireless sensor and actor networks," *Ad Hoc Networks J.*, vol. 2, pp. 351–367, 2004.

[2] A. Nayak and I. Stojmenovic, *Wireless Sensor and Actuator Networks: Algorithms and Protocols for Scalable Coordination and Data Communication*. New York: Wiley, 2010.

[3] Y. Mei, Y.-H. Lu, Y. C. Hu, and C. S. G. Lee, "Reducing the number of mobile sensors for coverage tasks," in *Proc. IEEE/RSJ IROS*, 2005, pp. 1426–1431.

[4] Y. Paschalidis and P. Pennesi, "A distributed actor-critic algorithm and applications to mobile sensor network coordination problems," *IEEE Trans. Autom. Control*, vol. 52, no. 2, pp. 492–497, Feb. 2010.

[5] P. Corke, R. Peterson, and D. Rus, "Localization and navigation assisted by networked cooperating sensors and robots," *Int. J. Robot. Res.*, vol. 24, no. 9, pp. 771–786, 2005.

[6] A. Panousopoulou, E. Kolyvas, Y. Koveos, A. Tzes, and J. Lygeros, "Robot-assisted target localization using a cooperative scheme relying on wireless sensor networks," in *Proc. IEEE CDC*, 2006, pp. 1031–1036.

[7] X. Li, N. Santoro, and I. Stojmenovic, "Localized distance-sensitive service discovery in wireless sensor and actor networks," *IEEE Trans. Computers*, vol. 58, no. 9, pp. 1275–1288, Sep. 2009.

[8] Y. Mei, C. Xian, S. Das, Y. C. Hu, and Y.-H. Lu, "Replacing failed sensor nodes by mobile robots," in *Proc. Workshop Wireless Ad Hoc Sensor Networks (WWASN)*, 2006, p. 87.

[9] R. C. Shah, S. Roy, S. Jain, and W. Brunette, "Data mules: Modeling a three-tier architecture for sparse sensor networks," *Ad Hoc Netw. J.*, vol. 1, no. 2–3, pp. 215–233, 2003.

[10] W. Zhao, M. Ammar, and E. Zegura, "Controlling the mobility of multiple data transport ferries in a delay-tolerant network," in *Proc. IEEE INFOCOM*, 2005, pp. 1407–1418.

[11] R. Sugihara and R. K. Gupta, "Optimal speed control of mobile node for data collection in sensor networks," *IEEE Trans. Mobile Computing*, vol. 9, no. 1, pp. 127–139, Jan. 2010.

[12] G. Xing, T. Wang, W. Jia, and M. Li, "Rendezvous design algorithms for wireless sensor networks with a mobile base station," in *Proc. ACM MobiHoc*, 2008, pp. 231–240.

[13] M. Ma and Y. Yang, "Data gathering in wireless sensor networks with mobile collectors," in *Proc. IEEE IPDPS*, Miami, FL, Apr. 2008, pp. 1–9.

[14] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA: MIT Press, 2001.

[15] S. Nesamony, M. K. Vairamuthu, M. E. Orlowska, and S. W. Sadiq, "On Optimal Route Computation of Mobile Sink in a Wireless Sensor Network," ITEE, Univ. Queensland, Brisbane, Australia, Tech. Rep. 465, 2006.

[16] A. Dumitrescu and J. S. B. Mitchell, "Approximation algorithms for TSP with neighborhoods in the plane," in *Proc. ACM-SIAM SODA*, 2001, pp. 38–46.

[17] F. Aurenhammer, "Voronoi diagrams—A survey of a fundamental geometric data structure," *ACM Comput. Surv.*, vol. 23, no. 3, pp. 345–405, 1991.

[18] J. Kirk, Traveling Salesman Problem—Genetic Algorithm [Online]. Available: http://www.mathworks.fr/matlabcentral/fileexchange/13680

Video Surveillance Over Wireless Sensor and Actuator Networks Using Active Cameras

Dalei Wu, Song Ci, *Senior Member, IEEE*, Haiyan Luo, *Student Member, IEEE*, Yun Ye, and Haohong Wang, *Member, IEEE*

***Abstract*—Although there has been much work focused on the camera control issue on keeping tracking a target of interest, few has been done on jointly considering the video coding, video transmission, and camera control for effective and efficient video surveillance over wireless sensor and actuator networks (WSAN). In this work, we propose a framework for real-time video surveillance with pan-tilt cameras where the video coding and transmission as well as the automated camera control are jointly optimized by taking into account the surveillance video quality requirement and the resource constraint of WSANs. The main contributions of this work are: i) an automated camera control method is developed for moving target tracking based on the received surveillance video clip in consideration of the impact of video transmission delay on camera control decision making; ii) a content-aware video coding and transmission scheme is investigated to save network node resource and maximize the received video quality under the delay constraint of moving target monitoring. Both theoretical and experimental results demonstrate the superior performance of the proposed optimization framework over existing systems.**

***Index Terms*—Camera control, content-aware video coding and transmission, video tracking, wireless sensor networks.**

I. INTRODUCTION

Recently, as one type of the most popular wireless sensor and actuator networks (WSANs) [1], [2], wireless video sensor networks with active cameras have attracted a lot of research attentions and been adopted for various applications, such as intelligent transportation, environmental monitoring, homeland security, construction site monitoring, and public safety. Although significant amount of work has been done on wireless video surveillance in literature, major challenges still exist in transmitting videos over WSANs and automatically controlling cameras due to the fundamental limits of WSANs, such as, limitations on computation, memory, battery life, and network bandwidth at sensors, as well limitations on actuating speed, delay, and range at actuators.

Some work has been focused on automated camera control for video surveillance applications. In [3], an algorithm was proposed to provide automated control of a pan-tilt camera by using the captured visual information only to follow a person's face and keep the face image centered in the camera view. In [4], an image-based pan-tilt camera control method was proposed for automated surveillance systems with multiple cameras. The work in [5] focused on the control of a set of pan-tilt-zoom (PTZ) cameras for acquiring closeup views of subjects

Manuscript received April 03, 2010; revised March 18, 2011; accepted July 09, 2011. Date of publication August 08, 2011; date of current version October 05, 2011. This work was supported in part by the National Science Foundation (NSF) under Grant CCF-0830493. Recommended by Associate Editor I. Stojmenovic.

D. Wu, S. Ci, and Y. Ye are with the Department of Computer and Electronics Engineering, University of Nebraska-Lincoln, Omaha, NE 68182 USA (e-mail: dwu@unlnotes.unl.edu; sci@engr.unl.edu; yye@huskers.unl.edu).

H. Luo is with Cisco Systems, Austin, TX 79729 USA (e-mail: haiyan.luo@huskers.unl.edu).

H. Wang is with TCL Corporation, Santa Clara, CA 95054 USA (e-mail: haohong@ieee.org).

Color versions of one or more of the figures in this technical note are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TAC.2011.2164034