Optimal Sensor Scheduling Algorithms for Distributed Sensor Networks

Behrooz Safarinejadian*
Electrical and Electronics Engineering Department, Shiraz University of Technology, Shiraz, Iran
safarinejad@sutech.ac.ir
Abdolah Rahimi
Electrical and Electronics Engineering Department, Shiraz University of Technology, Shiraz, Iran
ab.rahimi@sutech.ac.ir

Received: 25/Jul/2013 Accepted: 31/Aug/2013

Abstract
In this paper, a sensor network is used to estimate the dynamic states of a system. At each time step, one (or multiple) sensors are available that can send its measured data to a central node, in which all of processing is done. We want to provide an optimal algorithm for scheduling sensor selection at every time step. Our goal is to select the appropriate sensor to reduce computations, optimize the energy consumption and enhance the network lifetime. To achieve this goal, we must reduce the error covariance. Three algorithms are used in this work: sliding window, thresholding and randomly chosen algorithms. Moreover, we will offer a new algorithm based on circular selection. Finally, a novel algorithm for selecting multiple sensors is proposed. Performance of the proposed algorithms is illustrated with numerical examples.

Keywords: Sensor scheduling, Sub-optimal algorithm, Offline optimization, Error covariance.

1. Introduction

In recent decades there has been much interest in using sensor networks to improve estimation process [1]. Nowadays; many projects have been defined based on sensor networks. Works such as the EYES project [2], WINS [3] and Smart Dust [4] are examples of systems implementing such networks. Such networks have a complex implementation. Wireless sensor networks have the potential to improve the estimation process. It is obvious that estimates obtained using several sensors will be better than estimates obtained from a single sensor. Furthermore, many systems are constructed based on using these networks. In some sensor networks, there are some problems in order to use data from multiple sensors at the same time. The main issue is selecting a sensor from multi sensors. The sensor management issues will be raised when we have problems in communication protocols or hardware sensor networks. Selection of the most appropriate sensor for optimal performance based on sensor capabilities and network characteristics is an important aspect of the problem. The choice of sensors is calculated off line, and then will be used by the system.

Sensor scheduling problems are used when one (or more) sensors have to be selected in N given sensors at every time step. This might be the case if there are echo-based sensors like sonars [5,6]. If the sensors observe a schedule and thus minimize simultaneous measurements, the total sensor power consumption can be reduced. Another situation where sensor scheduling is useful is in tracking problems, where radar can make different types of measurements by transmitting a suitable waveform each of which has a different power requirement. There might be shared communication resources (e.g., broadcast channels or a shared communication bus) that constrain the usage of many sensors at the same time. Such a situation arises in telemetry-data aerospace systems.

The systems that use Bluetooth technology can communicate with a single system at any moment [7]. In such cases, the main issue is scheduling of sensors to minimize the error covariance, or cost function. In this paper, we consider the optimal scheduling of sensors. In such a way that at each time step, one (or multiple) sensors are allowed to send their data. That can be caused by several problems such as communication problems, system hardware limitations and energy problems in the system.

There are several techniques for optimal scheduling of sensors in the literature. If we want to probe all options to choose the best arrangement of the sensors, we will be faced with a lot of computation. In order to avoid such a huge computation, sub-optimal algorithms can

* Corresponding Author

be used. For example, sliding window and thresholding algorithms with tree structure are proposed. Another method that can be used is choosing the sensors randomly according to some optimal probability distribution. In this paper, we will present a new algorithm which has less computational burden than other algorithms. The algorithm is based on ring selection and can be used as a sub-optimal method. In this algorithm we move on a circular path on which all off the sensor are located, and considering the covariance of the error, the sub-optimal sensor will be chosen.

In more cases, by selecting one sensor at each time step, the system accuracy will not be as good as expected; therefore, multiple sensors should be selected. When the number of sensors increases, high accuracy will be achieved, but more complex hardware and energy will be necessary too. In this paper, an algorithm will also be proposed in order to select multiple sensors at each time step.

The rest of the paper is organized as follows. In section 2, the problem formulation will be presented. In section 3, the question of choosing the optimal sensor schedule is considered and new algorithms will be proposed. In section 3, we will compare the performance of these algorithms using some examples. Finally, Section 4 concludes the paper.

2. MODELING AND PROBLEM FORMULATION

Suppose we have a linear discrete-time system given as follows [8,9,10]:

$$x[k+1] = Ax[k] + Bw[k] \quad (1)$$

$$y_i[k] = C_i x[k] + v_i[k] \quad (2)$$

Where $x[k] \in R^n$ represents the state and $y[k] \in R^m$ is the measurement vector. We have N sensors for state estimation so that $y_i[k]$ is output of i-th sensor in step k. It is assumed that $w[k]$, $v_i[k]$, and $x[0]$ are independent Gaussian random vectors and $w[k] \square N(0,Q)$, $v[k] \square N(0,R)$ and $x[0] \square N(0,\Pi)$, are positive definite matrices where $Q,R,\Pi > 0$. It is assumed that only one sensor can be used at any time. Note that each sensor can communicate with the fusion center in an error-free manner. We use the one-step Kaman filter to estimate the process states. Kalman filter equations are as follows:

$$\hat{x}[k+1] = A\hat{x}[k] + K[k](y_i[k] - C_i\hat{x}[k]) \quad (3)$$

$$k[k] = AP[k]C_i^T (C_i P[k] C_i^T + R_i)^{-1} \quad (4)$$

$$P[k+1] = (A - K[k]C_i)P[k](A - K[k]C_i)^T + BQB^T + K[k]R_i[k]K[k]^T \quad (5)$$

The optimal estimation is given by a Kalman filter assuming a time-varying measurement equation. Assuming that the i-th sensor takes the measurement at time step k, the covariance of the estimation error P[k] evolves according to the Riccati recursion:

$$p[k+1] = Ap[k]A^T + BQB^T - Ap[k]C_i^T (C_i p[k] C_i^T + R_i)^{-1} C_i p[k] A^T \quad (6)$$

In the above relations, $p[k]$ is the error covariance and $x[k]$ is the estimated state at time step k. In the next section, $p[k]$ will be considered as the cost function.

Now, we assume that we want to select m sensors throughout n sensors in a sensor network at each time step $(n \geq m)$ [11,12]. The following relations can be used for data fusion in a central node:

$part\,1(time\,update)$:

$$p^-[k+1] = A\,p^+[k]\,A^T + B\,Q\,B^T \quad (7)$$
$$x^-[k+1] = A\,x^+[k]$$

$part\,2(measurment\,update)$

$$K_i[k] = p^-[k+1]C_i^T (C_i\, p^-[k+1]C_i^T + R_i)^{-1}$$
$$x_i^+[k+1] = x^-[k+1] + k_i[k](y_i[k] - C_i\, x^-[k+1]) \quad (8)$$
$$P_i^+[k+1] = (I - k_i[k]C_i)\, p^-[k+1](I - k_i[k]C_i)^T + k_i[k]R_i\, k_i^T[k]$$

Each iteration of the Kalman filter consists of one iteration of time update equations and m iterations of measurement update equations, where m is number of selected sensors.

2.1 Optimization of Sensor Scheduling

In the analysis presented so far, it is assumed that scheduling of the sensors is available.

It is natural that the minimum error covariance that can be achieved is a function of the sensor schedule. We will try to find a sensor schedule in order to minimize the covariance matrix P[k], over a special time horizon.

In order to simplify the problem, we assume that we have only three sensors and the cost function is defined to be the sum of the error covariance matrices for the three sensors as follows [5, 6]:

$$J = \sum_{k=0}^{N} trace(p_1[k] + p_2[k] + p_3[k]) \quad (9)$$

At any time step, only one sensor is allowed to send its data to the central node. It is assumed that the system begins at time $k=0$ and goes on until $k=N$.

Generally, the covariance can be variously weighed in cost function, because there are some sensors which are more important than others. All possible schedules for the sensors can be seen in Figure 1 for the case of three sensors.

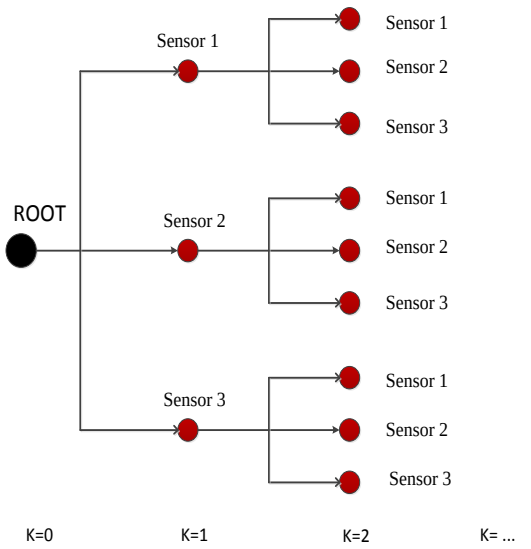

Figure 1. The tree structure defined by the various possible choices of sensor schedules illustrated for the case of 3 sensors.

At each time step, a branch of the tree will be selected according to the previous selected branch. If we want to check all possible options in order to find an optimal scheduling, we may face problems such as large computational burden and lack of memory. For example, in a schedule with only three sensors for N time step, we have 3^N selections. It should be noted that if x is the number of available sensors, to obtain an optimal scheduling, x^N different choices are available. So heavy computations are required to find the optimal sensor scheduling; therefore, we may use a sub-optimal method. In the following, four sub-optimal methods will be proposed for sensor selection.

If we want to use multiple sensors at each time step by using equation 8, the cost function will be as follows:

$$J = \sum_{k=1}^{N} trace(p_1^+[k] + p_2^+[k] + p_3^+[k]) \quad (10)$$

The above cost function can also be used in cases that one sensor is selected at each time step.

2.2 Sliding Window Algorithm

This algorithm is similar to a pseudo real time version of the Viterbi algorithm [13]. A window size d is defined where $d<N$. The algorithm proceeds as follows:

1) Initialization: Start from root node at $k = 0$
2) Traversal:
   a) Traverse all the possible paths in the tree for the next d levels from the current node.
   b) Find the sensor sequence $S_k, S_{k+1}, ..., S_{k+d+1}$ that gives the minimum cost at the end of this window of size d.
   c) Choose the first sensor S_k from the sequence.
3) Sliding the Window:
   a) If $k=N$ then quit, else go to the next step.
   b) Designate the sensor S_k as the root.
   c) Update time $k=k+1$.
   d) Repeat the traversal step.

It should be noted that the choice of the parameter d is arbitrary.

2.3 Thresholding

This algorithm is similar to the algorithm presented in [14], in the context of choosing the optimal controller from a set of many possible choices. We describe a factor f where $f \geq 1$. The algorithm steps are as follows:

1) Initialization: Start from root node with cost $J=0$.
2) Pruning:
   a) Extend the tree by one level (i.e. time step) through all possible paths from the current node.
   b) Calculate the minimum cost up to that time step.
   c) Remove the branches that have a cost greater than f times the minimum.
   d) For the remaining branches, denote the cost of the nodes as the cost achieved by moving down the tree till the node.

2.4 Randomly Chosen Sensors

In this algorithm, at each time step, the sensors will be chosen randomly based on some probability distribution. Then the probability distribution is chosen so as to minimize the expected steady state error covariance. Note that we can't compute the accurate value of the error covariance since it will depend on the specific sensor schedule chosen.

2.5 Circular selection algorithm

This algorithm is based on a circular selection; all sensors are placed on a ring. The computational burden of algorithm is less than

other algorithms because in each time step, it is possible to select more than one sensor.

Sensor selection order of the proposed algorithm is shown in Figure 2.

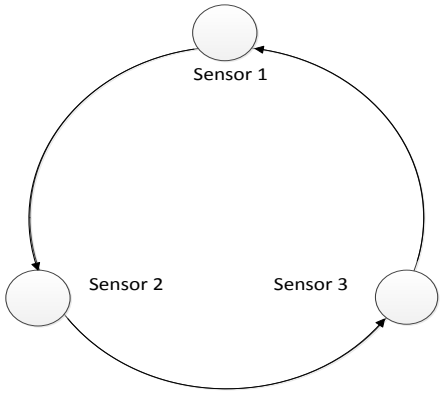

Figure 2. Sensor selection in the circular selection algorithm.

Proposed algorithm

1) Initialization: Start with the cost function $J > 0$ and a time step $k = 0$.
2) Calculation :
   a) Calculate the cost function of the first sensor. If the obtained cost function is less than J then it will be substituted by the new cost function, and the first sensor is chosen as the optimal sensor k. Then, $k = k + 1$ and number of the optimal sensor is stored in a vector z.
   b) Repeat the previous step for the next sensors.
3) If in above steps the optimal sensor isn't selected then the sensor with the smallest cost function is selected as optimal sensor between the sensors. J is substituted by cost function of the optimal sensor, and number of the optimal sensor is stored in z, and $k=k+1$.
4) if k ≥ N stop the iteration, else go to 2.

2.6 A new sensor scheduling algorithm for selecting multiple sensors at each time step

The main purpose of this sub-section is presenting a new algorithm that is able to select one or more than one sensor at each time step. Thus, the following algorithm is suggested. There are n sensors in the network which we want to select m sensors at each time step. The algorithm steps are as follows:

1) Initialization: Start from root node at $k=0$.
2) Traversal:
   a) Traverse all the possible paths in the tree for the next d levels from the current node.
   b) Find m string of sensors which have the least cost function and select them as optimal sensors at k'th time step.
3)
   a) a) If k=N then the algorithm is finished, otherwise go to the next .
   b) b) k = k+1
   c) c) go to the step 2.

3. Simulation Results

3.1 Simulation 1

In this part, we apply our algorithm to an application presented in [5] in which we have a system that track motion of an automobile in two dimensions x and y. Several sensors are located on the car that send information about position of automobile to the central node. In the central node, automobile velocity will be estimated. Equations of system are

$$x[k+1] = Ax[k] + Bw[k]$$

$$y_i[k] = C_i x[k] + v_i[k]$$

where

$$A=\begin{bmatrix} 1 & 0 & h & 0 \\ 0 & 1 & 0 & h \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, B=\begin{bmatrix} \frac{h^2}{2} & 0 \\ 0 & \frac{h^2}{2} \\ h & 0 \\ 0 & h \end{bmatrix}, X=\begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix}$$

where h=0.2 and $w[k]$ in equation (1) is a zero mean white Gaussian noise. The process noise is assumed to have covariance matrix Q given by:

$$Q=\begin{bmatrix} 1 & 0.25 \\ 0.25 & 1 \end{bmatrix}$$

We assume measurements taken by the three sensors being described by:

$$y_i[k] = C_i X[k] + v_i[k]$$

In this simulation, the matrix C_i will be constant as below

$$C_i=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

The terms $v_i[k]$ model the measurement noise, again assumed white, zero mean and Gaussian and also independent from each other and from w[k].We consider values of the sensor noise covariance's as:

$$R_1=\begin{bmatrix} 1.62 & 0 \\ 0 & 0.57 \end{bmatrix}, R_2=\begin{bmatrix} 0.59 & 0 \\ 0 & 1.65 \end{bmatrix}, R_3=\begin{bmatrix} 1.21 & 0 \\ 0 & 0.64 \end{bmatrix}$$

In the following figures, performance of the proposed algorithm and other algorithms can be observed:

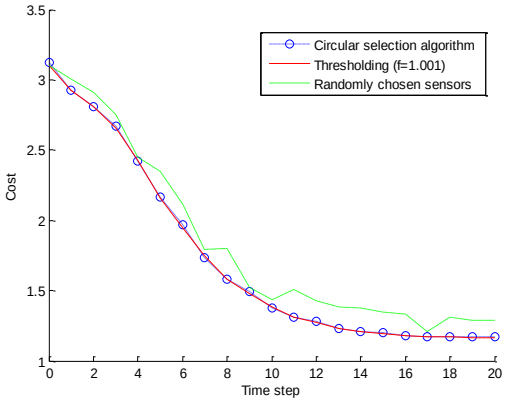

Figure 3: Comparison between circular selection, thresholding, and randomly chosen algorithms.

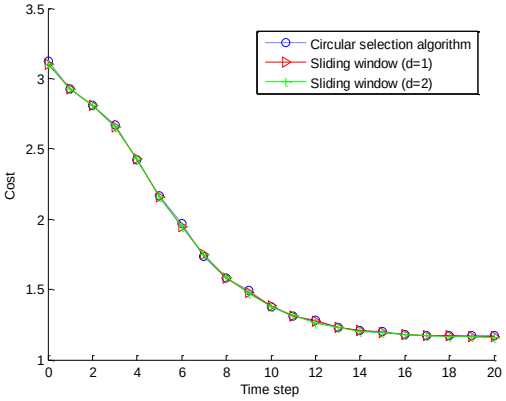

Figure 4: Comparison between circular selection, sliding window (d=1), and sliding window (d=2) algorithms.

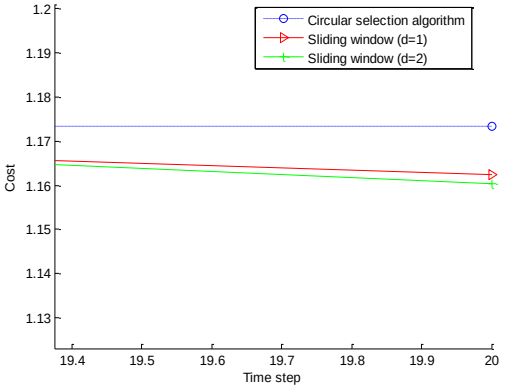

Figure 5: Comparison between circular selection algorithm, sliding window (d=1), and sliding window (d=2) at the final steps.

In Table I, performances of different algorithms are compared.

TABLE I: comparison of different algorithms.

	Sliding window $d=1$	Sliding window $d=2$	Threshold $f=1.001$	Randomly chosen sensors	Circular selection algorithm
Final cost function value	1.1624	1.1604	1.1614	1.3010	1.1735
elapsed time	0.005938	0.018012	0.009537	0.007952	0.005344

3.2 Simulation 2

Six sensors have been considered in this simulation.

$y_i[k] = C_i X[k] + v_i[k]$

In this simulation the matrix C_i will be constant as below

$$C_i = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

The previous model parameters have been used despite of the sensor noise covariance which is different in this case:

$$R_1 = \begin{bmatrix} 1.75 & 0 \\ 0 & 0.55 \end{bmatrix}, R_2 = \begin{bmatrix} 0.75 & 0 \\ 0 & 1.36 \end{bmatrix}, R_3 = \begin{bmatrix} 1.74 & 0 \\ 0 & 0.7 \end{bmatrix}$$

$$R_4 = \begin{bmatrix} 1.7 & 0 \\ 0 & 0.6 \end{bmatrix}, R_5 = \begin{bmatrix} 1.95 & 0 \\ 0 & 0.43 \end{bmatrix}, R_6 = \begin{bmatrix} 1.6 & 0 \\ 0 & 0.8 \end{bmatrix}$$

We compare the results of the mentioned algorithms:

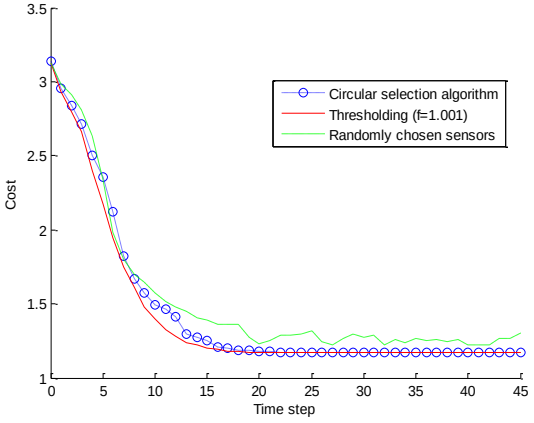

Figure. 6: Comparison between circular selection, thresholding, and randomly chosen algorithms.

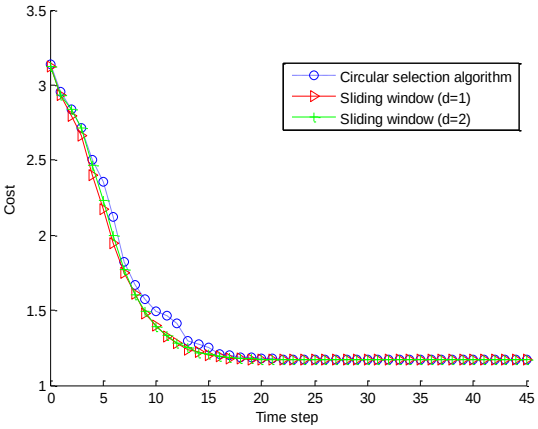

Figure. 7: Comparison between circular selection algorithm, sliding window (d=1), and sliding window (d=2).

In table II, the results of these algorithms can be viewed.

TABLE II: comparison of different algorithms

	Sliding window $d=1$	Sliding window $d=2$	Threshold $f=1$	Randomly chosen sensors	Proposed algorithm
Final cost function value	1.1686	1.1682	1.1686	1. 2323	1.1708
elapsed time	0.01455	0.1591	0.0293	0.0091	0.0085

From Tables I and II, it is clear that the circular selection algorithm has a good performance in terms of cost function reduction. Cost function graphs and the final cost function values of the sliding window, thresholding, and the circular selection algorithms are almost identical. However, in randomly chosen algorithm, the cost function value is higher than the other algorithms.

But in terms of reducing the computational burden, the circular selection algorithm has the best performance, even better than the randomly chosen algorithm. From Tables I and II, we can see that increasing number of sensors results in a better performance for the circular selection algorithm in terms of reduction of computation.

3.3 Simulation 3

In the following, the new algorithm is used for selecting multiple sensors assuming that there are four sensors in the network.

In this simulation, the matrix C_i will be as below

$$C_1=\begin{bmatrix}0&0&0&0\\0&1&0&0\end{bmatrix}, C_2=\begin{bmatrix}0&0&0&0\\0&1&0&0\end{bmatrix}$$

$$C_3=\begin{bmatrix}1&0&0&0\\0&1&0&0\end{bmatrix}, C_4=\begin{bmatrix}1&0&0&0\\0&0&0&0\end{bmatrix}$$

Measurement noise covariance's are as follows:

$$R_1=\begin{bmatrix}1.63&0\\0&0.55\end{bmatrix}, R_2=\begin{bmatrix}0.75&0\\0&1.37\end{bmatrix}$$

$$R_3=\begin{bmatrix}1.44&0\\0&0.6\end{bmatrix}, R_4=\begin{bmatrix}1.7&0\\0&0.57\end{bmatrix}$$

It is assumed that at each time step, two sensors are selected. Results are shown in Figure 8.

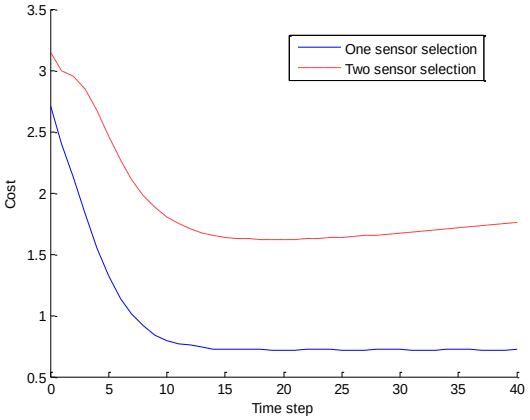

Figure 8: Comparison between proposed algorithm with two sensor selection in each time step and sliding window (d=1).

It is clear from Figure 8 that the proposed algorithm has better performance but it should be noted that using the proposed algorithm needs a more complex hardware.

3.4 Simulation 4

In this part, the matrix C is assumed to be constant and there are twelve sensors with the following covariance matrices

$$R_1=\begin{bmatrix}1.63&0\\0&1.55\end{bmatrix}, R_2=\begin{bmatrix}1.59&0\\0&1.69\end{bmatrix}, R_3=\begin{bmatrix}1.61&0\\0&1.59\end{bmatrix}$$

$$R_4=\begin{bmatrix}1.61&0\\0&1.57\end{bmatrix}, R_5=\begin{bmatrix}1.69&0\\0&1.53\end{bmatrix}, R_6=\begin{bmatrix}1.63&0\\0&1.51\end{bmatrix}$$

$$R_7=\begin{bmatrix}1.64&0\\0&1.55\end{bmatrix}, R_8=\begin{bmatrix}0.59&0\\0&2.75\end{bmatrix}, R_9=\begin{bmatrix}1.68&0\\0&1.58\end{bmatrix}$$

$$R_{10}=\begin{bmatrix}1.44&0\\0&1.6\end{bmatrix}, R_{11}=\begin{bmatrix}1.65&0\\0&1.55\end{bmatrix}, R_{12}=\begin{bmatrix}1.65&0\\0&1.53\end{bmatrix}$$

Using the proposed method for multi-sensor selection, one, two, three, and four optimal sensors are selected between twelve sensors.

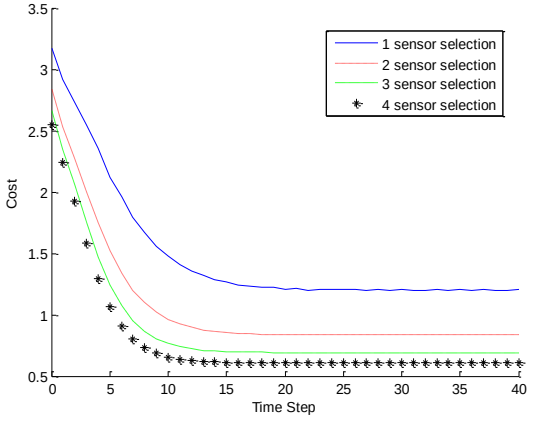

Figure 9: comparison between selection of one, two, three, and four sensors from twelve sensors.

4. Conclusions

It has been clarified that the circular selection algorithm was faster than the other algorithms such as sliding window and thresholding and it had a suitable performance in terms of finding optimum sensors. In other words, this algorithm reduces the computations while its performance in finding the optimal sensors is as good as other scheduling algorithms. According to the obtained results, it is possible to use the proposed algorithm instead of sliding window and thresholding algorithms.

In the case of selecting multiple sensors at each time step, it can be seen that the performance of the system has been improved but hardware problems will be greater. Number of selected sensors at each time step is related to system's expected accuracy and cost of the system.

References

[1] S. Roumeliotis and G. Bekey, "Distributed multi-robot localization," *IEEE Transactions on* Robotics and Automation, vol. 18, no. 5, pp. 781–795,Oct 2002.

[2] H. Karl, "Making sensor networks useful: Distributed services - the eyes project," *ESF Workshop*, La Spezia, Italy, 2002.

[3] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next CenturyChallenges: Scalable Coordination in Sensor Networks," *in Proceeding of the Fifth Annual International Conference on Mobile Computing and Networking*, Seattle, WA, USA, August 15-19, 1999.

[4] J. Kahn, R. Katz, and K. Pister, "Next Century Challenges: Mobile Networking for 'Smart Dust'," *in Proceeding of the fifth annual ACM/IEEE international conference on Mobile computing and networking*, Seattle, WA, USA, August 15-19, 1999.

[5] V. Gupta, T. H. Chung, B. Hassibi, and R. M. Murray, "On a Stochastic Sensor Selection Algorithm with Applications in Sensor Scheduling and Sensor Coverage," *Automatica*, vol. 42, no. 2, pp. 251-260, 2006.

[6] V. Gupta, T. Chung, B. Hassibi, and R. M. Murray. "Sensor scheduling algorithms requiring limited computation," *In Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, volume 3, 825-828, May 2004.

[7] J. Haartsen. "Bluetooth-the universal radio interface for ad hoc, wireless connectivity," *Ericsson Review*, vol. 3, pp. 110-117, 1998.

[8] T. Kailath, A. Sayed, and B. Hassibi, "*Linear Estimation*". Prentice-Hall,2000.

[9] V. Gupta, T. Chung, B. Hassibi and R. M. Murray, "Sensor scheduling algorithms requiring limited computation," *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Pasadena, CA, USA,2004.

[10] B. Safarinejadian, A. Rahimi, M. Mozaffari, "A new sensor scheduling method for distributed sensor network," *The 21st Iranian Conference on Electrical Engineering (ICEE)*, Mashhad, Iran, 2013.

[11] M. R. R. Khan and V. Tuzlukov, "Multisensor data fusion algorithms for estimation of a walking person position," *International Conference on Control Automation and Systems (ICCAS)*, pp. 863-867, 2010.

[12] D. Hall, "The Implementation of Data Fusion Systems," *Multisensor Fusion, Springer*, Vol. 70, pp. 419-433, 2002.

[13] J. G. D. Forney, "The Viterbi algorithm," *Proceedings of the IEEE*, Volume.61 , pp. 268–278, January 1973.

[14] B. Lincoln and B. Bernhardsson, "LQR optimization of linear system switching," *IEEE Transaction on Automatic Control*, vol. 47, pp. 1701-1705, 2002.

Behrooz Safarinejadian received his BS and MS degrees from the Electrical Engineering Department, Shiraz University, Shiraz, Iran, in 2002 and 2005, respectively. He received his Ph.D. degree from the Electrical Engineering Department, Amirkabir University of Technology, Tehran, Iran, in 2009. Since 2009, he has been with the Faculty of Electrical and Electronic Engineering, Shiraz University of Technology, Shiraz, Iran. His research interests include distributed sensor networks, estimation theory, statistical signal processing, computational intelligence, control systems theory and fault detection.

Abdolah Rahimi received his B.Sc. degree in telecommunication engineering from Azad University, Tehran, Iran, in 2006, and the M.Sc. degree in control engineering from Shiraz University of Technology, Shiraz, Iran, in 2013. His research interests include sensor scheduling, system identification and chaotic systems.