Collaborative Distributed Data Fusion Architecture Using Multi-Level Markov Decision Processes

D. Akselrod, A. Sinha and T. Kirubarajan
Electrical & Computer Engineering Dept.
McMaster university
Hamilton, ON L8S 4K1, Canada
Email: adimitry@grads.ece.mcmaster.ca,
abhijit@mail.ece.mcmaster.ca,
kiruba@mcmaster.ca

Abstract—Decentralized multisensor-multitarget tracking has numerous advantages over single-sensor or single-platform tracking. In this paper, we present a solution to one of the main problems of decentralized tracking, namely, distributed information transfer and fusion among the participating platforms. This paper presents a hierarchical multi-level decision mechanism for collaborative distributed data fusion that provides each platform with the required data for the fusion process while substantially reducing redundancy in the information flow in the overall system. We consider a distributed data fusion system consisting of platforms that are decentralized, heterogeneous, and potentially unreliable. The proposed approach, which is based on hierarchical Markov decision processes and decentralized lookup substrate, will control the information exchange and data fusion process based, among the other parameters, on maximizing performance metrics of individual platforms, thereby enhancing the whole distributed system’s reliability as well as that of each participating platform. Simulation examples demonstrate the operation and the performance results of the system.

Keywords: Distributed data fusion, communication management, information flow control, Markov decision process, multisensor-multitarget tracking

I. INTRODUCTION

Distributed tracking has many advantages over single-sensor (or single-platform) architectures. One major advantage is that a more complete, accurate and timely common tracking picture that covers targets beyond the visibility limits of a given platform can be shared among multiple platforms. In distributed tracking systems individual sensors may be placed at considerable distances from each other. In this case, we consider the problem of transferring significant amounts of data through communication channels that are subject to certain capacity limits, associated costs of data transfers, reliability and security issues.

A number of multisensor tracking systems and data fusion techniques have been introduced in the literature [1]–[5]. Several practical distributed tracking algorithms, including distributed track fusion, track fusion using tracklets and distributed composite tracking have been identified [1]. What is common to many of the above is that the data from each platform, which may correspond to a different type of data fusion or data format, are passed to all the other platforms. The assumption that the available data channel capacity will suffice to carry all the required data will not hold in applications that feature a large number of platforms with high volumes of data to be exchanged among them (e.g., a network of airports). A number of works present solutions for distributed sensor networks applications and network-centric data fusion where limited communication capacity is considered. An information graph approach is introduced in [6]. The approach of using graphical models was also used in [7] to develop distributed fusion algorithms reducing communication. An agent based fusion model is presented in [8]. Issues associated with distributed multiple target tracking for ad hoc sensor networks are discussed in [9]. A decision fusion rule based on the total number of detections made by local sensors, for wireless sensor networks with a large number of sensors was proposed in [10]. In [11], distributed fusion and communication management algorithms for target identification were presented. Techniques to solve data association problems arising in distributed sensing scenarios were presented in [12]. A comparison of two different approaches for sensor selection for distributed tracking was presented in [13]. An approach for fusing information from diverse sources based on the quality of the source was presented in [14]. An algorithm for determining the quality of sensor data in the fusion process was presented in [15]. An information valuation metric for sensor networks was described in [16].

Our current work considers a hierarchical distributed data fusion system consisting of a number of platforms that are completely decentralized, independent and heterogenous. Each such data fusion platform contains a local tracker and sensors supplying data in form of measurements. We form a number of non-overlapping clusters with each cluster containing a number of platforms. The assignment of platforms to a specific cluster is done based on proximity criteria. Both the platforms within the same cluster and those belonging to different clusters are assumed unreliable, i.e., they may join or leave the network at any time, are not committed to share the information, and may have unreliable communication channels with limited capacity. When requesting specific information from such a platform we do not know if the requested information will be provided as we have only the statistical characteristics describing the ability of the platform to provide
such information. Data transfer from a platform in the system may be interrupted at any moment. We are considering a situation where the decisions regarding the data flow of the information in the distributed data fusion system should be made sequentially based on the currently observable state that reflects fully or partially the state of the environment. The result of each decision cannot be fully predicted, but can be accounted for using available statistical information before the next decision is made.

This paper presents a hierarchal decision mechanism that provides each platform within the same cluster as well as all the distinct clusters with the required data for the distributed data fusion process subject to the available channel capacities both within the clusters and externally; and reducing redundancy in the information flow in the overall system. The proposed approach, which is based on hierarchial multi-level Markov decision processes and decentralized lookup substrate (i.e., the way to identify efficiently all the platforms in the distributed network that possess the required information), will control the data fusion and information exchange process based, among the other parameters, on information gain metrics of individual platforms, enhancing the total distributed system’s reliability as well as that of each participating platform. The paper includes a complete solution for the distributed data fusion architecture timely providing participating platforms and clusters with specific decisions regarding obtaining information that is needed for the data fusion.

The rest of this paper is structured as follows. Section II presents a solution for controlling the data fusion process and information flow in distributed data fusion architectures for multisensor-multitarget tracking based on hierarchial Markov decision processes and decentralized lookup substrate. Section III expresses the component elements of the Markov decision process in terms of the parameters of the optimization problem that we are facing. Simulation results are presented in Section IV. Section V concludes the paper.

II. DISTRIBUTED DATA FUSION ARCHITECTURE

The current section presents a solution to controlling the information flow in distributed data fusion architectures.

We can distinguish between the two cases of platforms interchanging data over communication network. Fig. 1 and Fig. 2 show two cases of such platforms. In Fig. 1, the data from each platform should be passed to all the other platforms. Obviously, the same data coming from a certain platform are requested by other platforms as well, which causes redundancy and overloads the communication channels. Moreover, the task of providing tracking information from each node (platform) to every other node may be just infeasible taking into account large number of the nodes in the network and significantly high data rates as opposed to the limitations of the communication channels capacity. Fig. 2 demonstrates a different approach to the problem. We are considering a node that requests specific information originating from other nodes. The figure depicts a special case in which node3 requests information originating from node 1. The most straightforward (and obviously not the most optimal) solution would be requesting this information from the source - i.e. node1. But as can be seen from the figure, the information originating from node1 is also transmitted to nodes 2 and 2. It should be noted though that the information (e.g., tracks, tracklets or AMRs) transmitted from a node to other nodes may differ in its characteristics and quality. Therefore the required information may be obtained from neighboring nodes as well, as depicted in Fig. 2, thus eliminating redundancy in the transmitted information, unnecessary load on the communication channels, time overhead in getting the information, higher refusal probability, etc. If we consider node3’s request for data originating from node1, the decision to be made in this case is which of the platforms — 1, 2 or 4 — should supply the requested information. We consider two separate tasks - controlling the data fusion process within the same cluster and controlling data fusion process among different clusters. When considering a target, it is optimizing the internal inside-cluster data fusion process that will maximize the quality of tracking information available for that target. Fig. 3 demonstrates distribution of the fusion platforms to different clusters. It is the optimal data fusion process within a cluster that determines what the tracking information quality regarding that target will eventually be as it is observed by the sensors belonging to the cluster’s platforms. Another level of optimization is decision making regarding the fusion data transfer among different clusters.

A. Markov Decision Processes

A Markov Decision Process (MDP) is a stochastic process controlled by a decision maker. An infinite horizon fully observable MDP is defined by the model $M = \langle S, A, P, R \rangle$ where $S$ is the finite set of world states, $A$ is the finite set of actions, $P$ is the transition probability function, and $R$ is the real-valued reward function.

In an MDP, a policy $\pi$ is a mapping from states to actions. Then, $\Pi : S \rightarrow A$. Because of the Markov property, the action to take depends on the current state only, not on any of the previous states. The MDP model is solved using dynamic

Fig. 1. Distributed data fusion example. The data from each platform should be passed to all the other platforms.
programming approaches, which have been described in \cite{17, 18, 19} as well as in many other works.

**B. Dynamic Programming Solution Approach for Infinite Horizon Discounted MDP**

The MDP problem can be solved using the theory of dynamic programming. The basic problem of decision for the infinite horizon case \cite{20} can be expressed as follows. Given a stationary discrete-time process:

\[ s_{t+1} = f(s_t, a_t, w_t), \quad t = 0, 1, \ldots \]  

(1)

where \( s_t \) is the state, \( a_t \) is the control or action that depends only on the current state \( s_t \), and \( w_t \) is the random disturbance, the problem is to find a stationary policy \( \pi \) that maps each \( s_t \in S \) to \( a_t \in A \), which, given an initial state \( s_0 \), maximizes the following value function:

\[ V_\pi(s_0) = \lim_{N \to \infty} \mathbb{E} \left[ \sum_{k=0}^{N-1} \gamma^k R(s_t, a_t) \right] \]  

(2)

where \( \gamma \) is a discount factor, \( 0 < \gamma < 1 \). The random disturbance \( w_t \) is represented by the transition probability \( P \) in an MDP. Then, the optimal value function is defined as:

\[ V^*(s) = \max_{\pi \in \Pi} V_\pi(s), \quad s \in S \]  

(3)

where the policy \( \pi \) is optimal if \( V_\pi(s) = V^*(s) \) for all the states \( s \in S \). The optimal \( V^* \) is unique and is the solution of the following Bellman equation \cite{18, 19}:

\[ V^*(s) = \max_{a \in A} \left( R(s_t, a_t) + \gamma V^*(f(s_t, a_t, w_t)) \right), \quad \forall s \in S \]  

(4)

Discount factor \( \gamma \) expresses the dependence of the value function on current rewards over future rewards. Using the elements of MDP model defined in Section II-A, (4) can be expressed as follows:

\[ V^*(s) = \max_{a \in A} \left( R(s_t, a_t) + \gamma \sum_{s_{t+1} \in S} P(s_t, a_t, s_{t+1}) V^*(s_{t+1}) \right), \quad \forall s \in S \]  

(5)

Given the optimal value of the function, the optimal policy is:

\[ \pi^*(s) = \arg \max_{a \in A} \left( R(s_t, a_t) + \gamma \sum_{s_{t+1} \in S} P(s_t, a_t, s_{t+1}) V^*(s_{t+1}) \right) \]  

(6)

One of the existing techniques to find the optimal policy is a value iteration method which is based on the Bellman equation. The iteration can be stopped when the difference between two successive values of state \( |V(s_{t+1}) - V(s_t)| \) is less than a pre-defined error \( \varepsilon \) \cite{20}. According to \cite{21} for any state \( s \):

\[ V^*(x) - V(x) \leq \frac{2\varepsilon}{1 - \gamma} \]  

(7)

So, from the discount factor \( \gamma \) and the pre-defined error \( \varepsilon \) we can calculate the difference between two successive values of the state, which, when reached, will indicate that the algorithm can be stopped. Computational complexity of MDPs was analyzed in \cite{22} and it was shown that, under any of the three cost criteria \cite{22, 23}, namely, the expected cost to target, expected discounted cumulative cost, and average expected cost per stage, the problem is P-complete.

**C. Policy Update and Termination Criteria**

When a certain policy is calculated, it is assumed that during its execution the conditions and the state of the environment that contributed to the specific policy do not change significantly, except for the changes in the environment as a direct consequence of the actions that are taken. For example, transition probability \( P \) and reward \( R \) matrices are supposed to reflect the environment all the time during the policy execution. Also, it is assumed that the locations of all information sources have not been changed. Obviously, these assumptions do not hold, at least not for a long period of time, which means that the policy should be re-evaluated from time to time as shown in Fig. 4.
D. MDP Reward Structure as Expected Information Gain

In this work which is the extension of [24], the authors utilize the approach of using information based objective function, originally proposed and described in [25], [26]. The objective function is based on the Fisher information measure:

$$ J = \sum_j \log |I_j(k[k])| = \sum_j \log |P_j(k[k])^{-1}| $$  \hspace{1cm} (8)

where $P_j(k[k])$ is the posterior covariance matrix of the state vector corresponding to target $j$ at time $k$. It is expressed as follows:

$$ P_j(k[k])^{-1} = P_j(k[k-1])^{-1} + Y_j(k) $$  \hspace{1cm} (9)

where $P_j(k[k-1])^{-1}$ is the predicted state information (inverse of the state prediction covariance matrix) and $Y_j(k)$ is the new information that is given by:

$$ Y_j(k) = H(k, s, j)^T R(k, s, j)^{-1} H(k, s, j) $$  \hspace{1cm} (10)

where $H(k, s, j)$ is the measurement matrix and $R(k, s, j)$ is the measurement covariance matrix at the time step $k$ corresponding to the sensor $s$ from which the incoming AMR has originated and target $j$. Then, the expected updated information $I_j(k[k])$ can be expressed as follows:

$$ I_j(k[k]) = I_j(k[k-1]) + H(k, s, j)^T R(k, s, j)^{-1} H(k, s, j) $$  \hspace{1cm} (11)

The reward associated with a AMR arriving from a remote sensor $s$ after being successfully associated with one of the tracks of the platform is therefore expressed as the expected information gain corresponding to sensor from which the measurements originated. If there are $N_s$ AMR’s arriving from the same remote sensor $s$ which are associated with the tracks of the receiving platform, then the expected information gain is given by

$$ J_{N_s}(k, s) = \sum_{j=1}^{N_s(k)} \log |I_{s,j}(k[k])| - \log |I_j(k[k-1])| $$  \hspace{1cm} (12)

where $I_j(k[k-1])$ is the predicted information matrix and $I_{s,j}(k[k])$ is the updated information matrix corresponding to target $j$.

E. Data Lookup

When discussing the decision making process, we assumed that the information regarding the availability of the required information among the nodes is known. In fact, one of the fundamental problems that has to be addressed in order to apply the aforementioned decision process is in identifying all the nodes that possess the required information. One solution would be to use one of the available distributed lookup protocols to efficiently identify the nodes that store the desired data. For this purpose we define a space of the keys $K$. To each node in the distributed system assigned is a key $n_i \in K$. We use a hash function $h$ to map any particular source of information into the space $K$ as well. For example, each sensor will be identified by a corresponding key $s_i \in K$.

Fig. 5 demonstrates a node look-up process in one of the Distributed Hash Table (DHT) architectures called Chord [27]. The shortest available look-up time is $O(\log N)$, where $N$ is total number of the data sources in the distributed system.

III. MDP Based Multisensor Fusion for Multitarget Tracking

In this section, we will express the component elements of the Markov decision process in terms of the parameters of the optimization problem that we are facing. Here, the multisensor fusion problem for multitarget tracking for each level (either internal or inter-cluster) is mapped into a collection of corresponding MDP problems, each one being solved by the corresponding node. Each node will have a corresponding set of MDP parameters $S, A, P, R$ reflecting the optimization problem this node needs to solve.

A. Set of states: $S$

- **Original node**: This field specifies the node from which the sought information originates.
- **Supplying node**: This field specifies all the nodes that currently receive the information originating from the Original node.
Data available: This field indicates whether the requested data from one of the Supplying nodes is currently arriving and available.

Refusals: This field contains the total number of refusals from the corresponding Supplying node.

B. Set of actions: A

The set A contains all the possible actions that a node can take in order to specify the requested information sources in its next step of decision making. In our case, we have $n + 1$ different actions in the set, expressing a request for information from any of the $n$ nodes possessing the required information and an additional action of not requesting information at all.

C. Transition probabilities: $P$

The transition probability matrix specifies the probability $P(s_{t+1}|s_t, a_t)$ of transition to a specific state $s_{t+1}$, provided the transition is done from another state $s_t$ while performing a certain action $a_t$. The platform requesting data from other platforms has information regarding the holders of the required information as well as the knowledge of other circumstances that may influence successful reception of this information — for example, distributed network channels capacity, current load of the mentioned channels, the load of the nodes that have to supply information. Also, a specific node requesting information may have a priority rating index that may be different for various nodes.

D. Real-valued reward function on states: $R$

The vector $R$ contains the values of the immediate rewards associated to being in a certain state. We can express an element of $R$ as:

$$R(s) = \Theta r(s) - (1 - \Theta) c(s), \quad 0 \leq \Theta \leq 1 \quad (13)$$

where $r(s)$ is the revenue associated with being in state $s$ and $c(s)$ is the cost associated with it. Coefficient $\Theta$ balances between considerations of importance to reach the state $s$ as well as the revenue this will produce and the considerations of the cost this will incur.

In the above, $r(s)$ and $c(s)$ are expressed as

$$r(s) = \sum_{i=1}^{N} r^w_i(s)r^{re}_i(s) = R^w_iR^{re}_i,$$

$$r^w_1(s) + ... + r^w_N(s) = 1 \quad (14)$$

$$c(s) = \sum_{j=1}^{M} c^w_j(s)c^{re}_j(s) = C^w_jC^{re}_j,$$

$$c^w_1(s) + ... + c^w_M(s) = 1 \quad (15)$$

where $R^{re}_i$ and $C^{re}_i$ are vectors containing contributing elements of the state revenue and state cost, respectively. Vectors $R^w_i = [r^w_1(s), ..., r^w_N(s)]^T$ and $C^w_j = [c^w_1(s), ..., c^w_M(s)]^T$ contain weights, which control the influence of $R^{re}$ and $C^{re}$ elements on the revenue $r(s)$ and the cost $c(s)$, respectively.

IV. Simulation Results

In this section, we present and discuss the simulated scenarios and the results achieved using a number of performance metrics. The simulated system consists of three clusters depicted in Fig. 3.

Each of the clusters contains three tracking platforms, which in turn contain a different number of sensors each. Each of the clusters $n$ is located shifted in space by offsets of $x^a = [0, 0]'$, $[0, -1000]'$, $[5000, -5000]'$ km, respectively, $n = 1, 2, 3$.

The internal MDP process will be demonstrated on cluster 1 which contains 3 platforms and 5 sensors. Each sensor belongs to a specific platform associated with a corresponding tracker. Sensors 1 and 2 are connected to tracker 1, sensors 3 and 4 to tracker 2 and sensor 5 is connected to tracker 3. The sensors and platforms connectivity scheme of cluster 1 is shown in Fig. 6. The solid lines designate local communication channels between each of the sensors and the corresponding tracker. These communication channels are assumed dedicated and reliable due to relatively short distance between the sensors and the corresponding tracker. The dashed lines designate communication channels connecting different platforms (trackers) of the cluster. A platform requesting data from another platform is not guaranteed to receive them due to various reasons such as overloaded communication channels, low priority of the requesting platform, etc. The sensors are located at $x^s = [65, 155]'$, $[80, 140]'$, $[10, -65]'$, $[30, -40]'$, $[-80, 20]'$ km, respectively. The original measurements are obtained from the sensor $s$ in the form

$$[r(k, s, j) \ \Theta(k, s, j)]' \quad (16)$$

where $r(k, s, j)$ and $\Theta(k, s, j)$ are the range and the azimuth angle of the target $j$ supplied by sensor $s$ at time $t_k$ respectively. The measurement vector is assumed to contain independent additive Gaussian noise. The sensor $s$ range $r$ and bearing $\Theta$ standard deviations are $[\sigma^r_s, \sigma^\Theta_s] = [40, 2.5]'$, $[35, 2]'$, $[52.5, 3.5]'$, $[30, 2.5]'$, $[45, 4.5]'$ [m, mrad] respectively. The sampling intervals of the sensors are 2.5, 3.5, 2, 3, 1.5 [s] respectively. The false alarm is uniformly distributed in the coverage areas of the sensors with the number of false alarms Poisson distributed with means of 40, 100, 100, 50, 50 respectively. Two simulations included two close spaced targets. The scenario of the two target movements includes several constant velocity stages.
interleaved with coordinated turn maneuvers performed at rates \(|\omega| = 4 \, \text{[°/s]}\). The initial positions of the targets are \(\left[\xi_j, \eta_j\right] = [5, 68.6] \, \text{[km]}, \, [5, 69.1] \, \text{[km]}\) respectively and the initial targets velocity is \(300 \, \text{[m/s]}\). The Fig. 7 shows the coverage areas of the sensors and the targets trajectory. In the simulation, measurements associated with existing tracks, or Associated Measurement Reports (AMR’s), are transmitted between the platforms controlled by internal MDP processes. The transmitted measurements are naturally independent of the tracks of the corresponding platforms. The track maintenance is performed by a 2-D measurement to track association, performed by the auction algorithm [28], and track update is performed using a Kalman filter. A white noise acceleration model is assumed for the targets with process noise standard deviation \(\sigma_v\) of \(15 \, \text{m/s}^2\). The state of the target \(j\) is of the following form:

\[
x^j = [x^j, \dot{x}^j, y^j, \dot{y}^j]^T
\]

where \(x^j\) and \(y^j\) are \(x\) and \(y\) Cartesian coordinates of the target \(j\) and \(\dot{x}^j\) and \(\dot{y}^j\) its \(x\) and \(y\) velocity components respectively. We convert the original measurements obtained from the sensor \(s\) in the form

\[
[r(k, s, j), \theta(k, s, j)]^T
\]

where \(r(k, s, j)\), \(\theta(k, s, j)\) and \(\dot{r}(k, s, j)\) are the range and the azimuth angle of the target \(j\) supplied by sensor \(s\) at time \(t_k\) respectively, to the measurement vector of the following form,

\[
z^j = [r^j, \cos \Theta^j, \, r^j, \sin \Theta^j]^T
\]

using the standard coordinate conversion [29]:

The measurement covariance matrix \(R\) corresponding to the converted measurement is given by [29]:

\[
R_L = \begin{bmatrix} R_{11}^{11} & R_{12}^{12} \\ R_{21}^{12} & R_{22}^{12} \end{bmatrix}
\]

(21)

The elements of \(R_L\) are:

\[
R_{11}^{11} = r^2 \sigma^2_\Theta \sin^2 \Theta + \sigma^2_\theta \cos^2 \Theta
\]

(22)

\[
R_{12}^{12} = (\sigma^2_\theta - r^2 \sigma^2_\Theta) \sin \Theta \cos \Theta
\]

(23)

\[
R_{22}^{12} = r^2 \sigma_\Theta \cos^2 \Theta + \sigma^2_\theta \sin^2 \Theta
\]

(24)

In the simulation two point track initialization is applied.

Below we show the results resulted from applying the following three communication policies:

1) All the AMR’s are shared among all the platforms. In this case, the associated measurements obtained at one platform are transmitted to all other platforms within the same cluster.

2) No information is shared among the platforms.

3) The process of information (AMR’s) sharing at each platform is controlled by the dedicated MDP. To each state of the platform corresponds an action of requesting AMR from one or more sensors of the platform or not requesting information at all.

Fig. 8 and Fig. 9 show the position and velocity RMSE of \(\text{target}_1\) state estimation respectively for the 1st policy when all the AMR’s shared among the platforms. Fig. 10 and Fig. 10 show the position and velocity RMSE of \(\text{target}_1\) state estimation respectively for the 2nd policy when no information is shared among the platforms. Fig. 12 and Fig. 13 show the position and velocity RMSE of \(\text{target}_3\) state estimation respectively for the 3rd policy when the data fusion is controlled by MDP.

Fig. 14 shows the information gain of \(\text{platform}_3\) for all the AMR’s arriving from the sensors of \(\text{platform}_3\) and \(\text{platform}_2\). Fig. 15 shows the integrated information gain values obtained by integrating the values of the information gain during the policy re-calculation stages during the following time intervals: \([5, 15]\), \([60, 75]\), and \([120, 135]\) [s]. The MDP
policy of platform\textsubscript{3} was updated several times during the simulation at the end of the policy re-calculation stages above.

Table I shows the time-averaged Position and Velocity RMSE in the targets state estimation of the platform\textsubscript{3} after 100 Monte-Carlo runs. The performance of no information sharing mode is the worst. The performance of the mode in which all the AMR’s are transmitted among all the sensors is the best. We can see that the performance of the MDP controlled mode is much better that that of the 1st mode but still worse that that of the 2nd mode. In MDP controlled mode the maximal number of sensors that could transmit AMR’s to platform\textsubscript{3} was restricted to 2 sensors. That we can see that the information flow in the system was reduced by half which in many cases may justify the reduction in performance. In many cases the situation in which all the platforms transmit AMR’s to all the other platforms is infeasible. The performance may be increased though by increasing the maximal allowed number of sensors transmitting remote AMR’s to any given platform.

V. CONCLUSIONS

In this paper, we proposed a multi-level Markov decision process based algorithm for controlling the flow of information in a network-centric data fusion architecture. We utilized the approach of using information based objective function as one of the components in the formulated optimization problem. We

<table>
<thead>
<tr>
<th>Communication mode</th>
<th>Position RMSE (m)</th>
<th>Velocity RMSE (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>All AMR’s shared</td>
<td>73.2 79.6 38.4 39.5</td>
<td></td>
</tr>
<tr>
<td>No AMR’s shared</td>
<td>382.1 300.3 90.8 79.0</td>
<td></td>
</tr>
<tr>
<td>MDP controlled</td>
<td>103.8 109.7 44.4 45.8</td>
<td></td>
</tr>
</tbody>
</table>

TABLE I
PERFORMANCE METRICS SUMMARY.
demonstrated that the approach led to a substantial reduction in data flow volumes which also incurred a certain price in terms of reduction in performance. The node lookup is performed in $O(\log(N))$ and the computational complexity of solving the MDP is P-complete, which shows that the proposed method is computationally attractive for distributed data fusion applications.

REFERENCES


