Evaluating sensor allocations using equivalence classes of multi-target paths

Christian Mårtenson **Pontus Svenson**
Dept of Data and Information Fusion
Swedish Defence Research Agency
SE 164 90 Stockholm
Sweden
cmart,ponsve@foi.se
http://www.foi.se/fusion

***Abstract*— We present an algorithm for evaluating sensor allocations using simulation of equivalence classes of possible futures. Our method is meant to be used for pre-planning sensor allocations, *e.g.*, for choosing between several alternative flight-paths for UAV's, or deciding where to deploy ground sensor networks. The method can be used to choose the best sensor allocation with respect to any fusion method.**

In addition to the list of sensor allocations to evaluate, the algorithm requires knowledge of the terrain/road network of the region of interest. Additional doctrinal knowledge on the enemy's possible goals and rules of engagement can increase the speed of the method, but is not required.

Given a current situation picture, the method generates possible future paths for the objects of interest. For each considered sensor allocation, these futures are partitioned into equivalence classes. Two futures are considered equivalent with respect to a given sensor allocation if they would give rise to the same set of observations. For each such equivalence class, we run a fusion algorithm; in this paper, an emulated multi-target tracker. We use the output of this emulated filter to determine a fitness for each sensor allocation under evaluation. For small scenarios, we compare some different ways of calculating this fitness, concluding that an approximation introduced by us gives nearly the same result as an exact method.

We also introduce a formulation of the studied problem and the method used to solve it using random sets, and give several directions for future work.

I. Introduction

Sensor resource management (level 4 fusion in the JDL model [1]) is an important part of future information fusion systems [2], [3]. The need for good sensor management systems is particularly evident in situations where we have an inadequate number of sensor resources and lack detailed background and doctrinal information on adversary forces (*e.g.*, in international operations other than war).

In this paper, we introduce a new approach to sensor resource allocation by simulating *equivalence classes* of future multi-target paths. (The approach can also be used for threat prediction.) In this paper, we concentrate on the problem of evaluating a set of proposed sensor allocation schemes to determine which is the best to use in order to provide as good as possible input to a given fusion system. Here, we emulate a multi-target tracker as the fusion system; our formalism and method is however general and allows any fusion system to be used.

An equivalence class of future multi-target positions is defined with respect to a given sensor allocation scheme; the class consists of all those predicted paths that give rise to the same set of observations. We use this when simulating possible multi-target futures in order to cut down on the number of future paths that we need to consider. For each such equivalence class, we determine the fitness of each considered sensor scheme. The total fitness of a scheme is then the average over all equivalence classes of this fitness.

The method is based on a simpler version that was implemented in the FOI IFD03 information fusion demonstrator [4], [5]; that implementation did not make use of the equivalence classes of futures.

This paper is outlined as follows. Section II introduces the problem we want to solve, and formulates the method in terms of random sets and equivalence classes of future paths. In section III, we note the input requirements of the method, while section IV gives more details on the specific implementation presented here. Section V presents results from several different scenarios, while a discussion and suggestions for future work are contained in section VI.

II. Simulating futures using equivalence classes

In order to determine the best of the sensor allocation schemes, we simulate a possible future path of the units of interest and apply all the sensor schemes to it. For each sensor scheme, this gives us a list of simulated observations that we can input to a fusion module. For each fusion output, we calculate a fitness by comparing it to the "true" simulated future. By averaging over many possible future paths, we can determine a total fitness for each sensor allocation scheme. In order to speed up this process, we introduce the concept of equivalence classes of futures below, but first we describe the algorithm without using these.

In figure 1, we show a conceptual view of that we are doing. At the bottom, we see a simulated future path. This is converted into a set of observations shown in the middle, that are used to form the track shown at the top. To evaluate

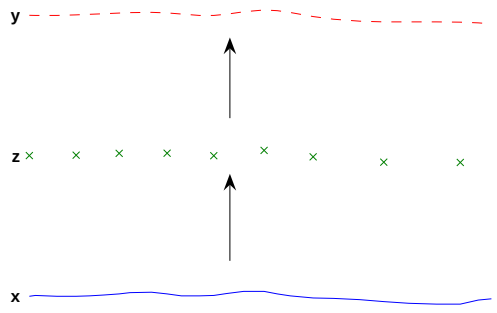

Fig. 1. Conceptual overview of how we use a simulated future x and a sensor scheme s to generate observations z which are input into a filter giving a track y which can be compared to x in order to evaluate s.

the sensor allocation scheme used to generate the fictitious observations, we compare the track with the simulated future.

Mathematically, the algorithm works as follows. A density vector x_0 is given, which describes the positions of the units of interest at time $t = 0$. A set S is defined, consisting of sensor allocation schemes and information about the terrain.

Three different random sets [6], [7] are used:

1) $\mathbf{X}(t)$ denotes the positions of the units of interest at time t, conditioned on them being at $\mathbf{x}_0$ at time 0. It can be seen as representing a simulation of ground truth: the instance $\hat{x}(t)$ of $\mathbf{X}(t)$ occurs with probability $P[\mathbf{X}(t) = \hat{x}(t)|\mathbf{X}(0) = x_0]$. For simplicity of notation, the conditioning on x_0 is not explicitly shown in the following.
2) For each sensor allocation scheme $s(t) \in S$ and instance $\hat{x}(t)$ of the future ground truth, a set of possible observations $\mathbf{Z}(x(t), s(t), t)$ is calculated at time t. $\mathbf{Z}$ is also a random set; note that it depends on the simulated ground truth as well as allocation scheme.
3) Finally, we determine what our view of ground truth would be, given the set of observations $\mathbf{Z}$. This gives rise to the final random set, $\mathbf{Y}(t)$. $\mathbf{Y}(t)$ is our fusion system's approximation of the (simulated) ground truth $\hat{x}(t)$ using the observations $\mathbf{Z}$ obtained by deploying sensors according to sensor allocation scheme $s(t)$.

Note that all of the random sets introduced are explicitly time-dependent. Here, an expression like $P[\mathbf{X}(t)]$ denotes the probability of the entire time-evolution of $\mathbf{X}(t)$, not just the probability at a specified time. $P[]$ can thus be seen as a "probability density functional" in the space of all explicitly time-dependent random sets.

Determining which sensor allocation scheme to use is now done simply by comparing the assumed ground truth $\hat{x}(t)$ to the fusion system's simulated view $\hat{y}(t)$. For each instance $\hat{x}(t)$ of $\mathbf{X}(t)$, the best $s(t)$ can easily be determined by averaging over the ensembles of observations $\mathbf{Z}$ and simulated fusion process output $\mathbf{Y}$ entailed by that simulated ground truth. An allocation scheme is good if the simulated filter gives a good approximation of the simulated ground truth.

The fit of a specific allocation scheme s for a certain simulated ground truth $\hat{x}(t)$ can be written as

$$\begin{aligned} &H(\hat{x}(t), s) = \\ &\int P[\mathbf{Z}(t) = \hat{z}(t)|\mathbf{X}(t) = \hat{x}(t), s] \times \\ &P[\mathbf{Y}(t) = \hat{y}(t)|\mathbf{Z}(t) = \hat{z}(t)] \times h(\hat{x}(t), \hat{y}(t)) d\hat{z}(t) d\hat{y}(t) \end{aligned} \quad (1)$$

where h is a functional that compares $\hat{x}(t)$ and $\hat{y}(t)$ and the integrals are functional integrals over all random sets $\hat{y}(t)$ and $\hat{z}(t)$. We use two different h-functionals: one which computes the entropy of $\hat{y}$ and one which calculates the L_1 distance between $\hat{x}$ and $\hat{y}$. The difference between the entropy-like measure and the distance measure is that the entropy measure rewards allocation schemes that give rise to peaked distributions, but might miss some of the tracked units. We also used the L_2 distance for some scenarios and found no significant difference in the results compared to the L_1 distance.

The overall best sensor allocation scheme is then determined by averaging also over the random set $\mathbf{X}(t)$, as

$$s^{\text{best}} = \arg \min_{s(t) \in S} \int P[\mathbf{X}(t) = \hat{x}(t)] H(\hat{x}(t), s(t)) dx(t) \quad (2)$$

Implementing Equations (1) and (2) would thus entail averaging over three different random sets, which is clearly computationally infeasible in most cases.

(We could also use alternate h-functionals, where we calculated h at a specific time. This would be useful if it is important to know where the objects we are tracking are at a certain time.)

There are several possible ways of approximating these equations. One way is to use approximations of the probabilities P appearing in them, perhaps employing some kind of Monte Carlo sampling instead of the ensemble averages.

A. Equivalent futures

Here we use a different idea in order to reduce the number of possible futures that need to be simulated. Consider two alternative future paths $\hat{x}_1(t)$ and $\hat{x}_2(t)$. If $\hat{x}_1(t)$ and $\hat{x}_2(t)$ are very similar, it might not be necessary to simulate them both. Consider, for example, the two sets of paths shown in figure 2. If the sensor is located as shown, it will not be possible for it to distinguish between the two paths. (We are here assuming that the sensors detection probablity $p_D = 1$; see below.)

Generalizing this idea leads us to the concept of *equivalence classes of future paths*. We define two possible future ground truths $\hat{x}_1(t)$ and $\hat{x}_2(t)$ to be equivalent with respect to the sensor allocation scheme $s(t)$ if $\mathbf{z}(\hat{x}_1(t), s(t), t) = \mathbf{z}(\hat{x}_2(t), s(t), t)$ for all t, and write that $\hat{x}_1(t) \sim_s \hat{x}_2(t)$

What this means is just that if there is no way for our sensor and fusion system to differentiate between two different predicted paths, there is no sense in simulating them both. Instead of averaging over all possible futures in equation 2, we average over all equivalence classes of futures:

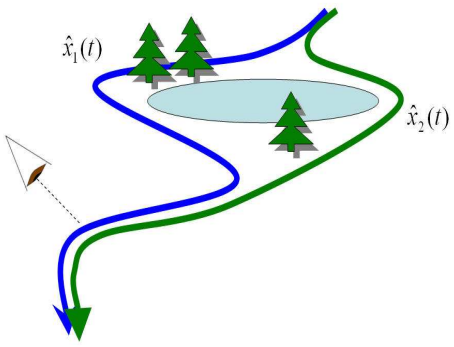

Fig. 2. This figure illustrates the concept of equivalent paths. Given the sensor location shown, it is impossible to distinguish between the two paths; hence they belong to the same equivalence class.

$$s^{\text{best}} = \arg\min_{s(t)\in S} \int_{\hat{x}(t)\in X_s} P[\hat{x}(t)]H(\hat{x}(t), s(t))d\hat{x}(t) \quad (3)$$

where X_s denotes the partition of X induced by the equivalence relation $\sim_s$ defined above.

This idea can be extended further. We could for instance relax the restriction that the observations must be equal for all t, and generate equivalence classes for specific time-windows. Another option would be to consider the threat posed by the a future $\hat{x}$ and define two futures to be equivalent if they entail the same threat for us. This and other extensions of the ideas presented here will be described in detail elsewhere.

Note that the definition of equivalence classes as given above is only correct for sensors with $p_D = 1$. In the algorithm and experiments described below, where we use $p_D < 1$, we also condition on a specific sequence of random numbers used in determining the fictitious observations; this will guarantee that our equivalence classes are proper. Formally, we express this by including the seed of the random number generator used in the sensor allocation scheme $s(t)$. Another way of exploiting the ideas presented here would be to group $\hat{x}_1(t)$ and $\hat{x}_2(t)$ together if their corresponding fictitious observations $\mathbf{Z}(\hat{x}_1(t), s(t), t)$ and $\mathbf{Z}(\hat{x}_1(t), s(t), t)$ are "sufficiently similar". This would, however, not give equivalence classes, since transitivity would not hold in that case.

Kadar et al [8] have recently described a system that predicts a future enemy path and uses this to determine figures-of-merit for dynamic sensor planning. In addition to our use of equivalence classes of multi-target paths, another major difference between that work and the method presented here is that we average over many predicted paths as well as over many realizations of the observations and of the emulated fusion module.

III. Inputs

Our sensor allocation scheme evaluation method requires a number of inputs. The following is a list of the different kinds of parameters required by our method:

- Sensor allocation policies to evaluate. This is of course the most crucial input. Sensor plans could be generated automatically or by a user.
- Object positions at start time. In a real system, this input could be a probability hypothesis density (PHD) [7] $\mathbf{X}_0$ from a particle filter or force aggregation system [4], [5].
- Terrain representation. In this paper, we have used a road network for this. Such networks could be generated automatically using GIS software. They can also be generated using methods such as those proposed in [9]
- Transition probabilities in road network. In our current implementation, this is based mostly on geographical information. The road network is represented below in terms of a transition matrix T, whose entries T_{ab} gives the probability to move from node b to node a in unit time. A real system could augment this with knowledge of possible enemy objectives, probably using input from a user/analyst. It is straightforward to include also more detailed information on enemy objectives in the transition matrix T; it could for instance be changed so that movement towards known goals are enhanced and movements away from these goals reduced. Doing this, however, necessitates having a detailed knowledge of enemy doctrine.

IV. Algorithms

Below we describe in detail the two methods for evaluating sensor allocations, the Exhaustive and the Monte Carlo method. The basic steps are the same:

1) Propagate the simulated enemy positions.
2) Generate one/all possible set(s) of fictitious observations, given the new enemy positions.
3) Generate equivalence classes of enemy positions with respect to the set of observations.
4) Propagate and update the emulated multi-target filter according to the set of observations.
5) Calculate the fitness of the updated filter.
6) Repeat from 1, unless the maximum number of propagation steps has been reached.

The Exhaustive algorithm will perform steps 3 to 6 for each observation set generated in step 2. In other words, it considers every single equivalence class, a set that grows exponentially with the number of observation opportunities. The Monte Carlo algorithm, in contrast, considers only a single set of observations for each propagation step. To compensate for this, the entire process of generating a future is repeated a number of times.

The steps (1-5) are further described in the paragraphs below. Pseudo-code for the algorithms are shown in Algorithm IV.1 and Algorithm IV.2.

A. Propagation of simulated ground truth

As described in section II, the enemy positions are defined by a random set $\mathbf{X}(t)$. In our current implementation, we choose to simplify this and instead work with a fixed number of targets. $\mathbf{X}(t)$ is implemented as a matrix, where column i

Algorithm IV.1: EXHAUSTIVE($t, t_{\mathbf{max}}, x, T, s$)

for $i \leftarrow 1$ **to** $N_{\mathbf{targets}}$
do x $\begin{cases} x_{ai}(t+1) \leftarrow \sum_b T_{ab} \cdot x_{bi}(t) \quad \forall a \quad \textit{** Propagate equiv. class} \\ z_{ai} \leftarrow s_a(t+1) \cdot x_{ai}(t+1) \quad \forall a \quad \textit{** Determine observation probabilities} \end{cases}$
$Z \leftarrow$ Set of all combinations with one non-zero z_{ai} for each i
for each $\hat{z} \in Z$ *** Split equiv. class into many and evolve each separately*
do
- **for** $i \leftarrow 1$ **to** $N_{\mathbf{targets}}$
  do
  - **if** target i is observed
    - **then** $x_{ai}(t+1) \leftarrow \hat{z}_{ai} \quad \forall a$
    - **else** $x_{ai}(t+1) \leftarrow x_{ai}(t+1) \cdot (1 - s_a(t+1)) \quad \forall a$
- $y_a(t+1) \leftarrow \sum_b T_{ab} \cdot y_b(t) \quad \forall a \quad \textit{** Update emulated filter}$
- $y_a(t+1) \leftarrow y_a(t+1) \cdot (1 - s_a(t+1)) + \hat{z}_a \quad \forall a$
- Calculate and store fitness
- **if** $t < t_{\mathbf{max}}$
  then EXHAUSTIVE($t+1, t_{\mathbf{max}}, x, T, s$)

Algorithm IV.2: MONTECARLO($t_{\mathbf{max}}, x, T, s, N_{\mathbf{samples}}$)

for $sample \leftarrow 1$ **to** $N_{\mathbf{samples}}$
do
- **for** $t \leftarrow 1$ **to** $t_{\mathbf{max}}$
  do
  - **for** $i \leftarrow 1$ **to** $N_{\mathbf{targets}}$
    do $\begin{cases} x_{ai}(t+1) \leftarrow \sum_b T_{ab} \cdot x_{bi}(t) \quad \forall a \quad \textit{** Propagate equiv. class} \\ z_{ai} \leftarrow s_a(t+1) \cdot x_{ai}(t+1) \quad \forall a \quad \textit{** Determine observation probabilities} \end{cases}$
  - $\hat{z}(t) \leftarrow$ Random combination of one non-zero z_i for each i
  - **for** $i \leftarrow 1$ **to** $N_{\mathbf{targets}}$
    do
    - **if** target i is observed
      - **then** $x_{ai}(t+1) \leftarrow \hat{z}_{ai} \quad \forall a.$
      - **else** $x_{ai}(t+1) \leftarrow x_{ai}(t+1) \cdot (1 - s_a(t+1)) \quad \forall a.$
  - $y_a(t+1) \leftarrow \sum_b T_{ab} \cdot y_b(t) \quad \forall a \quad \textit{** Update emulated filter}$
  - $y_a(t+1) \leftarrow y_a(t+1) \cdot (1 - s_a(t+1)) + \hat{z}_a \quad \forall a$
  - Calculate and store fitness

contains the probability density function over all nodes a for the position of target i. The targets of an equivalence class are propagated through the road network according to

$$x_{ai}(t+1) = \sum_b T_{ab} \cdot x_{bi}(t) \qquad \forall a, i. \tag{4}$$

Note that $x_{ai}(t+1)$ will be updated again in equations 6 and 7.

B. Generation of sensor observations

The sensor coverage for the entire network at time t is described by the sensor allocation scheme $s(t)$, which is a probability distribution over the network nodes. The value $s_a(t)$ gives the probability of detecting a target located at node a. The probability of detecting target i at node a and time t is thus given by

$$z_{ai}(t) = x_{ai}(t) \cdot s_a(t) \qquad \forall a, i. \tag{5}$$

This equation together with the probability of not detecting target i at any node, which we denote $z_{0i}(t) = 1 - \sum_a z_{ai}(t)$, can be used to generate the random set $\mathbf{Z}(t)$. An instance of $\mathbf{Z}(t)$ is represented by a binary matrix, $\hat{z}$, where $\hat{z}_{ai} = 1$ means that target i is observed at node a, and $\hat{z}_{ai} = 0$ that it is not. The corresponding probability is $P(\hat{z}) = P(\mathbf{Z}(t) = \hat{z} | \mathbf{X}(t), s(t))$.

In the Exhaustive Algorithm we need to find all instances of $\mathbf{Z}(t)$ with non-zero probability, *i.e.*, all possible combinations of observations and non-observations at a specific time t. These can be generated by choosing one non-zero element from each column (target) in z. (Recall that row 0 in the matrix corresponds to a target not being observed.) Each chosen element that corresponds to an actual observation (*i.e.*, not in row 0) yields a 1 in $\hat{z}$; $P(\hat{z})$ is determined by multiplying the elements' probabilities.

The Monte Carlo method uses only one set of observations in each time step. This can either be drawn from the set of all combinations with probability $P(\hat{z})$, or generated directly by

drawing one element from each column in z with probability z_{ai}.

C. Generating equivalence classes

The propagation of the target density described in section IV-A results in a description of all possible new enemy positions at $t+1$. These positions we now want to group into equivalence classes depending on which observations we are considering. Given one set of observations $\hat{z}$ we update x so that it only includes positions consistent with these observations.

For a target i that is observed the probability distribution is peaked,

$$x_{ai}(t+1) \leftarrow \hat{z}_{ai} \qquad \forall a. \tag{6}$$

The probability distribution of a target i that is not observed is updated according to the chance for this to happen at the node considered,

$$x_{ai}(t+1) \leftarrow x_{ai}(t+1) \cdot (1 - s_a(t+1)) \qquad \forall a. \tag{7}$$

The columns of the updated x are then normalized to preserve identity.

D. Filter update

The steps described above deal with the simulation of alternative future physical realities. Now we turn to the updating of the fusion system acting on these, in this case an emulated multi-target tracker.

The state of the tracker, y, is propagated and updated in a similar way as x. The main difference is that the fusion system cannot distinguish which targets generate which observations, which is why y is described by a PHD. In this case, to represent an instance of $\mathbf{Z}(t)$ it is enough to consider $\hat{z}_a := \sum_i \hat{z}_{ai}$.

Given a set of observations $\hat{z}$ we update the filter in three steps. First we propagate y through the road network with the propagation matrix T,

$$y_a(t+1) = \sum_b T_{ab} \cdot y_b(t) \qquad \forall a. \tag{8}$$

Secondly, in order to take negative information into account, we decrease the PHD where there is a positive probability of detection,

$$y_a(t+1) \leftarrow y_a(t+1) \cdot (1 - s_a(t+1)) \qquad \forall a. \tag{9}$$

Finally, we add the new observations to the PHD,

$$y_a(t+1) \leftarrow y_a(t+1) + \hat{z}_a \qquad \forall a. \tag{10}$$

These steps roughly correspond to the propagation and resampling of a PHD particle filter [5]. If the number of targets, $N_{\mathbf{targets}}$, is known , we require that $\sum y(t+1) = N_{\mathbf{targets}}$. This can be accomplished by normalizing y before performing equation 10, such that $\sum_a y_a(t+1) = (N_{\mathbf{targets}} - \sum_a \hat{z}_a)$.

E. Fitness calculations

For comparing the sensor allocation schemes we use two different measures, introduced in section II as alternative h-functionals for calculating the fitness. Here the measures can rather be viewed as a distance or inverse fitness, which we would like to minimize.

Both measures are applicable in the Monte Carlo as well as in the Exhaustive algorithm. In the Monte Carlo method the fitness is calculated by averaging the measure over all time steps and all runs. In the Exhaustive method we sum the measure for each equivalence class at each time step, weighting it with the probability for the equivalence class to occur.

- *XY-difference.* The XY-difference, D_{xy}, measures the mean L_1 distance between $y(t)$ and all instances $\hat{x}(t)$ of $\mathbf{X}(t)$,

$$D_{xy} = \sum_{\hat{x}(t) \in X(t)} P(\hat{x}) \cdot \|\hat{x} - y\|_1. \tag{11}$$

- *Entropy.* The entropy measure, H_y, determines how peaked $y(t)$ is. Since $y(t)$ is a PHD it does not have to sum to 1, and therefore has to be normalized before it can be inserted into the following formula for the entropy:

$$H_y = -\sum_a y_a \cdot \log y_a. \tag{12}$$

V. Experiments

Here we present results of the algorithms presented in sections II and IV for some different scenarios. All sensor allocation schemes considered here are assumed to be time-independent. For each scenario, we present a map giving an overview of the area where it takes place. The objective is to allocate a number of sensors in a way that minimizes the uncertainty of the estimated enemy positions over time.

To generate the enemy behavior a number of start and goal nodes are assumed to be known. The initial state, which is meant to be derived from the output of a PHD tracker, is determined by distributing the probability density functions of the targets randomly over the start nodes. All targets are assigned identical start distributions.

The motion model used to simulate the enemy behavior is described by the propagation matrix T. In all scenarios, T is constructed so that it is four times more likely to move to a node that is closer to a goal compared to the current node, than to a node that is more distant from the goals. A move to a node with equal distance to a goal node as the current is twice as likely as a move to a more distant node.

A. Härnösand scenario

Figure 3 shows the map for the first scenario. Three sensors are to be allocated to minimize the uncertainty of the positions over seven time steps when tracking three targets.

Results from the evaluation of 100 randomly generated sensor allocations, using 100 Monte Carlo samples, are shown in figure 4. The allocations are ordered after the results of the

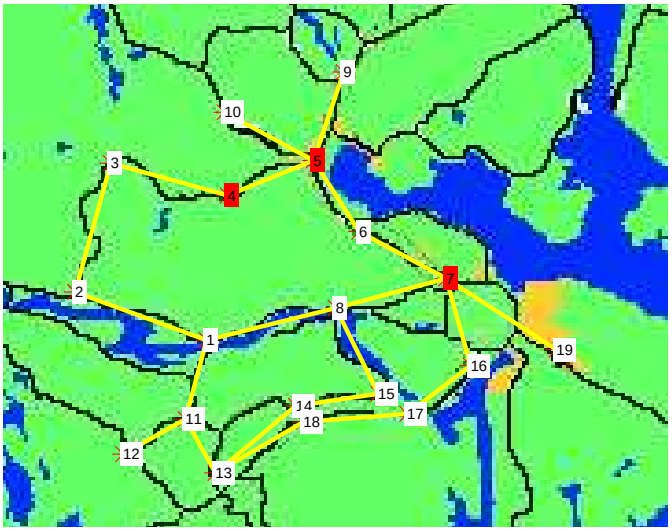

Fig. 3. The Härnösand scenario road network. Initial enemy positions are distributed on nodes 4, 5 and 9 (at the top of the map), and goal nodes are 12, 13 and 19 (at the bottom). The sensor positions of the best allocation are shown in red.

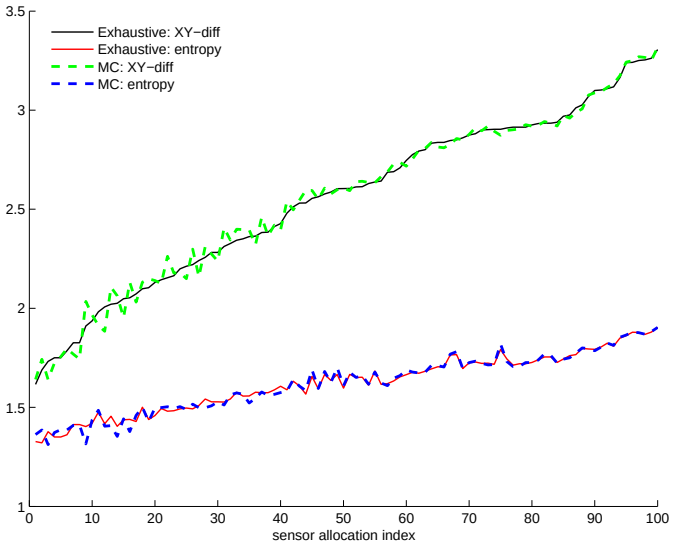

Fig. 4. Results from the evaluation of 100 sensor allocations for the Härnösand scenario, ordered after the Exhaustive method with the XY-difference measure. The ranking induced by the entropy measure is similar to that induced by the XY-difference.

Exhaustive method run with the D_{xy} measure, which can be regarded as an exact result. The Exhaustive method is also the most compute intense, as can be seen in figure 5. In a worst case scenario, the number of equivalence classes to consider grows exponentially both with the number of sensors and with the number of targets.

Two observations are therefore of crucial interest. The first is that the entropy measure H_y approximates the relative values of D_{xy} rather well. The second is that the Monte Carlo technique reduces the overall computation time without significant loss in precision. In other words, the Monte Carlo method

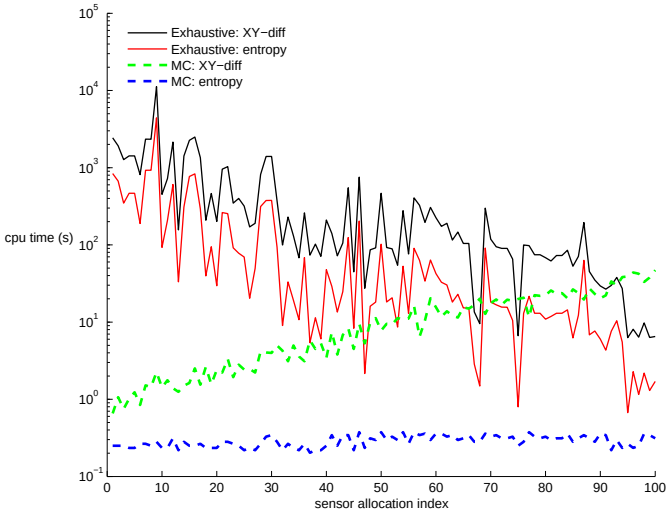

Fig. 5. The computation time for the four approaches on the 100 allocations for the Härnösand scenario. The allocations are ordered as in figure 4. Note the logarithmic scale on the time-axis.

Fig. 6. The Betaland scenario road network. Initial enemy positions are distributed on nodes 4, 7, 9 and 11 (to the left), and goal nodes are 30, 34 and 56 (to the right). The sensor positions of the best allocation are shown in red.

using the H_y measure gives us an acceptable approximation in the H¨arn¨osand scenario, with a large decrease in computation time. The time complexity for this combination is linear in all input parameters.

B. Betaland scenario

Figure 6 shows a map of the fictitious Betaland (in reality, the western part of Sweden) used by the Swedish Armed Forces to train and construct methodology for international operations. The scenario setup is the same as in the H¨arn¨osand scenario. Three sensors are to be allocated to track three targets during 7 time steps, and 100 randomly generated sensor allocation schemes are evaluated. The results are shown in

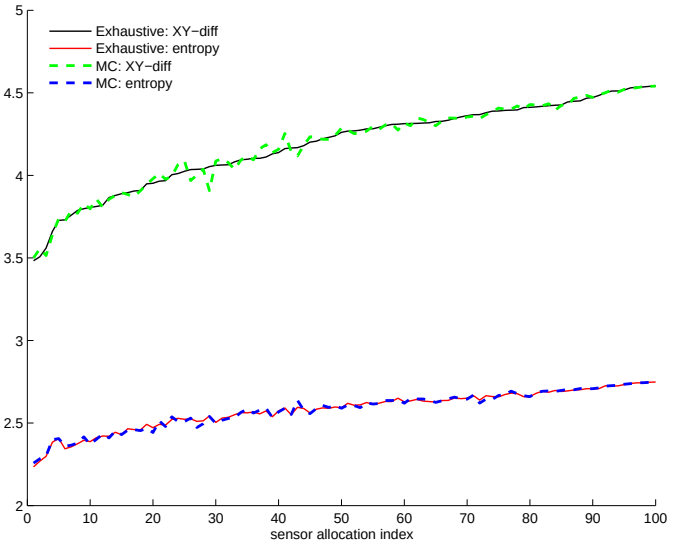

Fig. 7. Comparison of the fitness for the extensive and Monte Carlo approach for the Betaland scenario. For both methods, the xy-diff measure as well as the entropy are shown.

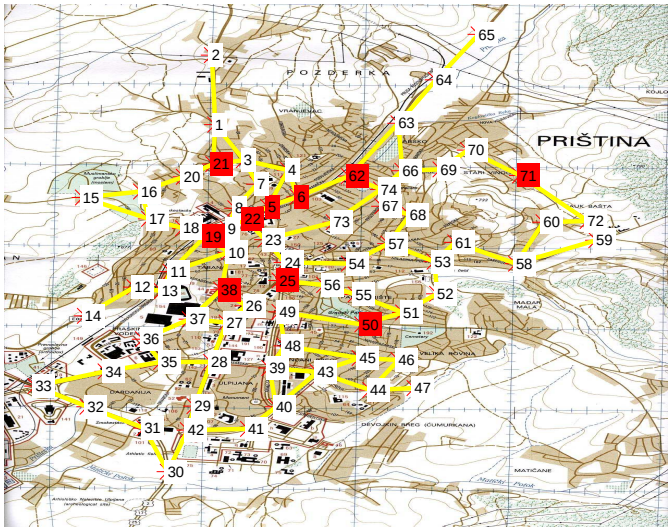

Fig. 8. The Priština scenario road network. Initial enemy positions are distributed on nodes 1, 2, 4, 63 and 64 (at the top of the map), and goal nodes are 30, 33 and 47 (at the bottom). The sensor positions of the best allocation with 10 sensors are shown in red.

figure 7. As can be seen, the behavior is almost identical to that of the H¨arn¨osand scenario, which gives us increased confidence in exploiting the proposed approximations.

C. Priština scenario

Figure 8 shows the map of our last scenario, which takes place in Priština in Kosovo. Here we choose to only explore the capability of the fast Monte Carlo entropy approach, so that a larger problem can be solved. 10 targets are tracked during 15 time steps, using setups with 5, 10 and 15 sensors respectively. For each setup we evaluate 1000 randomly generated allocations. The larger problem size calls for an increase of the number of Monte Carlo samples, which here was set to 1000. This choice is based on visual investigations of the convergence of the mean entropy.

The sensor positions of the best evaluated sensor allocation with 10 sensors are marked in figure 8. In figure 9 we plot the evolution over time of the entropy measure H_y. The top curve is a 'worst case' evolution, where no sensor data is registered at all. The other curves display the entropy of the best allocations from the setups with 5, 10 and 15 sensors respectively. Obviously, the entropy decreases when we increase the number of sensors. Our method found good sensor allocation schemes when tried on this scenario; however, since we test randomly generated schemes, we are not guaranteed to find the best. We are currently investigating using genetic algorithms with the method presented here used as fitness for evolving good sensor allocation schemes.

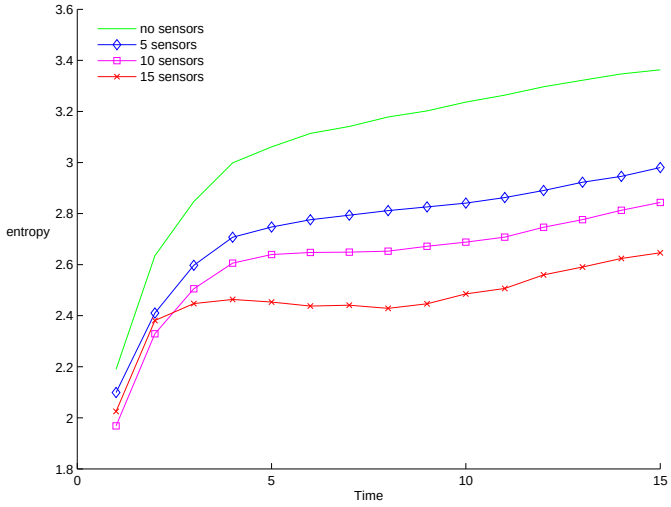

Fig. 9. The entropy evolution over time in the Priština scenario. The best sensor allocations from the setups with 5, 10 and 15 sensors are compared to the case with no sensor information at all.

VI. Discussion and future work

We have presented a method for evaluating different sensor allocation schemes. The method works with respect to a given fusion module, and ranks the given sensor schemes according to their ability to improve the output of this module.

The method is based on a random set formulation of possible future multi-target paths and the concept of *equivalent futures*. The use of these equivalence classes enable us to simulate more time-steps into the future using less computational resources, as can be seen in the figures of section V.

The benefit of using equivalent futures increases when

- network size increases
- number of sensors decreases.

The most obvious example of this is when no sensors are used: in this case, all the futures collapse into one equivalence class. At the other extreme: if we have one sensor on each road, equivalent futures are useless.

One way of thinking about the method for generating obser-

vations that we use to get the equivalence classes is to compare it with the continuous-time Monte Carlo algorithm [10], where the system is always updated and the time-step varies: system time is updated with the time-step needed to do the selected move. Continuous-time Monte Carlo could also be used to speed up the computation of future paths significantly.

There are several directions of future work related to the method described here.

More work needs to be done on automatically generating sensor allocation schemes. As mentioned before, we are currently considering a genetic algorithm approach to this. Another possibility would be to use a swarming or collective intelligence algorithm for this [11]

The method could also be used interactively by a user, who suggests partial or complete sensor schemes to the system. Here work needs to be done both on how to generate a complete plan from a partial one and on how to best interact with the user. It would be interesting to combine the method presented here with planning systems. Users can then input alternative plans and have them automatically evaluated by the computer system. This could be extended to be used for other things than sensor allocation, such as threat or possibility analysis, logistical planning, etc.

There are also many possible extensions of the algorithm that could make it even more efficient. Using more advanced Monte Carlo sampling of equivalence classes should enable us to speed up the algorithm considerably.

It would also be interesting to try to determine a stopping criterium for the simple Monte Carlo sampling performed in algorithm IV.2.

Since the approximate fitness seems to be very good for almost all sensor allocations, it should be possible to use the approximate fitness to determine the best 10 or 20 sensor allocation schemes and then run the exact method only on these.

It would be interesting to combine the methodology presented here with the movement prediction from [12]. The method given for determining reachable intervals there could possibly be used as input to our system. This would enable us to cut down on the number of x paths that we need to consider.

It would also be interesting to implement the idea of equivalence classes of futures in the rigorous FISST-based approach to dynamic sensor management recently introduced by Mahler and co-authors (*e.g.*, [13] and references therein).

We are currently exploring how to extend the concept of equivalence classes to take threat or impact into account. In other work, we will investigate a unification of the method here with that presented in [14], [15]

REFERENCES

[1] D. L. Hall and J. Llinas, editors. *Handbook of Multisensor Data Fusion*. CRC Press, Boca Raton, FL, USA, 2001.

[2] N. Xiong and P. Svensson. Sensor management for information fusion — issues and approaches. *Information Fusion*, 3(2):163–186, 2002.

[3] Ronnie Johansson and Ning Xiong. Perception management - an emerging concept for information fusion. *Information Fusion*, 4(3):231–234, 2003.

[4] S. Ahlberg, P. Hörling, K. Jöred, C. Mårtenson, G. Neider, J. Schubert, H. Sidenbladh, P. Svenson, P. Svensson, K. Undén, and J. Walter. The IFD03 information fusion demonstrator. In *Proceedings of the Seventh International Conference on Information Fusion, Stockholm, Sweden*, pages 936–943, Mountain View, CA, USA, 2004. International Society of Information Fusion.

[5] J. Schubert, C. Mårtenson, H. Sidenbladh, P. Svenson, and J. Walter. Methods and system design of the IFD03 information fusion demonstrator. In *CD Proceedings of the Ninth International Command and Control Research and Technology Symposium, Copenhagen, Denmark*, pages 1–29, Washington, DC, USA, Track 7.2, Paper 061, 2004. US Dept. of Defense CCRP.

[6] I. R. Goodman, R. P. S. Mahler, and H. T. Nguyen. *Mathematics of Data Fusion*. Kluwer Academic Publishers, Dordrecht, Netherlands, 1997.

[7] R. Mahler. *An Introduction to Multisource-Multitarget Statistics and its Applications*. Lockheed Martin Technical Monograph, 2000.

[8] Ivan Kadar, Kuo Chu Chang, Kristin O'Connor, and Martin Liggins. Figures-of-merit to bridge fusion, long term prediction and dynamic sensor management. In *Signal Processing, Sensor Fusion, and Target Recognition XIII*, volume 5429 of *SPIE*, page 343, 2004.

[9] R. Richbourg and W. K. Olson. A hybrid expert system that combines technologies to address the problem of military terrain analysis. *Expert Systems with Applications*, 11(2):207–225, 1996.

[10] A B Bortz, M H Kalos, and J L Lebowitz. A new algorithm for Monte Carlo simulation of Ising spin systems. *Journal of Computational Physics*, 17:10, 1975.

[11] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artifi cial Systems*. Oxford University Press, Oxford, UK, 1999.

[12] Mark J Carlotto, Mark A Nebrich, and Kristin O'Connor. Battlespace movement prediction for time critical targeting. In Ivan Kadar, editor, *Signal Processing, Sensor Fusion, and Target Recognition XIII*, volume 5429 of *Proceedings of SPIE*, page 326, 2004.

[13] Ronald P. Mahler. Sensor management with non-ideal sensor dynamics. In *Proceedings of the Seventh International Conference on Information Fusion*, volume II, pages 783–790, Mountain View, CA, Jun 2004. International Society of Information Fusion.

[14] Ronnie Johansson and Robert Suzić. Bridging the gap between information need and information acquisition. In *Proceedings of the 7th International Conference on Information Fusion*, volume 2, pages 1202–1209. International Society of Information Fusion, 2004.

[15] Robert Suzić and Ronnie Johansson. Realization of a bridge between high-level information need and sensor management using a common dbn. In *The 2004 IEEE International Conference on Information Reuse and Integration (IEEE IRI-2004)*. IEEE, November 2004.