Global/Local Dynamic Models

Avi Pfeffer
Harvard University
Cambridge, MA 02138
avi@eecs.harvard.edu

Subrata Das and **David Lawless**
Charles River Analytics, Inc.
Cambridge, MA 02138
sdas@cra.com

Brenda Ng
Lawrence Livermore Nat'l Lab
Livermore, CA 94551
bmng@llnl.gov

Abstract

Many dynamic systems involve a number of entities that are largely independent of each other but interact with each other via a subset of state variables. We present global/local dynamic models (GLDMs) to capture these kinds of systems. In a GLDM, the state of an entity is decomposed into a globally influenced state that depends on other entities, and a locally influenced state that depends only on the entity itself. We present an inference algorithm for GLDMs called global/local particle filtering, that introduces the principle of reasoning globally about global dynamics and locally about local dynamics. We have applied GLDMs to an asymmetric urban warfare environment, in which enemy units form teams to attack important targets, and the task is to detect such teams as they form. Experimental results for this application show that global/local particle filtering outperforms ordinary particle filtering and factored particle filtering.

1 Introduction

Many systems involve a number of entities that interact with each other. These entities may be largely independent of each other, and only interact with other entities via a subset of state variables. An example is a collection of companies, that interact via market conditions. Each company may have internal state that is conditionally independent of other companies given the market conditions. Another example is the spread of an infectious disease through a population. Each person may have an individual state, which corresponds to his or her symptoms, and people will interact via their infectious states.

We present a type of model called *global/local dynamic models (GLDMs)* that allows the representation of these kinds of systems. In a GLDM, the state of an entity is divided into two subsets: a *locally influenced state* that only depends on the state of the entity itself, and a *globally influenced state* that depends globally on the states of all entities. In this way, interactions between the entities are allowed, but much of the dynamic model is isolated within individual entities. Observations are restricted to depend only on the state of individual entities.

One would hope that in such a system, it will be possible to exploit the largely independent nature of the entities for efficient inference. We are concerned with the inference task of monitoring, i.e. the computation of the probability distribution over the state of the system at each point in time given the history of observations up to that time point. One popular approach to monitoring is particle filtering (PF) [Isard and Blake, 1998], in which the state is estimated by a set of particles. However in PF all the inference is performed globally, and the largely local structure is not exploited. In particular, when there are many entities the global state is high-dimensional, and PF will not perform well. Factored particle filtering [Ng *et al.*, 2002] is an approach that attempts to address this issue by decomposing particles into factors. However, all the dynamics propagation and conditioning on observations is still performed globally. As a result, factored PF still has a hard time inferring the correct local state from observations.

In *global/local particle filtering*, we introduce a simple inference principle: reason globally about global dynamics, and locally about local dynamics. Like in factored PF, particles are decomposed into factors. Unlike factored PF, however, only the dynamics propagation for globally influenced state is performed globally. Dynamics propagation for locally influenced state and conditioning on observations are performed locally. This allows global/local PF to more accurately take into account the observations when inferring the posterior distribution over the local state of each entity.

We present an application of our ideas to the task of monitoring goal formation of enemy units in an asymmetric urban warfare environment. In this task, there are a number of small, mobile enemy units moving around an urban environment. Sometimes the enemy units adopt the goal of attacking one of a number of possible targets. The units communicate with each other to form teams to attack a target. Our task is to detect when such teams have been formed, given the movements of the units and their communication.

We present experimental results for our application. Our results show that global/local PF considerably outperforms ordinary PF and factored PF at the task, and also outperforms a method that performs all its inference locally. Our method scales reasonably well both to situations with twenty units and to those with twenty target locations.

2 Global/local dynamic models

We are concerned with a dynamic system that evolves over time. We begin by describing a hidden Markov model (HMM)[Rabiner and Juang, 1986]. The state of the system at time t is represented by a variable X^t. At each time t there is an observation represented by the variable O^t. The dynamic system is defined by a transition model $P^t(X^t|X^{t-1})$; an observation model $P^t(O^t|X^t)$; and a distribution over the initial state $P^1(X^1)$. We do not assume that the system is homogeneous, i.e. the transition and observation models may vary from one time point to the next. The joint probability distribution over a sequence of states and observations is given by

$$P(x^1, o^1, ..., x^T, o^T) = P^1(x^1) \prod_{t=2}^{T} P^t(x^t|x^{t-1}) \prod_{t=1}^{T} P^t(o^t|x^t).$$

In a *global/local dynamic model (GLDM)*, we let $E_1, ..., E_n$ be a set of entities. We assume that the state of the system can be decomposed into a product of states of the individual entities. Each entity E_i has a *locally influenced state* U_i^t and a *globally influenced state* V_i^t. We denote the pair $\langle U_i^t, V_i^t \rangle$ by X_i^t and call it the *local state* of entity i at time t. Also the tuple $\langle U_1^t, ..., U_n^t \rangle$ will be denoted by $\mathbf{U}^t$ and $\langle V_1^t, ..., V_n^t \rangle$ by $\mathbf{V}^t$, and the variable representing the entire state by $\mathbf{X}^t$. Note that even the globally influenced state is called local state, because it pertains to a single entity. To summarize, the state space decomposes as

$$\mathbf{X}^t = \langle X_1^t, ..., X_n^t \rangle = \langle U_1^t, V_1^t, ..., U_n^t, V_n^t \rangle = \langle \mathbf{U}^t, \mathbf{V}^t \rangle.$$

In the transition model, the locally influenced state is constrained to depend only on the previous state of the individual entity, and the current globally influenced state of the entity. No restriction is made on the globally influenced state, and no decomposition of the distribution of that part of the state is assumed. Thus the transition model decomposes into

$$P^t(\mathbf{X}^t|\mathbf{X}^{t-1}) = P^t(\mathbf{V}^t|\mathbf{X}^{t-1}) \prod_{i=1}^{n} P_i^t(U_i^t|X_i^{t-1}, V_i^t).$$

Note that $P_i^t(U_i^t|X_i^{t-1}, V_i^t)$ may be different for different entities E_i.

The observation decomposes into a local observation O_i^t of each entity E_i that depends only on the local state of that entity. Thus the observation model decomposes into

$$P^t(\mathbf{O}^t|\mathbf{X}^t) = \prod_{i=1}^{n} P_i^t(O_i^t|X_i^t)$$

Just as the state of a HMM can be factored into the product of variables to produce a dynamic Bayesian network (DBN) [Dean and Kanazawa, 1989], so all the parts of a GLDM can be factored into variables. Thus the locally influenced state U_i^t is factored into variables $U_{i,1}, ..., U_{i,m}$, and similarly for the globally influenced state, and the local observation. The factoring may be different for different entities. The probabilistic models are factored into the product of conditional probabilities of variables given their parents in the usual way. The restrictions on the model are extended naturally from the non-factored case. The parents of a locally influenced variable may only be local variables of the same entity at the previous or current time step. The parents of a globally influenced variable may be any variable at the previous time step, and any globally influenced variable at the current time step. The parents of a local observation variable may be any locally or globally influenced variable of the same entity at the current time step.

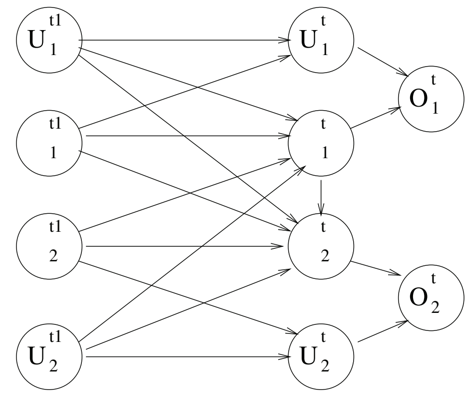

Figure 1: DBN representation of two-entity GLDM.

A GLDM can itself be viewed as a DBN, which is factored into variables representing the local states of individual entities, and the observations. This is the case even if the local states and observations are not themselves factored, as discussed in the previous paragraph. Such a DBN, for two entities, is shown in Figure 1. Note that there is an edge from V_1^t to V_2^t. No assumption is made that the V_i^t are conditionally independent.

For an example of a GLDM, consider a stock tracking and prediction application, where each entity may be a particular company. The locally influenced state may be the internal state of the company, while the globally influenced state may be the market conditions faced by the company. At each time point, the internal state of the company depends only on its previous internal state and on the market conditions it faces, whereas the market conditions are all dependent on all the previous market conditions. The observations may be the stock prices of individual companies.

For another example, consider an application of tracking the spread of an infectious disease through a population. Here the entities may be people, the locally influenced state may be the symptoms of the person, while the globally influenced state may be the stage of infection, if any, of the person. The transition model for the globally influenced state may specify that the infection stage of a person at time t depends on the infection stages at time $t-1$ of the people with whom the first person comes into contact at time t.

For a third example, which will be expanded on in Section 4, consider the task of monitoring enemy units moving around an urban environment. From time to time the units communicate and adopt goals of attacking possible targets. Here the locally influenced state of an entity may consist of its current position, while the globally influenced state is its goal. When a unit communicates with another unit, its goal becomes dependent on the goal and position of the other unit as well as itself. A unit's position, however, depends only on its previous position and its current goal.

Let us consider the expressive power of GLDMs. Trivially they can capture any HMM or DBN, since we can have a system with only one entity. It is more interesting to ask what can naturally be captured using the structure of the model. One question arises regarding the representation of a global state that applies to all entities, since each state in our representation is a local state of a particular entity. Global state can be captured in a GLDM by introducing an additional entity representing the global state, and making its state globally influenced. In the stocks example, there may be an entity representing general economic conditions, that influences the market conditions faced by each company. However, this is only legal if the global state does not influence locally influenced states (e.g. the internal states of the companies). Another question involves the representation of observations. There are no observations that depend on the state of more than one entity. We will see a way to get around this restriction in our application in Section 4. The reason for this restriction is that it will allow us, when performing inference as described in Section 3.1, to condition on observations locally.

GLDMs bear a superficial resemblance to factorial HMMs [Ghahramani and Jordan, 1997]. In a factorial HMM, there are a number of hidden state sequences, each of which evolves independently. The observation depends jointly on all the hidden states. In fact GLDMs and factorial HMMs are quite different. Factorial HMMs cannot easily be modeled as GLDMs, because the observation depends on the joint state of all entities and not on the hidden state of a single entity. On the other hand, in factorial HMMs all sequences evolve independently, so there is no globally influenced state; the state of each entity is completely locally influenced.

3 Inference

There are several inference tasks on dynamic systems, including diagnosing the past, predicting the future, and keeping track of the current state. In this paper, we will focus on the latter task, known variously as monitoring, filtering and state estimation. The filtering task is to compute, at each time point t, $P(X^t|o^1,...,o^t)$, where $o^1,...,o^t$ is the sequence of observations obtained up to time t. The quantity $P(X^t|o^1,...,o^t)$ is known as the belief state at time t. In principle, this can be computed simply using Bayesian updating:

$$P(X^t|o^1,...,o^t) \propto \sum_{x^{t-1}} P(x^{t-1}|o^1,...,o^{t-1})P^t(X^t|x^{t-1})P^t(o^t|X^t).$$

In practice this is very difficult because the state space may be very large. Even if the transition and observation models are represented in factored form as in a DBN, the belief state cannot be decomposed and must be represented as an explicit joint distribution over the state variables, which is exponential in the number of variables. The same holds for GLDMs. After a certain amount of time, the local states of all the entities become dependent on each other, and performing the filtering exactly requires a belief state which is a joint probability distribution over the local states of all entities. This is exponential in the number of entities.

Therefore approximate filtering algorithms are used. One standard algorithm for DBNs is the Boyen-Koller algorithm (BK) [Boyen and Koller, 1998]. In this algorithm, the variables of the DBN are partitioned into clusters. In a GLDM, each cluster will be the locally and globally influenced state of one entity. An approximate belief state $\hat{P}$ is maintained as a product of distributions $\hat{P}_i$ over the clusters, i.e.

$$\hat{P}(\mathbf{X}^t|\mathbf{o}^1,...,\mathbf{o}^t) = \prod_{i=1}^{n} \hat{P}_i(X_i^t|\mathbf{o}^1,...,\mathbf{o}^t).$$

In principle, the method works by beginning with the factored distributions $\hat{P}_i(X_i^{t-1}|\mathbf{o}^1,...,\mathbf{o}^{t-1})$, multiplying them to obtain $\hat{P}(\mathbf{X}^{t-1}|\mathbf{o}^1,...,\mathbf{o}^{t-1})$, propagating through the dynamics and conditioning on the observation to obtain $\hat{P}(\mathbf{X}^t|\mathbf{o}^1,...,\mathbf{o}^t)$, and then marginalizng onto the factors to obtain $\hat{P}_i(X_i^t|\mathbf{o}^1,...,\mathbf{o}^t)$. The joint distributions $\hat{P}(\mathbf{X}^{t-1}|\mathbf{o}^1,...,\mathbf{o}^{t-1})$ and $\hat{P}(\mathbf{X}^t|\mathbf{o}^1,...,\mathbf{o}^t)$ are not represented explicitly. Instead, the factored distributions $\hat{P}_i(X_i^t|\mathbf{o}^1,...,\mathbf{o}^t)$ are computed more efficiently. One way to do that is to create a junction tree representing two time slices of the DBN, in which each factor is contained in a clique at both the previous and current time points. In practice, even though this method often results in more efficient inference than the exact method, sometimes the cliques of the junction tree are too large and the method is still too expensive to be practical. This may particularly be a problem with some GLDMs, because no restrictions are placed on the way globally influenced variables evolve.

An alternative approach to approximate inference is to use particle filtering (PF) [Isard and Blake, 1998]. In PF, the joint distribution over the state variables is approximated by a set of samples, or *particles* as they are called. Each particle contains an assignment of values to the state variables. The probability of any property of the state is the fraction of particles that have that property. The basic steps of PF for a GLDM are as follows.

Begin with M particles $\mathbf{x}^{t-1,1}$, ...,$\mathbf{x}^{t-1,M}$.
For m = 1 to M:
  Propagate:
    Sample $\hat{\mathbf{v}}^{t,m}$ from $P^t(\mathbf{V}^t|\mathbf{x}^{t-1,m})$.
    For each entity E_i:
      Sample $\hat{u}_i^{t,m}$ from $P_i^t(U_i^t|x_i^{t-1,m}, \hat{v}_i^{t,m})$.
  Condition:
    $w_m \leftarrow \prod_{i=1}^n P_i^t(o_i^t|\hat{x}_i^{t,m})$.
  Resample:
    For $\ell = 1$ to M:
      Choose $\mathbf{x}^{t,\ell}$ from $\hat{\mathbf{x}}^{t,1}, ..., \hat{\mathbf{x}}^{t,M}$, with probability that $\hat{\mathbf{x}}^{t,m}$ is chosen being proportional to w_m.

The difficulty with PF for this problem is that the variance of the method is high and the number of particles required for a good approximation generally grows exponentially with the dimensionality of the problem. Therefore this approach does not scale well with the number of entities. An observation is that the different entities are somewhat independent of each other. The different entities interact with each other only through the globally influenced state. If these interactions are relatively weak, we might be able to take advantage of that fact. We might expect that instead of maintaining particles that assign values to all variables for all units, we can maintain local particles that only assign values to variables belonging to a single unit. This is the idea behind factored particle filtering [Ng *et al.*, 2002]. In factored particle filtering, the state variables are divided into factors. The joint distribution over all state variables is approximated by the product of marginal distributions over the factors, as in BK. Furthermore, the marginal factor distributions are approximated by a set of *factored particles*. Factored PF introduces two new steps into the PF process described above. The first joins factored particles together to produce global particles. The second projects global particles back down onto the factors. In between these two steps, all the usual steps of PF are performed. In particular, propagating through the dynamics and conditioning on the observations are done with global particles.

For this reason, ordinary factored PF is also not ideal for our situation. The problem is that in any global particle, it is highly likely that there will be some entities whose local state is far from the truth. Therefore, it will often be the case that for all global particles in the set of particles, the probability of the observation will be extremely low. Even if one entity's local state in the particle is good, other entities' states may be bad and so the observation will not confirm the first entity's state. As a result, inference about entities' true local states based on the observations will be poor.

3.1 Global/Local Particle Filtering

In order to allow observations about an entity to more effectively condition its local state, we introduce *global/local particle filtering*. Global/local PF is based on the principle of reasoning globally about global dynamics and locally about local dynamics. The method involves a simple change to factored PF, but one that makes a big difference. Instead of performing all the dynamics propagation globally, and conditioning on observations globally, and only then projecting down onto the individual factors, we project immediately after propagating the dynamics for the globally influenced variables. Propagation of dynamics for locally influenced variables and conditioning on observations are performed locally. The global/local PF process is as follows:

Begin with M factored particles $x_i^{t-1,1}, ..., x_i^{t-1,M}$ for each entity E_i.
Join the factored particles for different entities together to produce M global particles. (For details on the join process see [Ng *et al.*, 2002]).
For m = 1 to M:
  Propagate globally:
    Sample $\hat{\mathbf{v}}^{t,m}$ from $P^t(\mathbf{V}^t|\mathbf{x}^{t-1,m})$.
  Project:
    Project $\langle \mathbf{x}^{t-1,m}, \hat{\mathbf{v}}^{t,m} \rangle$ down to $\langle x_i^{t-1,m}, \hat{v}_i^{t,m} \rangle$ for each entity E_i.
For each entity E_i:
  For m = 1 to M:
    Propagate locally:
      Sample $\hat{u}_i^{t,m}$ from $P_i^t(U_i^t|x_i^{t-1,m}, \hat{v}_i^{t,m})$.
    Condition:
      $w_m \leftarrow \prod_{i=1}^n P_i^t(o_i^t|\hat{x}_i^{t,m})$.
  Resample:
    For $\ell = 1$ to M:
      Choose $x_i^{t,\ell}$ from $\hat{x}_i^{t,1}, ..., \hat{x}_i^{t,M}$, with probability that $\hat{x}_i^{t,m}$ is chosen being proportional to w_m.

Why does this method work? The key point is that in order for local propagation and conditioning to be successful, we don't need to have exactly the right joint distribution over all the globally influenced states. It is enough that the marginal distributions over individual entities' globally influenced states is approximately correct. If this happens, when we condition the local state of each entity on the local observation, we will produce an approximately correct posterior distribution over the local state. This is much easier to acheive than producing an approximately correct joint posterior distribution.

It is important to note that something is lost by propagating and conditioning locally. We lose the ability to reason from observations of one entity to another entity. For example, in the domain of detecting goals of enemy units, we are unable to reason about the fact that since unit 1 is moving towards target 4, and we have some reason to believe that unit 1 and unit 3 are on a team, then it is likely that unit 3 has a goal of attacking target 4. We do on the other hand successfully reason about the interaction between the units when reasoning globally, so we may infer that since unit 1 and unit 3 appear to have formed a team, they are both a priori likely to be attacking target 4. We are just unable to use the observation about unit 1 to confirm our beliefs about unit 3. The hope is that the inability to perform this type of reasoning is outweighed by the fact that inferences about individual units from their own observations are more accurate. The success of the global/local reasoning method will depend on this tradeoff, and how important this type of reasoning is in a particular application.

4 Application: Monitoring dynamic goals of enemy units

We have applied GLDMs to monitoring the dynamic goals of enemy units in an asymmetric urban warfare environment. In this scenario, units move about on a streetmap, and adopt goals of attacking one of a number of target locations. The motion of units depends on their goals; a unit with a goal of attacking a target will generally move in the direction of the target.

The goals of units can change dynamically. A unit may adopt a new goal in one of three ways: two units communicate and jointly agree to adopt a new goal; two units communicate and one invites the other to adopt its goal; or one unit spontaneously decides to adopt a goal. In all cases, the goal adoption decision is influenced by the proximity of the units to the goal; a unit will prefer to choose to attack a closer target.

Our task is to detect threats such as ambushes to targets, as soon as possible after they are formed. Two sources of evidence can be used. The first is a noisy sensor of the current position of each unit. The second is a noisy indication of whether or not a unit communicates. Communication provides some indication that two units are forming a team. This is a weak inference, however. Even when a unit communicates, usually the communication will not be related to goal adoption.

We model this scenario with a GLDM in which each entity corresponds to a unit. Since a unit's movement depends only on its own goal, the position of an entity is a locally influenced state. Units interact with other units in adopting goals, therefore the goal of a unit is a globally influenced state. In the dynamic model, the new goal of a unit depends on its previous goal and position, and the previous goal and position of a unit with which it communicates. Since this could be any unit, it depends on all the units' goals and positions.

The positional observation of each unit is a local observation. Whether or not two units communicate, however, depends on the goals and positions of both units, so the communication observations cannot be adequately captured in our framework as local observations. To get around this issue, we use the technique of *evidence reversal* [Kanazawa *et al.*, 1995]. Instead of having the observation depend on the state, we condition the transition model on the observation. That is, for each possible configuration of the communication sensors, we have a different transition model. This is allowed since we have not assumed that the dynamic model is homogeneous. In the inference process, we can sample from this conditioned dynamic model as follows: We first sample a configuration of actual communications given the noisy observations. We then sample which pairs of units communicate from the list of communicating units. For each pair of communicating units, we determine whether they jointly adopt a new goal or one joins the other's team, based on their previous goals and their proximity to the targets. Then, for each unit whose goal has not been determined by communication, we determine whether it spontaneously adopts a new goal.

5 Experiments

We tested the global/local particle filtering algorithm on simulated data generated from the model, and compared its performance to ordinary PF and factored PF. We also constructed an algorithm in which all interactions between units are ignored and all inference is performed locally, and compared our algorithm to that.

Each experiment consisted of one run of the system, with units moving about and choosing targets. Each run of the system lasted 100 time steps. A threat, which was defined to be four units sharing a common goal, was considered to be successfully detected if it was discovered within 12 time steps of its development. This was enough time for each unit to reach two intersections on average, and was considered to be the minimum amount of time in which our system could reasonably be expected to detect goal-directed behavior. If the threat was not detected within that time, the result was a false negative. If a threat was reported when none was present, the result was a false positive. For each experiment, we ran 500 runs and counted the number of true positives (TP), false positives (FP), and false negatives (FN). Our metrics are precision, which is $\frac{TP}{TP+FP}$, i.e the fraction of threats reported by the algorithm that were really threats, and recall, which is $\frac{TP}{TP+FN}$, i.e. the fraction of real threats caught by the algorithm. One parameter of the algorithm is the threshold of probability at which a threat is reported. We varied this probability in each experiment, thereby trading off precision for recall. In all experiments, we adjusted the number of particles allocated to each algorithm so that they all had approximately the same running time.

Figure 2(a) shows the precision-recall curves for each method for experiments with ten units and six target locations. The graph shows the recall that could be achieved for different levels of precision. Also shown for reference is the performance of random guessing. While all methods do better than random guessing, our method does best, getting much higher precision while still achieving high recall. At one point it achieves 56% precision with 87% recall. This is quite good performance considering the difficulty of the task. When there are a number of units moving about the map, it is quite likely that at some point in time several units will appear to be moving towards a target, even though in actual fact they have no intent of attacking it. Thus if we wish to achieve a high recall, we cannot avoid having a relatively low precision. Interestingly, factored PF performs very poorly, indicating that it is not simply the factoring that leads to the good performance of our method, but reasoning locally about unit positions. Also, note the relatively poor performance of the method that does all the reasoning locally. This shows that the global part of global/local PF is important.

Figure 2(b) shows how the algorithms scale up to a situation with 20 units. The task is more difficult, because there is more opportunity for units to appear to be heading towards a goal in the course of their business. Again our method does best, at one point achieving 57% precision with 76% recall. Figure 2(c) shows the performance when the number of targets is increased to 20. This is a much harder task, because some targets are close to each other and it is difficult to identify a unit's goals. Nevertheless, our method is able to achieve relatively good performance, getting 55% precision with 51% recall.

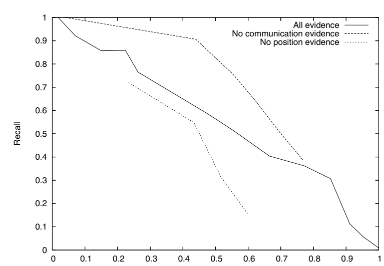

Figure 3: Comparison of performance without different evidence.

Figure 3 assesses the relative importance of each of the two sources of evidence, observations about the position of units, and about communications. We see that evidence from positional observations is more important, but taking communications into account is also useful. Surprisingly, the method that

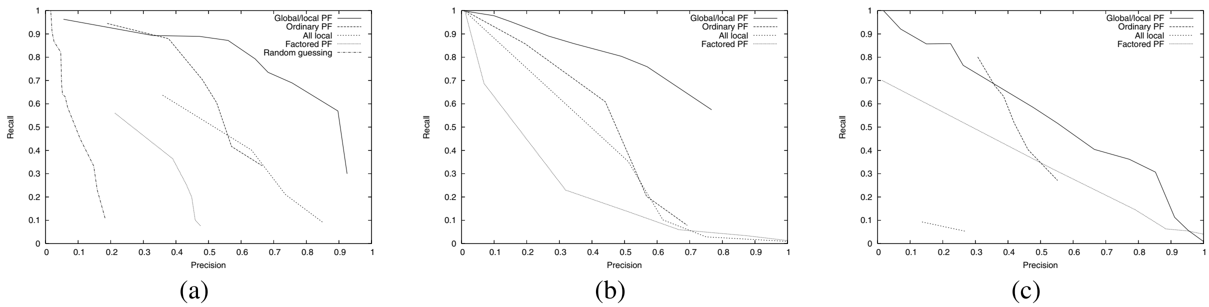

Figure 2: Comparison of methods: (a) 10 units, 6 targets; (b) 20 units, 6 targets; (c) 10 units, 20 targets.

does not take into account communication evidence performs better than the method that performs all the inference locally. The reason may be that although it does not take into account evidence pertaining to unit interactions, it still reasons about them and accounts for their possibility.

6 Conclusion

Many situations involve a number of entities that are largely independent of each other and only interact via a portion of their state. We have presented global/local dynamic models to represent these kinds of situation. We have presented global/local particle filtering, a monitoring algorithm based on the principle of reasoning globally about global dynamics and locally about local dynamics. We have applied our ideas to monitoring the goals of units in an urban warfare environment. We have shown experimentally that GLPF performs better than ordinary PF and other competitors on this application.

In future, it is important to explore whether global/local PF has benefits in other applications. We wish to explore ways in which GLDMs can be exploited for other types of inference algorithms. We would also like to extend our application to allow the number of units to change, and units to split or merge over time.

Acknowledgements

This work was funded by ONR contract N00014-05-C-0445, with thanks to Dr Wendy Martinez.

References

[Boyen and Koller, 1998] X. Boyen and D. Koller. Tractable inference for complex stochastic processes. In *Uncertainty in Artificial Intelligence*, 1998.

[Dean and Kanazawa, 1989] T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3), 1989.

[Ghahramani and Jordan, 1997] Z. Ghahramani and M.I. Jordan. Factorial hidden Markov models. *Machine Learning*, 29:245–273, 1997.

[Isard and Blake, 1998] M. Isard and A. Blake. Condensation — conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29:5–28, 1998.

[Kanazawa *et al.*, 1995] K. Kanazawa, D. Koller, and S. Russell. Stochastic simulation algorithms for dynamic probabilistic networks. In *Uncertainty in Artificial Intelligence*, 1995.

[Ng *et al.*, 2002] B. Ng, L. Peshkin, and A. Pfeffer. Factored particles for scalable monitoring. In *Uncertainty in Artificial Intelligence*, 2002.

[Rabiner and Juang, 1986] L. Rabiner and B. Juang. An introduction to hidden Markov models. *IEEE Acoustics, Speech and Signal Processing*, 3:4–16, 1986.