Factored Reasoning for Monitoring Dynamic Team and Goal Formation

Information Fusion 10 (2009) 99–106

Contents lists available at ScienceDirect

Information Fusion

journal homepage: www.elsevier.com/locate/inffus

Factored reasoning for monitoring dynamic team and goal formation

Avi Pfeffer [a,*], Subrata Das [b], David Lawless [b], Brenda Ng [c]

[a] School of Engineering and Applied Sciences, Harvard University, USA
[b] Charles River Analytics, USA
[c] Lawrence Livermore National Laboratory, USA

ARTICLE INFO

Article history:
Received 23 May 2006
Received in revised form 1 February 2008
Accepted 22 May 2008
Available online 14 July 2008

Keywords:
Particle filtering
Goal detection
Threat detection
Multiagent tracking

ABSTRACT

We study the problem of monitoring goals, team structure and state of agents, in dynamic systems where teams and goals change over time. The setting for our study is an asymmetric urban warfare environment in which uncoordinated or loosely coordinated units may attempt to attack an important target. The task is to detect a threat such as an ambush, as early as possible. We attempt to provide decision-makers with early warnings, by simultaneously monitoring the positions of units, the teams to which they belong, and the goals of units. The hope is that we can detect situations in which teams of units simultaneously make movements headed towards a target, and we can detect their goal before they get to the target. By reasoning about teams, we may be able to detect threats sooner than if we reasoned about units individually. We develop a model in which the state space is decomposed into individual units' positions, team assignments and team goals. When a unit belongs to a team it adopts the team's goal. An individual unit's movement depends only on its own goal, but different units interact as they form teams and adopt new goals. We present an algorithm that simultaneously tracks the positions of units, the team structure and team goals. Goals are inferred from two sources: individual units' behavior, which provides information about their goals, and communications by units, which provides evidence about team formation. Our algorithm reasons globally about interactions between units and team formation, and locally about individual units' behavior. We show that our algorithm performs well at the task, scaling to twenty units. It performs significantly better than several alternative algorithms: standard particle filtering, standard factored particle filtering, and an algorithm that performs all reasoning locally within the units.

1. Introduction

We study situations in which loosely coordinated agents dynamically form teams to achieve goals. The particular scenario that we study is an urban warfare environment, in which units collaborate to attack important targets. Our task is to simultaneously monitor the positions of units, the team structure and the goals as they evolve dynamically over time. Whatever method we use should scale up to a large number of units and targets.

We present a model for this scenario in which the state is decomposed into individual unit positions, team membership and team goals. Individual agents' movement is influenced only by their own goals, but agents interact by communicating with each other. A natural approach to inference in such a model is to use particle filtering (PF) [7,9,4], but the high dimensionality of the problem makes the number of particles needed for good monitoring very high. In particular, the probability that a particle will contain a good position estimate for all units will be very small, so all particles will have very small weight. Factored PF [11] represents the distribution over the state by a set of local particles for each unit. However it suffers from the same problem as PF because the local particles are always joined together before reasoning about the dynamics, and therefore reasoning about the dynamics is always done globally for all units together. We introduce a new inference principle: reason locally about unit positions, and globally about team structure and goals. By reasoning separately about each unit's position, we are able to effectively infer its goal from its behavior, which we would not have been able to do if we tried to reason about all units' positions together. By reasoning globally about team structure, we can capture interactions between the units.

We show experimentally that our global/local approach works very well. It significantly outperforms ordinary PF in which all the reasoning is done globally, and a method in which all reasoning is done locally. Our method scales up well to 20 units, and up to 20 targets. Several researchers have studied multiagent goal or plan recognition, but their focus has generally been different from ours. Most of the work looks at more complex goals and interactions between units, but the goals are static, and team structure is fixed in advance. For example, Intille and Bobick [6] recognize American football plays, while Devaney [3] recognizes military tactics involving many agents. Hongeng and Nevatia [5] provide hierarchical plan representations, as do Suria and Mahadevan [12], whose

* Corresponding author.
E-mail address: avi@eecs.harvard.edu (A. Pfeffer).

1566-2535/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.inffus.2008.05.005

approach is related to ours in that it uses dynamic Bayesian networks and particle filtering. In contrast to all of these, we allow agents to dynamically adopt goals and form teams as time elapses, but the plans are simpler. Liu and Chua [10] is more similar to our work, in that it does focus on dynamic goal formation. Their approach is based on HMMs, but they only demonstrate it for three units.

Other work does not consider the issue of goals, but does study tracking the movements of interacting agents. Kahn et al. [8] is similar to our work in that it proposes a particle filter to track multiple interacting targets. They use an MCMC-based approach to make the tracking problem tractable, whereas we apply the global/local principle. There are a number of differences between their work and ours. They consider only local interactions between targets that are close to each other, whereas in our work two far-apart targets may interact by sharing a common goal. Furthermore, because interactions are local in their work it is always known which targets are interacting, and the same targets interact in all particles, whereas we have uncertainty over which targets belong to the same team, and different particles have different interaction structure. Finally, team membership in our work is a long-running concept, to which there is no analogue in their work. Our work in this paper also greatly improves on our previous work [2]. In that work we studied the tracking of just two units using factored particle filtering. As we show in Section 5, that method does not scale up to a larger number of units. In this paper we improve on the method using the global/local principle, scaling up reasonably well to 20 units.

2. The problem

The setting is an urban environment in which there are a number of important locations, and some enemy units that may want to attack them. The goal of the system described in this paper is to detect malicious goals – when do the enemy units have a goal of attacking a location? In particular, we want to detect threats, when enemy units form teams to coordinate an attack on a location, such as an ambush. By detecting a threat early, we can enable friendly units defending the location to be more prepared, thereby potentially averting a dangerous situation. The hope is that we can detect situations in which teams of units simultaneously make movements headed towards a target, and we can detect their goal before they get to the target. By reasoning about teams, we may be able to detect threats sooner than if we reasoned about units individually.

The goal of a unit can be inferred from two basic sources of evidence. First, we can infer goals from behavior. This is the source of evidence normally used in goal recognition. In our scenario, the behavior of a unit is its movement. Thus we have to simultaneously track the location of a unit and reason about its goal. Since a unit that has a goal will generally move in the direction of the goal, our beliefs about its movement will affect our beliefs about its goal. At the same time, if we strongly believe the unit to be heading towards a particular goal, that will affect our tracking of the unit. The second source of evidence allows us to reason directly about team formation. Units will communicate with each other to form teams. Thus if we observe one unit communicating with another unit that we already believe intends to attack a location, our belief that the first unit will have the same goal will go up. Adequately reasoning about joint goals requires combining these two sources of evidence.

In our scenario there is a map consisting of streets and intersections. In our experiments, we use a map of central Baghdad that has 119 streets and 79 intersections. Each street has a "forward" and a "backward" direction. This does not mean the streets are one-way streets; a unit can travel in either the forward or backward direction. The directions are used to uniquely identify which way a unit is traveling at any point in time. Intersections on the map have an importance between 0 and 1; those with an importance >0 are potential targets. Units move around the map. At each point in time, we receive a noisy observation about the unit's position. We also receive an observation indicating whether or not the unit communicated. However we do not know with whom the unit communicated. This uncertainty about which units communicated with each other raises interesting inferential problems which will be addressed in Section 4.2.

3. The model

In the following presentation, units and teams are identified by integers. Units are indexed by i and j and teams are indexed by k. M is the number of units. Our model uses the following state variables:

- For each unit i, there is a position $\text{Position}_i = \langle \text{Street}_i, \text{Forward}_i, \text{Distance}_i, \text{Speed}_i \rangle$, where Street_i is the street the unit is on, Forward_i indicates whether the unit is moving in the forward or backward direction along the street, Distance_i is the distance the unit has traveled along the street, and Speed_i is the current speed of the unit.[1]
- Team_i is the team, if any, to which unit i belongs. The value is $\emptyset$ if i does not belong to a team. We assume that each unit belongs to at most one team. In some applications it may be desirable to allow a unit to belong to more than one team and have multiple concurrent goals, but allowing this would cause an exponential blowup in the number of possible team assignments of each unit. Relaxing this assumption is a topic for future research.
- TeamGoal_k is the goal of team k. A goal is any target location.

In addition to these state variables that influence and are influenced by the state at other time points, the model contains the following transient variables that only affect the current state:

- $\text{PairComm}_{i,j}$ is a flag indicating whether or not units i and j communicate.
- $\text{Content}_{i,j}$ describes the content of the communication, if any, between i and j. Its possible values are $\text{Invite}_{i \to j}$, which means that i is inviting j to join its team, $\text{Invite}_{j \to i}$, PairGoal[g], which means that i and j spontaneously adopt the new joint goal g, or NoContent, meaning that the communication is not related to forming a team to attack a target.
- $\text{Accepted}_{i,j}$ is a flag indicating whether or not the content, if any, of the communication between i and j was accepted.
- SingleComm_i is a flag indicating whether unit i communicates at all.
- SingleGoal_i is the new goal, if any, spontaneously adopted by a unit, not as a result of communication. It may be $\emptyset$, or any of the possible targets.
- Goal_i is the resulting goal, if any, of unit i, after any new goal has been adopted or old one maintained. The value is $\emptyset$ if i has no goal.

In our observation model, we assume that each unit is associated with a single positional observation. By no means do we consider the data association model to be solved; we acknowledge that it is a hard problem. However, the main goal of this work is

[1] To avoid confusion, we emphasize that there is not a variable for every street. Instead, there is a Street_i node for each unit i, whose possible values are numbers from 1 to 129, indicating the identity of the street the unit is on.

to focus on monitoring team and goal formation. For this purpose we assume that the data association is performed as a preprocessing step. It would be interesting to investigate simultaneously monitoring team and goal formation and performing data association but that is beyond the scope of the current work.

At each point in time, the following observations are received:

- A noisy observation $\text{ObsPos}_i = \langle X_i, Y_i \rangle$ of the coordinates of unit i. The positional observation is generated by a Gaussian distribution centered around the true position, with the x and y components independent. Our framework can easily accommodate more complex models for the positional observation. All that is required is that we can determine the density of the observation given the true position. Any model that satisfies this property can be used.
- An observation ObsComm_i indicating whether or not unit i communicated. When a communication does not actually happen, there is a small probability that a communication is erroneously reported. When a communication actually happens, it is detected with high probability.

The probabilistic model is represented using a dynamic Bayesian network (DBN). Rather than presenting the entire model in one go, we present it in fragments. We use the notation X^t to denote the value of variable X at time t. Fig. 1a shows the fragment describing the model of individual unit positions. The key point about this model is that an individual unit's position at time t depends only on its own goal at time t and its own position at time $t-1$. Also, the positional observation at time t depends only on the position at time t. The team structure and all other units' goals have no direct influence on the individual unit's movement. Of course, they do have indirect influence by influencing the unit's goal.

The goal affects the new position by influencing the unit's decisions when it reaches an intersection or might make a U-turn. The desirability of moving in a certain direction depends on the distance to the goal, as determined by following streets on the map. A unit heading towards a target will prefer to take the most direct route. When a unit reaches an intersection, the probability that it chooses a street is higher if the distance to the target along that street is shorter, but there is still some uncertainty. At the beginning of each iteration, a unit may choose to make a U-turn. The probability of this is small unless the unit has a goal and the route to the goal is shorter if it makes the U-turn. The dynamics of movement are defined as follows:

1. The unit may optionally execute a U-turn at the beginning of the iteration. If the unit does not have a goal, this happens with very small probability. If the unit has adopted a goal, and the route to the target is shorter, a U-turn happens with larger probability. In that case, it makes the U-turn with probability $\frac{qd_1^{-\beta}}{qd_1^{-\beta}+d_2^{-\beta}}$, where d_1 is the distance to the goal after making the U-turn, d_2 is the distance without the U-turn, and q is a small constant. A U-turn takes time, and the unit begins traveling in the opposite direction at low speed.
2. The unit then moves in whichever direction it is heading, beginning Distance^{k-1} along the street and traveling at Speed^{k-1}. The potential distance along the road at the end of the iteration is computed, with some Gaussian noise: $\overline{\text{Distance}}^k = \text{Distance}^{k-1} + \text{Speed}^{k-1} + \epsilon$.
3. If $\overline{\text{Distance}}^k$ is less than the length of the street, the unit does not reach an intersection, and Distance^k becomes $\overline{\text{Distance}}^k$. A new speed is computed by mixing the beginning speed with a target speed and adding Gaussian noise: $\text{Speed}^k = \theta\ \text{Speed}^{k-1} + (1-\theta)\text{Target speed} + \epsilon$.
4. If $\overline{\text{Distance}}^k$ is greater than the length of the street, the unit reaches an intersection. If the unit does not have a goal, the next street is then chosen with uniform probability. If the unit has a goal, it chooses to take street j with probability proportional to $d_j^{-\beta}$, where d_j is the distance to the goal taking the shortest route beginning with street j. When the unit makes a turn, the time elapsed before reaching the turn is computed, and the unit proceeds to move for the remaining time in the new direction, starting at a slow speed. A new speed is then computed as in step 3.

The movement model presented here is quite abstract. It could be refined in various ways. For example, we could introduce acceleration into the state, or we could model the fact that units slow down before intersections and U-turns. We could also introduce a non-uniform distribution over the street a unit chooses at an intersection when it has no goal, making straight more likely. Our inference approach can easily be adapted to a more refined movement model, as long as it maintains the property that the new position depends only on the old position and the unit's goal, and as long as it is possible to easily sample a new position given an old position. The problem may become more difficult, if it becomes more difficult to distinguish dangerous behavior from ordinary behavior, but the principle will remain the same: to reason locally about unit's behavior as much as possible, and globally about unit interactions.

Fig. 1b shows the DBN fragment defining the current goal of a unit. Goal_i depends on Team_i and all of the team goals. It is fully determined by its parents by the fact that the goal of i is the team goal of Team_i. The conditional probability distribution of Goal_i is a multiplexer, where Team_i selects which parent's value to use. Fig. 2 shows the DBN fragment defining a single unit's communication. SingleComm_i depends on all the $\text{PairComm}_{i,j}$ and $\text{PairComm}_{j,i}$ where $j \neq i$. It is fully determined by its parents: tautologically, a unit communicates if and only if it communicates with someone else. ObsComm_i is a noisy observation of SingleComm_i.

Fig. 3 shows a DBN fragment for determining which team a unit belongs to, and what the team goal is. First an explanation of the notation: $\text{TeamGoal}^t_{\text{Team}^t_i}$ means the goal, at time t, of the team to which i belongs at time t. $\text{TeamGoal}^{t-1}_{\text{Team}^t_i}$ means the goal, at time $t-1$, of the team to which i belongs at time t. Note that this might not be i's goal at time $t-1$, if i belonged to a different team at time

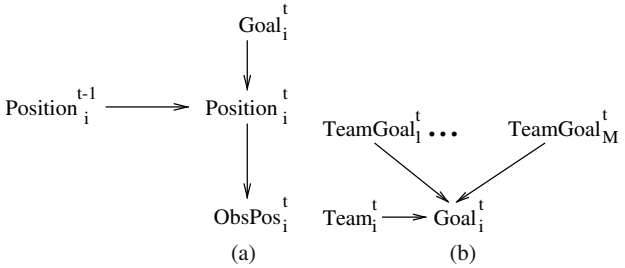


Fig. 1. DBN fragments for (a) movement model; (b) individual goal model.

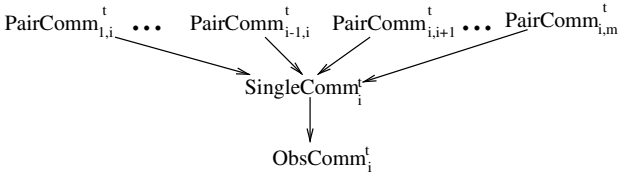


Fig. 2. DBN fragment for communication observation.

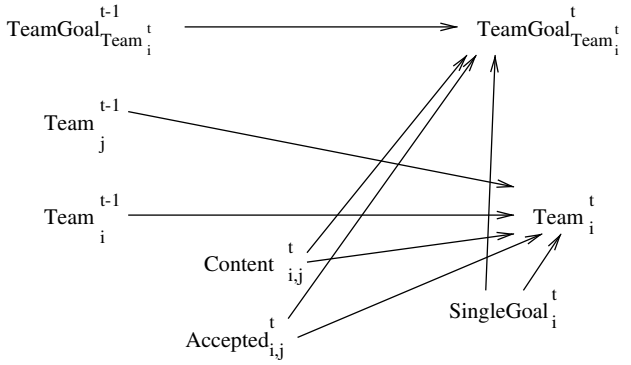


Fig. 3. DBN fragment for Team and TeamGoal variables.

$t-1$. We make the assumption that no unit can communicate with more than one other unit at any point in time. This assumption is appropriate if communication is by cellular phone, but may not be appropriate for other communication methods such as broadcast communications. The assumption greatly simplifies the reasoning process. As a result of this assumption there is at most one j for which $\text{Content}_{i,j}$ is interesting. (For convenience we will always refer to the variable as $\text{Content}_{i,j}$ even though i might actually be the second subscript.) That is the j that is relevant in deciding the new team of a unit. The team of a unit can change in one of three ways. First, a unit may accept an invitation from another unit to join its team. This is indicated when $\text{Content}_{i,j}$ is $\text{Invite}_{j\to i}$, and $\text{Accepted}_{i,j}$ is true. In this case Team_i^t becomes Team_j^{t-1}. $\text{TeamGoal}^t_{\text{Team}_i^t}$ is the same as $\text{TeamGoal}^{t-1}_{\text{Team}_i^t}$. This is because i belongs to j's team at time t (since it has just accepted an invitation from j), so Team_i^t is the same as Team_j^t, and the goal of j's team has not changed. Second, two units may spontaneously decide to form a team and adopt a new goal. This is indicated when $\text{Content}_{i,j}$ is PairGoal[g] and $\text{Accepted}_{i,j}$ is true. Third, a unit may, on its own, spontaneously decide to adopt a new goal g and form a team consisting of a single unit. This is indicated when SingleGoal $= g \neq \emptyset$. In both the second and third cases, Team_i^t was previously an empty team that did not contain any units. $\text{TeamGoal}^t_{\text{Team}_i^t}$ becomes g. In all of the above cases, the unit may previously have belonged to another team. If none of these happen, the team to which the unit belongs will stay the same as before, and its goal will stay the same. This model determines the teams of all units, and the goal of all teams to which at least one unit belongs. Since more than one unit may belong to a team, a team goal may be determined multiple times. However, no contradiction can arise. The only way a team goal can change is if one or two units spontaneously create it. But in that case the team must have been empty before the units joined it.

Fig. 4 shows a DBN fragment for determining whether the content of a communication is accepted and whether a unit spontaneously adopts a goal. The fitness of a goal is equal to h_g/d_g where h_g is the importance of target g, and d_g is the distance of the unit to target g. Whether or not a communication is accepted depends on its content. If the content is $\text{Invite}_{j\to i}$, the probability depends on the fitness of j's goal to unit i. If the content is PairGoal[g], the probability depends on the product of the fitness of g to both units. In all cases, the probability of acceptance is smaller if the unit previously had a goal, in which case the probability decreases with the fitness of the old goal.

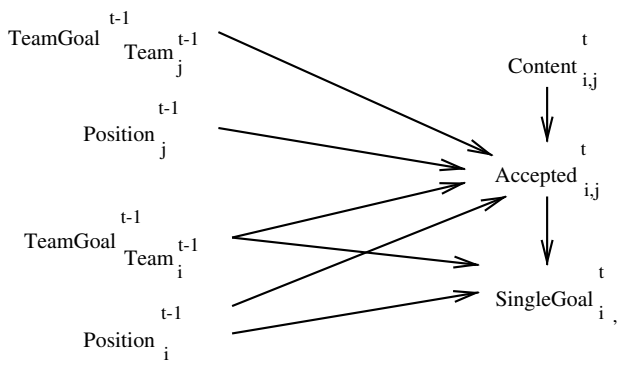


Fig. 4. DBN fragment for invitation acceptance and spontaneous goal formation.

This DBN fragment also defines the model for SingleGoal_i. If the content of any communication that i was involved in was accepted, i cannot spontaneously adopt a new goal. Otherwise the probability that it will adopt a new goal g is proportional to the fitness of g. The overall probability that a unit will spontaneously adopt a new goal is generally small but not negligible. Again, it is smaller if the unit already had a goal, the probability is proportional to the fitness of the new goal and inversely proportional to the fitness of the old goal.

We do not show the DBN fragment for determining whether or not two units communicate and what the content of the communication is. The reason is that all of the $\text{PairComm}_{i,j}$ and $\text{Content}_{i,j}$ variables must be determined together. This is due to the constraint that a unit cannot communicate with more than one other unit. Therefore we define a probability distribution over all legal complete communication configurations, i.e. all complete assignments to PairComm and Content. For each pair of units i and j, we define $\text{Score}(\text{PairComm}_{i,j},\text{Content}_{i,j})$. The score of a complete configuration will then be the product of scores for all possible i,j pairs.

$\text{Score}(\text{true},\text{Invite}_{j\to i})$ is 0 if j does not have a goal, otherwise it increases with the fitness of j's goal to i. Similarly, Score(true, PairGoal[g]) depends on the fitness of g to both units. Score(true, NoContent) is a fixed constant. Finally, we let Score(false,NoContent) be 1. Letting C be the collection of i,j pairs that communicate, this allows us only to consider communicating units when determining the probability:

$$P(C) \propto \prod_{\langle i,j\rangle} \text{Score}(\text{PairComm}_{i,j}, \text{Content}_{i,j}) = \prod_{\langle i,j\rangle \in C} \text{Score}(\text{true}, \text{Content}_{i,j})$$

4. Inference

Our task is to simultaneously track the unit positions, team structure and team goals. At each point in time we want a probability distribution over these variables given the history of position and communication observations. Doing this exactly requires maintaining a complete joint distribution over all state variables that influence the next time step. Considering the discrete variables alone, if the number of targets is L and the number of units is M, then there are $(2 \cdot 119 \cdot (L+1))^M$ discrete states, since a unit may be traveling forwards or backwards along any of the 119 streets, and its goal may be $\emptyset$ or any of the targets. Clearly, maintaining an explicit joint distribution over all these states is infeasible for more than two units.

Therefore we need to use an approximate monitoring algorithm. The basic framework for our algorithm is *particle filtering (PF)* [4]. 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. In particular, we are concerned with the probability that a threat exists to a target – this is simply the fraction of the particles for which such a threat exists. The basic

steps of PF are as follows. Let **x** denote the state variables, **v** the transient variables, and **y** the observations:

Begin with N particles $\mathbf{x}^{t-1,1}, \ldots, \mathbf{x}^{t-1,N}$.
For $n = 1$ to N:
  Propagate: Sample values for $\mathbf{v}^{t,n}$ and $\hat{\mathbf{x}}^{t,n}$ from $P(\mathbf{v}^{t,n}, \mathbf{x}^{t,n} | \mathbf{x}^{t-1,n})$.
  Condition: $w_n \leftarrow P(\mathbf{y}^t | \mathbf{v}^{t,n}, \hat{\mathbf{x}}^{t,n})$.
Resample: For $m = 1$ to N:
  Sample $\mathbf{x}^{t,m}$ from $\hat{\mathbf{x}}^{t,1}, \ldots, \hat{\mathbf{x}}^{t,N}$, with the probability that $\hat{\mathbf{x}}^{t,n}$ is chosen being proportional to w_n.

Propagating the particles through the dynamics and conditioning on the observations is non-trivial for our problem and will be discussed in Section 4.2.

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 units. An observation is that the different units are largely independent of each other. The movement of the different units is assumed to be independent; they only interact with each other by communicating and inviting each other to join teams or forming new teams together. Therefore 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 [11].

The basic premise of factored particle filtering is that all the state variables in the domain can be divided into a set of factors. The joint distribution over all state variables is approximated by the product of marginal distributions over the factors, in the style of Boyen and Koller [1]. In our application, the choice of factors is obvious: the variables pertaining to a single unit correspond to a factor. We believe that in many applications the correct factorization will be apparent and can be supplied by hand. We are currently looking into ways of factorizing a complex process algorithmically by automatically discovering weakly interacting components.

In addition to decomposing the state into factors, factored particle filtering additionally approximates the marginal factor distributions using 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. This step will be discussed in more detail in Section 4.1. 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 units whose position 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 positional observation will be extremely low. Even if one unit's position in the particle is good, other units' positions may be bad and so the observation will not confirm the first unit's position. As a result, inference about units' true positions based on the global positional observations will be poor. Furthermore, units' positions are important indications about their goals, so the reasoning about units' goals will also be poor.

One might suggest that since this is the case, doing global reasoning at all is a bad idea, and all the reasoning should be done locally. However, that will not allow us to reason about unit interactions and communication between units. Therefore we present an approach in which we combine global and local reasoning. We reason globally about unit interactions and team formation, and locally about the positions and individual goals of units. To avoid confusion, we use the notation $P_i^{t,n}$, $T_i^{t,n}$ and $G_i^{t,n}$ to refer to unit i's position, team and goal at time t in the nth local particle belonging to factor i, whereas $\text{Position}^{t,n}$ will stand for the vector of unit positions at time t in the nth global particle. The process is as follows:

For each factor i, begin with N factored particles $\langle P_i^{t-1,n}, T_i^{t-1,n}, G_i^{t-1,n} \rangle$.
Join the factors together to produce N global particles $\langle \text{Position}^{t-1,n}, \text{Team}^{t-1,n}, \text{TeamGoal}^{t-1,n} \rangle$, each with weight w^n.
Propagate goals: For $n = 1$ to N:
  Sample $\langle \text{PairComm}^{t,n}, \text{Content}^{t,n} \rangle$ given $\langle \text{Position}^{t-1,n}, \text{Team}^{t-1,n}, \text{TeamGoal}^{t-1,n} \rangle$.
  Compute $\text{SingleComm}^{t,n}$ from $\text{PairComm}^{t,n}$.
  $w^n \leftarrow w^n \prod_{i=1}^{M} P(\text{ObsComm}_i^t | \text{SingleComm}_i^{t,n})$. (*)
  Sample $\text{Accepted}^{t,n}$ given $\text{Content}^{t,n}$ and $\langle \text{Position}^{t-1,n}, \text{Team}^{t-1,n}, \text{TeamGoal}^{t-1,n} \rangle$.
  Sample $\text{SingleGoal}^{t,n}$ given $\text{Accepted}^{t,n}$ and $\langle \text{Position}^{t-1,n}, \text{Team}^{t-1,n}, \text{TeamGoal}^{t-1,n} \rangle$.
  Compute $\text{Team}^{t,n}$ and $\text{TeamGoal}^{t,n}$ from $\langle \text{Content}^{t,n}, \text{Accepted}^{t,n}, \text{SingleGoal}^{t,n} \rangle$ as well as $\langle \text{Team}^{t-1,n}, \text{TeamGoal}^{t-1,n} \rangle$.
Project: For $i = 1$ to M:
  For $n = 1$ to N:
    Project $\langle \text{Team}^{t,n}, \text{TeamGoal}^{t,n} \rangle$ to obtain $\langle T_i^{t,n}, G_i^{t,n} \rangle$.
  $w_i^n = \sqrt[M]{w^n}$.
Propagate positions: For $n = 1$ to N:
  Sample $P_i^{t,n}$ given $\langle P_i^{t-1,n}, G_i^{t,n} \rangle$.
  $w_i^n \leftarrow w_i^n P(\text{ObsPos}_i^t | P_i^{t,n})$.
Resample locally in each factor.

The weights for communication observation in (*) could also be computed locally, but in fact it does not make a difference because in Section 4.2 we show how we avoid computing these weights altogether.

4.1. Joining factored particles

As discussed earlier, the factored inference approaches introduce a join step and a projection step into the PF framework. The projection step is straightforward. A global state uniquely defines a local state for each unit. However the join step is more difficult. The join consists of all global particles, such that the projections of the global particle onto each of the factors is consistent with a local particle for that factor. This join is too large to store or compute, so instead we generate N samples from the result of the join. Since different units may belong to the same team, and therefore must have the same goal, we must make sure that the particles chosen from different factors are consistent with each other.

In Ng et al. [11], an importance sampling process was presented for sampling from the join. The process samples a particle from each factor in turn. When it samples a particle from a factor, it forces the variables that overlap with other factors to have the same value as previously sampled. Thus only a subset of the available particles are considered for sampling. To compensate for this, the method introduces an importance weight equal to the fraction of particles in the new factor that are consistent. The result of the process is a global particle with an importance weight.

Our method is similar, but there is a twist. Ng et al. [11] assumed that the overlapping variables between different factors are fixed and known in advance. This assumption is not valid here. Instead, the overlap is determined by the individual particles

chosen by sampling. Whether or not the factors for units 1 and 2 overlap depends on whether the units are on the same team in the chosen particles. This means that we cannot predefine a join process in which we know in advance which variables to constrain at each point in the process. We cannot optimize the join process to ask a set of predetermined queries. We have to be flexible and determine on the fly exactly how the different factors join together. We define a particle-dependent process for producing a single weighted sample from the join as follows. Notation: $P[V]$, $T[V]$ and $G[V]$ denote the values of P,T and G in particle V.

Set $\text{TeamGoal}_k = \emptyset$ for all k.
Let $w = 1$.
For $i = 1$ to M:
  Let $\mathbf{V}_i = \emptyset$.
  For $n = 1$ to N:
    If $\text{TeamGoal}_{T_i^n} = \emptyset$ or $G_i^n = \text{TeamGoal}_{T_i^n}$
      Add $\langle P_i^n, T_i^n, G_i^n \rangle$ to $\mathbf{V}_i$.
  Sample a factored particle V_i from $\mathbf{V}_i$.
  $\text{Position}_i = P[V_i]$.
  $\text{Team}_i = T[V_i]$.
  If $T[V_i] \neq \emptyset$
    $\text{TeamGoal}_{T[V_i]} = G[V_i]$.
  $w \leftarrow w \frac{|\mathbf{V}_i|}{N}$.
Return $(\langle \text{Position}, \text{Team}, \text{TeamGoal} \rangle, w)$.

Unfortunately, the time complexity of the sample-join is $O(N^2M)$. We need to perform the importance sampling process once for each of the N particles produced. The process goes through all M units, and for each unit needs to go through a process of deciding which of the N factored particles are consistent with the previously sampled factored particles. One thing that might be done is to precompute, for each factor, all sets of particles consistent with each possible assignment of goals in previous factors, so that when we need to sample from the factor, we know immediately which particles to use. However, this must be done for all possible previous team goal assignments, which is exponential in the number of units. In comparison, the cost of PF is $O(MN\log N)$. This is an advantage of PF, but PF may require many more particles.

4.2. Sampling communication configurations

A key step in the sampling process is to sample a communication configuration, given the observation about which units have communicated. Of course, we cannot possibly enumerate all possible communication configurations, so we cannot compute the probability of any single configuration. We cannot directly generate a sample configuration, because each pair of units' communication depends on all the other pairs. Furthermore, even if we could generate a configuration, the probability of the observed communication would be very low for all but a small fraction of configurations. We could use Markov chain Monte Carlo techniques to sample a particular configuration, given the observations. However each sample would be quite costly to generate. Instead, we use the technique of evidence reversal [9], together with approximating assumptions about the model, to generate samples directly given the evidence.

The first assumption is that every time a unit communicates, the communication sensor will detect it.[2] It is still possible for the sensor to report a communication when there is none. Making this assumption allows us to restrict attention to units that we have observed communicating. Furthermore, the only pairs we need to consider are those where we have observed both units communicating. The second assumption is that the fitness of goals does not determine whether or not units communicate in the first place, only the content of the communication and whether or not it is accepted. As we said earlier, the goal fitness shows up in multiple places in the model. Therefore ignoring it in one place does not mean that we are ignoring it altogether. We emphasize that these assumptions are made only by the reasoning process, not in the model itself.

Now, given a set of observed communicating units, we want to sample a set of pairwise communications. Each assignment will match some pairs of units that have been observed to communicate, while others will be left unmatched. Under the assumptions, any pairwise assignment with the same number of matching pairs must have the same probability. How do we ensure this? We introduce a parameter p_i, which intuitively represents the probability that a unit actually communicates with one particular other unit when there are i units to communicate with. Thus the total probability that a unit communicates with any other unit is $i\ p_i$. Let $q_i = 1 - i\ p_i$; thus q_i is the probability that a unit does not communicate with any other unit when there are i units to communicate with.

Now suppose we start with one unit, and do not match it with any other unit. This happens with probability q_i. We then go to another unit, and match it with a third unit. This happens with probability p_{i-1}. That gives us a configuration of three units with one unmatched and two matched. Now suppose that instead we match the first unit with a second unit, which happens with probability p_i, then go to a third unit and do not match it, which happens with probability q_{i-2}. That also gives us a configuration of three units with one unmatched and two matched. No matter what configuration we choose for the remaining units, either of these starts will produce the same number of matching pairs in the total configuration. Thus both these starts must have the same probability. This implies that $p_i q_{i-2} = q_i\ p_{i-1}$, for $i \geqslant 2$. Therefore $p_i = \frac{q_i p_{i-1}}{q_{i-2}}$. Substituting in the definition of q_i, we have

$$p_i = \frac{(1 - ip_i)p_{i-1}}{1 - i - 2p_{i-2}}$$

Solving yields the recurrence relationship

$$p_i = \frac{p_{i-1}}{1 - (i-2)p_{i-2} + ip_{i-1}} \quad \text{for } i \geqslant 2$$

To start the recurrence, we need a base case p_1, which is a free parameter. p_1 can be understood as the probability that one unit communicates with another unit when there are no other units to communicate with. We do not need p_0 to derive p_2 because p_{i-2} is multiplied by $i - 2$ in the formula. The values of the p_i are precomputed before any monitoring begins.

The process for sampling a configuration is now easy. At the beginning, each unit is marked as unprocessed. We start by considering a unit. Let the number of other units be m. With probability q_m we do not match the unit being considered to any unit. Otherwise we choose another unit to match it to uniformly at random; thus each other unit has probability p_m of being chosen. We then mark the unit as being processed. If we matched it to another unit, we also mark the other unit as being processed. We then repeat the process until all units have been processed. The following pseudocode describes the process:

Pairs $\leftarrow \emptyset$
For each unit i:
  Mark i as unprocessed.
Repeat until all units have been processed:
  If there is only one unprocessed unit i:
    Mark i as processed.

[2] This does not contradict the statement in Section 3 that a communication is observed with high probability, but not all the time. The assumption made here is an approximation to the truth, made for the purpose of making the sampling of communication configurations tractable.

Else:
  Choose an unprocessed unit i uniformly at random.
  Let the number of other unprocessed units be m.
  With probability q_m:
    // i is not matched to any other unit
    Mark i as processed.
  Otherwise:
    // i is matched to some other unit
    Choose an unprocessed unit j different from i uniformly at random.
    $Pairs \leftarrow Pairs \cup (i,j)$.
    Mark i and j as processed.
Return *Pairs*.

5. Experimental results

We tested our algorithm on simulated data generated from the model, and compared its performance to ordinary PF, factored PF, and an algorithm that performs all the reasoning locally and never joins the factored particles. 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. 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, except experiment (d), we ran 500 runs and counted the number of true positives (TP), false positives (FP), and false negatives (FN). Experiment (d) used 100 runs. 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. In each experiment we varied the threshold of probability required for an algorithm to report a threat, 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. The first column of Fig. 6f shows the number of particles used for most of our experiments; the second column shows the number used for experiment (d).

Fig. 5a 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 (standard error 3.7%). A system with this level of performance would be very useful in practice. 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, the relatively poor performance of the method that does all the reasoning locally shows the importance of reasoning globally about unit interactions.

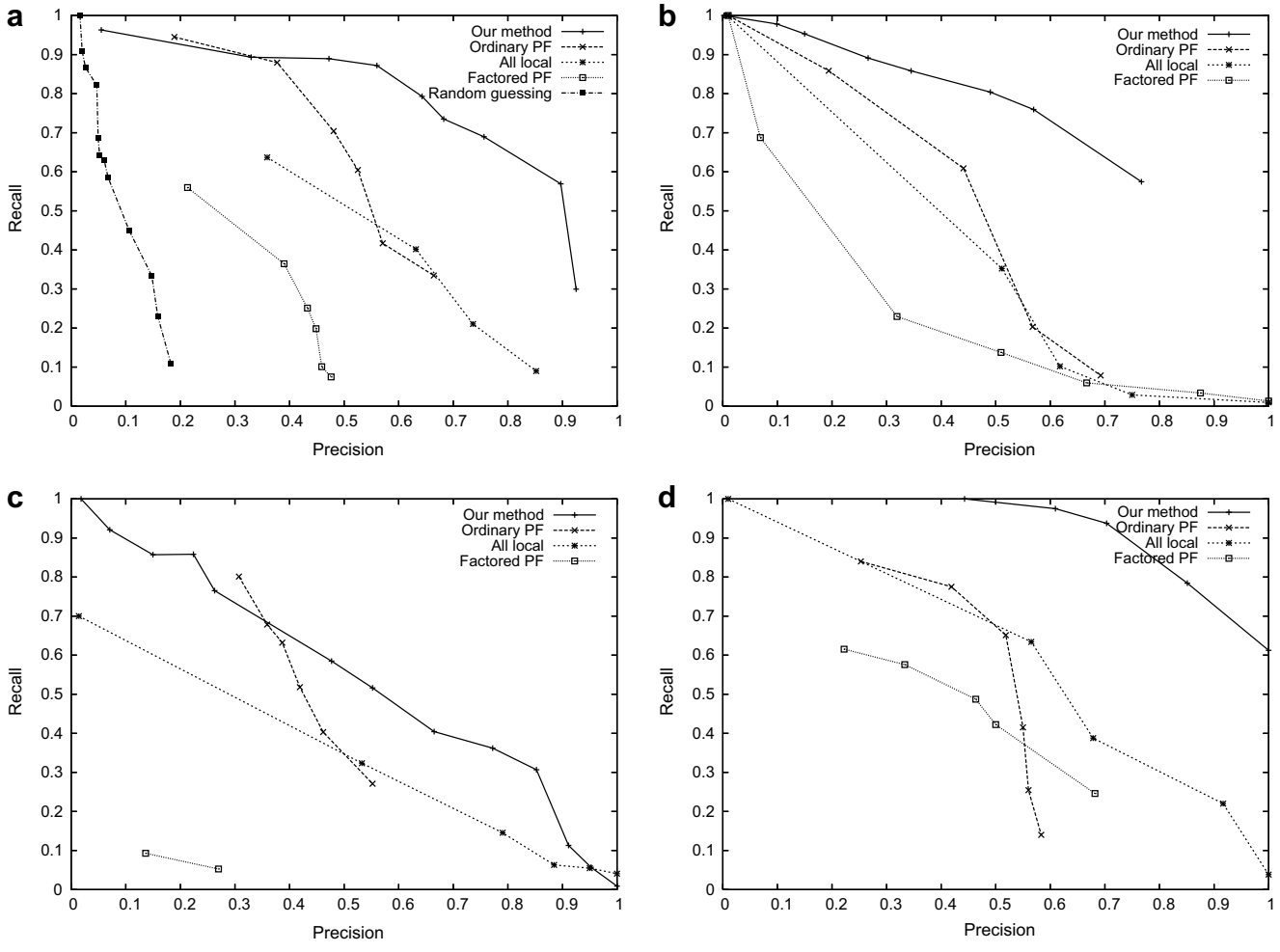
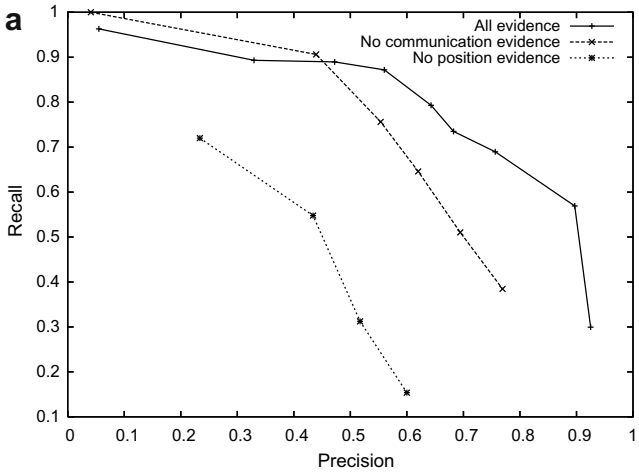


Fig. 5. Comparison of methods: (a) 10 units, 6 targets; (b) 20 units, 6 targets; (c) 10 units, 20 targets; (d) 10 units, 6 targets with more particles.

a



b

	(a)	(d)
Our method	100	300
Ordinary PF	190	1550
Factored PF	115	300
All local	120	880
Time per iteration	0.25	2

Fig. 6. (a) Comparison of performance of our method with all evidence and without different sources of evidence; (b) number of particles used and running time per iteration in seconds in the different experiments.

Fig. 5b shows how the algorithms scale up to a situation with 20 units. Again our method does best, at one point achieving 57% precision with 76% recall (standard error 4.0%). Fig. 5c 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 reasonably good performance, at one point getting 55% precision with 51% recall (standard error 4.5%).

As we discussed, one of the disadvantages of our method is that it is quadratic in the number of particles. This might lead us to believe that if we allocated more particles to the different algorithms, the gap between them would close. Fig. 5d shows that this is not the case. The performance of our algorithm improves significantly with more particles. Strangely, ordinary PF does somewhat worse, although this may be noise resulting from the fact that only 100 runs were used for this experiment. The standard errors for this curve range from 8 to 10%, indicating that there is a fair amount of noise, but the trend is still clear. At the very least, we can say that being able to use many more particles does not seem to provide an advantage to PF. Fig. 6a assesses the relative importance of each of the two sources of evidence. We see that evidence from positional observations is more important, but taking communications into account is also useful. Surprisingly, the method that does not take into account communication evidence performs better than the all local method, perhaps because it still considers potential unit interactions.

6. Conclusion

We have presented a new model of dynamic team and goal formation and an algorithm for dynamically monitoring the positions of units, the team structure and goals, and applied them to an asymmetric urban warfare domain. Our method, based on the principle of reasoning locally about individual units' actions and globally about unit interactions, has been shown to be successful. This principle is a general one, and can be applied to any situation in which units operate individually but interact with each other.

We have shown that our method scales up to reasonably large situations, involving up to 20 units or 20 targets. An important next step is to try to further scale up our method, to situations involving possibly hundreds of units. Another next step is to extend the method to situations in which the number of units changes, and units can split or combine dynamically. We would also like to allow different units to have different roles on a team. Our basic principle should still hold in both cases. It would also be good to find a way to approximately compute the join in time that is less than quadratic in the number of particles.

Acknowledgements

This work has been performed under several contracts funded by ONR and OSD, with special thanks to Dr. Wendy Martinez.

References

[1] X. Boyen, D. Koller, Tractable inference for complex stochastic processes, in: Uncertainty in Artificial Intelligence (UAI), 1998.
[2] S. Das, D. Lawless, B. Ng, A. Pfeffer, Factored particle filtering for data fusion and situation assessment in urban environments, in: International Conference on Information Fusion, 2005.
[3] M. Devaney, Plan recognition in large-scale multi-agent tactical domains, PhD Thesis, College of Computing, Georgia Institute of Technology, 2003.
[4] A. Doucet, N. de Freitas, N. Gordon (Eds.), Sequential Monte Carle Methods in Practice, Springer-Verlag, 2001.
[5] S. Hongeng, R. Nevatia, Multi-agent event recognition, in: International Conference on Computer Vision, 2001.
[6] S.S. Intille, A.F. Bobick, A framework for recognizing multiagent action from visual evidence, in: National Conference on Artificial Intelligence (AAAI), 1999.
[7] M. Isard, A. Blake, Condensation – conditional density propagation for visual tracking, International Journal of Computer Vision 29 (1998) 5–28.
[8] Z. Kahn, T. Balch, F. Dellaert, An MCMC-based particle filter for tracking multiple interacting targets, in: European Conference on Computer Vision (ECCV), 2004.
[9] K. Kanazawa, D. Koller, S. Russell, Stochastic simulation algorithms for dynamic probabilistic networks, in: Uncertainty in Artificial Intelligence (UAI), 1995.
[10] X. Liu, C.-S. Chua, Multi-agent activity recognition using observation decomposed hidden markov model, in: International Conference on Computer Vision, 2003.
[11] B. Ng, L. Peshkin, A. Pfeffer, Factored particles for scalable monitoring, in: Uncertainty in Artificial Intelligence (UAI), 2002.
[12] S. Suria, S. Mahadevan, Probabilistic plan recognition in multiagent systems, in: International Conference on Automated Planning and Scheduling, 2004.