Agent Interaction, Multiple Perspectives, and Swarming Simulation

H. Van Dyke Parunak Robert Bisson Sven A. Brueckner
Vector Research Center, division of TTGSI
3520 Green Court, Suite 250
Ann Arbor, MI 48105
+1 734 302 4684 +1 734 302 4648 +1 734 302 4683

{van.parunak, robert.bisson, sven.brueckner}@newvectors.net

ABSTRACT

Agents in a multi-agent system do not act in a vacuum. The outcome of their efforts depends on the environment in which they seek to act, and in particular on the efforts of other agents with whom they share the environment. We review previous efforts to address this problem, including active environments, concurrency modeling, recursive reasoning, and stochastic processes. Then we propose an approach that combines active environments and stochastic processes while addressing their limitations: a swarming agent simulation (which maintains transition probabilities dynamically, avoiding the static assumptions most convenient with traditional Markov models), applied concurrently to multiple perspectives (thus partitioning the active environment and addressing its scalability challenges). We demonstrate this method on a simple example.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Distributed Artificial Intelligence – *multiagent systems.*

General Terms

Algorithms, Design, Experimentation, Theory

Keywords

Modeling, polyagents, biomimetic systems, agent interaction, prediction, agent environments, swarming

1. INTRODUCTION

"Ah, but a man's reach should exceed his grasp, or what's a heaven for?"—Robert Browning,

Browning's aphorism captures a notorious problem of agent-based planning. Most plans consist of multiple steps, and later steps often depend on the successful execution of earlier ones ("precursors"). Planning the execution of a precursor, or even attempting to execute it, does not guarantee that it will be accomplished. Agents are not omnipotent over their environment. Sometimes they are frustrated by the actions of other agents with whom they share the environment. In other cases, the environment has its own distributed dynamics (e.g., weather). Every agent action is only an attempt, whose success (along with the success of later steps that depend on that action) is contingent.

When other agents make the environment dynamic, the situation becomes more complicated. Those agents are also attempting to act, and their actions are just as contingent as the actions of the initial agent. Each agent's success at any given action depends on the success or failure of the other agents, which in turn depend on its success or failure, and so forth.

Reasoning explicitly about this recursive feedback is computationally prohibitive in many realistic domains, and may not converge. The most straightforward application of statistical models such as Markov processes make unrealistic assumptions about the shape and stationarity of the underlying distributions. Our approach is statistical, but maintains its distributions dynamically by means of coupled agent-based simulations. Most multi-agent models explore only one trajectory per run, and so cannot efficiently generate distributions to guide estimates of action success. A swarming approach avoids this problem.

Agent activity can often be viewed from multiple perspectives, such as the structure of the tasks that the agent is trying to execute, the agent's social environment, and actions in the physical world. Some of these perspectives are essentially private, but may be constrained by public perspectives. The technology we outline in this paper allows us to partition the modeling by perspective (thus simplifying the modeling problem) while providing sufficient information flow among the perspectives to resolve shared constraints.

Section 2 points to related literature on the problem of action in a multi-agent world. Section 3 summarizes our statistically-oriented agent-based modeling technology. Section 4 illustrates the approach with results from a concrete example. Section 5 discusses the relation of multi-perspective modeling to other ways of reasoning about action uncertainty.

2. REASONING ABOUT ACTION UNCERTAINTY

The problem of the uncertain outcome of an agent's actions has been articulated by a number of authors [4, 11], and several solutions have been proposed.

The most detailed analysis [11] recommends that simulation architectures include an **active environment**. The simulation cycle consists of alternating decisions by the population of agents (who post their desired actions to the environment) and resolution by the environment (which decides which actions succeed). This approach addresses the problem, but at the expense of programming a dynamically omniscient environment that becomes more and more complex as the size and semantic richness of a problem grows.

Formalisms such as CSP [7] and the Pi calculus [12] seek to **model the concurrent aspect** of multiple agent actions. An example of applying this approach in the early tradition of software agents [3] develops Recursive Petri Nets (RPN's) to

Cite as: Agent Interaction, Multiple Perspectives, and Swarming Simulation, Parunak, Bisson, and Brueckner, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. 549-556

account for concurrent actions of multiple agents. By deferring the refinement of an abstract transition until execution, RPN's allow agents to take into account the current circumstances in their choices. This formalism encourages interleaving of planning and execution (which in our view is good), but explicitly avoids the question of how to choose a refinement, and thus sidesteps issues of goal conflict among agents. It also does not identify actions that are so invariant under execution context that they can be modeled as abstract transitions whose place in the overall net structure will not be affected by differing outcomes of atomic actions. More generally, abstract concurrent modeling languages abstract away from details of agent decision processes and domain semantics that are important in modeling real-world situations.

Some researchers let each agent **reason recursively** about likely acts of other agents. Gmytrasiewicz [6] develops an elaborate game-theoretic formalism, and argues that the recursion will inevitably terminate due to lack of information. Łatek [9] lets agents incrementally increase the depth of recursive reasoning while monitoring the payoff they experience to find the number of levels that are actually useful. Recursive reasoning accounts for agents' intentionality, but does not account for outcomes that are not rationally driven (such as those of nature).

One can model the response of the environment (including other agents) as **stochastic**, subsuming all intentionality in probability distributions over alternative actions. (Gmytrasiewicz's Sub-Intentional Model allows this case for non-intentional agents such as laws of nature, and as a way to terminate recursion.) Decision tree analysis [19] is perhaps the simplest application of this approach. The nodes in the tree alternate between choices by the agent and responses of the environment. A probability is assigned to each environmental response, utilities are associated with the outcomes, and by propagating the utilities through the intervening probabilities, a value can be placed on each choice open to the agent. Another formalism for modeling dynamical systems involving uncertainty is the Markov process, which associates a probability of success $p_{a,s}$ with each action. At each step, the system decides with this probability whether the action succeeds or not. This approach is computationally attractive. For example, it is tractable, and it permits studying the convergence of a series of repeated actions by matrix products over the transition probabilities. However, it has several disadvantages.

1. Its purely numerical nature discards any insight that might be available from examining agent intentions.
2. Determining the probabilities in a stochastic model of a large real-world scenario is a daunting task.
3. These probabilities in general do not conform to any particular distribution (making sampling them difficult), and they are likely to be nonstationary, greatly complicating the analysis.

To make this last observation more concrete, consider a scenario in which a single agent seeks to move from one point to another in the face of a team of opposing agents. For a given strength of the opposition, what will the distribution of arrival times look like across multiple replications of the scenario? Intuitively (and confirmed by experiment),

- If the opposition is weak, the arrival time distribution will be unimodal, around a value that marginalizes over stochastic variations in the travel time experienced in different replications of the scenario.
- If the opposition is very strong, the distribution will again be unimodal and in fact univalued at infinity. No replication will yield a successful arrival.
- At intermediate values, some replications will succeed and some will fail, yielding a bimodal distribution. One mode, at infinity, will represent trials that did not succeed. The other will reflect arrival times for trials that did succeed. The location of the success mode and the relative size of the two modes will depend on the degree of opposition.

In such a situation, none of the previous approaches suffices. The complexity of a single active environment does not scale to large problems. Mere analysis of concurrency cannot capture the adversarial relation between the players. There is no natural termination to the recursion in the recursive rationality approach, and the non-stationary, non-canonical form of the underlying statistics frustrates the standard forms of statistical analysis.

We use multiple agents to represent each domain entity, an approach that has been called a "polyagent" [17] and that has parallels in other systems [18]. Each agent (a "ghost") explores a (possibly distinct) possible future for its entity. Ghosts record their movement through the environment by depositing a digital analog of insect pheromones, and interact with the ghosts of other entities, not directly, but by sampling the pheromone fields that they generate. These fields are, up to a normalizing constant, probability fields over the alternative futures explored by the ghosts [15]. The field for a single entity reflects all the possible futures explored by its ghosts, and when a ghost of another entity modulates its behavior based on that field, it is sampling over those futures. Thus the total number of system futures explored by a polyagent system with g ghosts for each of e entities is on the order of g^e, far larger than can be explored with single-trajectory simulations. Each entity has a supervisor agent (called an "avatar") that monitors its ghosts as they run into the future, and selects real-world actions based on those behaviors.

When agents' actions can be described as movement in a constrained environment, this construct offers an elegant solution to the problem of action planning.

- The environment is an instance of an active environment, as it maintains the fields modulated by the ghosts.
- Like concurrency models, it accounts for concurrent agent actions, since ghosts of all entities are active concurrently.
- Like models of recursive rationality, it provides an account of agent behavior, since the ghosts have behavior models that they follow in taking action in the shared environment.
- Also like recursive rationality, it accounts for the mutual feedback among agents, since each ghost sees the fields generated by the ghosts of all other agents and takes them into account in its decisions.
- Like statistical mechanisms, it is computationally tractable and amenable to theoretical analysis by way of the probability fields that it generates.

In addition, because it generates and maintains the probability fields dynamically, the polyagent approach does not require their manual analysis, is not confined to the structure of predefined distribution types, and is not limited to stationary distributions.

One weakness of polyagents as a general planning mechanism is that ghost actions, following their biological models, are preeminently movements in space. Planning often needs to accommodate actions that are not readily understood as physical movement of the agent performing the action.

Recently, polyagents have been demonstrated in an environment consisting of a hierarchical task network (HTN, specifically, a dialect of TÆMS) [1, 16]. Agent movement in this space can represent any series of actions, even those that are not simple movements. Multiple agents can share a TÆMS network, so a polyagent simulation embedded in a TÆMS network can support planning in a way analogous to simulations in a geospatial network. Each ghost can increment fields on method nodes[1] to indicate which methods it has sampled. As it explores its trajectory, it deposits an "urgency" field on each node that it has visited. This value is computed from the overall improvement to the quality of the solution that would be contributed by executing that node, including the impact of execution deadlines on methods that depend on the node's execution. The avatar then chooses as its next action the one preferred by its ghosts.

Just as the HTN captures some aspects of agent behavior that are not represented in a geospatial model, so the geospatial model captures some aspects (e.g., the impact of physical constraints such as roads, obstacles, gradients, and spatial density of agents) that the HTN cannot represent. The environment really needs to combine the task structure of the HTN with the physical constraints of the geospatial manifold.

There are other behavioral constraints that neither HTN's nor geospatial maps can express. Consider, for example, the decision sequencing represented in a decision tree, or the interpersonal exchanges of financial, physical, and political resources mediated by a social network. No single topology captures all aspects of agent planning. Each topology expresses a different *perspective* on the overall problem.

In any given problem, some perspectives are often more public than others, and thus more likely contexts for possible agent interaction. For example, all the tasks in a given HTN might belong to the same agent (a "private" HTN), but the execution of particular tasks may be constrained by agent interactions in another topology (say, a geospatial map or a social network). The shared topology serves as the integrating environment recommended by [11], but because it is constrained to a single perspective, its computational complexity is more manageable than a single overall shared environment. Realizing this vision requires a way to join multiple topologies together so that polyagents can generate appropriate distributions over alternative outcomes in each of them, under the influence of the distributions that are emerging at the same time in the other environments.

This paper gives an instance of this methodology with special emphasis on swarming agents. Specifically, we show how agents can swarm concurrently over linked task network and geospatial map topologies.

Some architectural constraints of maintaining multiple concurrent environments for agents have been discussed elsewhere [5].

3. ARCHITECTURE

The polyagent model of field-based agent collaboration was originally inspired by chemical pheromone systems in social insects such as ants and termites. The multi-perspective system in this paper can naturally be seen as a step in the successive refinements of that system. Insects, and the earliest applications of pheromone-based planning and prediction, deposit their pheromones (chemical for insects, digital for agents) on a single map, allowing coordination in space, but not in time. The next refinement of the agent application of digital pheromones uses a series of pheromone maps spaced through time, allowing coordination in both space and time. The multi-perspective technique outlined in this paper links multiple topologies at each temporal epoch.

3.1 Atemporal Maps

In the natural world, insects share a single "map," a physical surface, through time. A deposit made at time t, after evaporation and propagation, is sensed by later insects that visit that same location. The map provides no temporal information, only a constantly adjusted probability distribution[2] of where members of the colony have recently been. It coordinates the members of the community in real time, but offers no predictive capability.

This natural system is the basis of ant colony optimization [2], an off-line mechanism that has delivered impressive results in tasks such as route optimization. It is also the basis of an early application of polyagents in planning air combat missions [20], which has become the foundation for a control system for swarms of unmanned ground and air vehicles [21]. The ghosts in a polyagent system are an engineered parallel to real-world insects.

3.2 Temporally Indexed Geospatial Maps

We can move beyond coordination to prediction by maintaining a separate field map for each time step in a discrete-time model. In the most complete implementation of this approach [10], a fixed set of pages (the "book of maps"), indexed by τ, spans a period from a point in the recent past (the "insertion horizon") to some point in the future.

The fields on each page between the insertion horizon and $\tau = 0$ ("now") record the historical state of the world at the point in the past to which it corresponds. They are used to fit ghosts' behavior to the observed behavior of the domain entities that their avatars represent. The avatar inserts its ghosts at the insertion horizon with initial behavioral profiles that may be hand-crafted, or may be randomized. The ghosts move forward in time from page to page while executing their spatial movement decisions. As they do so, they interact with the past state, based on their behavioral parameters. These interactions mean that their fitness depends not just on their own actions, but also on the behaviors of the rest of the population. Because the ghosts move through the pages faster than real time, eventually they reach the page for which $\tau = 0$ (current wall-clock time).[3] At this point, each ghost is evaluated based on its location compared with the actual location of its corresponding real-world entity.

The fittest ghosts serve three functions.

[1] We use "method" in the sense used in TÆMS [8], to denote an atomic (non-decomposable) task.

[2] Interpretation of pheromone fields as probability fields makes it possible both to interface swarming methods with other machine reasoning technologies, and to develop a formal foundation to analyze their performance [15].

[3] Actually, because the pages discretize time, the condition for identifying the current page is $|\tau - t| < \varepsilon$, where ε is the interval between pages. We use $\tau = t$ as shorthand for this condition.

1. The personality of each entity's fittest ghost is reported to the rest of the system as the likely personality of that entity.
2. The fittest ghosts breed genetically and their offspring return to the insertion horizon to continue the fitting process.
3. The fittest ghosts for each entity run past the avatar's present (the page on which τ = 0) into the future. Each ghost that runs into the future explores a different possible future of its entity's interaction with the environment, analogous to how people plan ahead by mentally simulating different ways that a situation might unfold. Analysis of the behaviors of these different possible futures yields predictions.

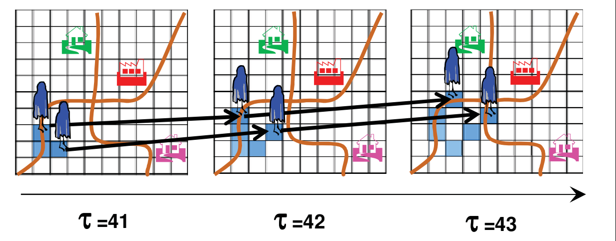

Figure 1: Temporally Indexed Geospatial Maps.—Ghosts move concurrently in both space and time.

Pages in the book of maps for which $\tau \geq 0$ have real fields only for relatively persistent environmental features such as topography or clan territories. Otherwise, the fields to which ghosts respond on these pages are built up by the ghosts themselves as they traverse them. The first ghosts to visit each page do not see any ghost-generated fields, and their behavior is constrained only by persistent features. To enable ghosts to respond to one another, avatars release them in *shifts*. In one application, each avatar releases a total of 200 ghosts over 100 shifts, two per shift. The ghosts in each shift respond to the state of the fields as modified by the previous shifts.[4] This shift mechanism is analogous to the recursive rationality model, where the number of shifts corresponds to the depth of the recursion: the n^{th} shift makes its decision based on the system's estimate of the decisions of the earlier $(n - 1)$ shifts.

At each time step, each avatar's ghosts move from one page to the next, and the avatar releases a new shift. At each step, each ghost

1. Evaluates the fields on its current page;
2. Increments the fields at its location on its current page;
3. Chooses an action based on the field strengths in step 1;
4. Executes the action while moving to the next page.

Thus each ghost step consists of a movement in time and (optionally) in space. For example, Figure 1 shows the two ghosts moving through the book of maps, one with a preference to follow roads, the other cutting across an open field. In addition, at each time step the system attenuates the fields on each page by a constant factor *E*, favoring more recent deposits over earlier ones.

The process outlined in the previous paragraphs takes place with a fixed mapping of pages to real-world times. In real-time planning applications, we update this mapping as time advances in the real world. The trigger for this change may be the passage of a fixed interval of real-world time, or some event such as an update from real-world sensors. We advance time in the following manner.

1. Each avatar takes action in the real world, based on the distributions generated by its ghosts.
2. The oldest past page is deleted.
3. A new page is added at the most remote future time. This step and the previous one keep the size of the book of maps constant.
4. The contents of the first page in the future (which up until now have been defined by fields generated by the ghosts) are replaced by currently sensed information.

Because this cycle advances the avatar's actions, we call the period during which the page-time mapping is fixed, an "avatar cycle." The repeated ghost cycles described in the previous paragraphs take place during each avatar cycle.

The temporal sequence of maps in the book of maps can be used to formulate predictions of agent movement through time—predictions that have proven more accurate than those produced, not only by other machine learning technologies, but also by professional experts in the domain being modeled [14].

3.3 Multi-Perspective Modeling

In multi-perspective polyagent modeling, each page contains, not a single topology, but topologies representing different perspectives, and ghosts move not only from one page to another, but also from one perspective to another. Figure 2 and Figure 3 outline a simple example, within a single avatar round.

Each page has three perspectives: a process graph with methods ma1-ma5 belonging to entity A, a process graph with methods mb1-mb5 belonging to entity B, and a shared geospatial map. The process graphs are shown as simple precedence relations among methods. Ghosts are represented by ovals. At each time step, all ghosts move in time (that is, from one page to another). Some also move in task space, geospace, or between perspectives. The task spaces are private to each entity, so entity interactions are limited to the geospatial map. In this example, the effect of an interaction in physical space is to delay the movement of both entities.

All ghosts begin at τ=41. Each ghost is located at a method in the process graph of its respective entity. Methods ma1, ma4, ma5, mb3, mb4, and mb5 have duration 1. The duration of other methods (shaded grey) depends on movement in physical space.

For clarity, we show the movements of two successive shifts in separate figures. Figure 2 shows the movement of two ghosts, one from entity A and the other from entity B, launched in shift 1. At τ=41, both ghosts begin on a method that requires physical movement, so they move to the shared geospatial map. Their methods require them to move due west. They evaluate the local fields, find none, and compute that the next time step will suffice for them to move to their destinations, shown as 'X'. (The distance moved in geospace depends on the agent's mobility, and can be more than a single cell, as here.)

At the next simulation step, both ghosts move to their destinations on the page for τ=42. As they move from east to west, they increments fields specific to their entities (indicated by vertical

[4] Ghosts in early shifts do not experience well-defined fields, so their movements do not estimate entity movement as reliably as those in later shifts, when the fields have converged. To accommodate this increase in accuracy over time, at each simulation step the field strengths on each page are attenuated by a constant factor (a process inspired by pheromone evaporation in insect systems). The effect is to weight the deposits by later shifts more strongly than those by earlier ones.

hatching for entity A, and horizontal hatching for entity B). By reaching their destination, they complete their respective methods, so they move back to their respective process graphs, which forward them to their successor methods. These methods also require spatial movement, so the ghosts move back to the geospatial map at locations dependent on their methods and their current state. It happens that both of their methods require them to move over the same path. Thus the ghosts are collocated in the geospatial map. However, since no ghosts have visited this region of the map before, the field strengths representing the two entities are both zero along the required path. So the ghosts determine that they can move without restriction to their destinations. In this step, the ghosts have moved in time, physical space, process space, and between perspectives.

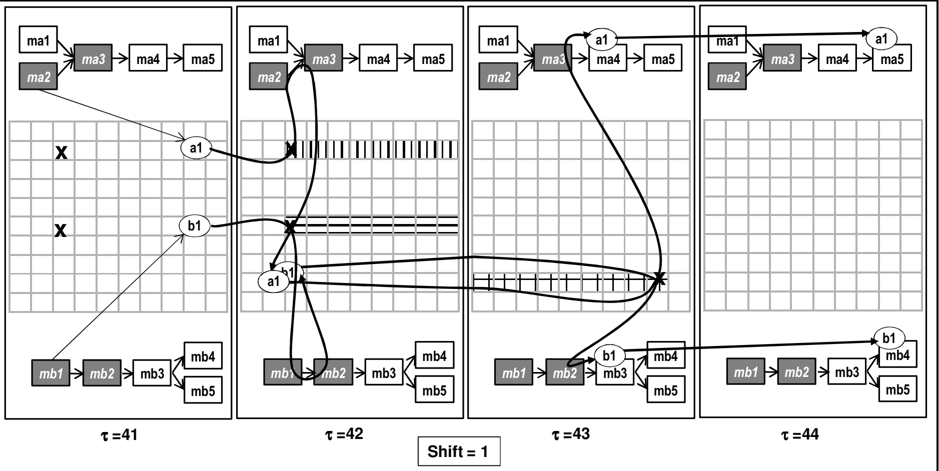

Figure 2: Multi-Perspective Modeling. Ghosts move in time, space, and perspective. Methods shaded grey require completion of a spatial movement. Only ghosts belonging to shift 1 are shown.

Next, the ghosts move to τ=43. Both ghosts reach their (coincidentally common) destination, incrementing their entities' fields along the path (marked by cross-hatching, combining the vertical and horizontal hatching of the individual ghosts). In both cases, they complete their methods, thus returning to their respective process graphs, where they then advance to their next methods. These methods do not require geospatial movement, so the ghosts wait for the duration indicated in the method.

In both cases, this duration is just one time step, so in the next simulation step, both ghosts move through time and process space to their successor methods in the page for τ=44.

The result of the movement of shift 1 is the development of fields at regions in the geospatial map visited by the ghosts, as well as fields (not shown) on the methods that they have executed. Note that the ghosts complete all their methods, both those requiring movement and those not requiring movement, in one time unit.

Now consider the evolution of shift 2, shown in Figure 3. Again, ghosts start on the page for τ=41. Ghost a1 is on a non-movement method with a duration of one time step, so it decides on its next step to move to the successor method. Ghost b2 is on a movement method, so it takes its position on the geospatial map, evaluates the fields in its vicinity (finding none), and decides that it can on the next step complete its movement to its destination (marked with an 'X') to its north-east.

At the next step, both ghosts move to the page for τ=42. Ghost b2 completes its movement and thus method mb1, and advances to mb2, a movement method that requires it to return to the geospatial map. Ghost a2 finds itself on a movement method and takes its place on the geospatial map as well. The starting locations and destinations of movement methods depend on the agent's state as well as the method, so though a2 and b2 are executing methods previously visited by a1 and b1, they need not follow the same path.

Each ghost finds its intended path marked by a field representing the presence of the other entity at this point in space and time. These fields were deposited by a1 and b1 in the previous shift. Ghosts b2 and a2 interpret these fields probabilistically, by flipping a coin weighted by the strength of the field. Ghost b2 samples the case that B meets A at this location, and decides that in the next time step, it will not be able to move very far. Ghost a2 samples the case that A does not meet B, and decides that it can reach its destination in the next time step. These stochastic decisions are repeated for each ghost that visits the location, so the proportion of ghosts sampling an encounter will be proportional to the strength of the field. However, unlike a static estimator like a Markov transition probability, the strength of the fields varies as successive shifts of ghosts traverse the pages and increment the fields based on their own estimates of where the entities may move.

Next, the ghosts move to τ=43. Both ghosts are moving on the

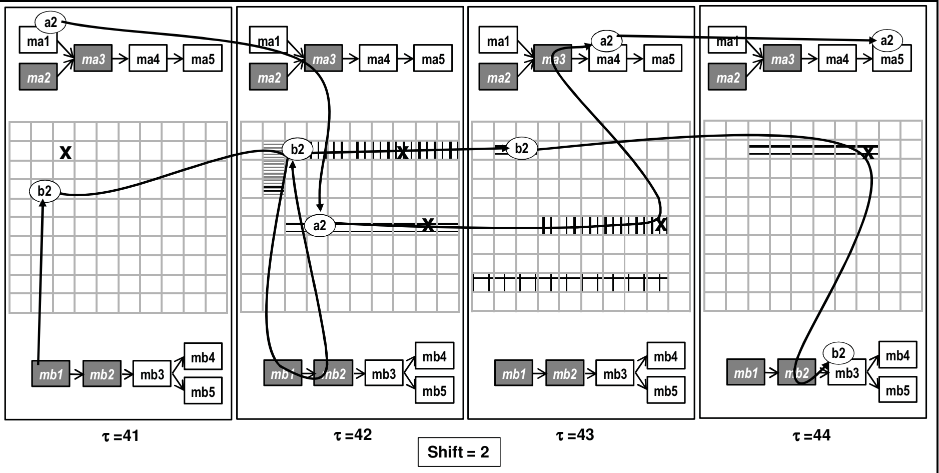

Figure 3: Multi-Perspective Modeling, Shift 2.—These ghosts start on the page for τ=41 one simulation step after those shown in Figure 2.

geospatial map, and increment their entities' fields accordingly. Ghost b2 moves only a short distance, because it is sampling a future in which entity B meets entity A and is delayed. Upon completing its move, it evaluates the fields in its vicinity. There are no fields near it for entity A at $\tau = 43$, so it computes that it can complete its move without delay in the next simulation step. Ghost a2 reaches its destination, because it is sampling a future in which A does not meet B. Upon reaching its destination, it completes method ma3, advances to ma4, observes the duration of that method, and waits for the designated period (one time step). Ghost b2 has moved in time and physical space. Ghost a2 has moved in time, physical space, process space, and between perspectives.

Now the ghosts move to the page for τ=44. For ghost a2, this is simply a movement in process space, from method ma4 to ma5. Ghost b2 completes its movement in the geospatial map, incrementing the field for Entity B. It returns to its process map, exiting mb2 and moving to mb3, where it computes the duration needed to execute its next action.

This simple example illustrates several crucial features of the multi-perspective approach to modeling agent interaction.

- A method's duration can depend on interactions, by sampling the interactions in a different perspective (in this case, the geospatial map).
- The probabilities that are used to estimate a method's duration are developed dynamically by the agents during the model's execution.
- Methods with no interaction contingencies are efficiently executed directly, based on recorded durations. In some cases, experience may allow us to define a closed-form distribution of execution times for a movement method, thus avoiding the need to send the ghosts for that method to the geospatial perspective.

In this example, the same agents move through both task space and physical space. In some cases, different classes of entity move in different spaces. For example, the entities that move through task and physical space might be stationary nodes in a social network. In that perspective, mobile agents could represent resources such as money or materials that are passed from one actor to another. Ghosts representing these resource agents would then drop down into the process graph to supply enabling resources [1] (not shown in the simplified process graphs of Figure 2 and Figure 3) and thus allow ghosts of actors to complete the methods supplied by those resources. In all cases, ghosts move between perspectives to explore interactions between their avatars, but multiple species of avatars may be involved.

4. EXAMPLE

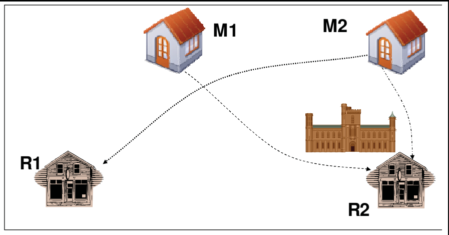

Figure 4: Urban delivery scenario. Two shops M1, M2 supply two retailers R1, R2. Retailer R2 is near the parliament building.

We illustrate this approach with an urban delivery scenario (Figure 4). A bicycle courier services two small manufacturing workshops (M1 and M2) and two retailers (R1 and R2). M1 makes products only for R2, while M2 makes products for both M1 and M2. The city is very congested, and sometimes demonstrations and street rallies prevent the courier from completing a delivery on time, in which case she is not paid. Such disruptions are particularly common in the vicinity of the parliament, which is close to R2. Her planning process thus must take into account not only the supply of products and the demand from the retailers, but also her estimate of the likelihood of heavy traffic in different parts of the town.

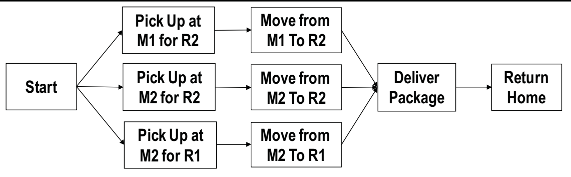

Figure 5: Process Graph for Delivery Example

Figure 5 shows a fragment of her process graph. The three " Move from X to Y" methods require her ghosts to swarm in the geospatial map in order to determine the duration of the method. She can choose to pick up either a shipment for R1 (which must come from M2), or a shipment for R2 (which can come from either workshop), then move to the appropriate retailer. A trivial process graph (not shown) for demonstrators determines the degree of traffic around the Parliament on a given day.

We demonstrate the behavior of our implementation of this system with two different situations. In one, there is no demonstration, and the two deliveries are of equal value. In the second, a demonstration around the parliament impedes access to R2.

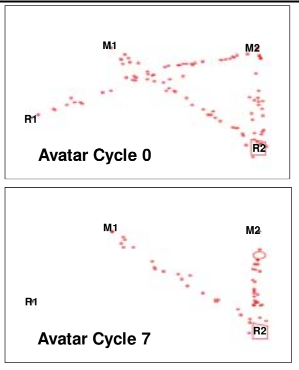

Figure 6: Ghost execution locations in geospace as function of avatar cycle (no demonstration)

In the first configuration, with no demonstration, the ghosts representing the courier initially explore all possible routes (Figure 6, top). The ghosts that explore the route from M2 to R1 travel a longer distance than those on the other two paths, leaving a weaker field. By the seventh avatar cycle (Figure 6, bottom), most ghosts are favoring the M2-to-R2 route. Figure 8 shows the total ghost activations in each avatar cycle for each destination.

Figure 7 shows the locations of ghost executions in geospace in the second configuration. The cluster of ghosts around R2 represents the avatars of the demonstrators. Again, initially (Figure 7, top), they explore all routes, but ghosts attempting to deliver to R2 are less successful and leave less reinforcement for successive ghosts, so that by avatar shift 7, only ghosts exploring paths from M1 (which can only deliver to R2) are attempting this path.

Figure 9 shows ghost executions by destination as a function of avatar cycle. In the initial cycle, successive shifts of ghosts reinforce the shorter paths, leading to a preference for R2 in spite of the demonstrators (though the number of R2 ghosts is lower than in the first scenario, due to the congestion near R2). But this benefit drops in subsequent avatar cycles, and most ghosts pursue the M2-R1 route, which the avatar follows.

We have exhibited only two extremes in which the courier's preference for the two destinations flips. As the strength of the demonstration varies, we obtain intermediate results between those of Figure 8 and Figure 9. In particular, we could find the point at which the courier's preference for the shorter route is balanced by the delay imposed by the demonstration.

5. DISCUSSION

The simple example in Section 4 illustrates how the behavior of the courier's avatar in process space is affected by the presence or absence of interactions with other agents (the demonstrators) in geospace. Without a multiperspective model, we would have to estimate an explicit distribution for the duration and success of the movement methods as a function of level of demonstration, a laborious task that would be difficult to validate. In our approach, we simply ask each ghost to step into the geospatial world and see what happens. The interaction effect is estimated constructively, by direct simulation in the appropriate domain, and communicated between perspectives by the movement of ghosts sampling alternative futures for their avatars. The large sample of futures accessed by the ghosts [15] increases our confidence in the robustness of the emergent effects that they exhibit.

The multi-perspective polyagent approach to modeling interactive agents draws from each of the four antecedents outlined in Section 2, while addressing the shortcomings we identified there.

Of the four techniques we discussed, the only one not directly reflected in our approach is the use of a concurrency formalism such as CSP, RPN's, or the Pi calculus. These techniques have their place in the meta-analysis of a multi-agent system [13], but we have not directly applied them in this work.

The set of topologies, one for each perspective, forms an **active environment.** Each perspective actively maintains the fields by aggregating the increments deposited by each ghost and attenuating the fields over time to favor the more recent shifts. Because the environment is partitioned into different perspectives, and because agent interactions typically involve only a subset of the perspectives, the complexity of the environment's computations is reduced compared with a single global environment.

As in models of **recursive rationality**, and unlike purely statistical methods, multi-perspective polyagents maintain explicit models of the decision-making of each agent. The successive waves of ghost agents that build up the probability fields correspond directly to the levels of recursion in recursive rationality. The convergence of the probability fields offers a quantitative way to determine how far to extend the recursion (i.e., how many shifts to send through the system), and the probabilistic nature of the result is a more realistic guide to prediction than the single result emerging from agents' deterministic recursive estimates of one another's decisions.

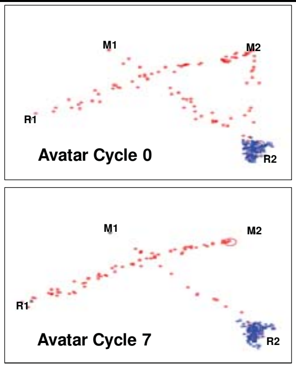

Figure 7: Ghost execution locations in geospace with demonstration

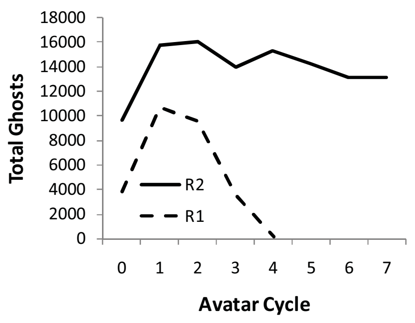

Figure 8: Total ghost executions by destination, no demonstration

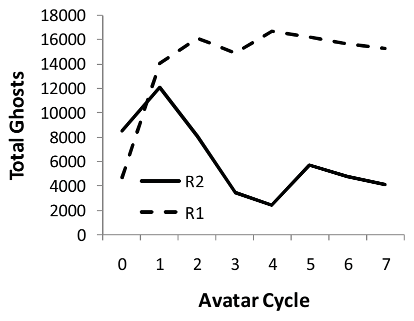

Figure 9: Total ghost executions by destination, with demonstration

The probabilistic nature of the ghosts' decision-making is at variance with most recursive reasoning approaches, and more closely resembles **statistical process models** such as Markov models. We share many benefits of these models, including the relative computational efficiency of numerical over symbolic reasoning, and the ease of interfacing with the many machine learning techniques that are based in probability theory. At the same time, we offer several advantages over conventional statistical models. Our model of individual agent behavior can be mapped readily onto rational decision models. By tracking individual agents, rather than simply transitions in the state of the overall system, we avoid the anomalies that often arise with mean-field methods [22, 23]. And the maintenance of probability fields by ghosts that are in turn evolved against observations from the domain avoids the limitations of static transition probabilities endemic to more traditional statistical approaches.

The approach outlined in this paper will increase in value as we develop further perspectives, in addition to geospatial maps and task networks. Our current priority is to incorporate reasoning over various social networks. As suggested above, the agents that swarm over a social network may be different from the domain actors that we have discussed in the examples in this paper. In a network representing financial flows, they may constitute financial resources. In a communication network, they may develop a field indicating the likelihood of communication between two actors at a given epoch. Both financial resources and coordinating messages naturally feed into the resource nodes that enable methods in a full rTÆMS network (not shown in the simplified examples in this paper, but discussed in [1]).

The modularity achieved by decomposing the environment into multiple perspectives allows a more general extension, in which the dynamics of some perspectives are computed, not by swarming agents, but by other techniques, such as difference or differential equations, conventional Markov models, or BDI mechanisms. In turn, distributions that can be derived from our swarms can provide non-swarming probabilistic reasoners with dynamically-varying, nonstationary estimates of important variables that would be difficult to estimate in any other way.

6. ACKNOWLEDGMENTS

This research was conducted with the support of the Office of Naval Research (ONR). The results presented do not necessarily reflect the opinions of the sponsor.

We are grateful for stimulating interactions with our colleagues Rainer Hilscher, Bob Bechtel, and Laura Hamel, who helped develop and implement the algorithms reported in this paper, and to a careful reviewer.

7. REFERENCES

[1] S. Brueckner, T. Belding, R. Bisson, E. Downs, and H. V. D. Parunak. Swarming Polyagents Executing Hierarchical Task Networks. In *Proceedings of Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2009)*, IEEE, 2009.

[2] M. Dorigo and T. Stuetzle. *Ant Colony Optimization*. Cambridge, MA, MIT Press, 2004.

[3] A. El Fallah Seghrouchni and S. Haddad. A Recursive Model for Distributed Planning. In *Proceedings of the 2nd International Conference on Multi-Agent Systems (ICMAS'96)*, pages 307-314, AAAI Press, 1996.

[4] J. Ferber and J.-P. Müller. Influences and Reactions: a Model of Situated Multiagent Systems. In *Proceedings of Second International Conference on Multi-Agent Systems (ICMAS-96)*, pages 72-79, AAAI, 1996.

[5] M. A. d. C. Gatti and C. J. P. d. Lucena. A Multi-Environment Multi-Agent Simulation Framework for Self-Organizing Systems. In *Proceedings of the 10th International Workshop on Multi-Agent-Based Simulation (MABS 2009)*, Springer, 2009.

[6] P. J. Gmytrasiewicz and E. H. Durfee. A Rigorous, Operational Formalization of Recursive Modeling. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS'95)*, pages 125-132, 1995.

[7] C. A. R. Hoare. *Communicating Sequential Processes*. Englewood Cliffs, NJ, Prentice-Hall, 1985.

[8] B. Horling, V. Lesser, R. Vincent, T. Wagner, A. Raja, S. Zhang, K. Decker, and A. Garvey. The Taems White Paper. Multi-Agent Systems Lab, University of Massachusetts, Amherst, MA, 2004. http://dis.cs.umass.edu/research/taems/white/.

[9] M. Łatek, R. L. Axtell, and B. Kaminski. Bounded rationality via recursion. In *Proceedings of Eighth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2009)*, pages 457-464, IFAAMAS, 2009.

[10] T. W. Lucas and J. A. Dinges. The Effect of Battle Circumstances on Fitting Lanchester Equations to the Battle of Kursk. *Military Operations Research*, 9(2):17-30, 2004.

[11] F. Michel. *Formalisme, méthodologie et outils pour la modélisation et la simulation de systèmes multi-agents.* Thesis at Université des Sciences et Techniques du Languedoc, Department of Informatique, 2004.

[12] R. Milner. *Communicating and Mobile Systems: the Pi-Calculus*. Cambridge, UK, Cambridge Univ. Press, 1999.

[13] H. V. D. Parunak. Manufacturing Experience with the Contract Net. In M. N. Huhns, Editor, *Distributed Artificial Intelligence*, pages 285-310. Pitman, London, 1987.

[14] H. V. D. Parunak. Real-Time Agent Characterization and Prediction. In *Proceedings of International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'07), Industrial Track*, pages 1421-1428, ACM, 2007.

[15] H. V. D. Parunak. Generation and Analysis of Multiple Futures with Swarming Agents. In *Proceedings of the International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2010)*, pages (forthcoming), IFAAMAS, 2010.

[16] H. V. D. Parunak, T. Belding, R. Bisson, S. Brueckner, E. Downs, R. Hilscher, and K. Decker. Stigmergic Modeling of Hierarchical Task Networks. In *Proceedings of the Tenth International Workshop on Multi-Agent-Based Simulation (MABS 2009, at AAMAS 2009)*, pages (forthcoming), Springer, 2009.

[17] H. V. D. Parunak and S. Brueckner. Concurrent Modeling of Alternative Worlds with Polyagents. In *Proceedings of the Seventh International Workshop on Multi-Agent-Based Simulation (MABS06, at AAMAS06)*, Springer, 2006.

[18] H. V. D. Parunak, S. Brueckner, D. Weyns, T. Holvoet, and P. Valckenaers. E Pluribus Unum: Polyagent and Delegate MAS Architectures. In *Proceedings of Eighth International Workshop on Multi-Agent-Based Simulation (MABS07)*, pages 36-51, Springer, 2007.

[19] H. Raiffa. *Decision Analysis: Introductory Lectures on Choices Under Uncertainty*. McGraw-Hill, 1997.

[20] J. A. Sauter, R. Matthews, H. V. D. Parunak, and S. A. Brueckner. Performance of Digital Pheromones for Swarming Vehicle Control. In *Proceedings of Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 903-910, ACM, 2005.

[21] J. A. Sauter, R. S. Matthews, J. S. Robinson, J. Moody, and S. P. Riddle. Swarming Unmanned Air and Ground Systems for Surveillance and Base Protection. In *Proceedings of AIAA Infotech@Aerospace 2009 Conference*, AIAA, 2009.

[22] N. M. Shnerb, Y. Louzoun, E. Bettelheim, and S. Solomon. The importance of being discrete: Life always wins on the surface. *Proc. Natl. Acad. Sci. USA*, 97(19 (September 12)):10322-10324, 2000.

[23] W. G. Wilson. Resolving Discrepancies between Deterministic Population Models and Individual-Based Simulations. *American Naturalist*, 151(2):116-134, 1998.