High-Density Pattern-Of-Life Modeling

Randolph M. Jones[1], J. T. Folsom-Kovarik[1], Pat McLaughlin[2], and Rich Frederiksen[1]
[1]Soar Technology, 3600 Green Court, Suite 600, Ann Arbor, MI, 48105 USA
[2]Scientific Systems Company, 24632 Monte Royale, Laguna Hills, CA, 92653 USA
rjones@soartech.com, jeremiah.folsom-kovarik@soartech.com, Pat.McLaughlin@ssci.com, rdf@soartech.com

Abstract - *Patterns of Life are general human behavioral patterns that emerge from individual behaviors and reflect group tendencies, especially sociocultural tendencies. Challenges for modeling patterns of life are somewhat different from challenges for traditional modeling of cognition, decision-making, and expertise. Modeling patterns of life requires solutions for background agents, which are not individually complex, but are individually unique and representative of rich sociocultural backgrounds; as well as solutions for foreground agents, which are unique and complex, but also blend in with the background behavior patterns. This paper presents a POL modeling architecture that addresses issues in authoring and executing such large populations of unique individuals.*

Keywords: Patterns of life, Large-scale behavior modeling, Automated scenario authoring

1 Patterns of life

We have pursued several projects that involve computational modeling of human *Patterns of Life* (POL). Schatz et al. [1] define POL as follows: "In the context of cultural training, patterns of life are the archetypal emergent properties of a complex sociocultural system." In addition, "…the emergent properties…can be organized into categories that describe classes of patterns of life that share similar general features, and we expect these same classes will manifest (likely with different nuanced characteristics) across all societies and cultures." Folsom-Kovarik et al. [2] proposed scalable POL models as a challenge problem for applied artificial intelligence. This paper describes an architecture and approach to respond to that challenge.

POL are properties of a society that determine how individuals function on a day-to-day basis. The US military uses POL for Stability, Security, Transition, and Reconstruction (SSTR) operations, to train US Forces who must operate in these communities, and to identify threats based on POL anomalies. The DARPA Insight program sponsored research into the structure of generative POL models to support a simulation and test environment for threat recognition algorithms. Typical applied and theoretical work in artificial intelligence focuses on high-level competence or intelligent decision-making of individuals. In contrast, POL modeling focuses on social, cultural, and everyday behavior. POL are themes that appear in a population's everyday actions and interactions. The ability to understand patterns of life, and thus to represent and instantiate POL models, is a critical emerging military requirement.

This paper describes technical issues of POL modeling and details of the approach we have developed across many projects, beginning with the sophisticated Insight simulation. POL modeling requires solutions for breadth, efficiency, and authoring of agent behaviors. We use cultural representations that explicitly allow for natural human variation, and a computational infrastructure that generates a continuum of relevant socio-cultural behaviors in two classes: efficient schedule-based scripts that support high-density populations and knowledge-rich individual models with the competence to adapt decisions based on goals and situational understanding.

2 Application areas

The Department of Defense (DOD) recognizes that POL analysis is an excellent mechanism for threat identification and engaging with a population in appropriate socio-cultural ways [3]. Those who live in or analyze foreign areas learn to recognize anomalies in typical patterns and react accordingly. The training challenge is to prepare those who are not in the area or do not have time to slowly become familiar with these threats. Consider an example where a U.S. Marine Corps squad leader is operating near a town of about 10,000 people, with reports of insurgent activity in the area. On a given morning, the marketplace includes a small number of people milling around (perhaps a dozen or so), a few bicycles and one car. Some stalls in the marketplace are empty. Is this a clear indication of danger or normal for this particular time?

Understanding the local POL is essential for the squad leader to build an overall assessment of this situation, determine whether any observed elements are unusual or dangerous, and predict what is likely to happen next. The squad leader's observations must cohere into high-level patterns that make sense. In this example, perhaps the traffic is typical for this time of the day and should increase

dramatically later. This is just one example of how POL analysis informs modern warfighting and peacekeeping. Modeling POL can support scenarios with increasing levels of behavior fidelity depending on the simulation need.

Realistic background context in training. POL can provide a realistically rich background for focused training activities. This can improve training simply by increasing user presence or putting the training into a realistic context, even if the POL itself is not a key part of the training.

Training of anomaly identification. For other types of training, POL is a key focus. A trainee may be learning to distinguish significant, anomalous behavior from the normal patterns in an area. In this case, *background* POL models include routine behaviors, while *foreground* models generate patterns that are similar, but characteristic of significant anomalous threats. A trainee must pick out the anomalous behavior patterns from the background.

Socio-cultural sense making. For analysis and situational understanding, POL models can define high-level, latent behavior clusters that explain several low-level, observable behaviors. For example, a crowd meeting at a particular location every weekday morning might represent a high-level behavior of going to a bus stop to get to school. In such an application, an SSTR planner might use this information to decide where to locate schools or build roads. A logistics planner might use these patterns to decide when and how to move people and goods through the space with minimal risk.

3 Requirements for patterns of life

The problem of simulating POLs is to provide the simulated environment with realistic *foreground* and *background* representations of the human terrain. Realistic *foreground* models require groups of entities to be intelligent enough to appear realistic under scrutiny. Threatening insurgent behavior often comes from groups who blend in with normal patterns most of the time, but step out carefully as required to achieve goals. Foreground models must be aware of their surroundings, react to local dynamics and proximal entities, and blend in with the background POL models, so they cannot be simple scripts. Without sufficient fidelity, foreground agents stand out from the background population, making them easy to spot and useless for training or planning. Realistic *background* models must generate sufficiently broad behavior patterns that accurately reflect the agents' sociocultural milieu. The models must also be *efficient to execute* and *easy to author*. Individually authoring entity behaviors is intractable for large populations, but random generation produces unrealistic behaviors, too easy for a human learner to detect.

The overall POL of a population describes the actions and interactions of many individuals taken as a whole. The patterns emerge from aggregated individual actions. An observer can recognize general trends that characterize the *group*, even if they are not true for any one *individual*. Individual decisions and actions that comprise group behavior arise from underlying facts and narratives pertaining to the sociocultural situation. These provide a causal context that an observer can use to infer latent meanings from relatively few visible actions. Together, the emergent group behaviors and the background facts and narratives make up the POL for a particular region and population.

There is a unique collection of technical challenges to modeling POL. The major focus must be on routine behaviors rather than complex decision making. However, the behavior must be appropriate to situational context, so the agents must have some level of situation-understanding competence. Similarly, in order for behaviors to adapt appropriately to context, they cannot simply be scripts that break down if the assumptions behind the scripts become violated. The range of behaviors must be sufficiently broad and varied to provide a rich population-level dynamic in which a large variety of group-level patterns are observable and "normal", but that there are not just a large number of agents doing exactly the same things. The behaviors must also be appropriately configurable to represent sub-populations, such as different cultural groups, economic and social roles, occupation types, social strata, etc. The simulation must be computationally efficient enough to generate hundreds of thousands of background agents, as well as perhaps dozens of more sophisticated (and presumably more expensive) foreground agents. Finally, the technical solution cannot reasonably require specific programming of each background agent. There must be ways to configure the behaviors using population-level demographic parameters.

4 POL modeling architecture

We have designed and partially implemented a POL modeling architecture that specifically addresses these technical challenges. The architecture allows us to build models of the activities of many individuals in a scalable way. It includes a library of goals and behavior patterns that generate a wide variety of behaviors that instantiate group-level patterns. It also provides mechanisms to minimize user input for configuring large populations of agents. Figure 1 displays the components and functions of the POL architecture. The architecture addresses two primary modes of operation: on the left, a scenario author carries out *scenario generation*. On the right, an operator takes the generated scenario and carries out *scenario execution*. In many situations, the author and operator are a single user. Generation and execution both rely on a POL knowledge representation that supports background and foreground agents. Background agents are highly scheduled, with configuration mostly or entirely determined by population-level parameters. Foreground agents generally have minimal scheduling, make their own decisions during execution, and have more detailed and unique goals and activities, rather than population-level parameters. However, in our design, any

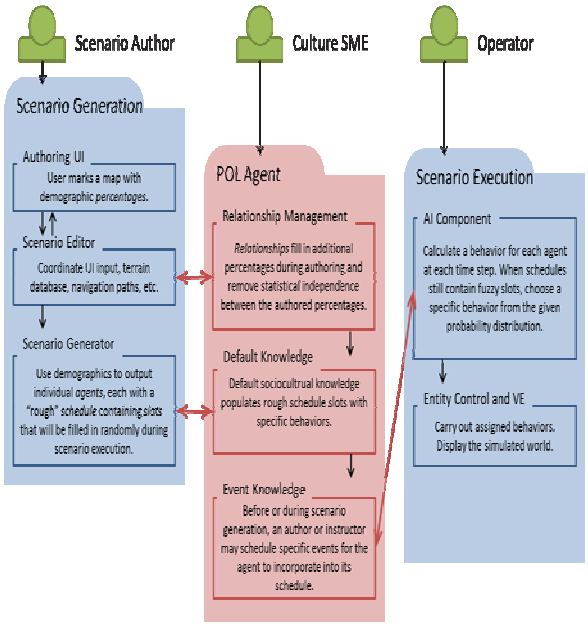

Figure 1. Schematic and functions of the POL architecture.

POL agent can be authored with a mix of population-level and detailed parameters, as well as more or less scheduling of specific activities. This enables rapid construction of the essential details and automated variation of the non-essential backgrounds. Because the background and foreground behaviors use the same sets of primitive actions and overlapping intermediate goals, the architecture supports the generation of foreground behaviors that blend appropriately with background behaviors.

5 POL knowledge representation

The knowledge representation includes primitive actions, goals, schedules, and configuration parameters. For this architecture, it is important to note that agent reasoning can take place during scenario authoring *and* execution. The output of the authoring process is a set of goals incorporated into a schedule. The agents know how to achieve the goals using the available primitive actions. The schedule may be complete or only partial. Background agent generation creates for each agent a rough schedule with some actions determined and some *slots*, where actions will be chosen from a reasonable range in order to create variation. Foreground agents have run-time knowledge for scheduling their own goals and actions. Configuration parameters influence the agents' choices of goals and the timing of the actions. In general, configuration parameters for background agents are used to generate choices during scenario authoring, while foreground agents may make some or most of their choices by reasoning about goals and parameters during scenario execution. The rest of this section provides more detail on each of these knowledge components.

Primitive actions. These define the interface between the agents and the simulation. Primitive actions generally provide the observable actions that a POL agent will generate, including movement, entering or exiting buildings and vehicles, using devices (e.g., to communicate), and interacting with other agents. Primitive actions can also be unobservable, such as a change in beliefs or achievement of a goal. In general, the primitive actions used by background agents are the same (or at least a large subset) of the primitive actions used by the foreground agents. This is because at a surface level there must be a possibility of confusing observable background and foreground behavior, so there should be nothing (or very little) observable that is unique to either type of agent.

Goal hierarchies. An important property of the POL architecture is that schedules are not merely timed scripts of primitive actions. Rather, they are composed of goals that can be decomposed into actions. The use of goals allows the agents to adapt their actions to changes in the scenario. As a simple example, if the schedules were composed only of actions, a schedule item might be something like "Starting at noon, drive Car X to Location Y". This is a specific and brittle command, which could fail if any assumptions behind the command become violated during scenario execution. For example, what happens if Car X becomes unavailable, or the agent is not able to get to Car X before noon? To address such possibilities, the POL schedules refer to goals, such as "As soon as possible before lunch time, drive to a nearby restaurant, with Car X being the preferred vehicle and Location Y being the preferred restaurant." This specification of schedules in terms of goals and preferences allows each agent to adapt its behavior when assumptions break down.

POL agents also organize their behavior around goals that can contain both scheduled and unscheduled actions. The primary difference between background and foreground agents is that the goals for background agents are generally at a level of abstraction closer to the primitive actions, while goals for foreground agents are more abstract. Both types of agents have goals centered on group activities that are fluid and contain unscheduled elements. Therefore, they can even carry out complex tasks such as planning a covert attack.

Schedules. Agents achieve goals based on schedules. Each agent has a schedule that tells it where and when to achieve its goals. A schedule covers a day, and agents can have more than one schedule to govern different types of days such as weekdays and weekends. Variation between daily schedules is accomplished with probability distributions over action alternatives. This is implemented via efficient fuzzy state machines, with extensions to improve expressivity, adaptivity, look-ahead reasoning, and behavior authoring. For example, a schedule might specify a 10% chance of going out to lunch and 90% chance of staying in the office. As another example, a *leave_for_work* goal's start time may vary about a 7:00AM center point. Each schedule items dictates a goal to achieve, with some indication of the time to start pursuing or

to aim to achieve the goal (or some other temporal relationship). Depending on the situation and the parameters, there is room for variation in goal achievement. POL agents have the knowledge to refine goals into primitive actions. Simpler goals that generally populate the schedules of background agents have simple achievement actions. Additionally, the schedules include fuzzy adaptation parameters to reflect natural human variation. Without this, every employee of an office might stand up to leave at exactly the same time or use the same route to get to work. This approach enables us to simulate thousands of diverse background agents on a single machine.

Parameters. Depending on the type of goal and/or how the goal will be achieved, there can be additional parameters that need to be specified for each instance of a schedule goal. Example parameters include:

- Location of goal achievement
- Start time to pursue the goal
- End time to achieve the goal
- Type of goal
- Transportation plan to get to location, including
  - Sequence of waypoints to achieve
  - Transportation mode to use for waypoint sequences
- Equipment to be used to achieve the goal
- Target agent for communication or interaction goals

Foreground agents do not require values for all of their goal parameters. The scenario author and operator may leave the decision up to the agent to instantiate the parameters. Depending on the parameter type, the scenario, and the run-time situation, the agent may make the decision at authoring or execution time. In addition, the goals for foreground agents can be more abstract and complex, meaning they might be refined into other goals, rather than being refined directly to primitive actions.

In addition to the goal parameters specified above, each agent can have associated population and role parameters that are used at scenario authoring time to assist in automatically generating goals and schedules. For example, the architecture provides the ability to generate schedules for all agents in a region based on percentages the author assigns to that region. The author might specify that a certain percentage of the population in a region adhere to a particular religion, are employed, are married, own cars, etc. The generated schedules follow rules to keep them internally consistent. For example, agents are not scheduled to work far from their homes if they have no transportation to get them to work.

POL agents can also form groups and act together when all members of a group except one have schedules that specify they should follow the movements directed by one other agent. POL agents have social network links that specify relationships, and they use these links to form groups when appropriate. For example, when an agent goes to work on a weekday it goes alone, but when it takes the family to a park on the weekend, the entire family pursues the goal together. Configuration parameters are not only associated with goals and agents. They are also associated with the geography and regional infrastructure. This allows the scenario-authoring component to associate authoring choices together based on mixes of parameters. The next section presents examples of how parameters drive scenario-authoring decisions.

6 POL authoring

When discussing behavior models for thousands, or possibly hundreds of thousands, of agents, it is infeasible to take an approach that requires detailed specification of each individual agent. Thus, it is necessary for a POL architecture to support population-level specification of scenario authoring choices. In our architecture, scenario authoring consists of three basic activities:

- Geography and infrastructure. This involves setting up the simulated infrastructure of the scenario, including location and functions of buildings, as well as available objects (such as vehicles or devices).

- Background agent specification. This involves specifying population-level parameters that are used automatically to generate schedules for the individual background agents.

- Foreground agent specification. This is a more interactive process of creating more complex goals, schedules, and decision points for foreground agents. Still, the author is not required to script every action for any character. Also, foreground agent behaviors can be derived from templates for foreground agents that have similar goals.

Scenario authoring begins by creating a geographical terrain with specified building locations and functions. Buildings can be residential, commercial, industrial, religious, educational, or governmental. Next, the scenario author specifies demographic parameters for regions of the terrain. For example, an author can outline a neighborhood and enter parameters that tell how many of the people generated in that region are employed, how many own a car, how many belong to a particular religion, etc.

Once these regions have been defined, the authoring component automatically generates population individuals, characteristics, and relationships. For residential buildings, the demographic parameters are used to generate populations that occupy each building (using parameters such as average family size, together with sex and age distributions). Demographic parameters are also used to determine ownership of goods (such as vehicles and devices) for each agent. For non-residential buildings, the POL authoring component generates job slots for each building, based on building size and job-related parameters for the region. Using employment parameters, the authoring component stochastically assigns available job slots to "adult" members of the population. It is possible to bias employment assignments using factors such as employment rate, ethnicity, religion, residential location, vehicle ownership, etc.

Once the basic background population has been generated, organized into families (based on residence), and assigned jobs, additional demographic parameters are used to compute additional types of relationships between individuals. These relationships include:

- Non-immediate family connections
- Religious connections
- Political connections
- Miscellaneous "interest group" connections
- Social connections
- Educational connections
- Other economic connections

These simulated relationships and group memberships are then used to generate schedules for the background agents. Without the relations, the system would produce a background population that exhibits the specified population statistics at random. However, demographic facts in reality have dependencies between categories that make some characteristics predict others. In order to reflect these patterns, the POL system applies causal and correlative information to eliminate unwarranted independence and to introduce underlying causality. For example, if it is known that employment in an urban neighborhood is correlated with education level, the authoring component can use that knowledge to fill in blanks in the authored percentages and to impose additional patterns that mere percentages do not capture. The output of the authoring component is a background population bearing realistic simulated demographics that produce rational behavior patterns.

In addition, cultural baseline knowledge helps transform population demographics into agents with plans and goals. This process converts otherwise random assignments into population-level patterns. Agent generation creates rough schedules with some actions determined and some *slots*, where actions will be chosen from a reasonable range in order to allow variation. For the scenario author, using the POL architecture's baseline knowledge reduces or eliminates the need to hand-code any portion of the background agent schedules. This background knowledge imposes consistency and causality on the schedules. For example, in some cultures, it is rare for an individual to fail to attend religious ceremonies at least once a week. Without cultural knowledge, the choice to skip attendance would be random and carry no meaning for a human to understand. Cultural knowledge adds causation to the event. It changes the output of the random generator so the agent more closely follows the norm for its assigned culture, religion, ethnicity, or other relationship. Alternatively, it might add parameters labeling some population members as religiously unobservant, a fact that would lead to further constraints on agent schedules.

6.1 Foreground agent authoring

The basic characteristics of foreground agents can be defined in the same ways that background agents are configured. However, because foreground agents engage in more complex and specialized behaviors, the POL authoring component contains additional tools for building their more complicated schedules. Figure 2 depicts the general framework for representing foreground agent knowledge and generating foreground agent schedules. Foreground agents are implemented within the Soar cognitive architecture [4][5] to allow them to reason about their environment and adjust their goal-directed behavior accordingly. A comprehensive *long-term knowledge specification* contains the goals, tasks, conditional logic, and goal decompositions available to the foreground agents. An author runs the *scenario/behavior planning user interface* to configure foreground agents. This interface uses the long-term knowledge specification to assist in creating a behavior schedule for the foreground agents. For example, if the user specifies that the scenario should include agents with the goal "achieve-covert-attack", the user interface will automatically decompose that mission into a supporting set of subgoals, primitive actions, and parameter slots. The user interface will allow the user to examine the mission structure and fill parameters, including scheduling times to achieve goals. As the user fills parameters, the user interface performs further task decompositions and displays them to the user. This also identifies additional parameters that the user can fill. The user has the option of filling each parameter or allowing the agent to fill it during scenario execution. Ultimately, the output of the planning process is a partial schedule of goals for the foreground agent to achieve, with unspecified parameters left as choice points for the agent during execution. In the extreme, but atypical, case where the user selects values for *all* parameters in the schedule, the output is a scripted schedule similar to the background agents. However, the schedule includes a full goal hierarchy, which increases adaptivity and traceability in dynamic situations.

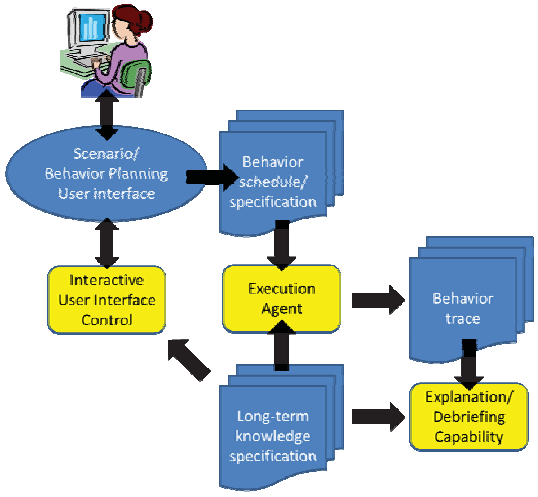

Figure 2. Multiple interpretations of HVI knowledge and behavior specifications.

7 POL agent execution

At run time, the *execution agent* follows the goals and actions dictated in each behavior schedule. For background agents, this typically executes a schedule of low-level goals. However, even low-level goals include adaptive constraints, such as pedestrians yielding to moving vehicles. These execute with high computational efficiency, because they do not involve any sophisticated sensing, understanding, or intelligent decision making. Rather, they require only efficient primitive functions for activities like route planning and schedule estimation. Foreground agents usually include schedule elements with unspecified parameter values or goal decompositions. The execution agent uses its decision logic and long-term knowledge specification to make choices, with the choices being sensitive to the current state of the execution environment. This allows foreground behavior that is not scripted, but is adaptive to the current situation. The execution agent fills in any details necessary to execute the mission. This type of adaptive, intelligent reasoning and situation understanding requires more computational resources than the simple schedules executed by background agents. Thus, we expect a scenario to include on the order of dozens of foreground agents.

After execution, an *explanation/debriefing* interface can examine population behavior patterns and apply long-term knowledge to generate explanations for the parameters, goals, and situational elements that contributed to decisions and actions generated by the execution agent. The interface uses the background knowledge base to reflect, after the fact, on the decisions that a foreground agent made during execution time [6]. This improves scenario creation and evaluation, by supporting traceability of the behavior to knowledge.

8 Current status

The POL architecture continues development under several DOD-funded projects, together with internal research and development efforts. We implemented the initial prototype under a DOD training program and DARPA Insight. Figure 3 shows a demonstration simulation environment running the initial POL prototype implementation. The prototype implemented background and foreground agents. The background engine interpreted schedules, translating goals into primitive actions using functions for path planning and action sequencing. It modeled several thousand agents per CPU core. The foreground engine used the Soar cognitive architecture [4] to implement situation understanding, hierarchical goal decomposition, and parameter-value selection at domain-specific choice points. We are extending this initial prototype through a collection of active projects, with specific recent focus on authoring.

Figure 3. A sample pattern-of-life simulation.

Currently, users author the scenarios, with a user developing background and foreground schedules with population parameters used to generate some of the schedule goals, timings, and parameters. The full capabilities for relationship management and interactive behavior planning are still in the design phase. We have evaluated the initial prototype on a set of use cases in a schoolhouse environment, demonstrating the ability to deploy large numbers of background agents, together with foreground agents that perform team-based insurgent activities. Background and foreground agents blend in the simulation environment. We have demonstrated that demographic parameters allow us to configure up to 320,000 background agents. We have run example scenarios populated with 20,000 background agents per CPU core, plus small numbers of foreground agents that blend in.

9 Related work

The POL architecture we have presented addresses the unique collection of requirements and applications described at the beginning of this paper. The design and implementation of the architecture has benefited from lessons learned from other research efforts that address portions of the POL problem.

9.1 Crowd modeling

Crowd modeling portrays realistic movements for low-fidelity, identical individuals in relatively small spaces [7]. Typical crowd models implement thousands of individuals moving in areas up to the size of a football stadium. Academic models, for example, analyze the best way to design exits to quickly empty an auditorium. The individuals have concepts of personal space and other data that makes small-area movement realistic, but they are interchangeable in that they do not have individualized choices. All agents exhibit essentially the same behavior with small variations, such as attempting to use different doors or different loitering areas. Thus, looking carefully at individuals in these crowds reveals they are merely walking around random points and not following any latent patterns. They do not possess reasons for their choices or actions, and they do not implement demographically based constraints and decision logic.

9.2 Social network modeling

Social network modeling describes groups of individuals that differ in their properties and relationships. These models represent relationships between individuals as connection graphs [8][9]. The networks are often randomized to create a particular overall graph structure. These models, may have many different individuals, but are often valid only in the aggregate. For example, such models in the academic world have yielded results about what mechanisms for making and breaking links are typical in the network as a whole. These models focus on social dynamics, but typically ignore POL dynamics related to the physical world.

Random graph generation implies a lack of realism when observed closely. As an example, in a real-world social network, advertisers have recently begun to value having people "like" their products on Facebook. This has led to unscrupulous providers creating fake individuals and selling their "likes." However, random generation does not follow real patterns of life, so it is possible to detect these fake individuals – one product might be liked almost entirely by people in a single foreign city, and another might be liked by one person in each city, with no overlap. Both of these mistakes reflect a failure to model the POLs that should geographically cluster individuals with remote, second- and third-order similarities and interpersonal connections.

9.3 Abstract population models

Abstract population models simulate large groups with low detail. The population may include a few powerful individuals, or no individuals and only groups. For example, one such simulation models political, military, and other kinds of power as stocks and flows [10]. Some abstract population models include goals and beliefs associated with each stock that direct the flows to achieve goals, but most models do not. Abstract population models in general focus on group relationships and influences, and they do not model individual behaviors that would fit into an overall pattern of life.

10 Conclusions and future work

POL modeling differs from typical AI modeling in its focus on breadth and density of behavior, as well as the importance of agent behaviors organizing into appropriate high-level patterns. POL requires solutions that emphasize efficiency in behavior authoring as well as execution, and that model a continuum that blends background and foreground behaviors. Our POL architecture design addresses these requirement and currently implements components sufficiently to demonstrate efficient generation of high-density background populations that represent varied cultural demographics and provide a suitable context for sophisticated foreground agents to blend into.

An important next step is automatic generation of POL based on interaction with standard representations of a population. The vision is to ingest a geographical and statistical profile, available from open-source intelligence reports, and combine this with socio-cultural representations to generate background agents automatically. Once achieved, whether for training or for planning, the system will ingest just-in-time information and support high-fidelity simulations of scenarios for direct relevance to ongoing missions. A second consideration is modeling communications that affect POL. Communications have significant impact on behavior. Locals flee when they hear police are coming. Crowds form when they learn of a controversial rally through twitter. These drivers of POL deviations make a difference in practice, and we plan to implement authoring and run-time representations for these effects in the next iterations of our work.

11 References

[1] Schatz, S., Folsom-Kovarik, J. T., Bartlett, K., Wray, R. E., & Solina, D. 2012. Archteypal patterns of life for military training simulations. *Proceedings of I/ITSEC*. Orlando, FL.

[2] Folsom-Kovarik, J. T., Schatz, S., Jones, R. M., Bartlett, K., & Wray, R. E. (2014). AI challenge problem: Scalable models for patterns of life. *AI Magazine 35*(1), 10–14.

[3] Carpenter, L. W. 2012. Emerald Warrior 2012: Joint, coalition exercise offers special operations forces irregular warfare training. *Airman.*

[4] Laird, J. E. 2012. *The Soar cognitive architecture.* MIT Press.

[5] Wray, R. E., & Jones, R. M. 2005. An introduction to Soar as an agent architecture. In R. Sun (Ed.), *Cognition and multi-agent interaction: From cognitive modeling to social simulation,* 53–78. Cambridge, UK: Cambridge University Press.

[6] Taylor, G., Jones, R. M., Goldstein, M., Frederiksen, R., & Wray, R. E. 2002. VISTA: A generic toolkit for visualizing agent behavior. *Proceedings of the Eleventh Conference on Computer Generated Forces and Behavior Representation.* Orlando, FL.

[7] Helbing, D. and Molnar, P. 1995. Social force model for pedestrian dynamics. *Physical review E, 51*(5):4282. Hughes, R. 2002. A continuum theory for the flow of pedestrians. *Transportation Research Part B: Methodological, 36*(6):507-535.

[8] McPherson, M., Smith-Lovin, L., & Cook, J. M. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 415-444.

[9] Mislove, A., Marcon, M., Gummadi, K. P., Druschel, P., & Bhattacharjee, B. 2007. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement* (pp. 29-42). ACM.

[10] Taylor, G., Bechtel, R., Knudsen, K., Waltz, E., & White, J. 2008. PSTK: A toolkit for modeling dynamic power structures. *Proceedings of the Seventeenth Conference on Behavior Representation in Modeling and Simulation. Providence*. Providence, RI.