Semi-Automated Construction of Adversarial Agents for Trainable Automated Forces

(Demonstration)

Robert G. Abbott*
Sandia National Labs
Albuquerque, NM
rgabbot@sandia.gov

Kiran Lakkaraju
Sandia National Labs
Albuquerque, NM
klakkar@sandia.gov

Christina Warrender
Sandia National Labs
Albuquerque, NM
cewarr@sandia.gov

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Intelligent Agents

Keywords

Live,Virtual,Constructive

ABSTRACT

Virtual agents are used by the military for extensive training of pilots. However, creating virtual agents with the appropriate behaviors is a lengthy and specialized process. We describe a new tool (TAF) to allow rapid creation of virtual adversarial agents by non-specialized personnel. In TAF, users can draw diagrams of agent behavior which are matched within a pre-existing knowledge base. By combining several instances of similar behaviors a behavior model for a new agent is created that matches the diagram provided by the user.

1. INTRODUCTION

Modern combat jets are complex entities that require thousands of hours of training before pilots are proficient enough to fly in real missions. To train, students spend thousands of hours in simulators (as well as many hours in real jets) fighting against virtual adversarial agents in high fidelity simulation environments.

"Pucksters" are teachers who, behind the scenes, manually control adversary agent behavior to fulfill the scenarios training objectives. As training scenarios become lengthier and more complex it is becoming expensive to maintain the required personnel to support training within simulations.

Thus recent simulation platforms (particularly the "Next Generation Threat System" (NGTS)) have allowed users to create behaviors for virtual agents in a flow-chart like visual language. The resulting agent models are called "Behavior Transition Network" (BTN) (Figure 2).

Compared to earlier systems in which agents automated only through writing conventional computer code, BTNs are a significant step forward in usability. Still, because of the complexity of BTNs (which can have up to 100s of nodes/edges to respond to a wide variety of situations), specialized personnel are required for development.

Current BTN development processes are costly, and do not efficiently adapt over time to the needs of end users, new platforms, or changes in tactics, techniques and procedures (TTPs). Specialized personnel requirements results in high support costs over the life-cycle of the simulation system.

Trainable Automated Forces (TAF) addresses the difficulty of specifying behaviors for agents by inferring the details required to produce realistic and responsive behaviors, based on a simple sketch provided by the user. Our vision is for the user to plan a training mission like an American Football coach explaining a new play to his team, by drawing a sketch on a whiteboard. The players (agents) are able to interpret the sketch because of their background knowledge of the sport, and the commonality among almost all plays. And as with a coach, some information beyond the sketch is required.

2. THE TAF APPROACH

This section describes the process of creating a new agent behavior using TAF (as shown in the accompanying demonstration video) and adds more detail on the approaches used in the implementation.

First, the user creates a mission sketch in the TAF Sandbox (Figure 1) which is based on the "Relational BlackBoard" (RBB) [1] [1] The Sandbox is essentially a temporal drawing program - that is, each data point has an associated time, and the trajectory has bounded start and end times. The Sandbox offers a palette of agent types (which for NGTS includes different types of aircraft on each side of the engagement). For each agent, the user selects a type and draws its trajectory. After sketching, the user searches the TAF Knowledge Base (KB) for relevant scenario traces. The KB is a collection of NGTS scenario executions generated from hand-crafted BTNs included with NGTS. The KB contains both the time-varying state of each simulation (e.g. agent positions) as well as a model trace of the BTN

*Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. This research was supported by ONR award N0001414IP20021.

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*

[1]RBB is available at http://rbb.sandia.gov

execution for each agent that produced the observed behavior. The essential function of TAF is to create the reverse mapping, i.e. identifying a sequence of NGTS nodes that will produce a desired outcome. This is how TAF generates a detailed behavior from a high-level specification, such as a sketch. So the success of TAF hinges on the completeness of its knowledge base and the relevance of results from its search algorithm.

Matching Algorithm

The search algorithm first locates mission executions in the KB with the desired number and type of agents. KB scenarios could potentially include large numbers of agents over a long period of time, and TAF considers all assignments of agents in the query to entities of the same type in the KB scenario. [2]

Next the algorithm extracts time-varying spatial features derived from agent positions, such as distance between agents and the difference in heading. The on-the-fly feature extraction creates a multi-dimensional trajectory through feature space. Users can select and de-select features to improve search results.

The extracted features are rotationally invariant but scale-dependent, since range is a crucial aspect of air engagement - e.g. various sensors, weapons, and aircraft are distinguished largely by how far they can reach, etc.

Rate (or velocity) is a special challenge when deriving behaviors from sketches. Aircraft velocity is key in fighter combat, but sketching in proportion to actual desired velocities is not feasible. Yet the sketch may contain some useful information on relative rates (such as one fighter catching up to another), and turn rates. The TAF matching algorithm selectively discards rate information by resampling paths (from both the sketch and the KB) at uniform increments of distance (e.g. 1 km) rather than time.

Next, TAF matches individual maneuvers to segments of trajectories. TAF uses the Dynamic Time Warping algorithm to perform segmentation and matching. We have extended the implementation from [2] to perform segmentation by matching against multiple patterns at every time step, keeping track of the best match at each time step, and reporting it as a match (and eliminating all tentative lower-scoring matches) as soon as it is determined that a match cannot be exceeded by any future match (the patterns have finite length).

Generating Agent Behaviors TAF provides a list of matches to the user, who can visually examine the matches and view the model trace that generated the matching trajectory. The model trace is the sequence of NGTS nodes whose names and meanings have been determined by a task analysis and decomposition, so they are human-readable.

Creating a new, cohesive behavior from a collection of snippets is non-trivial. For example, many NGTS nodes require a specific agent to be the current target; but this may occur before the start of the matching portion of the trace in most cases. Selecting a target is a complex process that normally involves using various radar models.

[2]This creates a combinatorial problem which we currently avoid by populating the KB only with small, self-contained "mini-scenarios"; segmenting interacting entities in a large scenario into small subsets of entities that actually interact is another possibility.

Using a Bayesian model of the frequency of invocation for behaviors in the hand-coded BTNs, and the common links between them, we can match against a formalization of the preconditions for the commands in the new BTN to compose a behavior that is self-contained and self-consistent.

We view the model trace as being generated by a Markov Chain [3] (with each node being a state). We create an adjacency matrix from the transition matrix of the Markov Chain by thresholding on the transition properties. Post processing is done to add necessary nodes and metadata to the BTN.

Finally, the generated BTN is provided to the user for their use.

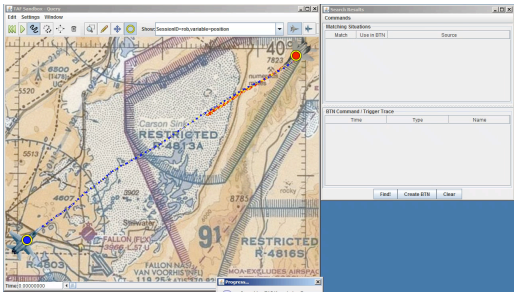

Figure 1: The TAF Sandbox application.

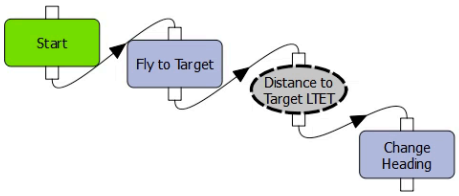

Figure 2: An example BTN. Nodes are primitive behaviors provided by NGTS. Node descriptions: *Start node* = beginning of the tree. *Fly to Target* = instructs agent to fly to a particular target. *Distance to target LTET* = a conditional (or *trigger* in BTN terms) that fires when the agent is Less Than or Equal To some predefined distance to the target. *Changes Heading* = Agent changes heading when previous node fires.

3. REFERENCES

[1] R. G. Abbott. The relational blackboard. In *Proceedings of the BRIMS Conference*, 2013.

[2] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 262–270, New York, NY, USA, 2012. ACM.

[3] S. M. Ross. *Introduction to probability models*. Academic Press, Amsterdam; Boston, 2010.