VIPER: A Text Based Environment for Intelligent Agents

Jeremiah Hiam (jwhiam@psu.edu), Changkun Zhao (cuz111@psu.edu), and
Frank E. Ritter (frank.ritter@psu.edu)

Overview

VIPER, a virtual implementation of plural environmental representations, provides a lightweight spatial world to model the evolution and behavior of complex social networks where you would like to simulate how agents move in a 2d world to generate and exercise networks. VIPER provides a way for agents (including human ones using a text interface) to move and talk with each other, as well a uniform and easy way to connect multiple agent types (e.g., heavier agents such as Soar, ACT-R) The world can, of course, also be populated with included lighter Java agents. Because agents can connect from outside of VIPER, it can support large networks spanning multiple machines (e.g., using high-performance computing). We have created simulations with over 1,000 agents, but we believe the limit is much higher.

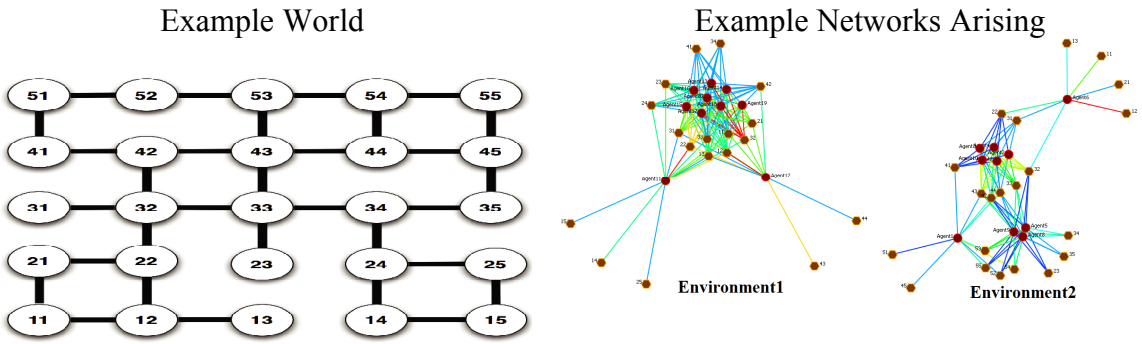

Figure 1. Simulated world and networks that arise.

The use of external agents supports studying network formation and evolution by enabling multiple cognitive agents to interact. This network representation arises from and in the agent's declarative representation of its network of friends and is arises from the agents path and who they meet. These memory structures can be subject to human-like memory decay and thus vary over time. Because these social networks are constrained by memory and, to some extent, the agent's path planning capabilities, they reflect some key limitations associated with embodied social networks. Further, this approach provides a way to study how agent types, environments, and other factors influence a network's evolution and connectedness.

Viper supports the creation and use of multiple maps; this feature enables researchers to examine interactions at varying levels of complexity. The analyst creates a room map or uses a provided map. Rooms can represent rooms or other units of space. The size of a map or room can vary from a room in an office building to a block on a city street. The map defines the connectivity of these rooms, and thus defines a powerful constraint on interaction. Agents can look, move, and talk to either the room (all the agents currently occupying the room) or to individual agents in the room. Updates and commands are passed through a text interface similar to Adventure or MUDs.

Included Example Agents

Three types of independent agents are included with VIPER for populating the world. The agents are implemented in ACT-R and Common Lisp and use Telnet to interact.

(1) Random-Move Agent. The random-move agent is implemented in ACT-R 6. It randomly explores the environment. When the agent comes into a room, it will randomly pick an available direction to move. It keeps no spatial memory of the environment.

(2) Intelligent-Agent. A preliminary intelligent-agent simulates human exploring and navigating using declarative memory mechanisms in ACT-R. This agent has the ability to build a symbolic representation of the rooms it has encountered. This agent currently explores the environment and tries to reach every room in the environment. In this task, when the agent comes into a new room, it will recall its memory and make a decision to go to the room that it has not yet reached. In later experiments, we can test how memory influences the behavior of agents by changing such parameters as previous knowledge, noise in cognition, and memory activation constraints.

(3) Herbal-Random-Move Agent. The Herbal/ACT-R agent provides a simple ACT-R model (a random-move agent) created using the Herbal high-level language. We augmented the Herbal/ACT-R complier to support creating this model.

Technical details

VIPER is created in C based on Naked MUD. The simulation handles complex computations server side, but uses additional Python scripting. This Python scripting allows for an extremely flexible implementation of ideas, including persistent objects (modifiable objects that retain their changes even across runs). Further, this scripting model allows for the simulation itself to call on outside programs to launch and create events within the environment. An example of this feature would be a specific testing area that would call for Java program to populate itself with scripted agents. These additional agents provide additional layers of complexity by providing many more interactions for the intelligent agents. Agents connect and interact in VIPER using the Telnet protocol (RFC 854). Languages such as Java and LISP provide robust Telnet libraries. VIPER runs asynchronously; the speed and frequency of communication is determined by each agent. VIPER is designed so that variations in performance originate from the agents participating in the environment, as opposed to being a function of the environment.

VIPER does not and cannot record the internal state of agents in it because it has no access to this information. VIPER, however, does record temporal and spatial behaviors of agents, as well as agent interactions including strings passed between agents. Logs are saved in a CSV file for use in tools like ORA.