Swarm Works! Details Inside

Paul E. Johnson, Dept. of Political Science, University of Kansas & Swarm Development Group

http://lark.cc.ukans.edu/~pauljohn/Swarm

Model Objectives: To demonstrate the scheduling mechanisms available in Swarm and validate the implementation of models with decentralized, autonomous agent behavior. This example employs a model of public opinion and interaction for illustration.

Background: While all Swarm models are agent-based models, some models will orchestrate their actions from high level objects while others will allow the agents to "schedule themselves" in a decentralized way. The Swarm class is vital because it controls the sequence of actions in the model. Swarms can be created at several levels so that the actions they imply are nested. Roughly speaking, there are two designs for agent-based simulation models.

Top Down Scheduling: Programs can be designed with "top down" schedules, setups in which a higher level Swarm tells each agent to move (take a "step"). This approach is useful for traditional models of cellular automata, such as the Game of Life. These models can repeatedly iterate through the list of agents, either in a fixed or random order. The models are supposed to represent "bottom up" processes, such as emergent properties, but the schedules themselves are imposed from the top.

Bottom-up (Decentralized, Dynamic) Scheduling: A Swarm program can be designed with "bottom up" scheduling, a setup in which each individual agent has its own scheduling apparatus. Each agent decides individually when it wants to act. This is especially meaningful if time is thought of as a series of days or years, each of which is composed of smaller time steps. The Swarm Activity library is designed to merge these individual actions into the overall time framework in a meaningful way. If agents schedule themselves to act at the same time, their actions can be executed in a first-come/first-serve manner or in a randomly sorted order.

Software: In Objective-C, variants of Robert Axelrod's model of public opinion/culture have been implemented.

The Model: Each agent has an array of discrete-valued opinions. The agents "wander" about, looking for other agents that they can interact with. In this version of the model, agents employ a social comparison approach when they decide whether to change their opinions after they disagree. When asked to change an opinion, the agent will check with other agents with whom it has interacted in a "friendly" way, that is, they agree more than one half of the time. The agent will change an opinion only when a majority of the friendly contacts support the change.

Implementation: C preprocessor flags are used to control which scheduling mechanism is used in a given build. The "top down" schedule approach creates a single schedule and it periodically instructs agents to plan their interactions and add them to the "master schedule." The "bottom up" model creates each individual agent from the Swarm class. Each agent has a schedule, and it places its own actions into that schedule. At the beginning of each day, agents plan their actions, and the last action they plan is a "wake up" call which will prompt them to repeat the cycle. It is then the responsibility of the Swarm library to make sure that the autonomous actions of the agents are executed in a meaningful sequence.

Screenshot: The model's display includes graphs summarizing the perceptions of the agents about the diversity of their society, objective measures of opinion diversity (entropy), as well as grids which show the state of opinion in some locations.

Validation: Many bottom-up simulations are impossible to validate because there is no benchmark against which their performance is compared. Suppose, for example, that a bottom up model is built on the premise that each agent has a unique number of time steps per day. There is no way to write a top down model that would represent that premise. Rather, decentralized scheduling is required and we rely on the Swarm library to integrate the disparate actions into a meaningful sequence. It is necessary to employ a decentralized scheduling approach. However, after designing the simulation, one has no easy way to be sure that the sequence of actions is a correct representation of what is intended.

In order to build confidence in the decentralized scheduling that Swarm provides, it is important to consider cases which can be validated. This model is designed so that the "top down" and "bottom up" designs should generate numerically similar results. Each agent's day consists of 10 time steps, and in the first time step of each day, each agent randomly chooses a time point during which to interact with another agent. The only difference in the sets of runs is that the agent behavior is prompted by the master schedule in the top down model, while the bottom up model has fully decentralized agent scheduling.

A comparison of the runs under the two setups allows us to validate the scheduling machinery included in the Swarm libraries. Observe that, across 500 runs of the model, the histograms showing the number of iterations to convergence and the average level of perceived homogeneity are extremely similar. All other summary measures for the models also support the contention that the decentralized scheduling of Swarm generates the intended sequence of actions.