

## Policy Analysis using $Q$ -learning

Ceyhan Eksin

University of Pennsylvania

Ackoff Collaboratory for Advancement of the Systems Approach(ACASA)

Department of Electrical and Systems Engineering

220 South 33rd Street

Philadelphia, PA 19104-6315

ceksin@seas.upenn.edu

**ABSTRACT:** *Building of complex white box models brings the need to build tools that guide validation, verification and analysis processes. The goal of this study is to develop an automated tool for policy analysis. The tool utilizes an approximate reinforcement algorithm to improve the behavior of the simulation according to predefined objectives. The Stochastic and complex nature of the model makes approximate learning algorithms a good fit for the problem. The approximation technique requires a summary of information about the model that the user finds essential. This information is subjective. Hence, depending on the run results, user may verify whether user's understanding of the model overlaps with the model's representation of the system. Therefore, the tool merges a policy analysis phase with the verification and validation phases.*

### 1. Introduction

Recently, there has been a strong initiative in many fields such as economics, decision sciences, and psychology etc. to have descriptive white box approach to modeling socio-economic systems. Specifically, agent-based simulation has been one of the popular methodologies to model social phenomena. Combined with the white box approach, agent-based models focus on descriptive representations of human behavior which further introduce complexity to socio-economic models. Complexity in the models comes deliberately from the desire to capture and explain the dynamics of systems. It is often impossible even for an expert familiar with the model to interpret and analyze the results such complex computational models. This is the main reason why these models did not meet the expectations of many scholars (Richiardi et al., 2006).

According to Richiardi et al. (2006), the underlying problem of complex white box models is the lack of evolved, automated and standardized analysis tools that help verify, validate, fine tune, and design policies. Like Richiardi et. al., there are papers that call for formal methodological guidelines to model building process i.e. verification, validation, calibration and/or sensitivity analysis specifically for agent based models (Windrum et al., 2007). These papers spot the reasons for the need of rigorous methodology and either raise questions or provide suggestions on how to proceed. There are also papers that provide theoretical guidelines to validation and/or analysis processes (Gonenc and van Daalen, 2009, Glenn et al., 2004). These papers elaborate on the questions they raise and provide theoretical answers but they "do not provide precise prescriptions" (Gonenc and van Daalen, 2009). There are supplementary papers that provide tools along some theoretical guidelines (Moss, 2008,

Kase and Ritter, 2009, Schreiber and Carley, 2007). General consensus in the literature is that verification, validation and policy analysis are essential phases of model building and require structured protocols and guided tools.

This paper proposes a tool that can be useful in model validation, scenario and policy analysis. Current application is on policy analysis. A policy analysis is the process of designing applicable policies that improve performance according to predefined objectives. Given certain objectives in the simulation world, our aim is to guarantee some improvement compared to benchmark runs using a learning algorithm. Our particular goal is to have a tool that can guarantee reasonable improvement in expected performance for a stochastic model without having to simulate the model many times with multiple steps. The main reason for trying to minimize the number of runs is concern for computation time. For this purpose, we use  $Q$ -learning algorithm which requires single training run. The learning algorithm replaces the decision making mechanism of a particular agent. Hence, the application looks for plausible policies for that agent. The application is on a complex agent based model of a country developed using PMFServ (Silverman et al., 2006), a software for building agent-based models with socio-cognitive agents.

The paper is organized as follows. First, we introduce Approximate  $Q$ -learning algorithm. The following section goes over PMFServ and the country model. Application section will define the model specific properties of the algorithm and discuss the results. The final section concludes with a discussion of reflections of the tool to model validation.

## 2. Approximate $Q$ -Learning Algorithm

In  $Q$ -learning (Watkins and Dayan, 1992), the algorithm learns which action is profitable for a given state.  $Q$ -learning algorithm requires a single run for training. Then trained function is used for performance improvement. Given a state, an agent can switch to another state by taking an action,  $u \in U$ . Each state has a certain cost associated with it. The goal of the agent is to minimize the total cost.

$Q$ -learning algorithm is a function that maps combination of state space,  $\mathbf{S}$  and action space to real space,  $Q : \mathbf{S} \times \mathbf{U} \mapsto \mathcal{R}$ .

$$Q(i, u) = \sum_{j=1}^n p_{ij}(u)(c(i, u, j) + \alpha J_{\mu}(j)) \quad (1)$$

where  $c(i, u, j)$  is the cost of transition from  $i$  to  $j$  by taking action  $u$  (Bertsekas, 2005a).  $J_{\mu}(j)$  is the cost-to-go with policy  $\mu$  and generally can be defined as below:

$$J_{\mu}(i) = E_w \left\{ c_k(i, \mu(i), w) + \alpha J_{\mu}(f(i, \mu(i), w)) \right\} \\ \text{for all } i \in \mathbf{S}$$

where  $w$  is a random variable and  $f$  is the function that represents the model i.e.  $i + 1 = f_k(i, \mu(i), w)$  (Bertsekas, 2005b). Notice that the models we are interested in do not have mathematical representations for  $f$ . Additionally, we are not given transition probabilities for computing the expectation as done in equation 1. Hence, we introduce a parametric architecture for approximation of  $Q$ ,  $\tilde{Q}(i, u, r)$ . We have  $\tilde{Q}(i, u, r) \approx Q(i, u)$  in linear form,

$$\tilde{Q}(i, u, r) = \phi(i, u)'r \quad (2)$$

where  $r = (r_1, \dots, r_m)$  is the parameter vector.  $\phi(i, u)$  is called features vector, a vector with known scalars,  $\phi_k(i, u)$ , that depend on state  $i$  and action  $u$ . This type of approximation is called feature extraction. It is a process that maps the state  $i$  and action  $u$  into some other vector  $\phi(i, u)$ .

$$\phi(i, u)' = (\phi_1(i, u), \dots, \phi_m(i, u))$$

These features are handcrafted based on insight and experience on the model. They are meant to capture the most important aspects of the current state. For example, in chess where the state is the current position of the pieces on the board, appropriate features can be balance of pieces, their mobility, king safety, etc (Shannon, 1950). Eventhough approximation is linear, we can capture nonlinearities in the model by crafting features well (Bertsekas, 2005a).

Once approximate  $Q$ -factors are obtained, we can use the minimization

$$\bar{\mu}(i) = \arg \min_{u \in \mathbf{U}(i)} \tilde{Q}(i, u, r) \quad (3)$$

to obtain the optimal policy.

The algorithm is very similar to the optimistic approximate policy iteration methods based on temporal difference(TD). The only difference is it uses approximate values of  $Q$ . The

pseudocode for the algorithm is given as such (Bertsekas, 2005b):

At the beginning of iteration  $k$ , simulation is at some state  $i_k$ , agent has chosen a  $u_k$ , and we have the current parameter vector  $r_k$ . Then:

We simulate the next transition  $(i_k, i_{k+1})$ . We generate the action  $u_{k+1}$  by using the minimization

$$u_{k+1} = \arg \min_{u \in \mathbf{U}(i)} \tilde{Q}(i, u, r)$$

We calculate the TD

$$d_k = c(i_k, u_k, i_{k+1}) + \alpha \tilde{Q}(i_{k+1}, u_{k+1}, r_k) - \tilde{Q}(i_k, u_k, r_k)$$

Then parameter vector is updated using

$$r_{k+1} = r_k + \gamma_k d_k \nabla \tilde{Q}(i_k, u_k, r_k)$$

where  $\gamma_k > 0$  stands for the step size. Then the process is repeated after replacing  $r_k$ ,  $i_k$ , and  $u_k$  with  $r_{k+1}$ ,  $i_{k+1}$ , and  $u_{k+1}$ , respectively (Bertsekas, 2005b).

We say that the algorithm has converged when  $d_k$  approaches 0. When  $d_k$  reaches zero, we can say that parameter vector,  $r$  is learned and we can use  $\tilde{Q}(i, u, r)$  for policy analysis. Literature has varying suggestions for choice of algorithm specific variables such as discount factor,  $\alpha$ , and step sizew,  $\gamma$ . Through out the study, we have them as constants where  $\alpha$  is 0.9 and  $\gamma$  is 0.1.

## 3. PMFServ and Model Definition

PMFServ is a human behavior emulator that drives agents in simulated gameworlds. This software was developed over the past 11 years at the University of Pennsylvania as a "model of models" architecture to synthesize many best available models and best practice theories of human behavior modeling (Silverman et al., 2006). PMFServ models profile the traits, cognitions, and reasoning of agents to capture the cognitive-affective state and reasoning abilities of agents. PMFServ agents can play the roles of leaders, follower archetypes, and institutional ministers that allocate services to others based on cultural norms, corruption, and other inefficiencies.

The country model (Silverman et al., 2009) is built using agents in PMFServ. The agents in the country base their decisions solely on the current state of the world. Each agent's action has a certain impact on determining the next state of the world. The next state of the world only depends on the actions taken in the previous step. Each agent perceives the state of the world, and other agents around. The agents are socio-cognitive i.e. they are aware of the agents around them, and have feelings of their own and toward other agents. They develop emotions based on their profile (traits, norms, relations etc.) and the actions of their own and others. For further discussion and mathematical underpinnings of profiling leaders and followers refer to (Silverman and Bharathy, 2005) and (Silverman et al., 2007a).

The country model includes all the important political and

ethnic groups in the region. There are two types of agents within a group: Follower and Leader. Each group can have multiple followers but only one leader. Similarly, leader agent can only lead one group but a follower agent can be a member of multiple groups. Groups have relations with other groups corresponding to socio-economic, political and ethnical conflicts which has role in determining the action space of agents. For example, an action to attack is not available if you are perceiving your friend. Leaders are the agents that take action on behalf of the group. Leader manages the *resources* (Security, Politics, Economy etc.), and in and out group *relations*. In-group relations stand for relations between leader and followers. Followers show their support for the group leader via a property called *membership*. Additionally, welfare of the followers is important for the leader if she wants the support of the followers. Leader also incorporates *Capital* i.e. economic situation of the group into her decision mechanism. Out-group relations are the actions based on group's relations and *relative power* to the target group(s). Leader's decision making is affected by whether the leader feels *superior* and whether the leader's group has higher relative power. Additionally, all agents have an aggregate variable called *VID* (Vulnerability, Injustice, Distrust). *VID* is a directed metric i.e. an agent has *VID* against all groups which shows whether she feels vulnerable toward that group or she is treated unjustly by that group or she trusts the group or not. The values for this parameter are negative meaning the more negative they are, she feels less vulnerable toward that group. Further descriptions and mathematical representations of leader/follower modeling can be found in (Silverman et al., 2007b), (Silverman et al., 2008) and (Silverman et al., 2009). All of these parameters mentioned above create the context the agent is in. Context can be considered as the circumstances or the state that the agent is in. We are specifically interested in the circumstances right at the time of the agent's decision. Given the context, the agent decides based on maximizing her subjective expected utility, *SEU*, that depends on her personality. The word subjective comes from the fact that each agent has different traits and norms which are reflected as the weights of the utility function. For example, given the same context two agents would decide to act differently because of the difference in their profile. Each action satisfies these norms and traits with certain probability; therefore, we consider expected utility.

Additionally, the model is stochastic. Stochastic nature of the model comes from the randomness in the result and effects of the actions. Hence, an action such as Give Economy (Economic Aid) might fail under certain circumstances with a given probability.

## 4. Application

This section explains the application of the algorithm to the country model. First, we will parametrize the model information discussed in Model Definition section and then define features using them. Second, we will define the cost function i.e. the objectives for policy analysis. Final subsection will provide the computational results and discuss methodological ideas based on computational results.

### 4.1. Defining Features

The set of features ( $\phi$ ) was based on majority of the variables discussed in the model description. Notice that these variables are already aggregated variables that summarize certain parts of the state. These variables do not exhaust the variables that make up the state space. They also do not exhaustively cover the information that can be extracted from the model. They were chosen so that they contain the sufficient information for the algorithm to converge and provide good policies. Choice of features depends on the researcher and is limited with his available insight and experience. Hence, there is no correct set of features but there is set of features that work.

We start by properly parametrizing state variables of interest to be able to define features.  $g \in \mathcal{G}$  denotes a group.  $x \in \mathcal{A}$  denotes an agent.  $\mathbf{VID}_k(x, g) \in (0, 1)$  is the *vulnerability*, *injustice* and *distrusted* at time  $k$  of  $x \in \mathcal{A}$  directed towards group,  $g$ .  $\mathbf{R}_k(g_1, g_2) \in (-1, 1)$  is the *relationship* between  $g_1 \in \mathcal{G}$  and  $g_2 \in \mathcal{G}$ .  $\mathbf{RP}_k(g_1, g_2) \in (-1, 0)$  is the *relative power* of  $g_1$  over  $g_2$ . The negative number indicates a stronger  $g_1$  than  $g_2$ .  $\mathbf{GP}_k(g) \in (0, \infty)$  stands for amount of "good" properties which is a sum of group's *capital* divided by 52 (each step is a week and there are 52 steps in a year) and group's property *economic output*. In other words, it is another economic indicator. Leader cannot take certain actions if they have insufficient capital.  $\mathbf{RS}_k(g) \in (0, \infty)$  stands for the total resources of group  $g$  at step  $k$ .  $\mathbf{S}_k(x, g) \in (-1, 1)$  stands for how *superior* the agent  $x$  feels over the group  $g$ . This is a summary of agent's emotions toward groups.  $\mathbf{FVID}_k(f, g) \in (0, 1)$  basically stands for the same thing as  $\mathbf{VID}_k(x, g)$ .  $\mathbf{f}$  stands for the follower agents of the group,  $\mathbf{f} \in \mathcal{F}$  where  $\mathcal{F} \cup \mathcal{L} = \mathcal{A}$  and  $\mathcal{F} \cap \mathcal{L} = \emptyset$ .  $\mathbf{FM}_k(f, g) \in (0, 1)$  looks at a follower agent's *membership level* toward a group,  $g$  at time  $k$ .  $\mathbf{FW}_k(f) \in (0, 1)$  denotes the *welfare* of a follower agent,  $f \in \mathcal{F}$ . Notice that this is not directed to any group as it represents the current situation of the follower. The parameters that sum up to  $\mathbf{FW}_k(f) \in (0, 1)$  are *BasicNeedsLevel*, *Capital*, *EducationLevel*, *SuppressionLevel*, *HealthLevel*, *JobsLevel*, *LawLevel*. These are the properties of the follower which the leader have direct influence on.  $\mathbf{SEU}_k(u) \in (0, 1)$  is the subjective expected utility associated with the decision at time  $k$ . As mentioned in the previous section, each agent differs in her utility function from others based on her profile.

These are the aggregate elements that summarize the huge state space,  $x$ . Hence, we can think of a function  $\Psi : \mathbf{X} \mapsto \varphi$

where

$$\begin{aligned}
 \varphi(1) &= \sum_{g \in \mathcal{G}} \frac{\text{VID}_k(x, g)}{\text{card}\mathcal{G}} \\
 \varphi(2) &= \sum_{g_2 \in \mathcal{G}} \frac{\text{R}_k(g_1, g_2)}{\text{card}\mathcal{G}} \\
 \varphi(3) &= \sum_{g_2 \in \mathcal{E}} \frac{\text{RP}_k(g_1, g_2)}{\text{card}\mathcal{E}} \text{ where } \mathcal{E} \subset \mathcal{G} \text{ are enemies of } g_1 \\
 \varphi(4) &= \frac{\text{RS}_k(g)}{\sum_{k=1}^t \text{GP}_k(g)/t} \\
 \varphi(5) &= \frac{\text{GP}_k(g)}{\sum_{k=1}^t \text{GP}_k(g)/t} \\
 \varphi(6) &= \sum_{g \in \mathcal{G}} \frac{\text{S}_k(x, g)}{\text{card}\mathcal{G}} \\
 \varphi(7) &= \sum_{f \in \mathcal{F}} \frac{\text{FVID}_k(f, g)}{\text{card}\mathcal{F}} \\
 \varphi(8) &= \sum_{f \in \mathcal{F}} \frac{\text{FM}_k(f, g)}{\text{card}\mathcal{F}} \\
 \varphi(9) &= \sum_{f \in \mathcal{F}} \frac{\text{FW}_k(f, g)}{\text{card}\mathcal{F}} \\
 \varphi(10) &= \text{SEU}_k(u)
 \end{aligned}$$

And we can summarize,

$$\phi(i, u)' = (f_1(\varphi(1), u), \dots, f_{10}(\varphi(10), u)) \quad (4)$$

Next, each action was divided into positive diplomacy ( $\mathcal{U}_{DP}$ ), positive economy ( $\mathcal{U}_{EP}$ ), positive military ( $\mathcal{U}_{MP}$ ), negative diplomacy ( $\mathcal{U}_{DN}$ ), negative economy ( $\mathcal{U}_{EN}$ ), and negative military ( $\mathcal{U}_{MN}$ ). Then, using insight about the model, we tried to spot certain situations where taking actions from a certain set would be advantageous. Functions  $f_1, f_2, \dots, f_{10}$  map the conditions,  $\varphi(\cdot)$ , and actions,  $u$  to real numbers. For example, if the leader agent feels superior and powerful with respect to her enemies and has follower support then she might be inclined to take risky aggressive actions for the purpose of increasing her group's resources. Hence, features vector is a 10 by 1 vector where value of each  $\varphi(\cdot)$  would define a context in which a certain action is favorable. I denote this features vector  $\phi(i, u)_C$  standing for a features vector.

## 4.2. Defining Cost Function

Before going into analysis, we need to define cost function,  $c(i, u, j)$ . It is defined as the total sum of the resources at the current step for the chosen leader agent,  $l$ , plus some penalty ( $f_c$ ) related to leader's actions;

$$c(i, u, j) = -\text{RS}_k(g) + f_c(u) \quad (5)$$

where  $f_c(u) = -\text{SEU}_k(u) + p(u)$ . And finally  $p(u)$  depends on the action set that  $u$  belongs to. Specifically,

$$f(n) = \begin{cases} 0 & \text{if}(u \in \mathcal{U}_{DP}) \\ 0.2 & \text{if}(u \in \mathcal{U}_{DN}) \\ 0.2 & \text{if}(u \in \mathcal{U}_{EP}) \\ 0.4 & \text{if}(u \in \mathcal{U}_{EN}) \\ 0.4 & \text{if}(u \in \mathcal{U}_{MP}) \\ 0.6 & \text{o/w} \end{cases}$$

This way more peaceful and diplomatic actions are preferable than negative or military actions unless they really have a high utility. Notice that the cost function does not depend on the following state,  $j$ . The cost function is designed so that leader takes actions to increase her resources.

## 4.3. Results

This section summarizes and discusses the results obtained from the experiments with feature vectors. The results are preliminary and they require further investigation. The plots of parameter vector  $r$ , and  $d_k$  are provided. There are 3 training runs made for 52 steps. Elements of the parameter vector seem to converge to same point (Figure 1). This shows us that only a single training run is enough to obtain the parameter vectors. Additionally, we see that  $d_k$ s converge to zero for all training runs (Figure 2).

Figure 3 summarizes the results of benchmark, training and trained runs. Benchmark runs constitute of runs of the model without the training algorithm i.e. the agent of interest acts according to the same decision making mechanism as the other agents. The benchmark decision making mechanism is based on picking the action that maximizes SEU. Notice that SEU maximization has no direct relation to maximization of resource levels. Maximization of SEU represents the action that fits best with the leader's views and norms. In training runs, the Q-learning algorithm replaces the subjective expected utility maximization for the agent of interest. For the trained run, the agent acts according to the action that minimizes  $\tilde{Q}$ . Since the aim is to obtain a policy that will increase the total resource level of the chosen agent, the performance measure is the Total Resource Level.

Finally, looking at the resources, training runs obtain higher resource values than the benchmark run (Figure 3). Of course to guarantee an improvement in performance and develop trust on the policy, we need to look at multiple benchmark runs since the model is stochastic. We have done 3 benchmark runs and observed that in all of these runs resources

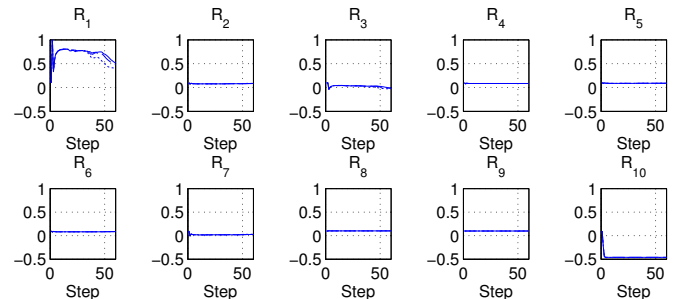


Figure 1. Three runs of the algorithm for  $\phi(i, u)_C$ , plots the 10 elements of the parameter vector,  $r$

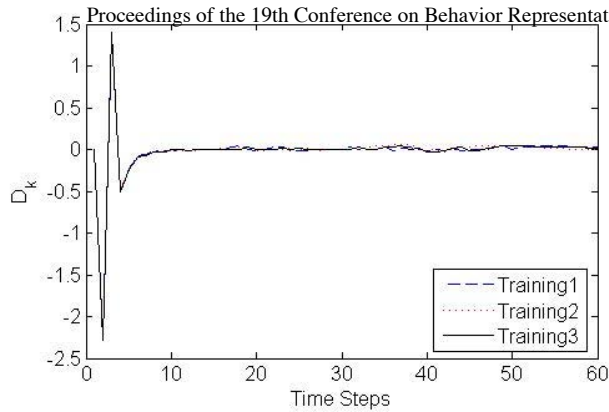


Figure 2. Three runs of the algorithm for  $\phi(i, u)_C$ , plots  $d_k$

are strictly less than resources in training runs. The run with trained parameter vectors obtain a reasonable improvement. Trained  $\tilde{Q}(i, u, r)$  is the resulting policy function that will dictate the actions to take. We need further runs with the trained  $\tilde{Q}(i, u, r)$  to show that the algorithm did not converge prematurely. We also need additional runs to show that the trained  $\tilde{Q}(i, u, r)$  works for different initial conditions. Current results suffice to say that there has been a reasonable improvement in resource levels when the leader adheres to algorithm's decisions.

So far, we discussed computational results. However, these are not the most interesting parts of the results. More interesting results come from the nature of the approximate  $Q$ -learning algorithm. Specifically, the way we define features vector reflect our insight on the model (recall the chess example). They correspond to which information we feel is important to take actions towards reaching the desired objectives. Looking at Figure 1, we observe that  $R_1$  and  $R_{10}$  corresponding to functions (see Equation 4) that depend on **VID** and **SEU** are the most influential in the policy. This means next time we develop features vector for the same model and objective, we might consider a simpler parameter vector that consider these two variables and a combination of the others. Moreover, when the algorithm

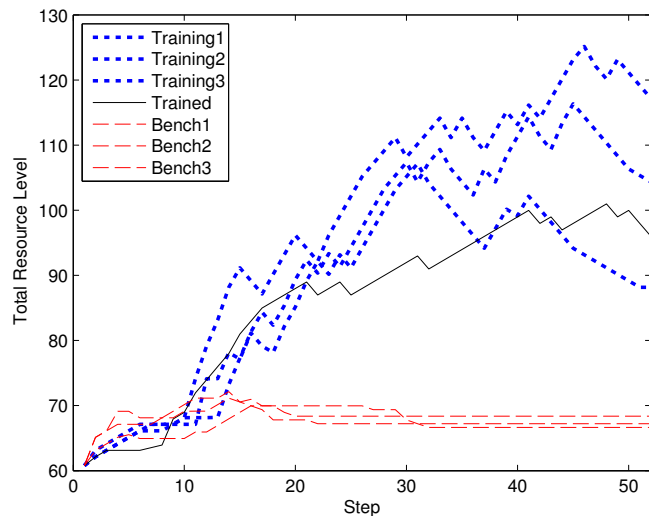


Figure 3. Total Resource Level for three benchmark runs(dashed), three training runs of the algorithm for  $\phi(i, u)_C$  (dotted) and single run with trained  $\tilde{Q}$ (solid)

converges and the results show improvement, that reassures our understanding. This can be considered a sanity check for the model. Furthermore, we need to look at how the leader agent's actions differ from the benchmark runs in trained runs. This corresponds to validating the means to achieve goals. If the actions taken to minimize cost do not make sense then we can infer that there is something wrong with the model. This sanity check is a way to poke structural representation (representation of underlying mechanisms) of our model. Valid structural representation makes sure that the goal of the model is not "just replication but also explanation" (Gonenc and van Daalen, 2009).

## 5. Discussion and Conclusion

We implemented a reinforcement learning algorithm to achieve certain goals in the model by letting the algorithm decide for the agent. One generalization is to make the algorithm control multiple agents. In that case, the algorithm returns a vector of actions that has cardinality equal to the number of agents. Although the implementation seems easy, it will be harder to define features.

Throughout the paper, we have avoided the case where convergence fails. This is simply because convergence is achieved in this study. However, if the results do not converge, then we might have to reconsider our understanding of the model and/or the structural representation of the model. The worst case scenario for convergence is when the model has a lot of volatility and the state space is huge. In that case, training runs might take infinite steps for convergence. This might fool us to question our understanding of the model i.e. features selection. This would be a false rejection of our correct understanding and representation of the system.

The tool is proposed for policy analysis. Yet, we see that both success or failure to achieve convergence can leave us with valuable information about the model. The design of the algorithm gives room to the experimenter to reflect her insight about the model. Although this might sometimes be cumbersome, it enforces the experimenter (usually the model builder) to reflect and summarize her ideas once more and cross check them with the model during policy analysis. Hence, policy analysis is added to the iterative loop of model verification and validation.

## References

Bertsekas, D. (2005a). *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific, 3. edition.  
 Bertsekas, D. (2005b). *Dynamic Programming and Optimal Control*, volume 2. Athena Scientific, 3. edition.  
 Glenn, F., Neville, K., Stokes, J., and Ryder, J. (2004). Validation and calibration of human performance models to support simulation-based acquisition. In *Proceedings of the 2004 Winter Simulation Conference*, pages 1533–1540.  
 Gonenc, Y. and van Daalen, E. (2009). An objective-based perspective on assessment of model-supported policy processes. *Journal of Artificial Societies and Social Simulation*, 12(4):3.

- Kase, S. and Ritter, F. (2009). A High Performance Approach to Model Calibration and Validation. In *Proceedings of the Eighteenth Conference on Behavior Representation in Modeling and Simulation (BRIMS)*, pages 07–BRIMS–10.
- Moss, S. (2008). Alternative approaches to the empirical validation of agent-based models. *Journal of Artificial Societies and Social Simulation*, 11(1):5.
- Richiardi, M., Leombruni, R., Saam, N. J., and Sonnessa, M. (2006). A common protocol for agent-based social simulation. *Journal of Artificial Societies and Social Simulation*, 9(1):15.
- Schreiber, C. and Carley, C. (2007). Agent Interactions in Construct: An Empirical Validation using Calibrated Grounding. In *Proceedings of the Sixteenth Conference on Behavior Representation in Modeling and Simulation (BRIMS)*, pages 07–BRIMS–54. Orlando, FL: Simulation Interoperability Standards Organization.
- Shannon, C. (1950). Programming a digital computer for playing chess. *Phil. Mag.*, 41:356–375.
- Silverman, B. and Bharathy, G. (2005). Modeling the Personality and Cognition of Leaders. In *Proceedings of the Fourteenth Conference on Behavior Representation in Modeling and Simulation (BRIMS)*.
- Silverman, B., Bharathy, G., and Kim, G. (2009). The New Frontier of Agent-Based Modeling and Simulation of Social Systems with Country Databases, Newsfeeds, and Expert Surveys. In Uhrmacher, A. and Weyns, D., editors, *Agents, Simulation and Applications*. Taylor and Francis.
- Silverman, B., Bharathy, G., and Nye, B. (2007a). Personality Profiling is Politically Correct. In *IITSEC*.
- Silverman, B., Bharathy, G., Nye, B., and Eidelson, R. (2007b). Modeling factions for 'effects based operations': Part i - leader and follower behaviors. *Journal Computational and Mathematical Organization Theory*, 13(4):379–406.
- Silverman, B., Johns, M., Cornwell, J., and O'Brien, K. (2006). Human behavior models for agents in simulators and games: Part i - enabling science with pmserv. *Presence*, 15(2):139–162.
- Silverman, B., Johns, M., Cornwell, J., and O'Brien, K. (2008). Human behavior models for agents in simulators and games: Part i - enabling science with pmserv. *Presence*, 14(2):120–155.
- Watkins, C. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8:279–292.
- Windrum, P., Fagiolo, G., and Moneta, A. (2007). Empirical validation of agent-based models: Alternatives and prospects. *Journal of Artificial Societies and Social Simulation*, 10(2):8.

## Author Biographies

**CEYHUN EKSIN** is a Ph.D. candidate in the Department of Electrical and Systems Engineering at University of Pennsylvania. He is affiliated with Ackoff Collaboratory for Advancement of the Systems Approach(ACASA).