Automatic Generation of Agent Behavior Models from Raw Observational Data

Bridgette Parsons[1], José M Vidal[1], Nathan Huynh[2], and Rita Snyder[3]

[1] Department of Computer Science and Engineering
[2] Department of Civil and Environmental Engineering
[3] College of Nursing's Healthcare Process Redesign Center
University of South Carolina, Columbia, SC, USA
parsons@email.sc.edu

Abstract. Agent-based modeling is used to simulate human behaviors in different fields. The process of building believable models of human behavior requires that domain experts and Artificial Intelligence experts work closely together to build custom models for each domain, which requires significant effort. The aim of this study is to automate at least some parts of this process. We present an algorithm called MAGIC, which produces an agent behavioral model from raw observational data. It calculates transition probabilities between actions and identifies decision points at which the agent requires additional information in order to choose the appropriate action. Our experiments using synthetically-generated data and real-world data from a hospital setting show that the MAGIC algorithm can automatically produce an agent decision process. The agent's underlying behavior can then be modified by domain experts, thus reducing the complexity of producing believable agent behavior from field data.

1 Introduction

Agent-based modeling has been used to simulate traffic patterns, markets, supply chains, wildlife ecology, and networking. It is a popular method for simulating complex systems because of its ability to show emergent behaviors, or behaviors that arise from the interaction between the different agents. Unfortunately, the creation of an agent-based behavioral model can be a difficult task, especially when modeling humans that are involved in complex processes. Frequently, simulation models involving human decision processes are created using observed behavior sequences. This model development paradigm requires that both the programmer and the domain expert work together to create a computational model which correctly reflects the observed behavior.

In this paper, we present MAGIC (Models Automatically Generated from Information Collected), an algorithm for extracting behavior models from raw observational data consisting of time-stamped sequential observations of the subject's behavior. Our behavior model, described in Section 3, resembles a Markov Decision Process (MDP), but with added support for cyclic behavior and additional nodes known as **decision points** that indicate when the agent requires

outside input in order to proceed. In order to demonstrate the ease of modification of the behavior model for use in simulation, we have also developed an editing tool that allows the model to be altered, and illustrated its use in a 3D simulation of a nurse administering medications to patients on a hospital floor.

We test our algorithm both in a synthetic test setting and a hospital setting where we build a simulation of a nurse as she carries out a medication administration process in a hospital. Data for the nursing simulation was gathered by following several nurses for 6 weeks as they administered medications to their patients [8, 7, 13]. Our experimental results demonstrate that the MAGIC algorithm can automatically build appropriate behavioral models.

2 Related Work

One possible method of building a model of human behavior is by deep analysis of the human decision process and human cognition. This is, in essence, the goal of cognitive psychology, which tells us that the heuristics humans use to make decisions are highly varied and individualized [4]. This lack of a clear model of the human decision-making process made an alternate method of deriving a decision process an attractive alternative.

Agent decision processes that have instead been derived from sequences of behavior observed over time have proved successful in many areas, including human behavior modeling. For example, in smart home studies, sensor pattern readings have been used to determine human behavior patterns in order to automate heating and lighting systems in accordance with the owners' lifestyle [10, 6, 2]. In the RoboCup competition, a framework was developed not only to learn from logged human behavior, but to then train other agents by using the behavior it had learned [3]. The 2012 BotPrize competition, an Unreal Tournament DeathMatch-style game where human judges attempt to distinguish between AI-players and humans, had a tie for first place between two bots that used mirrored human behavior sequences, fooling more than 50 percent of the judges in the competition [12]. Thus, there is ample evidence in different settings that agents that effectively and believably simulate human behavior can be built by deriving decision processes from observed sequences of human behavior.

In robotic planning, there has also been some success in deriving decision processes from observed behavior. The learning of primitives [1] or low-level actions [5] using variations of HMM's enables robots to learn by imitating behavior, although these methods necessitate online rather than offline learning. More recently, a method has been proposed to enable robots to learn offline using human-readable text files [9]. This method, however, requires natural language processing and the careful construction of an appropriate ontology, unlike our research, in which the task names are provided by domain experts, and behavior is recorded by trained observers.

In simulation, agents have required the specialized skills of AI experts working together with domain experts to create the needed agent behaviors. In contrast, our research aims to develop algorithms and tools to automatically build these

agents' behaviors using the raw observational data. That is, we want to take observed workflow data as input and output a generalized behavior model. Also, since these models will almost certainly require some modification, we propose to develop tools for domain experts, who are not AI experts or developers, to be able to modify these behaviors as needed.

3 Behavioral Model: Sequential Compressed Markov Decision Process

The type of behaviors we wish to model can almost be captured using a Markov Decision Process [11]. However, since MDP's do not allow for an internal state, they cannot be used to represent a finite loop of a length prescribed by an outside input. For example, in our healthcare domain, we need to represent the fact that a nurse will administer a fixed number of medications to a patient, so she will repeat a finite set of tasks some fixed number of times, such as 5 steps for each one of the 3 medications. We need a behavior model that can also represent these repeated sequences.

In this study, we created a variation of the Markov Decision Process that we will refer to as a Sequential Compressed Markov Decision Process, or **SCMDP**. This SCMDP extends the basic MDP by including decision points which have direct links to other states based on external inputs instead of a transition probability.

Definition 1. *(Sequential Compressed Markov Decision Process) An SCMDP consists of an initial state s_0 and an end state s_n both taken from a set S of states where $|S| = n$, a transition function $T(s, p, s')$, a set of decision points $D \subset S$, and a set of decision point transitions $P(d, s, e)$ where $d \in D$ and e is some external input.*

In the SCMDP, states correspond to tasks performed by the agent, such as "wash hands" or "enter room." The transition function T gives the probability p that the agent will transition from one state to another, therefore doing the corresponding task. All transition probabilities from any given state will always add to 1, as they do in an MDP. Start and end states s_0 and s_n are designated to account for the fact that only certain tasks are likely to occur at the beginning or end of a sequence.

The decision points D are a set of special states within the decision process. They represent the entrance to a cycle. Each decision point has at least two edges extending out from it. One edge goes to the first state in the cycle, and the other to the action that is to be taken after the cycle ends. The cycle begins and ends due to some external information e. The transitions out of decision points are represented by P. For example, a nurse agent might repeat the same set of tasks for each medication that must be administered to a patient. The external information in this case is the number of medications that the patient requires. The decision point keeps track of how many medications have been administered thus far and ends the cycle when there are no more medications

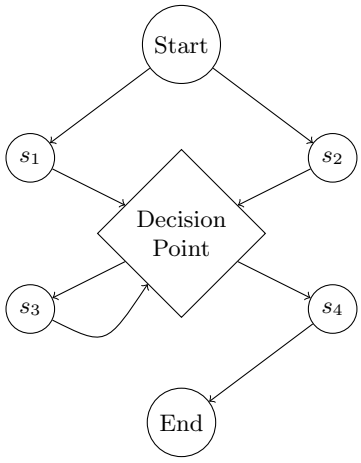

Fig. 1. Example SCMDP.

to administer. It is possible to have more than one transition out of a decision point, therefore requiring more than one piece of external information, such as whether the medication the patient needs is available, and whether it is located in the medication room or the pharmacy.

Figure 1 shows a simple example of an SCMDP. Note that at the decision point, the agent can either repeat the cycle by going back to s_3 or end the sequence by choosing to go to s_4. In this example, the cycle consists of only one state, s_3, but there could be any number of states before the agent gets back to the decision point. The decision between s_3 and s_4 is made using external information not shown in the diagram. In practice, this external information will depend upon the domain that the SCMDP is modeling.

3.1 The MAGIC Algorithm

The MAGIC algorithm, shown in Figure 2, takes as input a text file of sequential task observations and outputs an SCMDP. This input text file consists of a sequence of observations O, where each observation $o \in O$ is a sequence of tasks, $o_i = (t_1, t_2, \ldots t_{ki})$, that we have observed a person perform. For example, one observation corresponds to the sequence of tasks that we watched a nurse perform from the time she entered a patient's room on Monday 9:32 am until the time that she left the room. We assume that all of the observations have recognizable start and end points. In the nursing example, these start and end states correspond to a change in the patient's room number.

The MAGIC algorithm tries to identify and extract cycles in the raw input data, which is especially difficult given the fact that the data might contain errors in the form of transposed tasks. For example, in the observation $t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_3, t_5, t_4, t_8$ the set of tasks t_3, t_4, t_5 should be recognized as

```
MAGIC (O)
1  C = [ ] // List of cycles
2  m = k // Maximum number of tasks in a cycle, defined by user
3  O' = [] // Updated List of Observations
4  S = [] // List of lists of tasks that have been replaced with cycle pointers
5  for o ∈ O
6        o, S, C = MAGIC-ASSISTANT(o, S, m, C)
7        O'.append(o)
8  T = CALCULATE-TRANSITIONS(O', S)
9  return T, S

CONTAINS(o, s)
1  if ∃_{0≤i≤j≤|o|} o[i..j] ∩ s! = ∅ // Does o contain s, sequentially?
2        return i, j // If so, return the start and end points in o.
3  return NIL

MAGIC-ASSISTANT(o, S, m, C)
 1  t = ∅ // List of repeated tasks
 2  for c ∈ C
 3        for j = 0 to |o| − (|c| + 1)
 4              s, e = CONTAINS(o, c)
 5              if s, e ≠ ∅
 6                    o[s..e] = c // Replace the list of tasks with a pointer to the cycle in the cycle list
 7                    S.append(o[s..e]) // Add list of repeated tasks to list for transition calculations
 8  while m > 2
 9        for i = 0 to |o| − (m + 1)
10              j = i + m − 1
11              t = o[i..j] // Set of contiguous tasks taken from the observation
12              s, e = CONTAINS(o, t)
13              if s, e ≠ ∅
14                    if ¬∃_{c∈C} t ⊆ c // If t is not a subset of an old cycle
15                          C.append(t)
16                    o[s..e] = t // Replace the list of tasks with a pointer to the appropriate cycle
17                    S.append(o[s..e]) // Add list of repeated tasks to list for transition calculations
18        m = m − 1
19  return o, S, C
```

Fig. 2. The MAGIC algorithm. The CONTAINS procedure tells us if the list of observations o contains the set of tasks t anywhere within it, but contiguously. The MAGIC-ASSISTANT procedure identifies cycles, checks if they are subsets of existing cycles, and records any new cycles found.

a cycle because they appear twice, even if in different order: the first time as t_3, t_4, t_5 and the second time as t_3, t_5, t_4. This match is performed by the CONTAINS procedure, shown in Figure 2, which tells us if the list of observations o contains the set of tasks t anywhere within it, contiguously, and then returns the indexes i, j within o that mark the start and end of the set of tasks t, or NIL if they are not contained in o.

The MAGIC-ASSISTANT procedure takes as input a single observation o, the list of tasks S that have been replaced by a cycle, the current list of cycles found C, and an integer m which is the maximum number of tasks that we will allow in a cycle. MAGIC-ASSISTANT first checks to see if o contains any cycles that are already in the cycle list C, as seen in lines 6–7. If any are found, then it modifies o so that tasks that we recognized as belonging to c are replaced with a pointer to c in C (see line 6). The list of tasks that have been replaced is appended to S, so that the transition probabilities within the cycle can also be calculated.

MAGIC-ASSISTANT then steps through the observation sequence (see lines 8–18) selecting the maximum number of tasks in a cycle, converting them to a set s where $s = o[i..|s| - 1]$ and using the CONTAINS helper function to check for repetitions of that set of tasks in the same observation, that is, checking for a cycle. If a new cycle is found, we determine if it is a subset of one of the cycles that is already on the cycle list C. If the cycle is not yet on the list, it is added to C. The cycle is then replaced in the observation o with a reference to its location on the cycle list C. Finally, MAGIC-ASSISTANT returns the new modified observation o and the list of tasks S that have been replaced by a cycle $c \in C$.

The MAGIC procedure repeatedly calls MAGIC-ASSISTANT for each observation o and appends the new modified observations to O'. Finally, it calculates the transition probabilites T using the new O' and the list S by adding how many times a state follows another one and using the proportions as probabilities. In other words, if state s_6 appears right after s_2 in 1/3 of the observations where we see s_2, then we set $T(s_2, 1/3, s_6)$.

go-out, *throw-ball*, *chase-dog*, *throw-ball*, *chase-dog*, *throw-ball*, *chase-dog*, *go-in*
go-out, *throw-ball*, *chase-dog*, *throw-ball*, *chase-dog*, *go-in*
go-out, *throw-ball*, *go-in*
go-out, *throw-ball*, *chase-dog*, *throw-ball*, *chase-dog*, *throw-ball*, *chase-dog*, *go-in*
go-out, *throw-ball*, *chase-dog*, *chase-dog*, *go-in*
go-out, *chase-dog*, *throw-ball*, *throw-ball*, *chase-dog*, *throw-ball*, *chase-dog*, *go-in*
go-out, *throw-ball*, *chase-dog*, *throw-ball*, *chase-dog*, *throw-ball*, *go-in*
go-out, *throw-ball*, *chase-dog*, *throw-ball*, *chase-dog*, *throw-ball*, *chase-dog*, *go-in*

Fig. 3. Example input data for MAGIC algorithm.

As an illustration of the way that the MAGIC algorithm functions, consider the set of observations in Figure 3, which simulates the attempt to play fetch with a dog who doesn't seem to understand the concept of giving the ball back.

Since the length of the maximum observation is 8, we know the longest possible cycle will be 3, because the start and end states cannot be in a cycle. Thus, we set $m = 3$ in MAGIC. However, there are no cycles 3 tasks in length. The first and only cycle found is $c = (\textit{throw-ball}, \textit{chase-dog})$, which also matches the set $(\textit{chase-dog}, \textit{throw-ball}.)$ Each time c is found, the list of tasks that are replaced by the pointer to c in the list of cycles C is added to the list of lists of tasks S, to be used in transition calculations inside of the cycle. An illustration of the SCMDP produced by MAGIC is shown in Figure 4. At the decision point the agent needs the external knowledge of whether or not it has the ball, and whether or not the dog wants to play. If the agent has the ball, it can throw the ball. If not, it must chase the dog to get the ball. If the dog doesn't want to play any longer, the agent will go inside. Going outside is always the first event in the sequence, and going back inside is always the last event.

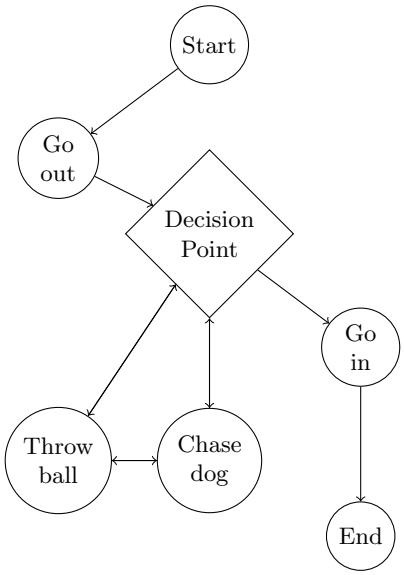

Fig. 4. Example SCMDP produced using MAGIC algorithm

The cycle created by our decision point ensures that, after completing the tasks of throwing the ball and chasing the dog, the agent returns to the decision point to once again make a decision based upon who has the ball, and whether or not the dog wants to play. This allows behavior that is based upon the human behavior pattern, but does not necessarily repeat one particular logged observation. For instance, if the agent goes outside and the dog does not want to play, the agent will go inside again. Likewise, the agent would continue playing fetch with the dog for more than three cycles if the dog still wants to play.

The modeler and the domain expert must choose the specific external inputs needed at the decision nodes. In this simple case, it is easy to determine that the input is simply whether or not the dog wants to play. In the case of a more complex model, however, the domain expert may need to tell the modeler what the agent would need to know in order to proceed. Well-named tasks in the logged data make this process simpler, so it is important for trained observers who are logging behavior to be as accurate and clear as possible in naming tasks. It is likewise important that they remain consistent. If the same task is given two different names by observers, it will appear as different tasks in the final model.

4 Validation Using Synthetic Data

In order to test how well MAGIC can extract cycles from raw data, we performed a test in which we created a synthetic model of a simple agent from which we could generate observational sequences. We then used the MAGIC algorithm to attempt to recover the original model from the observations.

The SCMDP we created mimics a player's movements in a first-person shooter "capture the flag" game. The agent has a single decision point called *Idle*. At this point, the agent needs to know if it is injured, needs ammunition, sees its opponent, or is at the checkpoint that must be seized in order to win the game. There are two possible initial actions: crouch or duck, and there are two possible final stages: win or die. The SCMDP used for this test is shown in Figure 5.

We used a Python script to generate 10,000 strings from the SCMDP and fed these as input 10 times to the MAGIC algorithm, for a total of 100,000 randomly generated strings. The resulting SCMDP mirrored the original's pattern, providing an appropriate decision graph for an agent in a first-person shooter "capture the flag" game. The transition probabilities found by MAGIC were, on average, within 0.19 percent of the ones in the original SCDMDP, with a variance of 0.08 percent, as shown by the black numbers (below) in Figure 5.

Our results show that, with the use of 10,000 strings, we are able to closely approximate the original pattern with minimal deviation between individual test runs. The low error rate in transition values indicated that, by adding enough data, we were able to overcome the disadvantage of unusual behavior patterns, allowing us to recover the correct pattern of behavior using the MAGIC algorithm. The identification of task cycles enabled us to determine the location of the decision point, indicating that the *Idle* state is a state where the agent would require further information before making a decision, rather than simply relying upon a percentage chance of a transition.

We then performed further tests on this SCMDP by adding Gaussian white noise with 1% variance to the input data, meant to simulate the type of errors we might encounter in data gathering and subject observation. The addition of this noise did not disrupt the location of the identification of the decision point. It did cause a minimal error in transition values, which was easily correctable by removing transitions that had less than one percent chance of occuring. This

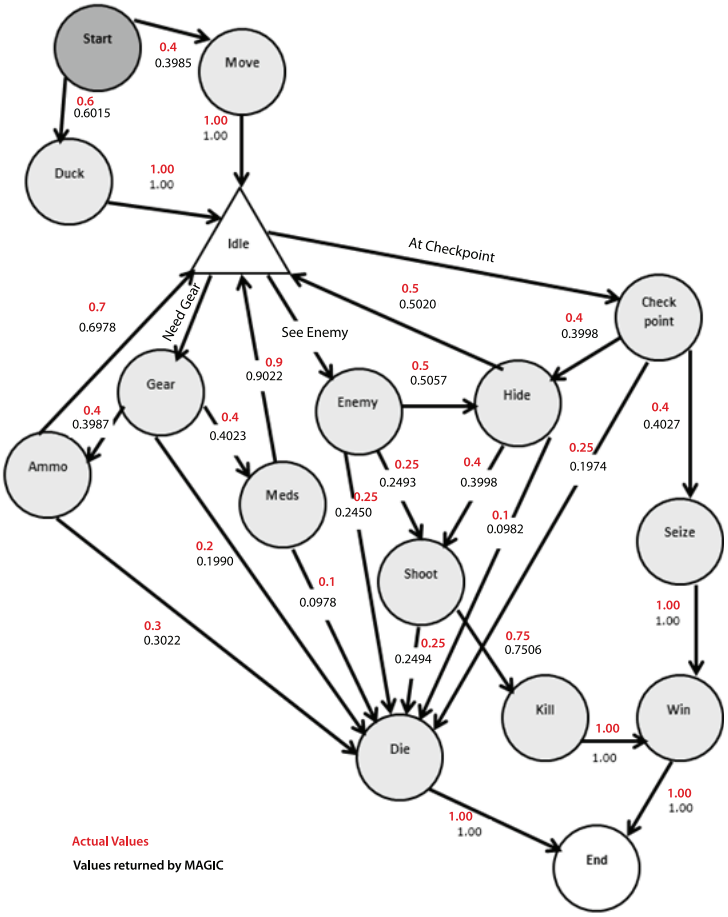

Fig. 5. SCMDP used for testing the MAGIC algorithm. The numbers in red (above) are the original transition probabilities in the SCMDP. The numbers in black (below) are the probabilities found by MAGIC.

slight adjustment to transition calculations also enabled better compensation for occasional unusual behavior patterns.

4.1 Validation With Real-World Data

A pilot study of the nurse medication administration process was conducted in a hospital setting [8, 7, 13]. In this study, over a 6 week period of time, nurses were shadowed by trained observers, and their activities were recorded using an iPad application. The actions used by the observers were chosen by domain experts. Observation data from 6 of the 17 observed nurses were used for the study, and the resulting files were combined into a CSV file. The start and end of

each observation sequence was determined by when a nurse entered and exited a room, as evidenced by the room number in the log files. In total, there were 10,391 tasks recorded which together comprised 313 observations.

An example of a subset of the data used is shown in Table 4.1.

Table 1. Example of nursing data.

Room Number	Behavior
628	enter_room
628	greet_patient
628	scan_patient_id
628	review_patient_computer_record
628	review_patient_med_box
628	scan_patient_meds
628	document_med_admin
628	scan_patient_meds
628	scan_patient_meds
628	document_med_admin
628	scan_patient_meds
628	document_med_admin
628	review_patient_med_box
628	scan_patient_meds
628	prepare_meds_for_admin
628	administer_meds
628	prepare_meds_for_admin
628	setup_for_med_admin
628	administer_meds
628	other_care

Despite the limited amount of sample data, we were able to achieve some success using the MAGIC algorithm. We were able to identify 12 decision points needing external information, such as the number of medications the patient required, or whether or not the patient needed special medication. Some of these were less obvious in nature, such as whether or not the nurse needed to wear gloves, whether or not the patient needed the medication explained, or whether the patient refused to take the medication.

The nursing study was particularly interesting because the nurses had two distinct approaches to patient care, as identified by domain experts (clinicians, in this case). We have referred to these approaches as bundled and unbundled. Nurses that took the unbundled approach visited a patient's room to administer medication, and then returned later to perform any other necessary tasks, while nurses that took the bundled approach performed all required tasks during the same visit. The SCMDP we obtained from the test data reflected the fact that it contained both methods, as indicated by the decision point that requires knowledge of whether or not the patient requires other care than simply administering

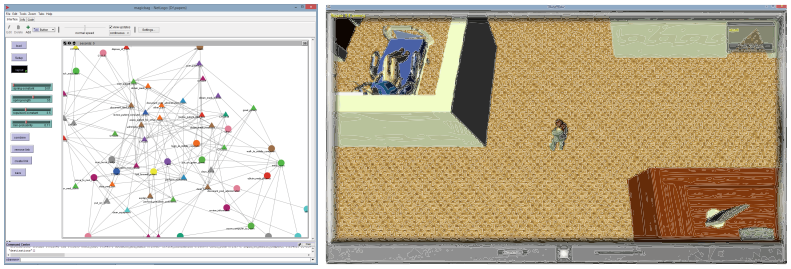

Fig. 6. The MAGICBAG Tool (left) and the NurseView simulation (right). Simulation video at `http://youtu.be/JH94PolDhZQ`

medications. While this pilot data set provided us with the location of the appropriate decision points, because of the difference in approaches to patient care, it will be necessary to have a greater number of observations to ensure the correct transition values.

Despite the smaller size of the data set, by using this format, we were able to create simulations using both NetLogo and the Unity3D game engine that can read the text file and use it as a logic controller for the nurse agent's behavior, as seen in Figure 6. This allowed the nurse domain experts to visualize current medication administration processes.

5 Summary

As cognitive modeling is difficult, imitation is a viable alternative to achieve believable human behavior in simulation. Statistical analysis of observed data allows us to achieve a pattern of human actions, essentially simulating human behavior by mimicking human behavior.

While building behavior models by hand can be complex and time-consuming, there is a better alternative. We have shown that it is possible to derive an agent decision process using the MAGIC algorithm which encapsulates the observational data in a small behavior model (SCMDP) that responds to external input, provided there is sufficient data, and tasks are labelled consistently.

Even with an automatically generated decision process, it will be necessary for an expert in the area that is being modeled to review the results. The process, however, will be less complex and time-consuming than making all of the necessary calculations by hand. The simple, standardized output format used in this study is easy to parse, allowing adjustments to be made quickly, and making it easy to load in a wide variety of simulation environments. Therefore, as part of our ongoing work, we have created a NetLogo tool, called MAGICBAG (MAGIC Behavior Adjustment Graph), which allows modelers to adjust the graph in a more visual and intuitive manner. We are continuing to refine this tool, and to test the MAGIC algorithm in different domains in order to further confirm its capability to work as a generic tool, rather than being domain-specific. We are

also developing methods to determine decision points that are not cycle-specific in order to alleviate more of the modifications to the model that must be made by the domain expert, thereby further reducing modeling time.

References

1. Bentivegna, D.C., Atkenson, C.G., Cheng, G.: Learning tasks from observation and practice. Robotics and Autonomous Systems 47(2), 163–169 (2004)
2. Cook, D.J., Youngblood, M., III, E.O.H., Gopalratnam, K., Rao, S., Litvin, A., Khawaja, F.: Mavhome: An agent-based smart home. In: Proceedings of the First IEEE International Conference on Pervasive Computing and Communications. pp. 521–524 (2003)
3. Floyd, M.W., Esfandiari, B., Lam, K.: A case-based reasoning approach to imitating robocup players. In: Proceedings of the 21st International Florida Artificial Intelligence Research Society Conference. pp. 251–256 (2008)
4. Gigerenzer, G., Gaissmaier, W.: Heuristic decision making. Annual review of Psychology 62, 451–482 (2011)
5. Guillory, A., Nguyen, H., Balch, T., Charles Lee Isbell, J.: Learning executable agent behaviors from observation. In: ACM The Fifth International Joint Conference on Autonomous Agents and Multiagent Systems. pp. 795–797 (2006)
6. Guralnik, V., Haigh, K.Z.: Learning models of human behaviour with sequential patterns. In: Proceedings of the AAAI-02 workshop "Automation as Caregiver". pp. 24–30 (2002)
7. Huynh, N., Snyder, R., Vidal, J.M., Tavakoli, A.S., Cai, B.: Application of computer simulation modeling to medication administration process redesign. Journal of Healthcare Engineering 3(4), 649–662 (2012), `http://jmvidal.cse.sc.edu/papers/huynh12c.pdf`
8. Huynh, N., Snyder, R., Vidal, J.M., Tavakoli, A.S., Cai, B.: Application of computer simulation modeling to medication administration process redesign. In: Chyu, M.C. (ed.) Advances in Engineering for Healthcare Safety, pp. 129–142. Multi-Science Publishing Co. Ltd. (2013)
9. Kaiser, P., Lewis, M., Petrick, R.P.A., Asfour, T., Steedman, M.: Extracting common sense knowledge from text for robot planning. In: IEEE International Conference on Robotics and Automation (ICRA) (2014)
10. Leon, E., Clarke, G., Callaghan, V., Doctor, F.: Affect-aware behaviour modelling and control inside an intelligent environment. Pervasive and Mobile Computing 6(5), 559–574 (2010)
11. Papadimitriou, C.H., Tsitsiklis, J.N.: The complexity of markov decision processes. Mathematics of Operations Research 12(3), 441–450 (1987)
12. Schrum, J., Karpov, I.V., Miikkulainen, R.: Ut^2: Human-like behavior via neuroevolution of combat behavior and replay of human traces. In: IEEE Conference on Computational Intelligence and Games. pp. 329–336 (2011)
13. Snyder, R., Huynh, N., Cai, B., Vidal, J., Bennett, K.: Effective healthcare process redesign through and interdisciplinary team approach. In: Studies in Healthcare Technology and Informatics, vol. 192. MEDINFO 2013 (2013), `http://jmvidal.cse.sc.edu/papers/snyder13a.pdf`