J. Indian Inst. Sci.
A Multidisciplinary Reviews Journal
ISSN: 0970-4140 Coden-JIISAD

REVIEW ARTICLE

Contextualized Behavior Recommendation from Complex Agent-Based Simulations of Disasters

Nidhi Parikh[1], Madhav V. Marathe[2] and Samarth Swarup[3]*

Abstract | We present an approach for generating contextualized behavior recommendations from a large, data-driven, complex agent-based simulation. We extend a previous method for generating a summary description by decomposing the output of a simulation into a tree of causally-relevant states, and show how behavior recommendations can be generated by ranking these causally relevant states in terms of their impact on an outcome of interest. An end-user can provide a query specifying a partial state description, which is used to retrieve the appropriate set of states from the summary description. The structure of the tree is used to generate the contexts that differentiate the behavior recommendations. We apply our method to a very complex simulation of a disaster in a major urban area and present results for multiple queries.

Keywords: *Behavior recommendation, Disaster preparedness, Agent-based simulation, Simulation analytics*

1 Introduction

Disasters are complex phenomena, where outcomes are driven by a combination of human behaviors, response efforts, physical circumstances, and preparedness and advance planning[1]. The complexity of the interactions between social and technical systems can lead to unforeseen consequences, which makes planning for disasters especially difficult. Agent-based simulations are increasingly being used to aid in understanding these complexities and in developing appropriate plans. In this work, we present a method for extracting behavioral recommendations from a large-scale complex agent-based simulation to help with planning. A behavioral recommendation is an answer to the question, what should people be doing in a particular situation?

In a complex disaster situation, the answer to this question is highly context-dependent. For example, the hypothetical disaster we study in this work is the detonation of a 10 kT improvised nuclear device in Washington DC (see Sect. 4.1 for details). In this case, we might reasonably expect that what people should be doing in the aftermath will depend on many factors, such as how close they are to ground zero (i.e., the point on the earth's surface directly above or below an exploding nuclear bomb), what their health status is, whether they are together with their household members, whether they are in the path of the fallout, and much more. Our approach here is not to try to find optimal behaviors in such a complex and dynamic multi-agent environment, but rather to try to understand if some *natural* behaviors are better than others.

By natural behaviors, we mean the behaviors that people are naturally likely to exhibit in disaster situations. To develop effective plans, it is important to understand when and where people's behaviors are helpful or harmful, with the goal of channeling their natural instincts in beneficial directions. For instance, restoring communication can be a relatively passive intervention which allows people to get in touch with their family members and can have the effect of making them more amenable to responding to

[1] *Information Systems and Modeling Group (A-1), Analytics, Intelligence and Technology Division, Los Alamos National Laboratory, Los Alamos, NM, USA.*
[2] *Biocomplexity Institute and Initiative, and Department of Computer Science, University of Virginia, Charlottesville, VA, USA.*
[3] *Biocomplexity Institute and Initiative, University of Virginia, Charlottesville, VA, USA.*
**nidhip@lanl.gov*

requests to shelter or evacuate[2, 3]. On the other hand, identifying behaviors that put people at risk is very useful for focusing the efforts of emergency responders.

Our approach is to use the agent-based simulation outputs, which show the overall results of the various natural behaviors that are modeled, and rank behaviors in terms of their effects on outcomes such as health. However, it may be the case that behaviors have different effects in different contexts, e.g., in the aftermath of a nuclear detonation scenario, seeking healthcare may improve an injured agent's health, but if the same agent is close to the blast area and seeking healthcare early on, then they may be exposed to more radiation and this behavior can actually be harmful to health. Any model that does not take into account contextual information would have to take the average across all contexts. This may lead to an inaccurate estimation of effects and an inaccurate ranking of behaviors, and may even estimate no effect for behaviors that have contradictory effects in different contexts. Therefore, we term this problem, *contextual behavior ranking*.

By contextual information, we mean any information (agent and environmental states) that may lead to different outcomes for the same behavior. This may include the agent state at the current time step as well as any previous state or behaviors that may lead to different trajectory of outcomes. Parikh et al.[4] proposed an algorithm to summarize simulation results by extracting causally-relevant states—states that have a measurable effect on the final outcomes. These causally-relevant states are the contextual information mentioned above.

The rest of this paper is organized as follows. We begin with an overview of simulation summarization[4], which forms the basis of our approach in this work. The summary representation that is constructed from the outputs of a simulation consists of a tree-structured set of causally-relevant states, as we explain in the next section. Then we describe how we can improve the prior simulation summarization method by extending it to extract causally-relevant states that capture temporal history. Then we present our method for contextualized behavior ranking using these new causally-relevant states. Then we show the results of applying this approach to a complex agent-based simulation of the nuclear disaster mentioned earlier. We end with a discussion of this and related problems in the larger context of simulation analytics.

2 Simulation Summarization

The simulation summarization approach is based on the computational mechanics framework[5, 6]. Let us first consider a stochastic process denoted by a sequence of random variables X_t, drawn from a discrete alphabet, $\mathcal{A}$. $\overleftarrow{X}$ denotes the "past" of the sequence, i.e., $X_{-\infty} \cdots X_{t-2}X_{t-1}X_t$, and $\overrightarrow{X}$ denotes the "future" of the sequence, i.e., $X_{t+1}X_{t+2} \cdots X_{\infty}$, following Crutchfield et al.[7, 8].

Crutchfield and Young[5] came up with an elegant model for such a time series: they group all the histories that predict the same future into a "causal state". They showed how to construct a state machine from these causal states, which they call an ϵ-machine[8]:

$$\epsilon\left(\overleftarrow{x}\right) = \{\overleftarrow{x}' | Pr\left(\overrightarrow{X} \mid \overleftarrow{x}\right) = Pr\left(\overrightarrow{X} \mid \overleftarrow{x}'\right)\}, \tag{1}$$

where $\overleftarrow{x}$ and $\overleftarrow{x}'$ are actual past sequences. They showed that this construction renders $\overleftarrow{X}$ statistically independent of $\overrightarrow{X}$ given the current causal state, thus making the ϵ-machine process Markovian. They also showed that ϵ-machines capture all of the information $\overleftarrow{X}$ contains about $\overrightarrow{X}$, thus making them optimally predictive.

Shalizi and Shalizi[9] developed an algorithm known as *Causal State Splitting Reconstruction* (CSSR) for learning an ϵ-machine representation from a given time series. CSSR incrementally constructs an ϵ-machine as a Hidden Markov Model (HMM)[10]. The HMM starts with just one hidden state, and more hidden states are added by using a statistical test to determine if the current set of states is insufficient to determine the distribution over future states.

The CSSR algorithm proceeds by evaluating longer and longer past sequences. In each case, it conducts a statistical test to compare the distribution over the next symbol. Let L be the length of the past sequences considered so far, and let Σ be the set of causal states estimated so far. CSSR, in the next step, looks at sequences of length $L+1$. If a sequence of the form αx^L, where x^L is a sequence of length L and $\alpha \in \mathcal{A}$ is a symbol, belongs to the same causal state as x^L, then we would have[9],

$$Pr(X_t | \alpha x^L) = Pr(X_t | \hat{S} = \epsilon(x^L)), \tag{2}$$

where $\hat{S}$ is the current estimate of the causal state to which x^L belongs. This can be tested using a statistical test, such as the Kolmogorov–Smirnov test. If these two distributions are found to be significantly different, then CSSR tries to match the

sequence ax^L with all the other causal states estimated so far. If $Pr(X_t|\alpha x^L)$ turns out to be significantly different in all cases, CSSR creates a new causal state and assigns αx^L to it. This process is repeated up to some length $L_{\max}$.

Parikh et al.[4] adapted the causal state formalism to large agent-based simulations. There are two key assumptions of CSSR, which are changed in this setting. Instead of relying on having a very long, stationary time series to be able to estimate the probability distributions, they rely on having a very large number of agents. Second, since a simulation is not a stationary process, they construct the optimal set of clusters at each time step, with respect to a final outcome, as explained below.

In their approach, an agent-based simulation is formalized as a set of agents. An agent is described by its current state, which is defined by a k-dimensional state vector $\mathbf{x}(t) = [x_1(t), x_2(t), \ldots x_k(t)]^\intercal$, which evolves over time. Let d_i be the number of possible values x_i can take. The simulation proceeds in discrete time steps from $t = 0$ to $t = T$. Let the number of agents be denoted by N.

The term 'state' is used in a broad sense. It can include, e.g., the agent's behavior, or functions of other agents in this agent's neighborhood. It can also include temporal state, e.g., if the agent has ever received an emergency broadcast (in emergency situations, the government may send out an emergency broadcast such as a text message with a recommended action for the given situation, for example in the nuclear disaster scenario, the emergency broadcast may suggest people to shelter in place to avoid radiation exposure), or the cumulative value of a variable (such as the total amount of radiation exposure).

The goal of summarization is to compress agents' trajectories through state space to a small number of key states that have a significant impact on the outcomes of interest. Let the outcome variable for agent a be denoted by y_a. We assume that y_a is an instance of a random variable Y. The summarization algorithm for discovering these causally-relevant states proceeds as follows.

The agent population is partitioned into a set of clusters, $C(t) = \{C_1(t), C_2(t), \ldots, C_m(t)\}$ at each time step. Initially, all the agents are grouped into just one cluster, i.e., $m = 1$ at $t = 0$. At each subsequent time step, the state of each agent changes because at least one of $x_1, \ldots x_k$ changes. The number of ways in which $\mathbf{x}$ can change is $d = d_1 \times d_2 \times \ldots \times d_k$.

Consider an arbitrary cluster of agents, $C_i(t)$. At time step $t + 1$, it can split into at most d groups, based on how each agent's state changes. However, some of these changes may be irrelevant if they fail to have a significant impact on the outcome variable. To test this, each group derived from $C_i(t)$ is treated as a candidate cluster, denoted by $\mathrm{CC}_{i,j}(t + 1)$, where $j \in 1 \ldots d$. At each step, $Pr(Y|C_i(t))$ is compared with $Pr(Y|\,\mathrm{CC}_{i,j}(t + 1))$ using the Kolmogorov–Smirnov test. The null hypothesis (analogous to Eq. (2)) is,

$$Pr(Y|\,\mathrm{CC}_{i,j}(t + 1)) = Pr(Y|C_i(t)). \tag{3}$$

If the null hypothesis is rejected (at a sufficient significance level) and $D_{\mathrm{KL}}(Pr(Y|C_i(t))||Pr(Y|\,\mathrm{CC}_{i,j}(t + 1)) > \delta$ $|\,\mathrm{CC}_{i,j}(t + 1)) > \delta$ (where D_{KL} denotes the Kullback–Leibler divergence), which is a parameter corresponding to effect size, then candidate cluster $\mathrm{CC}_{i,j}(t + 1)$ is accepted as a new cluster at time step $t + 1$. If none of the candidate clusters at time step $t + 1$ are accepted, then $C_i(t)$ is added to the set of clusters for time step $t + 1$.

The outcome of summarization is a decomposition of the entire simulation output into a tree structure of agent clusters. Each cluster splits from its parent only when the corresponding state change is informative about the final outcome of concern. In this sense, it is considered causally-relevant. This idea is relevant to our problem of contextualized behavior recommendation because it will allow us to define contexts in a meaningful sense as well as a direct score for ranking each state, which is the change in the outcome variable in this state. The trajectory of each agent traces a path through this tree structure. The trajectory is compressed compared to the full trajectory of the agent, since it retains only those time steps at which the cluster to which the agent belongs splits off from its parent cluster. The parameter δ allows control over how many new clusters are formed at each step, and consequently, how much compression of trajectories is achieved. Setting δ to a high value will retain only the clusters which have a large difference in outcomes from their parent clusters.

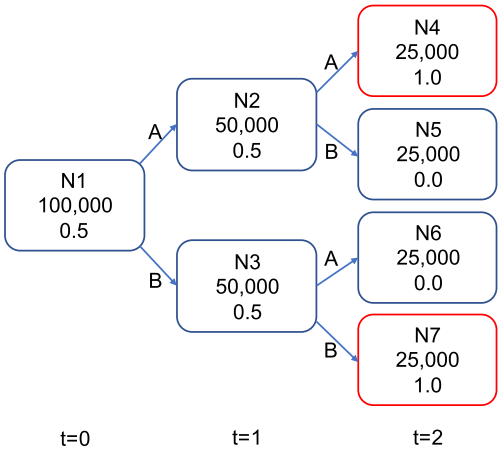

Figure 1: Tree structure of trajectories for example 1. Here, each node label consists of a *node id*, the number of agents belonging to that node, and the fraction of these agents that received reward. Labels on edges are actions. Nodes corresponding to trajectories that lead to a reward are marked red. Here, neither of the actions 'A' or 'B' at time steps 2 matter unless we had the information about their previous behavior and hence one needs to take into account this context information while ranking behaviors.

3 Approach

We first discuss the need to extend the method of the prior section to state sequences. Consider the following toy example, illustrated in Fig. 1. We have a large population of agents (say, 100,000) that start the simulation in state $N1$. The simulation runs for just two time steps and at each time step, each agent must choose one of two actions, A or B. An agent that takes the same action twice in a row gets a reward of 1 at the end of the second time step. An agent that takes two different actions in the two time steps gets a reward of 0. Figure 1 shows a tree structure of agent trajectories. An agent belongs to a node if it followed the trajectory (starting from the root node) leading upto that node. Nodes corresponding to trajectories that lead to a reward are marked red. In this example, exactly half the agents take each possible action in each state. In the figure, each node label consists of a *node id*, the number of agents belonging to that node, and the fraction of these agents that received reward. Labels on edges are actions.

The summarization approach of the previous section will proceed as follows on this example. It will first construct the final distribution over rewards for all agents grouped together. In this case, the distribution is [0.5, 0.5] as exactly half the agents end up with reward 1 (the proportion of agents who end up in states $N4$ and $N7$) and half with reward 0 (the proportion of agents who end up in states $N5$ and $N6$).

In the next round, the summarization algorithm will consider all the agents who took action A in time step 1, moving from $N1$ to $N2$. The distribution of rewards for this subset of the agents is once again [0.5, 0.5] as exactly half of these agents end up with reward 1 (in state $N4$) and half with reward 0 (in state $N5$). Since the distribution is exactly the same, the algorithm will not split the initial cluster from time step 0 into two clusters (causal states) in time step 1.

Now moving to time step 2, the summarization algorithm will once again consider all the agents who took action A. These are the agents who move from $N2$ to $N4$ in Fig. 1 and the ones who move from $N3$ to $N7$. Once again, exactly half end up with reward 1 and half with reward 0, resulting in the same distribution of rewards. The same applies to agents who took action B in either time step 1 or time step 2.

Thus, the summarization algorithm of the previous section fails to create any causal state clusters beyond the initial one corresponding to time step 0, and fails to reveal the difference between taking the same action in two consecutive time steps vs. taking two different actions.

For the behavior recommendation problem, suppose we want to rank behaviors at iteration 2. In absence of any contextual information, the effect of behavior 'A' at iteration 2 is 0.5 as half of the agents who were engaged in behavior 'A' at iteration 2 received reward. Similarly the effect of behavior 'B' is also 0.5 and hence both behaviors are ranked the same. Instead, if we could take into account the context information about the previous behavior and rank them separately for each possible context, behavior 'A' would be ranked higher if the previous behavior was also 'A' and vice-versa.

A more physically relevant (hypothetical) example is, in the nuclear blast scenario, sheltering at any given time step may not have any significant effect, but sheltering for a sufficiently long period of time may shield from radiation and prevent radiation sickness. Such causally-relevant sequences of states also provide the

required context and hence, we adapt the summarization algorithm to not only capture causally-relevant states but also causally-relevant sequences. These causally-relevant sequences are organized in a tree structure, which is then matched against the query for contextual ranking of behaviors and discovering the additional context required.

3.1 Finding Causally-Relevant State Sequences

We now adapt the summarization algorithm to capture effects of a sequence of states (or actions). Intuitively, the only change from the algorithm of Sect. 2 is that we need to keep track of state sequences when testing if new clusters need be formed. This process is explained in detail below.

Extending the notation established in Sect. 2, let $\mathbf{x}(t, t+k) = \mathbf{x}(t)\mathbf{x}(t+1)\cdots\mathbf{x}(t+k)$ denote the agent trajectory from time step t to $t+k$. As before, at each time step t, agents are partitioned into a set of clusters, $C(t) = \{C_1(t), C_2(t), \ldots, C_m(t)\}$. Initially, all the agents are grouped into just one cluster, i.e., $m = 1$ at $t = 0$. As time progresses, some of these clusters may split into more clusters as described in the next paragraphs. Along with each cluster $C_i(t)$, we now also associate the last time step when this cluster was split from its parent cluster, denoted as $C_i(t).t_{ls}$. For the initial cluster, $C_1(0)$ at time step $t = 0, C_1(0).t_{ls} = 0$.

Consider an arbitrary cluster of agents $C_i(t)$ at time step t. At the next time step, state of an agent can change in d possible ways and hence to evaluate possible effects of each change, the summarization algorithm of Sect. 2 creates d candidate clusters. Our goal is to not only to find causally-relevant states but also the causally-relevant state sequences, hence while creating candidate clusters at the next time step $t+1$, we consider increasingly longer past state sequences, $\mathbf{x}(t, t)$, $\mathbf{x}(t-1, t)$, $\mathbf{x}(t-2, t)$, and so on. Whenever a cluster splits from its parent cluster, any effects that the state variables upto that time step had on the final outcome have already been captured. So while creating candidate clusters for cluster $C_i(t)$, we only need to look at the past state sequences upto time step $C_i(t).t_{ls} + 1$. The candidate cluster associated with state sequence $\mathbf{x}(t-k, t)$ consists of all agents from its parent cluster whose trajectory suffixes match the state sequence $\mathbf{x}(t-k, t)$. The total number of states at any time step t is d and hence for $C_i(t)$, the maximum number of candidate clusters at time step $t+1$ are $d^{t+1-C_i(t).t_{ls}}$. The state sequence, associated with each candidate cluster $\text{CC}_{i,j}(t+1)$ where $j \in [1 \ldots d^{(t+1-C_i(t).t_{ls})}]$, is denoted by $\text{CC}_{i,j}(t+1)$.seq.

To check if the state sequence associated with a candidate cluster $\text{CC}_{i,j}(t+1)$ affects the final outcome, we compare the probability distribution over the final outcomes for this candidate cluster to the probability distribution over the final outcomes for its ancestor cluster at time step just before the start time of the state sequence associated with this candidate cluster (i.e., at time step $t+1-l$ where l is the length of $\text{CC}_{i,j}(t+1)$.seq). Let this ancestor cluster be denoted by $C_i'(t+1-l)$. We use the Kolmogorov–Smirnov (KS) test to evaluate the null hypothesis that the associated state sequence does not affect the probability distribution over the final outcomes, as shown below:

$$Pr(Y|\,\text{CC}_{i,j}(t+1)) = Pr(Y|C_i'(t+1-l)) \tag{4}$$

where l is the length of $\text{CC}_{i,j}(t+1)$.seq. As before, there is also a threshold δ on the "effect size" which is measured as the total variation distance between $Pr(Y|\,\text{CC}_{i,j}(t+1))$ and $Pr(Y|C_i'(t+1-l))$. For each candidate cluster $\text{CC}_{i,j}$, we associate its total variation distance from its ancestor cluster as $\text{CC}_{i,j}.\tau$. If the null hypothesis is rejected at a level α (say 0.001) and the total variation distance ($\text{CC}_{i,j}.\tau$) between $Pr(Y|\,\text{CC}_{i,j}(t+1))$ and $Pr(Y|C_i'(t+1-l))$ is greater than δ, then candidate cluster $\text{CC}_{i,j}(t+1)$ is added to a set of candidate splits CS for cluster $C_i(t)$.

Since we look at increasingly longer past sequences, the state sequence associated with one candidate cluster may be a suffix of the state sequence associated with some other cluster. Hence there may be an overlap (in terms of agents associated with these sequences) among some of these candidate splits. So among the overlapping candidate splits, we select the candidate split that has the most effect (i.e., the candidate split that changes the probability distribution over the final outcomes the most) using algorithm ChooseFromCandidateSplits (as shown in Algorithm 2).

In case of a tie in the effect size, we select the candidate split that requires the least amount of information (i.e., the candidate split that has the shortest state sequence associated with it) to predict the final outcomes. Hence the algorithm ChooseFromCandidateSplits first sorts all candidate splits by their total variation distances in descending order and length of the associated sequence in ascending order. Also, two candidate splits can overlap if and only if the state sequence associated with one of them is a suffix of the state sequence associated with the other candidate split. Hence the algorithm processes each candidate split in sorted order (i.e., starting with the candidate split with the highest total variation distance) and checks if the state sequence associated with it is a suffix of the state sequence associated with any of the candidate splits already accepted as the clusters and vice-versa. If not, then the current candidate split does not overlap with any of the candidate splits which are already accepted as clusters and hence is accepted as a new cluster and added to the set of clusters S as derived from the parent cluster *prevC* at time t. We also need to track the agents from the previous cluster *prevC* for whom the associated state sequence did not affect the probability distribution over the final outcomes (i.e., agents from the previous cluster *prevC* that do not belong to any of the clusters (set of clusters S) derived from it) for future computation and hence cluster c' is created of such agents. Finally, both set of clusters S and c' are added to the set of clusters at time t, $C(t)$.

As before, the entire simulation is decomposed into a tree structure of agent clusters. Now each cluster splits only when the change associated with corresponding state sequence is informative about the final outcome of interest. The trajectory of each agent traces a path through this tree structure.[A] With each node (cluster) in the tree, in addition to the associated state sequence, we also associate the time step at which it was split from its parent cluster and expected change (which serves as the score of the associated behavior sequence in that context) in the final outcomes (as compared to the parent cluster) that it lead to.

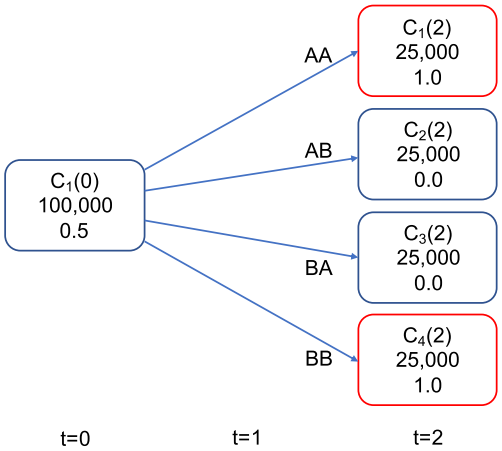

Figure 2: Causal state decomposition of the toy example. Each node in the tree is a causal state. The number in parentheses is the time step and the suffix is a numbering of the causal states in that time step. The second line in each box is the number of agents in that causal state, and the third line is the fraction of agents that end up with reward 1. The labels on the edges show the behavior sequences associated with the corresponding causal states.

Once again, the parameter δ allows us to control how many new clusters are formed at each step, and consequently, how much compression of trajectories we achieve. Setting δ to a high value will retain only the clusters which have a large difference in outcomes from their parent clusters. We also set the minimum size of clusters be 30 so that the number of samples in a cluster is enough for performing a statistical test. This poses a limit on splitting and hence identifying states that do not appear enough number of times. The algorithm for extracting causally-relevant state sequences is presented using pseudo-code in Algorithm 1.

[A] It may seem that similar to the CCSR algorithm[9], we could potentially merge all clusters that lead to the same probability distribution over the final outcomes at each time step t. However, as shown by a toy example in Appendix C of[11], merging clusters this way could potentially lead to suboptimal causal state sequences in the following time steps. Also, merging clusters would lead to nodes with multiple parents (i.e., simulation would be decompose into a graph that is not a tree) and the behavior recommendation algorithm described in Sect. 3.3, requires a tree structure to extract ancestor information (or historical context) for the clusters matching the query.

Algorithm 1: Algorithm for extracting causally-relevant state sequences.

input : $\mathbf{x}_a(1,T), y_a$, where $a \in 1 \ldots N$: trajectories up to time step t and outcome of interest for each agent a

output : $C(t), t \in 0 \ldots T$: a set of clusters for each time step, organized as a tree over time

parameters : α: significance level for Kolmogorov-Smirnov (KS) test

δ: "effect size" threshold on total variation distance

Initialization: $C(0) \leftarrow$ all agents

for $t \leftarrow 1$ ***to*** T **do**

  for ***each*** $prevC$ ***in*** $C(t-1)$ **do**

    $maxL \leftarrow t - prevC.t_{ls}$

    $CS \leftarrow \phi$

    for $l \leftarrow 1$ ***to*** $maxL$ **do**

      $j \leftarrow 1$

      for ***each*** *distinct* $\mathbf{x}(t-l,t)$ *for agents* $\in prevC$ **do**

        $CC_{i,j}(t) \leftarrow \{a | a \in prevC,$ a matches $\mathbf{x}(t-l,t)\}$

        $CC_{i,j}(t).seq \leftarrow \mathbf{x}(t-l,t)$

        Test the null hypothesis (Eqn. 4) for candidate cluster $CC_{i,j}(t)$ and its ancestor cluster $C'_i(t-l)$ at time step $t-l$

        if *null hypothesis is rejected at level* α **then**

          $\tau \leftarrow$ TotalVariationDist$(Pr(Y|C'_i(t-l)), Pr(Y|CC_{i,j}(t)))$

          $CC_{i,j}(t).\tau \leftarrow \tau$

          $CC_{i,j}(t).t_{ls} \leftarrow t$

          if $\tau > \delta$ **then**

            $CS \leftarrow CS \cup CC_{i,j}(t)$

      $j \leftarrow j+1$

    $S \leftarrow$ ChooseFromCandidateSplits(CS)

    $c' \leftarrow \{a | a \in prevC$ and $\nexists c'' \in S$ s.t. $a \in c''\}$

    $c'.t_{ls} \leftarrow prevC.t_{ls}$

    $C(t) \leftarrow C(t) \cup S \cup c'$

Algorithm 2: ChooseFromCandidateSplits.

input : CS: a set of candidate splits which may overlap with each other
output : C: a set of non-overlapping clusters

Sort CS by $CC_{i,j}.\tau$ in descending order and length of $CC_{i,j}.seq$ in ascending order
$C \leftarrow \phi$
while $CS \neq \phi$ **do**
  $s \leftarrow CS.first()$
  $l1 \leftarrow len(s.seq)$
  $included \leftarrow False$
  for c'**in** C **do**
    if $included = False$ **then**
      $l2 \leftarrow len(c'.seq)$
      if $l1 < l2$ **then**
        $d \leftarrow l2 - l1$
        if $c'.seq[d, d + l1) = c.seq(0, l1)$ **then**
          $included \leftarrow True$
      else if $l2 < l1$ **then**
        $d \leftarrow l1 - l2$
        if $c.seq[d, d + l2) = c'.seq(0, l2)$ **then**
          $included \leftarrow True$
  if not$inlcuded$ **then**
    $C \leftarrow C \cup s$

3.2 Toy Example Redux

We now revisit the toy example from Fig. 1 to show how it is handled by the extended summarization algorithm from Sect. 3.1.

As before, the algorithm starts with all the agents grouped into one cluster at time step 0, i.e., $C(0) = \{C_1(0)\}$ and we calculate the distribution over final outcomes, which comes out to be [0.5, 0.5], since exactly half the agents end with reward 0 and 1, respectively, as we have seen before. At time step 1, two candidate clusters are created, corresponding to the agents who do actions A and B. These are denoted as $\text{CC}_{1,1}(1)$ and $\text{CC}_{1,2}(1)$, respectively, where $\text{CC}_{1,1}(1).\text{seq} = A$ and $\text{CC}_{1,2}(1).\text{seq} = B$. However, the distribution over final outcomes for each of these candidate clusters is [0.5, 0.5], since within each of these clusters we find that, once again, exactly half the agents end up with reward 0 or 1. Since the final distribution is unchanged, we discard these candidate clusters.

At time step 2, since we are now tracking state sequences, we now create four candidate clusters, $\text{CC}_{1,1}(2)$ with $\text{CC}_{1,1}(2).seq = AA$, $\text{CC}_{1,2}(2)$ with $\text{CC}_{1,2}(2).seq = AB$ and so on, as illustrated in Fig. 2. The distribution over final reward for each of these clusters is evaluated and found to be significantly different from the distribution associated with $C_1(0)$. The distribution is [0, 1] for candidate clusters $\text{CC}_{1,1}(2)$ and $\text{CC}_{1,4}(2)$, since all the agents in these clusters get a final reward of 1, and it is [1, 0] for candidate clusters $\text{CC}_{1,2}(2)$ and $\text{CC}_{1,3}(2)$, since all the agents in these clusters get a final reward of 0. Thus, all of these candidate clusters are accepted as causal states as shown in Fig. 2 ($C_1(2)$ through $C_4(2)$).

Note that there are no causal states at time step 1. This is because the action taken at time step 0 actually has no effect on the probability of getting a reward at time step 2. However, the algorithm correctly identifies the four possible causal paths through the simulation, where the earlier simulation summarization algorithm fails.

3.3 Ranking Behaviors

We now present our approach for ranking behaviors for generating behavioral recommendations. The algorithm for contextual ranking of behaviors is as shown in Algorithm 3. It takes causal tree T (tree of causally-relevant state sequences generated by Algorithm 1 which provides the context required for ranking behavior) and a query q as input. Each cluster (or node) in

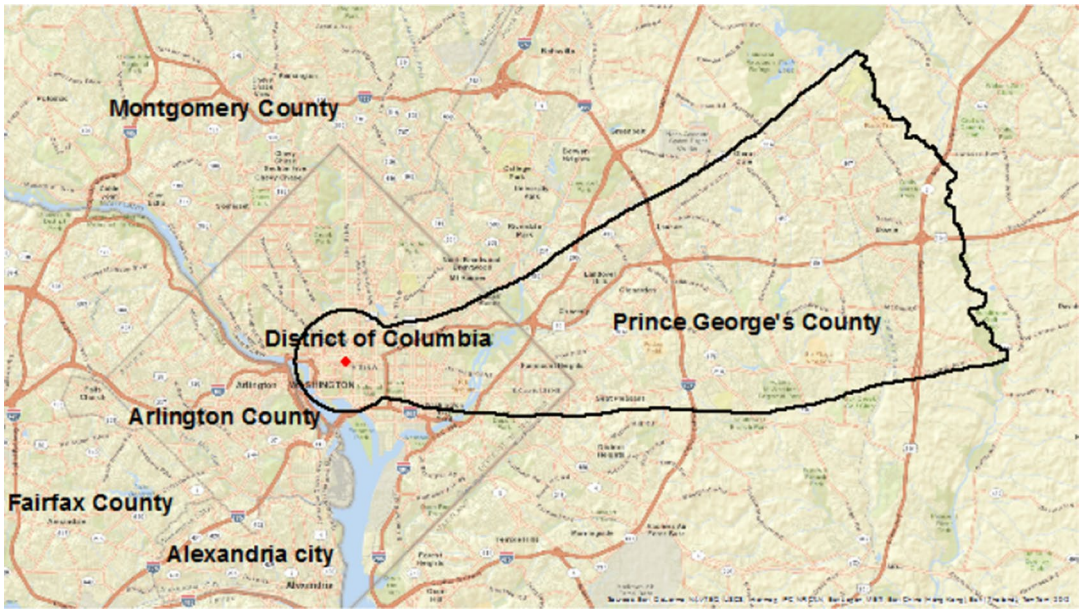

Figure 3: The detailed study area (DSA).

the causal tree contains information about the state sequence associated with it, the time stamp when it was split from the previous cluster, and its score (the expected change in the final outcome that it (or associated state sequence) lead to). Query q specifies time step t when we want to rank behaviors and optionally partial context (values for some of the state variables), for example, in a nuclear disaster simulation, the query may ask to rank behaviors after 1 h for people who are injured and are within 1 mile of the blast area. Let *qCls* denote the set of clusters matching the given query.

The idea behind behavior recommendation is simply to rank the clusters (causal states) in qCls by their scores. The corresponding seqs are the recommended behavioral partial ordering. We explain this in a little more detail below to provide some bookkeeping details.

The complete context for each of the clusters $c \in$ qCls is the path from the root of the tree T to c. However, some of the clusters in qCls may have some common ancestors and the context associated with these common ancestors does not really differentiate these clusters. Hence, to get the context that differentiates these clusters we only need to look back up to their least common ancestor (lca) of all the clusters in qCls.

The only clusters that share the same context are siblings and hence they need to be ranked by their scores in descending order. The common context for siblings is the path from lca to their parent. Let length of the context for cluster c be defined as sum over length of state sequence associated with each node n in the path from lca to its parent. So higher length gives more detailed context information. For clusters which are not siblings, they may have different length of context and it may happen that the context for one is subset of the other. So while ranking non-siblings, ideally we want to rank clusters with more detailed context first. So let *qClsSorted* denote the set of clusters qCls sorted in descending order by lengths of their contexts.

The while loop in Algorithm 3 processes clusters in this sorted order. It first extracts the cluster c with the most detailed context information (ties are broken arbitrarily). Let p be the parent of c. The context x for c is the path from lca to p and hence state sequence information associated with each node in this path is added to the context x. Cluster c share this context x with all of its siblings sbl and hence they are ranked by their score in descending order and this ranking is added to the context x. Since clusters associated with all siblings are already ranked, they are removed from the *qClsSorted* and context x is added to the list of contextual rankings CL.

Algorithm 3: Algorithm for ranking behaviors.

input : Causal tree T where each node is labeled with causal state, iteration when this causal state was split, and score (expected change in the expected outcome that the associated causal state lead to)
Query q

output : list CL of contexts and associated ranking for context matching query in descending order of the length of context

$qCls \leftarrow \{c | c \text{ matches } q\}$
$lca \leftarrow$leastCommonAncestor$(qCls)$
$qClsSorted \leftarrow$ sort $qCls$ by length of their context in descending order, where length of the context of $cls = \sum_{n \in \text{shortestPath}(lca, parent(cls))}$ length of causal state of n
$CL \leftarrow \phi$
while *$qClsSorted$ is not empty* **do**
  $c \leftarrow qClsSorted.first()$
  $p \leftarrow c.parent$
  $path \leftarrow$shortestPath(lca, p)
  $x \leftarrow \phi$
  for n ***in*** $path$ **do**
    $x \leftarrow x \cup n$
  $sbl \leftarrow p.children()$
  $sblR \leftarrow$ sort sbl in descending order by their scores
  $x.ranking \leftarrow sblR$
  $CL \leftarrow CL + x$
  for n ***in*** sbl **do**
    $qClsSorted.remove(n)$
return CL

3.4 Behavior Recommendation in the Toy Example

Returning to our toy example, the tree T is given by Fig. 2. Suppose the query $q :=$'$t = 2$', i.e., it asks for a ranking of behaviors at time $t = 2$. There are four nodes in Fig. 2 matching $t = 2$: qCls $= \{C_1(2), C_2(2), C_3(2), C_4(2)\}$. The least common ancestor of these four nodes is the initial node, $C_1(0)$. The length of the context for each of the four nodes in qCls is 2, so we can process them in any order. The context x for each of these nodes is the path from the lca to their parent (which is also $C_1(0)$). Thus $x = \varepsilon$ (the empty string). Ranking the four nodes by their score gives a partial order: $[\{C_1(2), C_4(2)\}, \{C_2(2), C_3(2)\}]$. The corresponding behavioral recommendation is $[\{AA, BB\}, \{AB, BA\}]$, i.e., that starting from the beginning of the simulation, it is better to do AA or BB than to do AB or BA.

Note that if the query were $q :=$'$t = 1$', i.e., to rank behaviors at time $t = 1$, the output of the algorithm would be to express no preference over behaviors (i.e., the output would be the empty set), since there are no causal states at $t = 1$. This is the expected output also, since both behaviors are equally good at this time step (they are equally likely to lead to a reward of 1).

4 Large-Scale Disaster Simulation

Next, we apply our contextual behavioral ranking algorithm to a very complex multiagent simulation of the aftermath of a disaster in a major urban area, developed by Barrett et al.[12]. We briefly describe the simulation below before discussing our experiments with ranking behaviors.

4.1 Scenario

The scenario is a hypothetical detonation of a low yield (10 kT) improvised nuclear device at ground level on a weekday morning in Washington DC at the corner of 16th and K street. The blast causes thermal fluence as well as radioactive fallout cloud which spreads mostly eastward and east-by-north-eastward. The simulation focuses on a region called the detailed study area (DSA) as shown in Fig. 3. It is the area under the largest thermal effect polygon (circle) and the area under the widest boundary of the fallout contours within DC region county boundaries (which includes the

Table 1: Simulation variables used in the experiments below.

Variable name	Values	Variable full name
HS	0: dead	Agent health status
	1-4: severe to moderate injury	
	5-7: mild injury to full health	
B	1: household reconstitution	Agent behavior
	2: evacuation	
	3: shelter-seeking	
	4: healthcare-seeking	
	5: worry	
	6: aid & assist	
EBR	0: no	Emergency broadcast received?
	1: yes	
E	0: low	Cumulative radiation exposure
	1: medium	
	2: high	
	3: very high	
TR	0: no	Treatment received?
	1: yes	
D	0: within 0.6 miles	Distance from ground zero
	1: between 0.6 and 1 mile	
	2: more than 1 mile	

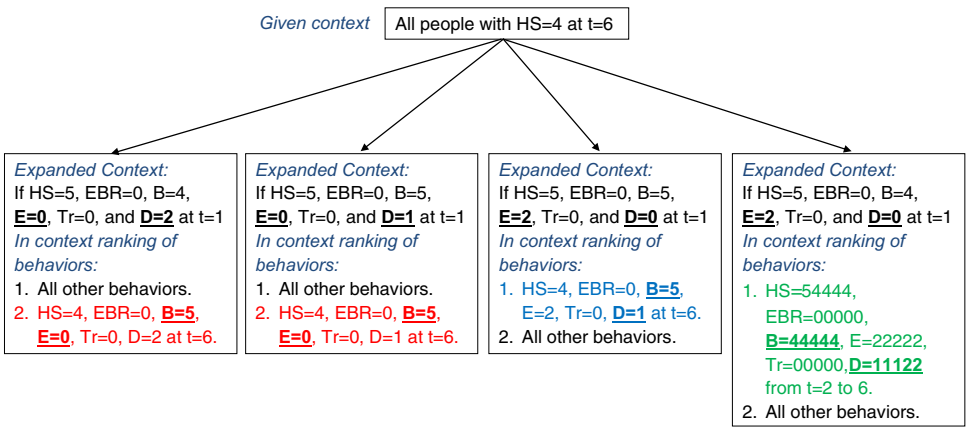

Figure 4: Results for behavioral ranking for query 1. Please refer to Table 1 for state variable abbreviations and values. The ranking shown are from best to worst.

District of Columbia and surrounding counties from neighboring states Virginia and Maryland).

In addition to the human casualties, the blast also causes significant damage to the infrastructure including roads, buildings, cellphone communication towers, the transportation network, and the power system. The simulation uses detailed data about these infrastructures to model blocked roads and injuries due rubble and debris, amount of radiation protection due to building construction material, reduction in radiation protection due to building damage, altered movements due to road damage and traffic jams, and cellphone call and text message capacity.

4.2 Agent Design and Behavior

Initial health and behavior of an agent depends upon its demographics as well as its location in the immediate aftermath of the disaster. For

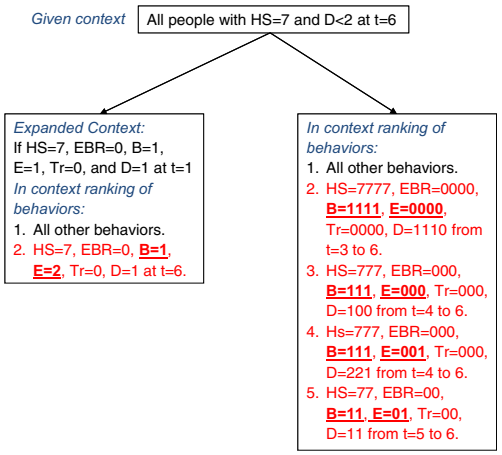

Figure 5: Results for behavioral ranking for query 2. Please refer to Table 1 for state variable abbreviations and values. The ranking shown are from best to worst.

example, agents close to the blast area likely to have low health as compared to the agents away from the blast area. Similarly, if an agent knows about one of its family members being injured, it may try to go towards that family member's location. Initial location and demographics information are obtained from a synthetic population, which is a high fidelity, individual-based representation of the population of a region along with their demographic (e.g, age, income, family structure) and daily activity-related information (e.g. type of activity, location, start time). A detailed description about creating synthetic populations can be found in[13].

Apart from the demographics and location, agents are defined by a number of other variables like health (modeled on a 0–7 range where 0 is dead and 7 corresponds to full health), behavior (described in the next paragaph), whether the agent is out of the affected area, whether the agent is the group leader, whether the agent has received an emergency broadcast (EBR), whether the agent is at a healthcare location, whether the agent has received treatment, time of the last call, if the last call was successful, the agent's exposure to radiation, agent's distance from ground zero, etc.

Each agent also keeps track of knowledge about family members' health states which is updated whenever it makes a successful call to a family member or meets him/her in person. Follow-the-leader behavior is also modeled, i.e., once

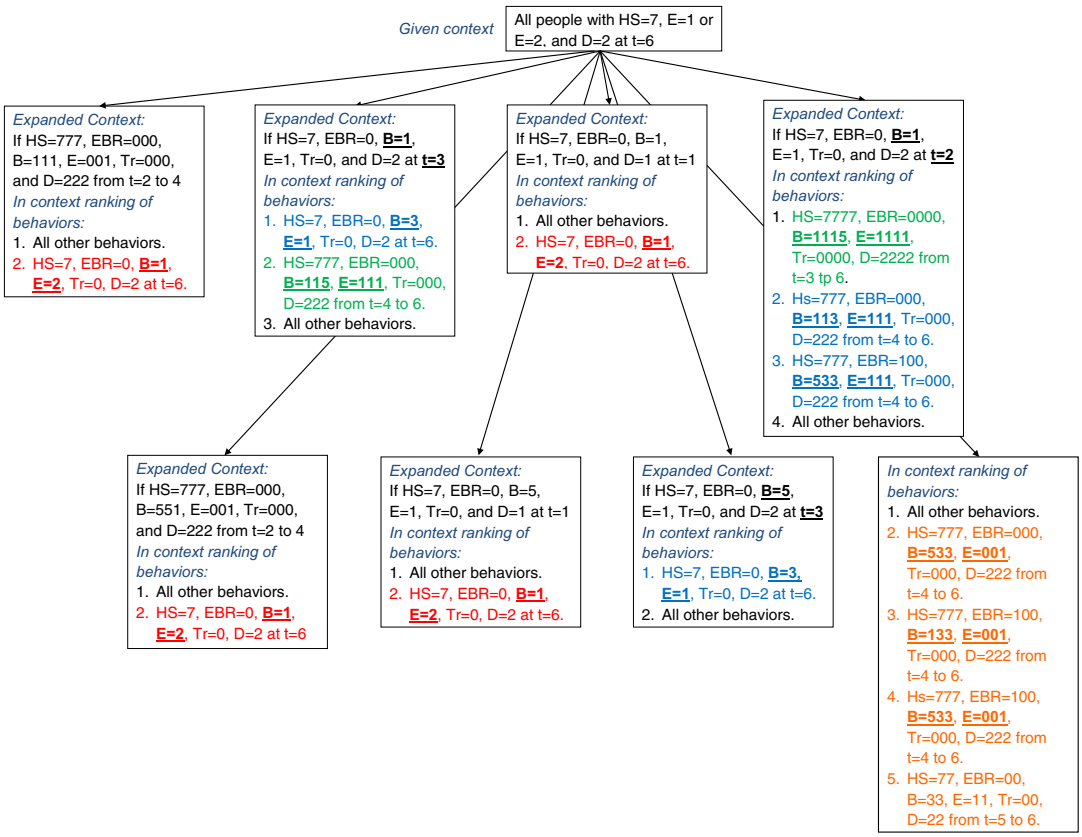

Figure 6: Results for behavioral ranking for query 3. Please refer to Table 1 for state variable abbreviations and values. The ranking shown are from best to worst.

family members encounter each other, they move together from there on. One of them becomes the leader and others follow him. This kind of behavior is well-documented in emergency situations. Also when a person is rescued by someone he travels with that person until he reaches a hospital or meets his family members.

Agent behavior is conceptually based on the formalism of decentralized semi-Markov decision process (Dec-SMDP) with communication[14] using the framework of options[15]. In this framework, high level behaviors are modeled as options, which are policies with initiation and termination conditions. These high level options correspond to low level action plans. Six behavioral options are modeled: household reconstitution, evacuation, shelter-seeking, healthcare-seeking, worry, and aid and assist. Low level actions are: call, text or move. Whom to call or text and where to move depends upon the current behavior option, e.g., in household reconstitution option, a person tries to move towards a family member and/or call family members while in healthcare-seeking option, a person tries call 911 or move towards a hospital. These mappings from high level behaviors to low level actions model dependancies of behaviors with infrastructural models. Detailed descriptions of the behavior model can be found in[3,16].

4.3 Experiments

Next, we present results of applying our contextual ranking algorithm to the disaster simulation. Agents and locations that they visit are represented by about 40 variables which could take binary, categorical or continuous values, leading to a very large state space. Here, we focus on six of these variables as context as described in Table 1. We use a threshold on effect size, $\delta = 0.5$, for extracting causally-relevant states and to create a causal tree. Finally, we apply our ranking algorithm to rank behaviors in multiple contexts and show that it can identify meaningful rankings as well as any additional context that may affect the rankings as shown in the queries below.

Query 1 Rank behaviors after 1 h (time step 6) for all people who are moderately injured (i.e., with health state 4).

As shown in Fig. 4, the results show that for people who started close to ground zero (within 0.6 miles of the blast area) and with high exposure to radiation early on, if they have managed to get farther from ground zero (farther than 1 mile) then healthcare-seeking is the best behavior but if they are still within 1 mile of ground zero, then worry is the best behavior. In the worry behavior people run outside looking for information, call 911 or go to the nearest healthcare location. If the 911 call is successful, some of them may be transported to the nearest healthcare location. Since the radiation exposure is already high, reaching healthcare location and receiving treatment is the best thing for them to do.

On the other hand, for people who started farther than 0.6 miles from ground zero and have low radiation exposure after 1 h, all behaviors are better than worrying. This is because the worry behavior may make them run outside looking for information or move towards nearest healthcare location which would then expose them to more radiation.

Query 2 Rank behaviors after 1 h (time step 6) for all people who are close to ground zero (within 1 mile) but in full health (health state 7).

The results (Fig. 5) show that for people who are in full health and close to ground zero, household reconstitution is ranked lower than all other behaviors, irrespective of their radiation exposure. This is because in household reconstitution behavior, people call or move towards their family members. This may lead them to be exposed to more radiation, which may still be quite high early on.

Query 3 Rank behaviors after 1 h (time step 6) for all people who are far from ground zero (i.e., further than 1 mile from ground zero) and in full health (health state 7) though with medium or high radiation exposure (E = 1 or 2).

As shown in Fig. 6, the results show that for people who are far from ground zero and in full health, though with high radiation exposure, household reconstitution behavior is ranked the worst among all behaviors as it makes them go outside and exposed to more radiation. For people with medium exposure and who were worrying early on, shelter-seeking (which also includes sheltering in place) is better than all other behaviors. For people with medium exposure and who were engaged in household reconstitution behavior previously, shelter-seeking and worry behaviors are ranked better than all other behaviors. This is because these behaviors shield them from radiation exposure or lead to a healthcare location.

Results also show that for people who were engaged in shelter-seeking behavior (which includes sheltering in place) in the previous time step and whose radiation exposure has increased from low to medium in the current time step,

all other behaviors are better than sheltering in place. Here, increase in radiation exposure while engaged in shelter-seeking behavior suggests that these people are either sheltering in place at a location that does not provide enough protection against radiation (due to the damage to the building or type of construction material) or they are still searching for shelter and hence exposed to radiation.

As seen from above query results, our algorithm finds meaningful ranking of behaviors as well as any additional context that may affect the behavioral ranking. Here, we have applied the contextualized behavior ranking algorithm to the output of one simulation run. However, the algorithm can be easily applied to the output of multiple simulation runs. Parikh et al.[11] compared the results of simulation summarization from their previous paper[4] for single and multiple simulation runs and the results were quite similar. We believe similar results will hold for behavior ranking as it uses the output of the simulation summarization algorithm (i.e., the causal tree) for ranking.

5 Conclusion

Large-scale complex data-driven agent-based simulations are becoming increasingly common for studying complex social phenomena such as disasters. A clear benefit of this approach is that the simulations can be very realistic, incorporating many sources of data, including procedural data. Agent-based simulations also yield a natural and human-comprehensible perspective of the complex dynamic social interactions. However, their very complexity also makes them harder to understand, since they produce very large, richly-structured data sets as output. Managing this double-edged challenge in modeling requires careful design and analysis[17]. One approach to addressing this issue is to develop new methods for extracting useful information from such simulation-generated data sets. The earlier work of Parikh et al.[4] and our current work are both in this spirit.

In this work, we described an approach to generating contextualized behavioral recommendations from a complex agent-based simulation. This is really a question from the perspective of an end-user such as an emergency planner. Faced with a complex simulation, which purports to model a hypothetical disaster, they may well wish to query the results both as a means of developing response and preparedness plans, and also as a means of developing trust in the model itself. In this sense, the behavioral recommendations also serve as a kind of explanation that shows the user the situations in which the same behavior can have different outcomes.

There are many possibilities for future work along these lines. The stochasticity that is built into the simulation will naturally result in some variability from one run to another, so an open question is, are these behavioral recommendations robust to this variability? How can we ensure that? There may also be variability of outcomes from agent to agent even though they have the same context (i.e., agents that end up in the same cluster). Some of this variability may be due to the selection of a subset of variables for extracting causally-relevant state sequences which may be necessary due to a large state space of a simulation. Could we use this variability to refine the set of variables selected? Similarly, robustness to parameter choices, such as the effect size parameter, also need to be studied.

Another useful direction of research is to work with actual response planners and see which aspects of the disaster they find the most useful to focus on and how behavioral recommendations shape their planning efforts. This requires developing appropriate user interfaces that would allow exploration of the simulation results, the corresponding causal states, as well as the behavioral recommendations in an intuitive and useful way.

Given the proliferation of agent-based simulation applications, we believe that questions such as the one addressed here will become increasingly important in ensuring that simulations are built with recognition of their use by non-computer scientists.

Publisher's Note

Acknowledgements

We thank our colleagues in the Network Systems Science and Advanced Computing Division at the Biocomplexity Institute and Initiative at the University of Virginia for many interesting discussions.

Declarations

Conflicts of Interest

On behalf of all authors, the corresponding author states that there is no conflict of interest.

Received: 25 June 2021 Accepted: 7 July 2021
Published online: 19 August 2021

References

1. Atun F (2014) Understanding effects of complexity in cities during disasters. In: Walloth C, Gurr JM, Schmidt JA (eds) Understanding complex urban systems: multidisciplinary approaches to modeling. Springer, Switzerland, pp 51–66
2. Chandan S, Saha S, Barrett C, Eubank S, Marathe A, Marathe M, Swarup S, Vullikanti AK (2013) Modeling the interactions between emergency communications and behavior in the aftermath of a disaster. In: The international conference on social computing, behavioral-cultural modeling, and prediction (SBP). Washington DC, USA
3. Parikh N, Swarup S, Stretz PE, Rivers CM, Lewis BL, Marathe MV, Eubank SG, Barrett CL, Lum K, Chungbaek Y (2013) Modeling human behavior in the aftermath of a hypothetical improvised nuclear detonation. In: Proceedings of the international conference on autonomous agents and multiagent systems (AAMAS). Saint Paul, MN, USA
4. Parikh N, Marathe MV, Swarup S (2016) Summarizing simulation results using causally-relevant states. In: Osman N, Sierra C (eds) Autonomous agents and multiagent systems: AAMAS 2016 workshops, visionary papers, no. 10003 in LNAI, pp 88–103. Springer
5. Crutchfield JP, Young K (1989) Inferring statistical complexity. Phys Rev Lett 63(2):105–108
6. Shalizi CR, Crutchfield JP (2001) Computational mechanics: pattern and prediction, structure and simplicity. J Stat Phys 104(3/4):817–879
7. Crutchfield JP, Ellison CJ, Mahoney JR (2009) Time's barbed arrow: irreversibility, crypticity, and stored information. Phys Rev Lett 103(9):094101
8. Ellison CJ, Mahoney JR, Crutchfield JP (2009) Prediction, retrodiction, and the amount of information stored in the present. J Stat Phys 136(6):1005–1034
9. Shalizi CR, Shalizi KL (2004) Blind construction of optimal nonlinear recursive predictors for discrete sequences. In: Chickering M, Halpern J (eds) Proceedings of the twentieth conference on uncertainty in artificial intelligence, pp 504–511. Banff, Canada
10. Rabiner L, Juang B (1986) An introduction to hidden markov models. IEEE ASSP Magaz 3(1):4–16. https://doi.org/10.1109/MASSP.1986.1165342
11. Parikh N (2017) Behavior modeling and analytics for urban computing: a synthetic information-based approach. Ph.D. thesis, Virginia Polytechnic Institute and State University
12. Barrett C, Bisset K, Chandan S, Chen J, Chungbaek Y, Eubank S, Evrenosoğlu Y, Lewis B, Lum K, Marathe A, Marathe M, Mortveit H, Parikh N, Phadke A, Reed J, Rivers C, Saha S, Stretz P, Swarup S, Thorp J, Vullikanti A, Xie D (2013) Planning and response in the aftermath of a large crisis: an agent-based informatics framework. In: Pasupathy R, Kim SH, Tolk A, Hill R, Kuhl ME (eds) Proceedings of the 2013 winter simulation conference
13. Barrett C, Beckman R, Berkbigler K, Bisset K, Bush B, Campbell K, Eubank S, Henson K, Hurford J, Kubicek D, Marathe M, Romero P, Smith J, Smith L, Speckman P, Stretz P, Thayer G, Eeckhout E, Williams MD (2001) TRANSIMS: transportation analysis simulation system. Tech. Rep. LA-UR-00-1725, Los Alamos National Laboratory Unclassified Report
14. Goldman CV, Zilberstein S (2008) Communication-based decomposition mechanisms for decentralized MDPs. J Artif Int Res 32(1):169–202
15. Sutton R, Precup D, Singh S (1999) Between MDPs and semi-MDPs: a framework for temporal abstraction in reinforcement learning. Artif Intell 112(1–2):181–211
16. Parikh N, Hayatnagarkar HG, Beckman RJ, Marathe MV, Swarup S (2016) A comparison of multiple behavior models in a simulation of the aftermath of an improvised nuclear detonation. Autonomous Agents Multi-agent Syst 30(6):1148–1174
17. Swarup S (2019) Adequacy: what makes a simulation good enough? In: Proceedings of the spring simulation conference (SpringSim). Tucson, AZ

Nidhi Parikh is a Scientist in the Information Systems and Modeling Group at Los Alamos National Laboratory. Her research interests are in agent-based modeling, data fusion, machine learning, simulation analytics, and network science.

Madhav V. Marathe is a Distinguished Professor in Biocomplexity, the division director of the Networks, Simulation Science and Advanced Computing Division at the Biocomplexity Institute and Initiative, and a Professor in the Department of Computer Science at the University of Virginia (UVA). His research interests are in network science, computational epidemiology, AI, foundations of computing, socially coupled system science and high performance computing. Before joining UVA, he held positions at Virginia Tech and the Los Alamos National Laboratory. He is a Fellow of the IEEE, ACM, SIAM and AAAS.

Samarth Swarup is a Research Associate Professor in the Networks, Simulation Science and Advanced Computing Division at the Biocomplexity Institute and Initiative at the University of Virginia (UVA). His research interests are in AI and machine learning, multi-agent systems, computational social science, agent-based modeling and simulation, and simulation analytics.