Predicting ABM Results with Covering Arrays and Random Forests*

Megan Olsen[1], M S Raunak[2] and D. Richard Kuhn[2]

[1] Loyola University MD, Baltimore MD 21210, USA mmolsen@loyola.edu
[2] NIST, Gaithersburg, MD 20899, USA {raunak, kuhn}@nist.gov

Abstract. Simulation is a useful and effective way to analyze and study complex, real-world systems. It allows researchers, practitioners, and decision makers to make sense of the inner working of a system that involves many factors often resulting in some sort of emergent behavior. The number of parameter value combinations grows exponentially and it quickly becomes infeasible to test them all or even to explore a suitable subset. How does one then efficiently identify the parameter value combinations that matter for a particular simulation study? In addition, is it possible to train a machine learning model to predict the outcome of an agent-based model (ABM) with a systematically chosen small subset of parameter value combinations? We explore these questions in this paper. We propose utilizing covering arrays to create t-way (t = 2, 3, etc.) combinations of parameter values to significantly reduce an ABM's parameter value exploration space. In our prior work we showed that covering arrays are useful for systematically decreasing an ABM's parameter space. We now build on that work by applying it to Wilensky's Heat Bugs model and training a random forest machine learning model to predict simulation results by using the covering arrays to select our training and test data. Our results show that a 2-way covering array provides sufficient training data to train our random forest to predict three different simulation outcomes. Our process of using covering arrays to decrease parameter space to then predict ABM results using machine learning is successful.

Keywords: agent-based modeling · machine learning · calibration

1 Introduction

Modeling and simulation is a useful and effective way to study complex, real world systems. Through modeling we can examine the inner working of an intricate system, and ask questions about how a change to one aspect of the system affects other aspects or the system as a whole. Scenarios such as the spread of a pandemic, the operations of a self-driving car, or the flow of patients in an emergency department can be studied with simulation models. Simulations allow us

* Supported by NSF MRI grant 1626262.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-35995-8_17

to study these complex scenarios without building the physical system, which can be costly, dangerous, or simply infeasible.

Agent-based modeling (ABM) is a popular modeling technique for studying these types of systems and phenomena through simulation. Agents are individual autonomous entities that make decisions about their actions and interactions within the environment. ABM can be a bottom-up approach, where the global behavior of a system emerges out of the individual decisions and actions of agents. A change to parameter values can affect model outcomes, sometimes in unexpected ways, as parameters affect the algorithms for how the agents make decisions, and how the system is updated over time or in response to those agent decisions. Thus it is important to understand what parameter values to use and explore while simulating and studying an agent based model. However, a typical ABM will include many parameters, each with a potentially very large set of possible values. The number of parameter value combinations grows exponentially with the addition of each parameter, quickly becoming infeasible to test all parameter combinations, or even to explore a suitable subset of them. This problem is not unique to ABM, and is present in any reasonably large system. How does one then efficiently identify the parameter value combinations that matter for a particular simulation study, or are likely to influence the output the most? These questions are crucial for developing a well-calibrated, valid simulation model.

There are many approaches for calibrating, testing, and validating ABMs, most of which rely on choosing a suitable subset of parameter values to examine. In this paper, we explore using covering arrays from the software testing literature to systematically cover an effective subset of the parameter space to choose the parameter value to test. We have previously shown the usefulness of covering arrays for reducing the parameter value space of simulation models [16,18]. Building on that work, after using covering arrays to thoughtfully reduce the parameter space, we ask the question: can we use machine learning (ML) to train a model on data from running an agent-based model on a small subset of its parameter space, and then predict the results for a larger set of new parameter combinations? This approach could significantly decrease the number of simulation experiments required to fully understand the impact of parameter changes on a model. We also analyze the relationships between parameter value combinations and simulation outputs. As far as we can tell, we are the first to explore this approach on simulation models. We test the approach by training random forest models on data from Wilensky's Heatbugs Netlogo agent based model [20], and our results show that it is feasible to make these predictions.

2 Related Work

One of the objectives of modeling and simulation is to understand the behavior of large, complex systems under different environmental conditions. Through the selection of parameter values, it can also allow us to perform predictive analysis of a system. For the predictions to be accurate and useful, the model has to reliably represent the real system and robust against parameter value changes such

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-35995-8_17

that it isn't too sensitive, resulting in uncertain output swings. Thus the rate and magnitude of changes in model output when the input parameter values change is an important aspect of developing useful simulation models. Researchers and practitioners perform uncertainty or risk analysis, i.e., estimation of output variance, parameter calibration, and sensitivity analysis of simulation models to gain better understanding of model accuracy and robustness[3,5]. These analyses require running the model under many parameter value combinations.

Model calibration adjusts the initial model to a reference system by tuning parameter values. Hoffmann showed that for large, complex models as well as for model federations with interdependent parameters, model calibration is an NP-complete problem [7], and consequently is too costly in many cases. Researchers are thus left to find more pragmatic approaches of exploring a small subset of parameter values. Sensitivity analysis towards validation and optimization of the model is another well studied area [9]. In his 2004 article Kleijnen commented, "Few statisticians have studied random simulations. And only some simulation analysts have focused on strategic issues, namely which scenarios to simulate, and how to analyze the resulting I/O data." [8] Kleijnen used his argument to motivate the need for using a Design of Experiment (DOE) study of the metamodels, using mathematical equations as a metamodel of computer simulation models. Multiple research works have studied DOE for deterministic metamodels such as the polynominal regression metamodel, Krigin models, and more [1,8]. None of these research studies, however, looked into systematically reducing the parameter value space to tackle the challenge of dealing with an exponentially large search space, especially on ABMs.

Covering arrays also have roots in DOE, and represent an approach to effectively cover the overall parameter value combination space [11]. These arrays include all t-way combinations of parameter values (typically for small t, i.e., 2 to 6), and are effective in designing software testing solutions [13,11]. In our earlier work, we showed that covering arrays can also be useful in focusing the parameter value space and in choosing a useful subset of values to test an ABM [16]. In this study we build on this approach by applying it on a different ABM, and investigating the effectiveness of using a Random Forest ML model to predict simulation outputs on unseen combinations of parameter values. In a similar vein, Lanus et. al. used covering array based approaches to describe how two different data sets used in training and testing of an ML model differ [15]. Their study used combinatorial set difference metrics to identify which features are most influential as a determining factor of how successful an ML model is going to be on a new data set. The objective and application of their study, however, was different than our work presented here.

We are unaware of prior work that attempts to predict the outcome of an ABM after training a machine learning model on a subset of that model's possible parameter combinations. Machine learning is instead used for processes such as to de-feature (feature simplification, removal, etc.) CAD models for simulation studies [6], or as part of building the model itself [21]. There are also numerous approaches for trying to find optimal parameter combinations for a model, such

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-35995-8_17

as genetic algorithms [17,4,19] and Robust Parameter Estimation [10]. However, this work is not focused on determining optimal combinations, but instead in making it easier to understand the type of behavior caused by any particular set of parameter values, and how a specific type of change to parameters would change model output, without needing to run the ABM on those values.

3 Approach

Our goal is to explore the feasibility and usefulness of using a combination of covering arrays and machine learning models for predicting results of an agent-based simulation model (ABM) within the vast parameter value combination space. The challenge is to select parameter values that are representative of the model's overall behavior, so that we can train the machine learning (ML) model to be able to correctly predict behavior on previously untested areas of the parameter space. We have chosen Wilensky's Heatbugs ABM in NetLogo [20] for our study. It is a simple model, amenable to quick data generation, with a limited number of outputs to predict, and with emergent behavior. This model therefore allows exploration of this new approach.

We utilize covering arrays to reduce the parameter value space systematically, run the model for each parameter set in the 2-way and 3-way covering arrays, train a random forest model on the 2-way data (33, 351 parameter combinations), and test its ability to predict the outcome of the simulation on the significantly larger 3-way data that was not seen during the training of the model (3, 971, 955 parameter combinations). This section provides details on each step.

3.1 Heatbugs Model

In Wilensky's Heatbugs ABM [20], agents move in a 2D grid in an attempt to find a location with their ideal temperature. Only one agent can exist in each location, and they have a random chance of moving at any given time step. Agents give off heat to the local environment, which is then diffused to neighboring squares. The heat also dissipates from each square at a given rate. The result is a level of unhappiness for each agent ($|ideal\_temp - current\_temp|$), with the goal of low unhappiness for all agents. The behavior of the agents and their environment are defined by the eight parameters in Table 1.

3.2 Choosing Parameters via Covering Arrays

As seen in Table 1, there are 5.6386e15 valid combinations for this model. If we hold the number of agents steady, we reduce this number to approximately 11 trillion parameter value combinations (1.1277e13). If we needed to test every combination this is the minimum number of times the model would need to run; however, it is infeasible to test more than a tiny fraction of these combinations.

An efficient approach for reducing the parameter space is to use covering arrays. A *t-way* covering array is a matrix of values that includes all *t-way*

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-35995-8_17

Parameter	Min Value	Max Value	Increment
Agent Number	1	500	1
Min Ideal Temp	0	200	1
Max Ideal Temp	0	200	1
Min Output Heat	0	100	1
Max Output Heat	0	100	1
Evaporation Rate	0.01	1	0.01
Diffusion Rate	0	1	0.1
Random Move Chance	0	100	1

Table 1: Heatbugs model parameters and their valid values. There are 2.28939e16 potential combinations. Given that each pair of min/max parameters must have the relationship min < max, there are 5.6386e15 valid combinations.

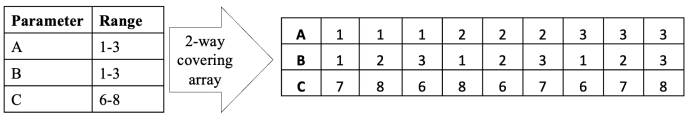


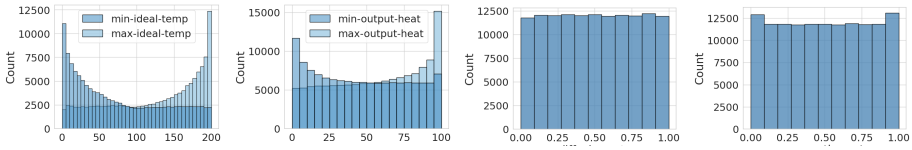
Fig. 1: Example of converting a simple 27 combination parameter space to a 2-way covering array of 9 combinations (each column is a combination).

combinations of parameter values. Suppose we have three parameters A, B, and C, and each of these parameters can take three values. Figure 1 shows the parameter values and a 2-way covering array constructed from those values. A 2-way covering array [11] as shown in the figure, includes every pair of parameter values at least once. As a larger example, if we instead have 10 binary parameters, there are $2^{10} = 1024$ possible value combinations one can use to run the model. However, all 3-way interactions of parameter values are included in a covering array of only 13 rows. Those 13 rows include every possible 3-way combination of parameter values, substantially reducing the search space. From the perspective of a simulation model study, that covering array allows systematic exploration of the parameter value space with only 13 simulation runs instead of 1024 runs.

We use the ACTS tool [2] to generate covering arrays of 2-way and 3-way combinations of parameter values to significantly reduce the parameter value exploration space while ensuring broad coverage of possible parameter interactions. When agent count is held steady for the Heatbugs model, there are 1.1277e13 valid parameter value combinations across the seven remaining parameter values. This number is reduced to 33,551 parameter value combinations where every possible 2-way interactions of the parameter values are present. Similarly, all 3-way interactions are captured by 3,972,000 value combinations.

Both 2-way and 3-way covering arrays reduce the parameter value space while maintaining good coverage of parameter interactions. We ensure that none of the 2-way rows are present in the 3-way data. Figure 2 shows the frequency of each parameter's values in the 2-way covering array. The diffusion and evaporation rates are evenly distributed across all potential values, while the pairs of ideal

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-35995-8_17



(a) ideal temp (b) output heat (c) diffusion rate (d) evaporation rate

Fig. 2: Histograms of the feature values as they exist in the 2-way covering array.

temperature and output heat parameters have higher rates on the lower and higher ends of their scales. This uneven application of values is expected due to the relationship between min and max for each pair, making the lower values of min and the higher values of max more likely to be compatible with values of the related variable.

3.3 Machine Learning

We learn random forest models to attempt to predict the outcome of the ABM. Random forests are a decision tree ensemble method. A decision tree is a supervised learning method that at each node branches based on the value of a particular feature, such as whether or not diffusion rate is less than 0.2. Each branch takes you closer to a prediction, and how many levels the tree needs will depend on the data. For a random forest, essentially multiple decision tree models are learned, and then the prediction that is most common among all learned trees is the prediction of the overall ensemble. This ensemble method is more likely to correctly predict outcomes than a single decision tree. Details of our implementation are in Section 4.

4 Experimental Setup

4.1 Data Gathering and Preparation

We run the simulation for each parameter combination in the 2-way and 3-way covering arrays four times, for 25,000 simulation steps. We run with four different random number seeds due to the stochasticity in the model, to make it more likely that the overall results produced are due to parameters instead of randomness. After each simulation ends we calculate the following metrics[3]:

- avg: the average unhappiness of the heat bugs
- $avgF$: the average unhappiness across the final 500 time steps
- $stdF$: standard deviation of unhappiness across the final 500 time steps
- $minF$ and $maxF$: minimum and maximum unhappiness of all agents across the final 500 time steps.

[3] The processed data used in this paper can be found at `https://data.nist.gov/`

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-35995-8_17

After all data are gathered, we prepare it for machine learning. Each set of four rows, one for each model run of the same parameters but different random number seeds, is combined into a single row of data and labeled based on whether that set of parameter values appears to achieve the outcome to be predicted. Each of our experiments prepare the data for a different type of prediction:

A) the model reaches low unhappiness with low variation of unhappiness across agents and time;
B) the model reaches a steady state; and
C) the average final unhappiness level of the agents.

In all experiments, each parameter combination appears once in the prepared data. We determine the thresholds for creating the class labels based on the 2-way (training) data, and then apply those thresholds to the 3-way (test) data. Each threshold creates a set of data with a different level of imbalance between the classes, so that we can test how imbalance affects predictive ability. In each experiment a threshold has been included that leads to balanced class labels on the 2-way data. In experiments A and B we predict a binary class, e.g. Class 1 represents meeting the criteria, and Class 0 represents not meeting that criteria. Experiment C is instead a multi-class classification problem.

Experiment A: Low Unhappiness and Low Variation: Experiment A attempts to predict if the model reaches low overall unhappiness with a low variation of unhappiness across agents during the final 500 time steps. In these experiments, the class of the parameter set is determined by $stdF < threshold$ for a particular percentage of the four simulations run for a particular parameter set. Class 1 label is applied to data below the threshold, as it represents low variation in standard deviation, which also correlates to relatively low overall unhappiness level for this ABM. We test our predictive ability on thresholds of $0.1, 0.15, 0.2, 0.5, 0.75, 1$. For each threshold we test two approaches for combining the four simulations of the same parameter combination: either 50% or 100% of the four simulation runs must meet the threshold criteria to be Class 1; otherwise, the parameter set is labeled as Class 0.

Experiment B: Steady State: Experiment B attempts to predict if the unhappiness level in the Heatbugs model reaches a steady state by the end of the simulation. We define steady state as $stdF/avgF < threshold$. The standard deviation on its own is insufficient as the overall averages vary widely, between 0 to 45,000, from run to run. The high average values are due to certain parameter combinations leading to overwhelming heat in the system well above the bugs' ideal temperature; low values occur when the situation keeps heat from building up too high. When the average is high, a relatively small standard deviation could still be in the hundreds; whereas when average is low, it would be single digits at most. Thus we found that the value of standard deviation can be misleading for this model. By dividing the standard deviation by the average, we can observe how much the unhappiness level is varying in terms of the overall average at that point in the simulation. We test our predictive ability on thresholds

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-35995-8_17

of $0.001, 0.002, 0.005, 0.0075, 0.01$. For each threshold we test three approaches for combining the four simulations of the same parameter combination: either 50%, 75%, or 100% of the four simulation runs must meet the threshold criteria for the parameter set to be labeled Class 1; otherwise, it is labeled Class 0.

Experiment C: Average Unhappiness: Experiment C attempts to predict the overall average unhappiness during the final 500 time steps, $avgF$. We create classes defined by quantiles of the $avgF$ column of the 2-way data:

1. 4 quantiles: < 30.91, < 74.08, < 125.19, $>= 125.195$
2. 6 quantiles: < 12.27, < 38.45, < 74.08, < 113.84, < 164.66, $>= 164.66$
3. 10 quantiles: < 12.27, < 24.24, < 38.45, < 55.60, < 74.08, < 92.76, < 113.84, < 137.98, < 164.66, $>= 164.66$

In this experiment we determine if it is possible to predict the general region of the average unhappiness for agents in the model. Although quantiles do not need to be used, it is a natural way to define the categories. To ensure a proper machine learning process is followed, we do not analyze quantiles of the 3-way data but apply the thresholds developed on the 2-way data to the 3-way data.

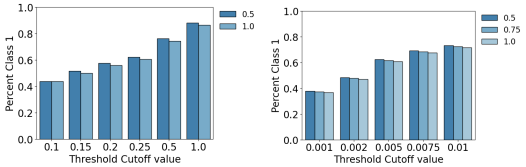
4.2 Machine Learning in All Experiments

We use the simulation results from the 2-way covering array data to generate training and validation data for each random forest model, then use the results from the 3-way covering array to test each model. This setup allows us to see if we can train a random forest model on a very small subset of the overall parameter space ($3.3551e4$ combinations), and then predict results for a significantly larger amount of new parameter combinations ($3.972e6$). In each case, the test data is processed in the same way as the equivalent training data. Each combination of thresholds and approaches to combining results of the same parameters results in a new set of data and a random forest model.

We use scikit-learn's implementation of random forests, and learn using a randomized grid search. A randomized grid search takes all of the various hyperparameter (e.g. parameters to the random forest) options and randomly chooses a subset of all possible hyperparameter combinations to search for the best model. We use 10 fold cross validation, with balanced accuracy as the scoring function on 100 hyperparameter combinations. Cross validation is the standard best practice to help reduce the chance of overfitting on the training data, with 10 folds being the generally agreed upon best choice. Balanced accuracy is used as our datasets are mostly imbalanced, which can affect how well the model learns.

We test the following hyperparameters: number of decision trees in ensemble (200, 288, 377, 466, 555, 644, 733, 822, 911, 1000); maximum number of features to consider at a branching (2, 7); maximum depth of learned trees (10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110); minimum samples required to branch (2, 5, 10); minimum samples required at a leaf node, e.g. at the point at which the decision tree is making a prediction, how many rows from the data should be represented

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-35995-8_17



(a) Experiment A (b) Experiment B

Fig. 3: The classes are generally unbalanced, with the most balanced data occurring when the cutoff threshold is 0.15 for Exp. A and 0.002 for Exp. B.

by the prior branch in this direction (1, 2, 4); Gini impurity to determine quality of split; and bootstrapping either on or off. The grid search tests 100 randomly chosen combinations of these hyperparameters. After completing the search, it determines the best hyperparameters and generates a final random forest model using those hyperparameters. The 3-way data is tested on that model.

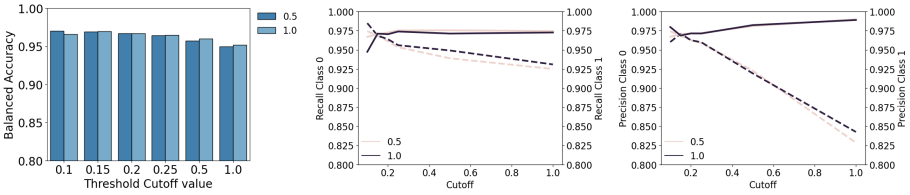
5 Results

We analyze *balanced accuracy*, *precision*, and *recall* as they are standard ML metrics that are particularly useful for unbalanced data. Unlike regular accuracy that only determines what percentage of predictions are correct, balanced accuracy is the sum of the number correct in each class divided by the number of classes. Balanced accuracy is thus more accurate when a large number of examples are from one class. As can be seen in Figure 3, our classes are imbalanced in most of our training data, which is to be expected. The recall and precision helps us to understand how that imbalance affects the predictions for each class. The recall tells us how likely we are to predict a given class if it actually is that class; so recall for Class 1 in Experiment B is how likely we are to predict that a simulation run with a specific set of parameter values will be steady when it actually is steady. Precision tells us how often a prediction is correct; so a high precision on Class 1 means that if we predict Class 1 then it is likely to actually be Class 1. As a reminder, in all scenarios, 2-way data trained the model; the results are from using that learned model to predict the 3-way results. High accuracy, precision, and recall implies that our overall process works for training a random forest to predict the reults of an agent-based model by only running a small percentage of potential parameter combinations.

5.1 Experiment A: Low Unhappiness and Low Variation

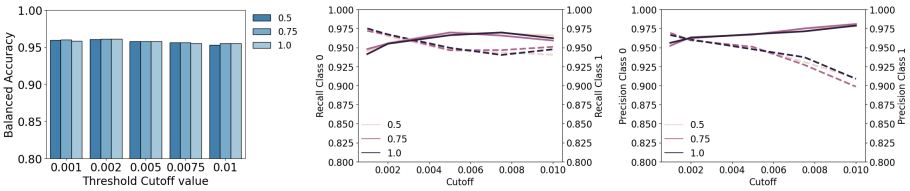
In Experiment A we predict whether the standard deviation over the last 500 time steps ($stdF$) crosses a given threshold. Class 1 represents situations where it is below the threshold, e.g. is steady and the overall unhappiness values are likely small. As can be seen in Figure 4, balanced accuracy varies between 95% and 97% on the 3-way data, with accuracy decreasing as we become more imbalanced

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-35995-8_17



(a) Balanced Accuracy (b) Recall (c) Precision

Fig. 4: Recall and Precision from Experiment A. The solid line is for Class 1 (e.g. below threshold), and the dotted line is Class 0 (e.g. at or above threshold). The colors denote what percentage of the four runs needed to be below the threshold for the parameter combination to be labeled as Class 1.



(a) Balanced Accuracy (b) Recall (c) Precision

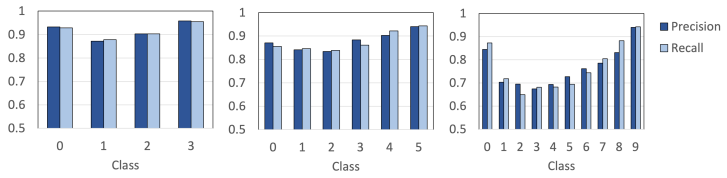
Fig. 5: Recall and Precision from Experiment B. The solid line is for Class 1 (e.g. below threshold), and the dotted line is Class 0 (e.g. at or above threshold). The colors denote what percentage of the four runs needed to be below the threshold for the parameter combination to be labeled as Class 1.

with a higher percentage of the data in Class 1. Our recall for both classes is actually quite good in all versions of the problem, meaning that we are getting most examples correct. The precision for Class 0 is closer to 80% for the more imbalanced situations, meaning that when we predict Class 0 we are less likely to be correct than when we predict Class 1. Overall, many values for standard deviation cutoff are predictable, even when the training data has significantly fewer examples in one class. It is feasible to predict if standard deviation will be low in the simulation by training on a small subset of the parameter space, although best results occur in a relatively balanced training set. How the four results from each parameter set are combined does not affect the result.

5.2 Experiment B: Steady Unhappiness

In this experiment we predict how steady the unhappiness level is across agents during the last 500 time steps. We defined *steadiness* as standard deviation divided by average, to take into account that some average unhappiness levels are orders of magnitude higher than others. As seen in Figure 5, the balanced accuracy score is high across all situations tested, always at least 95%. The precision

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-35995-8_17



(a) Four Quantiles (b) Six Quantiles (c) Ten Quantiles

Fig. 6: Recall and Precision from Experiment C for each class in each experiment.

and recall is always at least 90% as well, which is better than in Experiment A. We are very likely to get each class correct, although in the more imbalanced data we again see a dip in precision for Class 0 meaning that predictions for Class 0 are less trustworthy than predictions for Class 1. However, even for imbalanced data our models' predictions are generally accurate. Overall we seem able to predict steadiness of unhappiness level, with minimal impact by how steadiness is defined or how the four runs from each parameter set are combined.

5.3 Experiment C: Average Unhappiness

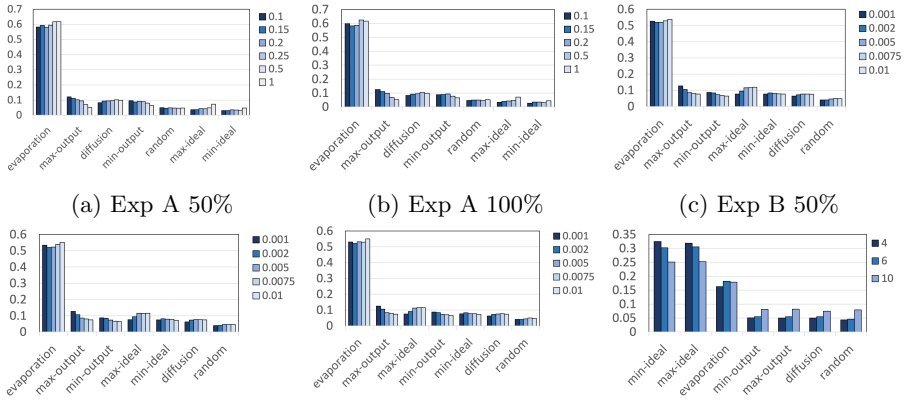
In experiment C we predict the overall average unhappiness level in categories defined by quantiles. Our balanced accuracy for four quantiles is 91.6%, for six quantiles is 87.8%, and for ten quantiles is 76.7%. We can see from balanced accuracy that the more fine grained our categories, the harder it is to predict. This result is not surprising, but is a confirmation of what may be most difficult to predict via machine learning. Figure 6 confirms that our prediction ability is high in the four quantile scenario, and steadily decreases as we increase the number of quantiles. As the number of quantiles increases we are increasing the difficulty of the prediction, as fewer categories means more precision on average. For ten quantiles, the random forest is not able to adequately predict any classes other than the smallest and largest categories. In all quantile experiments the highest average values are the most likely to be accurately predicted, meaning that it's easiest to determine if a high average unhappiness level will occur.

5.4 Feature Importance

One of the benefits of learning a random forest model is that the model can readily tell us how much influence each parameter had on the final result. Feature importance is a useful piece of information, as a goal in this process is to be able to better understand the parameter space and how those parameters affect the final outcomes in the simulation. We use the feature_importance feature of scikit-learn's implementation of Random Forests to compute this result.

In Figure 7 we see importance of each feature for determining which class a set of parameter combinations predict. A feature (e.g. parameter) importance of 1 would imply that the single feature alone can predict the results. As expected,

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-35995-8_17



(a) Exp A 50% (b) Exp A 100% (c) Exp B 50%

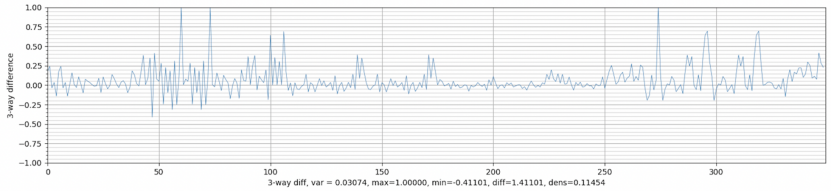
(d) Exp B 75% (e) Exp B 100% (f) Exp C

Fig. 7: Feature Importance. In Experiments A and B, evaporation rate has the strongest influence on classification. For all quartiles in Experiment C, the min and max ideal temp play the strongest role in determining classification. Neither percent nor threshold heavily influence this ranking.

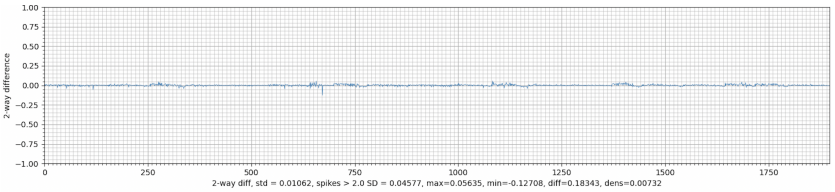
no feature has that high an impact, nor does any pair of features. However, in each experiment a single feature has a majority importance. For Experiments A and B, evaporation rate is the most important feature and is significantly more important than all of the other features with an importance over 0.5 (Figure 7). For Experiment C where we are predicting the averages, the minimum and maximum ideal temperature parameters have the most influence on the results, and are almost equally important with each other with an importance of at least 0.25 in each scenario (Figure 7f). The thresholds for determining the class label do not affect which feature is most important for making correct predictions; the feature importance seems tied instead to what we are trying to predict.

For a closer look at the importance of features, we can consider 2-way and 3-way combinations of feature values in the covering arrays, without machine learning. The notion of *combination frequency differences* (CFD) [14] identifies combinations of feature values that are more strongly associated with one class than another, an approach useful in explainable AI [13] and vulnerability analysis for physically unclonable functions [12]. The CFD value for a single attribute value combination is the difference between the rate of occurrence of a particular value combination in one class versus another class. For example, a particular combination of fur and eye color may occur in 75% of one dog breed as compared with 10% of another breed, for a difference of 0.65. Differences are computed for all attribute value combinations, for $v^t\binom{n}{t}$ t-way combinations of n attributes with v values each. Graphing these differences for every value combination in a data set produces a graph such as shown in Figure 8a, which shows a machine learning data set for clinical values related to diabetes. The graph shows the difference for all 3-way combinations in this data set. As can be seen in Figure

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-35995-8_17



(a) Diabetes combination frequency differences [14], for comparison



(b) Heatbugs 2-way combination frequency differences

Fig. 8: Combination frequency differences to identify combinations of features values that are more strongly associated with one class versus another. In (a) we see a comparison graph where some feature combinations strongly affect the outcome. In (b) we see that that no parameter combinations are strongly associated with the outcome for our 2-way data.

8a, three combinations occur in 100% of the positive cases (above center line), indicating a strong association with these combinations of clinical values [14].

Applying this method to the Heatbugs 2-way covering array for Experiment A produces a graph as shown in Figure 8b. A separate report (not shown due to space limitations) shows that, consistent with the random forest model, evaporation rate is the single most important attribute. However, the 2-way combination frequency analysis also shows that low values of *minimum ideal temp* in combination with evaporation rate are also strongly associated with class predictions. Similarly, a few other combinations are also found to be significant. This additional useful information is not apparent from the random forest model. Also note that the range of difference values is much narrower in Fig. 8b than in Fig. 8a, indicating that the Heatbugs problem is in some sense "harder" for machine learning than the diabetes prediction problem (which has $> 99\%$ accuracy), as there are no combinations that clearly indicate a particular class on their own.We plan to investigate this CFD approach further in future work, to determine how well it can aid in determining the impact of parameters on model results, and/or how likely a model is to be predictable via machine learning.

Both the ML feature importance results and the combination frequency difference results confirm that the ABM outcomes cannot be fully explained by 2-way combinations alone, reconfirming that training our ML model on 2-way data to predict 3-way data does mean we are able to predict situations previously unseen in the training (2-way) data. Pairs of variables are not enough to fully define the model's behaviors, although the 2-way data does allow us to train a model to predict a broader set of behavior than was already seen.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-35995-8_17

6 Conclusions

We propose a new process to systematically explore the parameter space of an ABM and predict outcomes of simulations using machine learning (ML) and combinatorial analysis. This process uses covering arrays to significantly reduce the parameter space from ∼11 trillion to $3.3551e4$ parameter combinations, allowing us to systematically explore, learn, and predict simulation outputs for almost 4 million unseen parameter combinations for the Heatbugs ABM. We train three different sets of random forest ML models using 2-way covering arrays of the parameter values, to predict three different types of outcomes of the simulation. We test the effectiveness of this approach on the 3-way covering array, which represents significantly more parameter combinations than the training data, to see if it is possible to train the model on a small part of the parameter space and then predict the results on a much larger part of the parameter space. We perform combinatorial analysis and feature importance analysis for insights into simulation output predictability, and the role of parameters in the predictions.

Overall our results show that this process works. All three experiments on Heatbugs had high success in predicting a) if the standard deviation of the average unhappiness level will be low, b) if the standard deviation of unhappiness will be low in relation to the average unhappiness, and c) the average unhappiness. We tested many different thresholds for our categories, all with high success. Our best success was when our classes were most balanced, although learning was still successful even with unbalanced data. These results indicate that the process could be effective on other ABMs as well, and that further work should be done on the predictability of ABM results.

In future studies we plan to explore what types of predictions can be made about ABMs, if the process will work on all types of models or only on models with specific properties, and if a different ML algorithm will garner better results. Random forests were a good fit for this model's data and our goals, but another ML algorithm may better fit other models. We also plan to further explore the use of CFD in analyzing results from the covering arrays in determine parameter importance in results prediction. We plan to continue this research to further define how and when our approach can be successfully applied, so that eventually one could use our process to explore a parameter space more effectively on many ABMs. Disclaimer: Commercial products may be identified in this document, but such identification does not imply recommendation or endorsement by NIST, nor that the products identified are necessarily the best available for the purpose.

References

1. A methodology for fitting and validating metamodels in simulation. European Journal of Operational Research **120**(1), 14–29 (2000)
2. Borazjany, M., Lei, Y., Kacker, R., Kuhn, R.: Combinatorial testing of acts: A case study. In: First Intl Workshop on Combinatorial Testing, IEEE Fifth Intl Conf on Software Testing, Verification and Validation (ICST 2012. pp. 591–600 (2012)
3. Cacuci, D., Ionescu-Bujor, M., Navon, I.M.: Sensitivity and Uncertainty Analysis: Applications to Large-Scale Systems. Taylor Francis Group, CRC Press (2005)

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-35995-8_17

4. Calvez, B., Hutzler, G.: Automatic tuning of agent-based models using genetic algorithms. In: International Workshop on Multi-Agent Systems and Agent-Based Simulation. pp. 41–57. Springer (2005)
5. Calvez, B., Hutzler, G., et al.: Adaptative dichotomic optimization: A new method for the calibration of agent-based models. In: Proceedings of the 2007 European Simulation and Modelling Conference (ESM'07). pp. 415–419 (2007)
6. Danglade, F., Pernot, J.P., Véron, P.: On the use of machine learning to defeature cad models for simulation. Computer-aided Design and Applications **11**, 358–368 (2014)
7. Hofmann, M.: On the complexity of parameter calibration in simulation models. The Journal of Defense Modeling and Simulation **2**(4), 217–226 (2005)
8. Kleijnen, J.: An overview of the design and analysis of simulation experiments for sensitivity analysis. European Journal of Operational Research **164**(2), 287–300 (2005)
9. Kleijnen, J.: Sensitivity analysis of simulation models: an overview. In: 6th International Conference on Sensitivity Analysis of Model Output (2010)
10. Kraußе, T., Cullmann, J.: Towards a more representative parametrisation of hydrologic models via synthesizing the strengths of particle swarm optimisation and robust parameter estimation. Hydrol. Earth Syst. Sci. **16**, 603—-629 (2012)
11. Kuhn, D.R., Kacker, R., Lei, Y.: Introduction to Combinatorial Testing. Chapman Hall / CRC (2013)
12. Kuhn, D.R., Raunak, M., Prado, C., Patil, V.C., Kacker, R.N.: Combination frequency differencing for identifying design weaknesses in physical unclonable functions. In: 2022 IEEE IEEE Intl Conf on Software Testing, Verification and Validation Workshops (ICSTW). pp. 110–117. IEEE (2022)
13. Kuhn, D., Kacker, R., Lei, Y.: Advanced combinatorial test methods for system reliability. Tech. rep., 2010, IEEE Reliability Society (January 2011)
14. Kuhn, D., Raunak, M.S., Kacker, R.: Combination frequency differencing. Tech. rep., National Institute of Standards and Technology (December 2021), `https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.12062021-draft.pdf`
15. Lanus, E., Freeman, L.J., Richard Kuhn, D., Kacker, R.N.: Combinatorial testing metrics for machine learning. In: 2021 IEEE Intl Conf on Software Testing, Verification and Validation Workshops (ICSTW). pp. 81–84 (2021)
16. Maalouf, C., Olsen, M., Raunak, M.S.: Combinatorial testing for parameter ealuation. In: Proceedings of the 2019 Winter Simulation Conference (WSC19). Society for Computer Simulation International (December 2019)
17. Olsen, M., Laspesa, J., Taylor-D'Ambrosio, T.: On genetic algorithm effectiveness for finding behaviors in agent-based predator prey models. In: Proceedings of the Summer Simulation Multi-Conference (SummerSim'18). Society for Computer Simulation International, Bordeaux, France (July 2018)
18. Olsen, M., Raunak, M.S.: Efficient parameter exploration of simulation studies. In: 2022 IEEE 29th Software Technology Conf. pp. 190–191 (2022)
19. Stonedahl, F.: Genetic Algorithms for the Exploration of Parameter Spaces in Agent Based Models. Ph.D. thesis, Northwestern University (2011)
20. Wilensky: Netlogo heatbugs model. http://ccl.northwestern.edu/netlogo/models/Heatbugs (2004)
21. Zhang, W., Valencia, A., Chang, N.B.: Synergistic integration between machine learning and agent-based modeling: A multidisciplinary review. IEEE Transactions on Neural Networks and Learning Systems pp. 1–21 (2021)

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-35995-8_17