# PREDICTING THE RESOURCE NEEDS AND OUTCOMES OF COMPUTATIONALLY INTENSIVE BIOLOGICAL SIMULATIONS

Andrew Fisher

Dept. of Computer Science
Lakehead University
Thunder Bay ON, Canada
afisher3@lakeheadu.ca

Bhisma Adhikari
Chao Zhai
Joshua E. Morgan

Dept. of Computer Science & Software Engineering
Miami University, Oxford OH, USA
{adhikab,zhaic,morgan42}@miamioh.edu

Vijay K. Mago

Dept. of Computer Science
Lakehead University
Thunder Bay ON, Canada
vmago@lakeheadu.ca

Philippe J. Giabbanelli

Dept. of Computer Science & Software Engineering
Miami University
Oxford OH, USA
giabbapj@miamioh.edu

## ABSTRACT

Accurately simulating viral dynamics within a human body is often computationally intensive, thus requiring dedicated computing infrastructures. This limits the use of simulations in personalized medicine since patients may not all have access to such infrastructures or the possibility of waiting weeks for predictions. Lowering the computational burden by simplifying the models is a challenge since simulation results must remain accurate in order to safely support the decision-making activities of patients. In this work, we use machine learning to predict the results of a simulation (i.e. surrogate model) and its computational resources (i.e. performance model). This allows patients to use machine learning as a computationally light proxy to simulations and to know how long an accurate simulation would take. We demonstrate that machine learning can build such surrogate and performance models with low errors for five previously developed models of HIV, while outperforming interpolation methods commonly used for surrogate modeling.

**Keywords:** Cellular Automata, HIV-1, Interpolation, Machine Learning, Regression

## 1 INTRODUCTION

A central tenet of personalized or 'individualized' medicine is the ability to accurately and efficiently identify the right intervention based on specific characteristics of a patient. This represents an important shift from "one size fits all" interventions or group-based interventions to developing a solution that is optimal for a target patient but not for others. The potential role of simulations in this setting was highlighted early on by Lesko, who pointed out the ability to compute an array of possible responses to clinical treatments while accounting for uncertainty (Lesko 2007). Since then, numerous simulation models have been developed in which the parameter values or input data can match a patient's characteristics. Examples include the

development of therapeutic unique devices, such as spinal implants based on patient-specific biomechanical simulations (Sabine and Dietrich 2017), or the growing area of pharmacogenomics in which models of the immune system are used to identify patient responses based on their specific genetic variations (Pappalardo et al. 2008, Sneha and Doss 2016). Simulation models used in this context face two challenges. First, they need to be safe and highly accurate for patients, as would be expected of any system with medical applications. Second, their computationally intensive nature requires a dedicated computing infrastructure, which is often discussed under the umbrella of handling 'big data' (Gligorijević et al. 2016). At a high-level, a human body has about 37.2 trillion cells and approximately 60,000 miles of blood vessels. Even if simulations necessarily simplify the number of elements to track and/or the temporal granularity at which interactions are represented, maintaining accuracy leads to computing on dedicated grid-based infrastructures (Halling-Brown et al. 2010) or, more recently, GPU-based platforms (Santhanam et al. 2016).

To expand access to simulations for personalized medicine, it is thus essential to explore trade-offs in which a sufficiently accurate solution can be found without a dedicated computing infrastructure. And, when highly accurate simulations must be performed, it is necessary to correctly estimate their computational resources such that patients are not waiting indefinitely for simulations to complete. We recently suggested that *machine learning* may hold the key to achieve both of these needs (Giabbanelli 2019). First, machine learning can be used to create a *surrogate* or 'meta-' model (Van Steenkiste et al. 2019). That is, characteristics of the patient would be given as input to a machine learning model trained to give an approximation of the simulated output – but at a fraction of the computational cost. While the use of surrogate models is well-established for optimization problems in engineering, there is a paucity of research using surrogates to predict the outcome of biological events within a human host. Second, machine learning can be used to create a *performance model* which predicts resource utilization such as the total run time or peak memory consumption. This is an active area of research in high-performance computing (Malakar et al. 2018), although the objective is usually to find the parameter values that optimize performances rather than telling a user exactly how long it will take to run the simulation (Thiagarajan et al. 2018).

In this paper, we use machine learning to build surrogate and performance models such that patients going to a point of care (*i*) may be offered approximate predictions (using surrogate models) based on their specific needs without requiring a dedicated computing infrastructure; or (*ii*) be informed (using performance models) on the time that would be needed to run a simulation for a more accurate result. We focus on simulation models for the Human Immunodeficiency Virus (HIV) which affects 37.9 million people worldwide and contributed to 770,000 deaths in 2018 alone ("UNAIDS" 2019). Using a single model of HIV would limit the generalizability of our results, as the ability to build accurate surrogate and performance models may be an artifact of the rules used in one specific model. Consequently, our study examines *five previously published models of HIV* which we recently re-implemented (https://osf.io/uxmkv/). To further mitigate the risk of implementation errors and avoid having to 'trust' our code, all implementations were verified (by replicating results from the original papers) and further assessed by an independent reproducibility committee which gave the 'reusable' and 'functional' ACM badges (Giabbanelli et al. 2019).

The machine learning task here is *regression* – that is, we predict a continuous value such as the time needed for a simulation to be completed or the number of dead cells within a human host. The machine learning process consists of first running simulations on some parameter values and recording their outcome as well as resource utilization, then training the regression model (i.e., *regressor*) on this data. The situation of a new patient is captured by parameter values that may not have been previously encountered, either falling within simulated parameter values (*interpolation*) or lying beyond (*extrapolation*). We note that machine learning is not the only way to predict continuous values: interpolation methods have been developed over decades in numerical analysis, and they are commonly used to build surrogates (Østergård et al. 2018). Ignoring such methods by only providing the results of regressors would thus be an incomplete or biased picture of the potential to use machine learning for biological simulations. Consequently, all results produced by machine learning in this paper are compared with results produced by commonly used interpolation methods.

The remainder of the paper is structured as follows. In section 2, we provide a succinct overview of models for HIV, assuming that the reader is familiar with the design of discrete simulation models. We detail the standard machine learning process used to create regressors and provide a summary of interpolation methods. Section 3 summarizes our process to create training data from the simulations and fine-tune several regression algorithms such as decision tree and support vector machines. Section 4 presents the accuracy of surrogate and performance models for both regressors and interpolation methods. These results are then discussed in section 5 together with the limitations of our findings and directions for future research.

## 2 BACKGROUND

### 2.1 Cellular Automata Models of the Human Immunodeficiency Virus (HIV) type 1

We focus on five Cellular Automata (CA) models identified in (Giabbanelli et al. 2019): three were found by a review as representing different model designs (Precharattana et al. 2010, González et al. 2013, Moonchai and Lenbury 2016), one is the foundational model in discrete simulations of HIV (dos Santos and Coutinho 2001), and one was added due to its uniqueness in simulating two modes of transmission for HIV between cells (Rana et al. 2015). In this section, we briefly introduce the design of CA models by emphasizing design choices for HIV research. Detailed, standardized descriptions are provided for each model at the start of the corresponding Jupyter notebook at https://osf.io/uxmkv/. For comprehensive introductions to the field, we refer readers to (Precharattana 2016) as the most recent discussion on CA models for HIV, and to (Willem et al. 2017, Herzog et al. 2017) for the use of simulations for infectious diseases including HIV.

A cellular automaton is a discrete model involving a set of *cells* that, for our purposes, are in a 2D plane. Each one of these cells is in one *state* from a small list of categorical possibilities. All five models include the states 'healthy, 'infected-A1' and 'infected-A2' (to represent two stages of infection), and 'dead'. As the years go, models tend to get more detailed through the addition of states to capture latency in infection (Figure 1f), drug therapy (Figure 1g), or drug adherence.

Cells are updated synchronously at each time tick, corresponding to one week of physical time. When each cell is assigned a color, a simulation can be visualized as a series of colored matrices (Figure 1a-d) (Giabbanelli and Baniukiewicz 2019). A cell is updated based on pre-defined *rules* which can be represented as flowcharts (Figure 1e-g). Since all HIV models are stochastic, some rules involve probabilities hence multiple simulations are needed to confidently characterize future states of a system. For instance, the two runs in Figure 1c-d are from the same model yet produce markedly different outcomes. A rule can involve time (e.g., an infected-A1 cell always becomes infected-A2 after four weeks) or the state of surrounding cells (i.e., the 'neighborhood configuration'). All models represent tissues of a human host and assume that any touching cells can pass on the virion, hence a *Moore neighborhood* is used consisting of the surrounding eight cells. At a high-level, all models capture the same disease progression thus their rules express how healthy cells can get infected and ultimately die. Differences in rules across the models are mostly a result of adding rules to cope with new states (e.g. drugs in Figure 1g) or, less commonly, changing the parameter values of a rule. Parameter values for each model are provided in a standardized table in the notebooks.

Finally, a CA has *boundaries* that determine how the rules work at the edge of the system. One is a *fixed* boundary which simply limits the cell interaction to within a grid. Cells at the edge of the grid have fewer neighbors than other cells, which creates irregularities in the system and can potentially cause unwanted artifacts around the edges. The other common type of boundary is *periodic*, which creates a toroidal topology so that, for example, cells on the left-most edge interact with those at the right-most edge. Four models use periodic boundaries (Precharattana et al. 2010, González et al. 2013, dos Santos and Coutinho 2001, Moonchai and Lenbury 2016), and one has fixed boundaries (Rana et al. 2015) which leads to the linear patterns observed in Figures 1c-d.

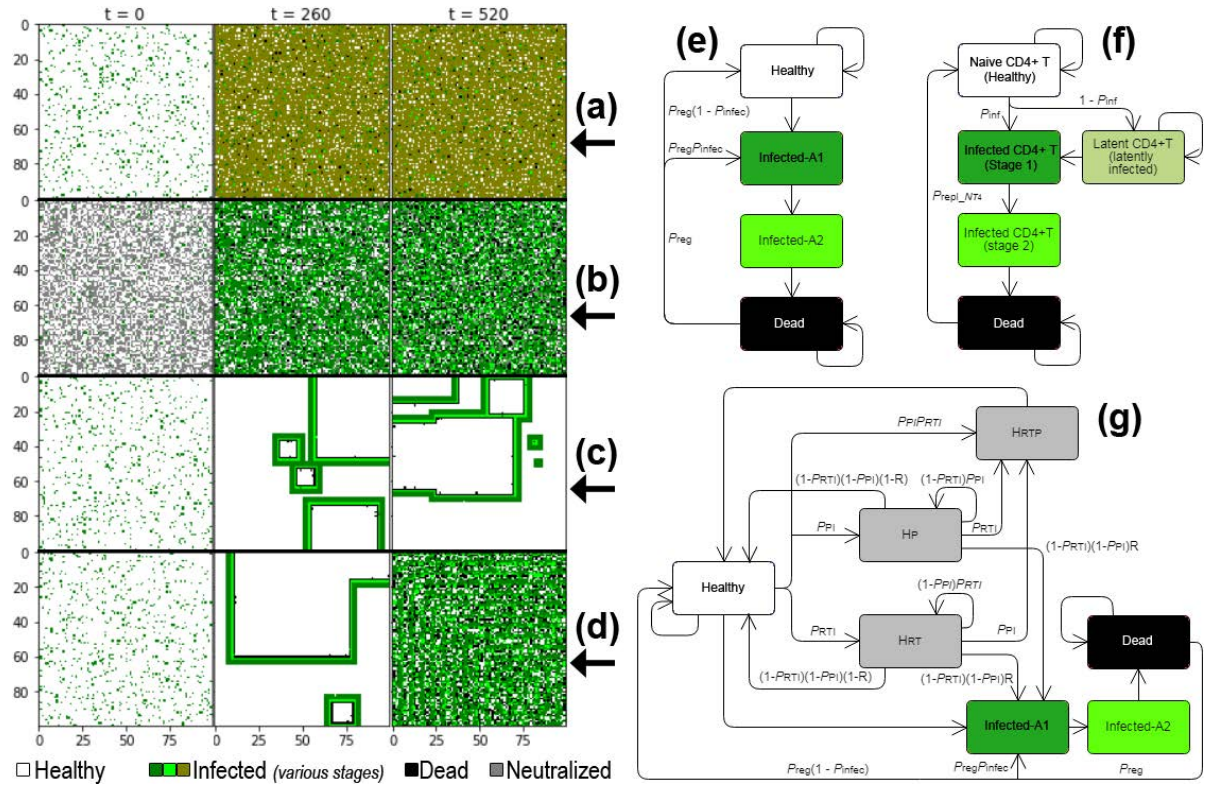


Figure 1: Sample runs from models by Precharratana *et al.* (a), Moonchai & Lenbury (b), and Rana *et al.* (c-d). Flowcharts of models by dos Santos & Coutinho (e), Precharratana *et al.* (f), and González *et al.* (e).

### 2.2 Machine Learning Process

*Supervised* machine learning tasks such as regression or classification proceed as follows. A dataset is obtained, consisting of *feature values* (e.g., parameter values for the simulation model) and associated outputs (e.g., simulation results, total time, and peak memory). As a data-driven driven approach, machine learning algorithms will be trained and evaluated using this dataset, similar to the process of calibration and validation in simulation research. The portion of the data used for *training* must be disjoint from the data withheld for *testing* otherwise the model's performances may be incorrectly high (i.e. *overfit*). Data is either split once with a holdout strategy (e.g., 80% for training and 20% for testing) or split $k$ times with *k-cross fold-validation*: $k-1$ parts are used for training, the remaining part used for testing, and the process is repeated until each part has served for training or testing. Although a cross fold-validation is more computationally intensive than a holdout, it serves as gold standard since it provides a more robust estimate of performances by averaging results across the $k$ splits instead of a single split.

When a machine learning algorithm has no parameters, it can be directly trained over the training data and then evaluated on testing. However, most algorithms have several parameters which need to be optimized (i.e., *hyper-parameter tuning*). An incorrect approach would be to have an additional loop which repeats the entire $k$-cross fold-validation across parameter values (Cawley and Talbot 2010): it would be similar to allowing a student to re-take the same exam until their study strategy finally leads the best score, hence overfitting. Rather, hyper-parameter tuning leads to a further subdivision of the training data into a training and *validation* subset. A model is repeatedly built on the training subset using different parameter values and evaluated on the validation subset. Once the best parameter values have been identified, the model is rebuilt on the whole training data (i.e., both training subsets and validation subsets) and then evaluated on the

testing data. *Nested cross-validation* handles the process of repeatedly dividing the data into training-testing (outer-fold) and, within each training portion, repeatedly dividing into a training subset and validation subset (inner-fold) for hyperparameter optimization. This process is very computationally intensive. For instance, imagine that 4 regression algorithms are used, each having two parameters with six values, and we do 10 outer-folds as well as 10 inner-folds. This already leads to building $4 \times 6 \times 6 \times 10 \times 10 = 14,400$ models.

It is not advisable to ignore potentially accurate regression algorithms or to compute less robust estimations by doing fewer folds. Savings are thus mostly obtained by carefully selecting which parameters to optimize, their possible values, or how to combine them (essentially to avoid the complete combination known as grid search) (Mehmani et al. 2014, Bergstra and Bengio 2012). Parameters and values depend on the algorithm. For instance, decision tree regressors are sensitive to constraints on depth or minimum number of instances (Rosso and Giabbanelli 2018), and support vector regressors depend on the choice of kernel function, each of which has its own parameters (Crutzen et al. 2015). In this paper, we use decision trees, support vector machines, and the Least Absolute Shrinkage and Selection Operator (Lasso). As the first two are commonly taught in introductory machine learning courses, we refer the reader to an intuitive introduction in (Crutzen and Giabbanelli 2014). Since the Lasso may need more explanations on penalized regressions, a thorough introduction is given in (Tibshirani 2011).

### 2.3 Interpolation Methods

Interpolations of irregularly-spaced data points have been studied for decades. In our study, we focus on three methods. The Radial Basis Function (RBF) represents the interpolating function as a linear combination of basis functions, one for each training point (Powell 1992). The basis functions only depend on the *distance* (or difference) from the prediction point to the training point. The Inverse Distance Weighting (IDW) is described as an interpolating method where the unknown points (i.e. test points) are calculated with the weighted average of the sampling points (i.e. training points) (Shepard 1968). Lastly, the Least Squares (LS) method fits a linear model with coefficients to minimize the sum of squares between the training data and the output from the linear approximation (Friedman et al. 2001).

We note that *Kriging* is a commonly used interpolation method in surrogate modeling for smooth surface responses because it is fast to train and often more accurate than other types of surrogate models. However, prediction time increases with the size of the dataset, and training can fail if the dataset is too large or poorly spaced – which limits the accuracy that is attainable (Hwang and Martins 2018). Variations of Kriging have been introduced, such as KPLS which also includes a partial least squares (PLS) approach, GE-KPLS which further features gradient enhancing, or RMTS which is efficient on low-dimensional problems with a large number of instances such as encountered here (Hwang and Martins 2018). Given the wide variety of Kriging-based approaches, which are not all supported by the library used here (Bouhlel et al. 2019), a comprehensive use of Kriging methods is beyond the scope of the present work.

## 3 METHODS

To provide full disclosure into our methods and support replication efforts, our repository (https://osf.io/uxmkv/) contains all artifacts from this study. The HIV models are available as Jupyter Notebooks (Python 3) under 'Models of HIV', while 'SpringSim20' holds their outputs, notebooks to build surrogate as well as performance models, and their accuracies. Our process is summarized in Figure 2 and detailed below.

The output of the simulation models (which serve as input to build surrogate and performance models) were generated as follows. In line with previous studies, we ran each of the five HIV models for 600 time steps (i.e. weeks) to capture the first two stages of HIV infection: the acute infection happening in the first weeks to months, and the chronic infection unfolding over years. Each combination of parameter values was run

repeatedly, to account for the stochastic nature of the models. Five parameters were varied in the simulations to represent differences between patients as well as uncertainty about the biological dynamics:

(1) The CA is simulated by a grid of size $N \times N$ in which each element represents a physical CD4+ T Cell. Previous studies have set the value of $N$ to 100 (Precharattana et al. 2010, Rana et al. 2015), 500 (Moonchai and Lenbury 2016), or 700 (González et al. 2013, dos Santos and Coutinho 2001). The real value is *much* larger, as there is an estimated total of $2.2 \times 10^{11}$ CD4+ T cells in lymphoid tissues and an additional $4.9 \times 10^{9}$ in blood (Zhang et al. 1998). In practice, the value of $N$ can thus be as large as one can afford to simulate. We thus tested values of $N = \{100, 200, 400, 800\}$.

(2) The percentage of initially infected cells (denoted in models as $P_{HIV}$) was varied to reflect that patients can have different baseline viral loads. We used the values 0.04, 0.05 (default for most models), and 0.06.

(3) The probability that a *healthy* cell replaces a dead cell (denoted $P_{repl}$) was lowered from 0.99 (default value) to 0.98 and 0.97. This reflects the possibility that some patients have a lower regenerative capacity, for instance due to aging.

(4) The probability that an *infected* cell replaces a dead cell (denoted $P_{infect}$) was set to either $1 \times 10^{-5}$ or $2 \times 10^{-5}$ to reflect uncertainty in the value.

(5) The number of time steps before an infected cell is discovered by the immune system ($\tau$) was tested at 3, 4, and 5 weeks. In reality, this may be at the scale of hours rather than weeks. However, some parameters in a model are set based on clinical data while others are calibrated. Thus, changing the other four parameters can bring uncertainty on the right value for this one.

For each HIV model, data from the simulation serves as input to the machine learning and interpolation methods. Feature values consist of the five simulation parameters aforementioned. Two performance models are built to predict the time (in seconds) and peak memory consumption (in Mb) recorded throughout a simulation. Up to four surrogate models are built to predict the number of cells in each state, recorded at the end of the simulation. While all HIV models have the states healthy, infected (covering both early infections $A_1$ and infections detected by the immune system $A_2$), and dead, a fourth category is added in some

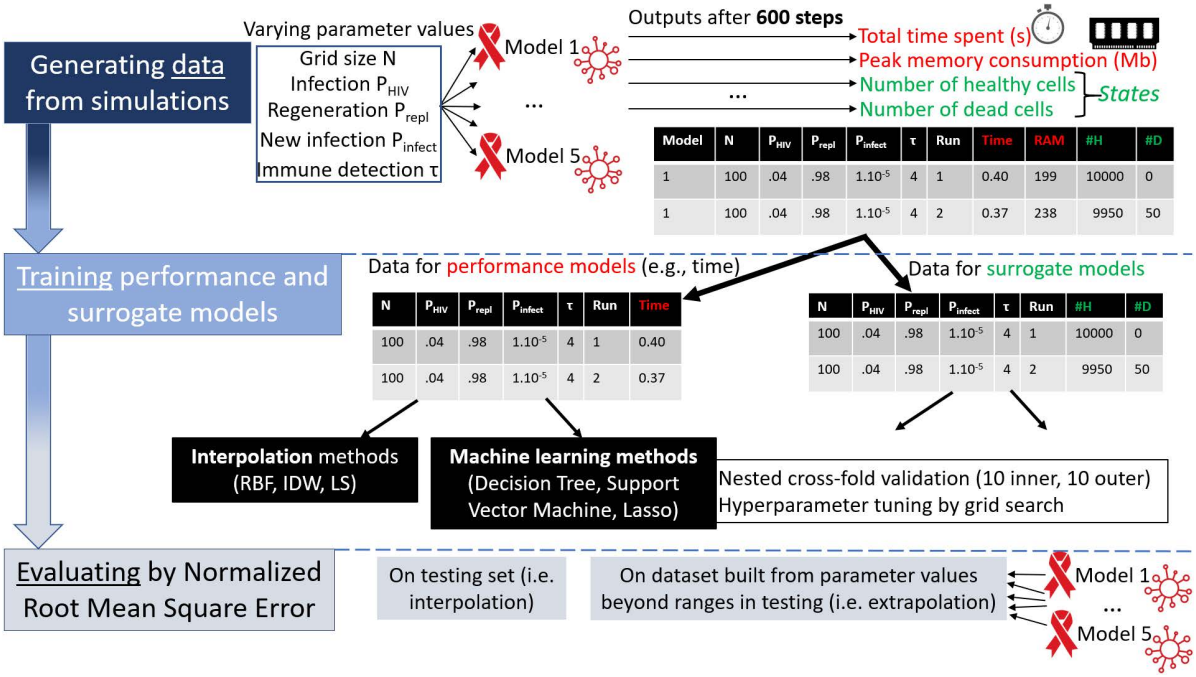


Figure 2: Summary of our methods from data generation to building and evaluating models.

models such as dormant infected cells. To build a model, we use three machine learning methods (decision trees, support vector machines, Lasso; c.f. section 2.2) and three interpolation methods (RBF, IDW, LS; c.f. section 2.3). Machine learning methods used the library scikit-learn version 0.21.3 and interpolation methods relied on the Surrogate Modeling Toolbox (https://github.com/SMTorg/SMT) (Bouhlel et al. 2019).

Machine learning methods used a nested cross-fold validation consisting of 10 inner folds and 10 outer folds, thus ensuring robust estimates at the expense of a computationally intensive process. To cope with the amount of computations, we used the Redhawk cluster from Miami University, consisting of 26 compute nodes, each equipped with 24 Intel Skylake Xeon Gold 6126 cores. We used a grid search for hyperparameter tuning. For the decision tree, we optimized the splitting criterion and type of splitter as well as the maximum depth and minimum number of samples to split. For the support vector machine, we tested with and without shrinking heuristic, and whether to scale the kernel coefficients.

Performances of each model were evaluated using the Normalized Root Mean Square Error (RMSE). We evaluate performances in two ways. First, a subset of data withheld for evaluation was used, coming from the same dataset as mentioned above (with a maximum grid size of 800). This leads to testing on *inter*polations as the parameter values of test cases are likely to be identical to, or contained within, some of the training data. Second, we performed additional simulations in which parameter values were outside the ranges previously seen. We used $N = \{1000, 1200, 1400, 1600\}$, $P_{HIV} = \{0.07, 0.08\}$, $P_{repl} = \{0.96\}$, and $\tau = \{6\}$ to model more extreme cases. We kept $P_{infec} = \{1 \times 10^{-5}, 2 \times 10^{-5}\}$ as there was no biological rationale for other values. By testing predictions on feature values that are outside of the training range, we are evaluating performances on a harder setting of *extra*polations.

## 4 RESULTS

We used each surrogate and performance model to predict a numerical outcome given features that were either from the same dataset but withheld when training the model (so the model has seen similar inputs), or from a dataset computed using feature values outside of the range used in training (thus forcing the model to extrapolate). Results from the first evaluation are shown in Figure 3, where low errors are colored in green. It is only when using the machine learning technique of decision trees that we produce an entirely green column, indicating strong results across all HIV simulations and all types of models. Support vector machines provide the second best performance, as they produce low errors for two HIV simulations, and only make one (large) error out of six predictions for another simulation. Lasso, which is entering the realm of regressions, already shows several issues as performances are only satisfactory for one simulation, and an error is made in two others. In contrast, there is not a single simulation in which the interpolation methods consistently produce low errors. IDW and RBF are more likely to make errors than to get the results right for the foundational HIV simulation and its direct successor. For the other three simulations, IDW and RBF create accurate surrogate models but inaccurate performance models.

There are two takeaways from these results. First, using machine learning such as decision trees or (to a lower extent) support vector machines, we *can* accurately predict the results of a simulation or the computational resources it needs. This is uniquely obtained through machine learning techniques and not through interpolation methods. Second, the problems of creating surrogate or performance models have different difficulties, as one technique could be accurate for one but not the other.

The difference was even more pronounced when evaluating models on extrapolations. This is a harder task by design, since models are forced to make predictions for inputs unlike anything that they were trained on. It is nonetheless a useful task in practice, for instance to predict the outcome of very large grids of cells without having to ever run one. Interpolation methods were expected to be challenged on extrapolations, hence the question is not whether errors will increase but rather by how much. Out of the 75 errors computed across the three interpolation methods, 3 resulted in medium errors (using LS to predict the time for three

| | | | Interpolation (numerical method) | | | Regressor (Machine Learning method) | | |
|---|---|---|---|---|---|---|---|---|
| **Simulation** | **Model type** | **Outcome to predict** | IDW | LS | RBF | DT | LASSO | SVM |
| **Dos Santos** | **Surrogate** | Class [A] | 0.1399 | 0.2428 | 0.1009 | 0.0520 | 0.1076 | 0.5260 |
| | | Class [D] | 0.5945 | 0.4745 | 0.4207 | 0.0108 | 0.1500 | 0.1078 |
| | | Class [H] | 0.1441 | 0.2311 | 0.1015 | 0.0623 | 0.1117 | 0.1050 |
| | **Performance** | RAM (MB) | 0.0212 | 0.2057 | 0.0258 | 0.0003 | 0.0937 | 0.0133 |
| | | Time | 0.1218 | 0.1344 | 0.0740 | 0.0000 | 0.0637 | 0.0000 |
| **Gonzalez** | **Surrogate** | Class [A] | 0.5062 | 0.4636 | 0.3854 | 0.0000 | 0.2896 | 0.0000 |
| | | Class [D] | 0.4403 | 0.3941 | 0.3304 | 0.0000 | 0.3177 | 0.0000 |
| | | Class [H] | 0.5206 | 0.4602 | 0.3894 | 0.0000 | 0.3113 | 0.0000 |
| | **Performance** | RAM (MB) | 0.0212 | 0.2052 | 0.0255 | 0.0003 | 0.0936 | 0.0133 |
| | | Time | 0.0993 | 0.1196 | 0.0598 | 0.0000 | 0.0584 | 0.0000 |
| **Moonchai** | **Surrogate** | Class_blood [A] | 0.0113 | 0.1358 | 0.0090 | 0.0004 | 0.0700 | 0.0600 |
| | | Class_blood [H] | 0.0050 | 0.1104 | 0.0039 | 0.0002 | 0.0633 | 0.0127 |
| | | Class_lymph [A] | 0.0118 | 0.1358 | 0.0093 | 0.0004 | 0.0702 | 0.0596 |
| | | Class_lymph [H] | 0.0055 | 0.1103 | 0.0042 | 0.0002 | 0.0634 | 0.0126 |
| | **Performance** | RAM (MB) | 0.0887 | 0.1182 | 0.0830 | 0.0003 | 0.0773 | 0.0008 |
| | | Time | 0.1186 | 0.1419 | 0.0846 | 0.0000 | 0.0683 | 0.0000 |
| **Precharattana** | **Surrogate** | Class [0] | 0.0004 | 0.1495 | 0.0003 | 0.0002 | 0.0779 | 0.5371 |
| | | Class [A] | 0.0148 | 0.1509 | 0.0118 | 0.0003 | 0.0760 | 0.0379 |
| | | Class [D] | 0.0052 | 0.1610 | 0.0039 | 0.0001 | 0.0761 | 0.0054 |
| | | Class [H] | 0.0073 | 0.1906 | 0.0054 | 0.0003 | 0.0832 | 0.0558 |
| | **Performance** | RAM (MB) | 0.0832 | 0.1471 | 0.0867 | 0.0004 | 0.0763 | 0.0022 |
| | | Time | 0.0390 | 0.1157 | 0.0319 | 0.0000 | 0.0608 | 0.0000 |
| **Rana** | **Surrogate** | Class [A1] | 0.0074 | 0.1726 | 0.0057 | 0.0020 | 0.0774 | 0.3735 |
| | | Class [A2] | 0.0074 | 0.1728 | 0.0058 | 0.0006 | 0.0768 | 0.0912 |
| | | Class [D] | 0.0338 | 0.1819 | 0.0272 | 0.0006 | 0.0825 | 0.0943 |
| | | Class [H] | 0.0060 | 0.1635 | 0.0048 | 0.0029 | 0.0763 | 0.0895 |
| | **Performance** | RAM (MB) | 0.0165 | 0.1558 | 0.0329 | 0.0001 | 0.0792 | 0.0017 |
| | | Time | 0.1189 | 0.1707 | 0.0816 | 0.0000 | 0.0735 | 0.0000 |

Normalized Root Mean Square Error — Low — Medium — High

Figure 3: Results for each method (columns), simulation and type of model (rows). Lower errors (green) indicate better results.

out of five simulations) and all others produced high errors. In other words, not a single case was still within a low error level when using interpolation methods. Figure 4 shows that every result from an interpolation method was beyond the tolerance threshold for error level. Note that errors were not exclusively made by interpolation methods since the Lasso method was also systematically above this level. As the logarithmic scale suggests, errors often became extreme. However, we note that machine learning methods (i.e., decision trees and support vector machines) can still provide accurate performance models (predictions for time and RAM always had low error levels) or, in a few cases, surrogate models with low to medium errors.

## 5 DISCUSSION AND CONCLUSION

Simulations can estimate the disease dynamics of a patient in reaction to various treatments while taking into account the specific characteristics of the patient and accounting for strutural or parametric uncertainty (Lesko 2007). While these predictive abilities suggest that simulations can support personalized medicine, there is a computational challenge: accurate results require large and dedicated computing infrastructures (Halling-Brown et al. 2010, Santhanam et al. 2016) which are beyond the reach of many patients. There is thus a trade-off between democratizing access by lowering the computational burden and keeping accuracy at a level that is safe to support the decision-making activities of patients. We previously suggested

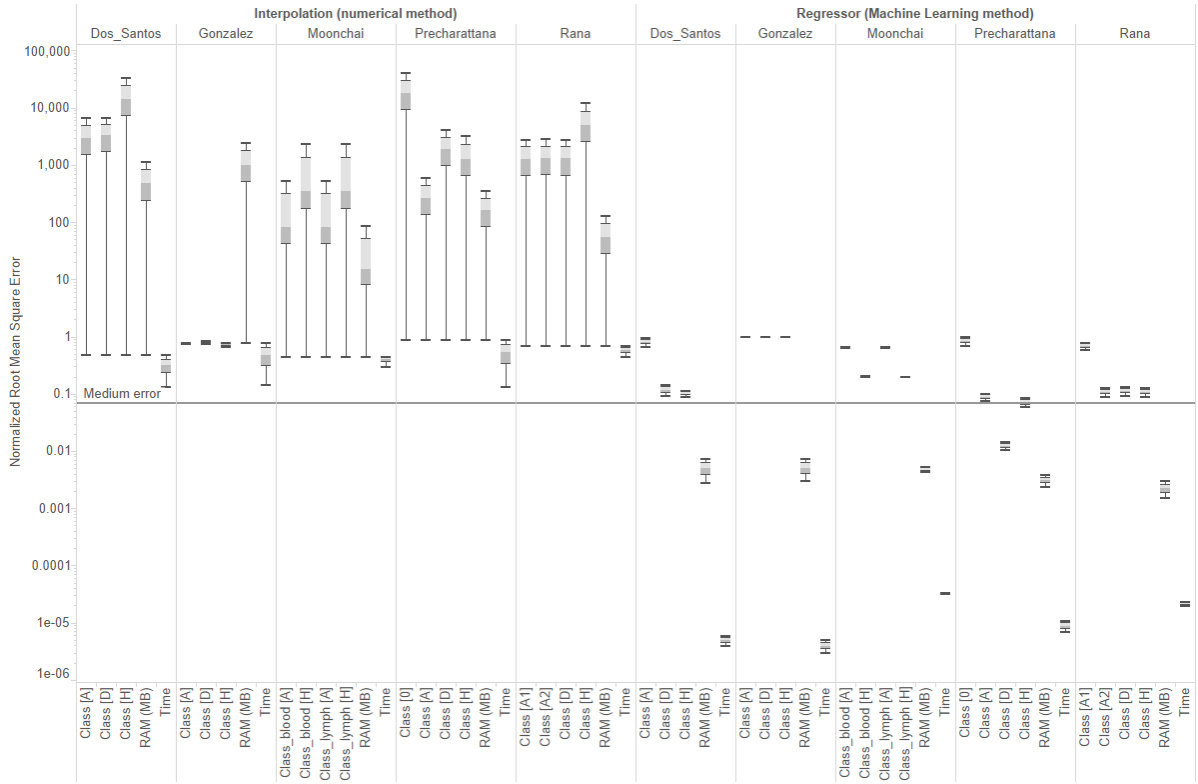


Figure 4: Error across algorithms for the extrapolation dataset. We use a logarithmic scale (y-axis) due to massive errors from interpolation methods. The Lasso method is at a comparable level to the worst results for interpolation but was not displayed as its results are an outlier among machine learning methods.

machine learning as an option (Giabbanelli 2019) to build surrogate models (to approximate the simulations' outpus) and performance models (to estimate the resources necessary to get an exact answer when needed). In this paper, we created the first surrogate and performance models for biological simulations of HIV.

We used five previously developed models, whose implementations were verified and are accessible. For each of the five HIV simulations, we derived surrogate and performance models using both numerical methods (interpolation) and machine learning methods (regressors). We followed a rigorous machine learning process, including a nested cross-fold validation and hyperparameter tuning, and we evaluated the prediction errors on data points similar to training as well as on data points beyond the training range (i.e. extrapolation). Results demonstrate that we *can* predict the outcome of complex and computationally intensive HIV simulations. Such accurate predictions can only be made by machine learning methods: decision trees resulted in low errors for every task and simulation, while support vector machines were always accurate for performance models but made large errors on surrogate models. In contrast, there was no simulation or category or models in which an interpolation method would always produce acceptably small errors. The dominance of decision trees and support vector machines was further demonstrated in the harder evaluation scenario of extrapolations, for instance by estimating the outcomes of larger grids than encountered in the training data. The two machine learning methods still produced accurate performance models and some surrogate models with low to medium levels of errors. In contrast, the best three predictions for interpolation methods had a medium level of errors and the remaining 72 predictions all resulted in large errors. Although our focus was on errors, we note that the *validity* of models is also a concern and an important goal for future work. Indeed, approximating a simulation model by building a metamodel on its output (i.e.,

bottom-up metamodel) may also compromise the model's validity, which may be addressed in part by the new approach of enhanced metamodels used by Gore and colleagues for regressions (Gore et al. 2017).

In conclusion, we demonstrated that machine learning can create an accurate proxy to a simulation model and estimate its computational needs. However, many steps are necessary before this combination of machine learning and simulation can result in products for patients. In particular, the specific characteristics of a patient can be unlike any patient encountered before on at least one aspect. Based on the extrapolation scenario which we examined, predictions for such a patient may be inaccurate in terms of viral dynamics and only accurate to predict computational needs. Improving predictions in the extrapolation scenario is thus needed to handle the real-world diversity of patient cases and hence truly support personalized medicine.

## CONTRIBUTIONS

The project was designed and supervised by PJG, with additional input from VKM. AF generated the training data, applied interpolation methods, and reviewed the literature. BA, CZ, and JEM applied machine learning methods. AF and PJG wrote the manuscript, with input from BA, CZ, and JEM.

## ACKNOWLEDGEMENTS

The authors are indebted to MITACS Canada for supporting this research through the Globalink programme which funds AF while supervised by VKM and PJG.

## REFERENCES

Bergstra, J., and Y. Bengio. 2012. "Random search for hyper-parameter optimization". *Journal of Machine Learning Research* vol. 13 (Feb), pp. 281–305.

Bouhlel, M. A., J. T. Hwang, N. Bartoli, R. Lafage, J. Morlier, and J. R. R. A. Martins. 2019. "A Python surrogate modeling framework with derivatives". *Advances in Engineering Software*, pp. 102662.

Cawley, G. C., and N. L. Talbot. 2010. "On over-fitting in model selection and subsequent selection bias in performance evaluation". *Journal of Machine Learning Research* vol. 11 (Jul), pp. 2079–2107.

Crutzen, R., and P. Giabbanelli. 2014. "Using classifiers to identify binge drinkers based on drinking motives". *Substance use & misuse* vol. 49 (1-2), pp. 110–115.

Crutzen, R., P. J. Giabbanelli, A. Jander, L. Mercken, and H. de Vries. 2015. "Identifying binge drinkers based on parenting dimensions and alcohol-specific parenting practices: building classifiers on adolescent-parent paired data". *BMC public health* vol. 15 (1), pp. 747.

dos Santos, R., and S. Coutinho. 2001. "Dynamics of HIV infection: A cellular automata approach". *Physical review letters* vol. 87 (16).

Friedman, J., T. Hastie, and R. Tibshirani. 2001. *The elements of statistical learning*, Volume 1. Springer series in statistics New York.

Giabbanelli, P. J. 2019. "Solving challenges at the interface of simulation and big data using machine learning". In *2019 Winter Simulation Conference (WSC)*. IEEE.

Giabbanelli, P. J., and M. Baniukiewicz. 2019. "Visual Analytics to Identify Temporal Patterns and Variability in Simulations from Cellular Automata". *ACM T MODEL COMPUT S* vol. 29 (1), pp. 5.

Giabbanelli, P. J., C. Freeman, J. A. Devita, N. Rosso, and Z. L. Brumme. 2019. "Mechanisms for Cell-to-cell and Cell-free Spread of HIV-1 in Cellular Automata Models". In *Proceedings of the 2019 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, pp. 103–114. ACM.

Gligorijević, V., N. Malod-Dognin, and N. Pržulj. 2016. "Integrative methods for analyzing big data in precision medicine". *Proteomics* vol. 16 (5), pp. 741–758.

González, R. et al. 2013. "Dynamics of the HIV infection under antiretroviral therapy: A cellular automata approach". *Physica A: Statistical Mechanics and its Applications* vol. 392 (19), pp. 4701–4716.

Gore, R. J., S. Diallo, C. Lynch, and J. Padilla. 2017. "Augmenting bottom-up metamodels with predicates". *J. Artif. Soc. Soc. Simul.* vol. 20 (1).

Halling-Brown, M., F. Pappalardo, N. Rapin, P. Zhang, D. Alemani, A. Emerson, F. Castiglione, P. Duroux, M. Pennisi, O. Miotto et al. 2010. "ImmunoGrid: towards agent-based simulations of the human immune system at a natural scale". *PHILOS T R SOC A* vol. 368 (1920), pp. 2799–2815.

Herzog, S. A., S. Blaizot, and N. Hens. 2017. "Mathematical models used to inform study design or surveillance systems in infectious diseases: a systematic review". *BMC Infect. Dis.* vol. 17 (1), pp. 775.

Hwang, J. T., and J. R. Martins. 2018. "A fast-prediction surrogate model for large datasets". *Aerospace Science and Technology* vol. 75, pp. 74–87.

Lesko, L. 2007. "Personalized medicine: elusive dream or imminent reality?". *Clinical Pharmacology & Therapeutics* vol. 81 (6), pp. 807–816.

Malakar, P., P. Balaprakash, V. Vishwanath, V. Morozov, and K. Kumaran. 2018. "Benchmarking Machine Learning Methods for Performance Modeling of Scientific Applications". In *Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems*, pp. 33–44. IEEE.

Mehmani, A., S. Chowdhury, J. Zhang, and A. Messac. 2014. "A novel approach to simultaneous selection of surrogate models, constitutive kernels, and hyper-parameter values". In *10th AIAA Multidisciplinary Design Optimization Conference*, pp. 1487.

Moonchai, S., and Y. Lenbury. 2016. "Investigating Combined Drug and Plasma Apheresis Therapy of HIV Infection by Double Compartment Cellular Automata Simulation". *Int J of Computer Theory and Engineering* vol. 8 (3), pp. 190.

Østergård, T., R. L. Jensen, and S. E. Maagaard. 2018. "A comparison of six metamodeling techniques applied to building performance simulations". *Applied Energy* vol. 211, pp. 89–103.

Pappalardo, F., P. Zhang, M. Halling-Brown, K. Basford, A. Scalia, A. Shepherd, D. Moss, S. Motta, and V. Brusic. 2008. "Computational simulations of the immune system for personalized medicine: state of the art and challenges". *Current Pharmacogenomics and Personalized Medicine* vol. 6 (4), pp. 260–271.

Powell, M. J. 1992. "The theory of radial basis function approximation in 1990". *Advances in numerical analysis*, pp. 105–210.

Precharattana, M. 2016. "Stochastic modeling for dynamics of HIV-1 infection using cellular automata: A review". *Journal of bioinformatics and computational biology* vol. 14 (01).

Precharattana, M. et al. 2010. "Investigation of spatial pattern formation involving CD4+ T cells in HIV/AIDS dynamics by a Stochastic cellular automata model". *Int J of Mathematics and Computers in Simulation* vol. 4 (4).

Rana, E. et al. 2015. "Exploring the Relationship Between Adherence to Treatment and Viral Load Through a New Discrete Simulation Model of HIV Infectivity". In *Proc. of the 3rd ACM SIGSIM Conf on Principles of Advanced Discrete Simulation*, SIGSIM PADS '15, pp. 145–156, ACM.

Rosso, N., and P. Giabbanelli. 2018. "Accurately inferring compliance to five major food guidelines through simplified surveys: applying data mining to the UK National Diet and Nutrition Survey". *JMIR public health and surveillance* vol. 4 (2), pp. e56.

Sabine, B., and P. Dietrich. 2017. "Computational simulation as an innovative approach in personalized medicine". *Innovations in Spinal Deformities and Postural Disorders*, pp. 47.

Santhanam, A. P., J. Neylon, J. D. Eldredge, J. Teran, E. P. Dutson, and P. Benharash. 2016. "GPU-Based Parallelized Solver for Large Scale Vascular Blood Flow Modeling and Simulations.". In *Medicine Meets Virtual Reality 22*, pp. 345–351, IOS Press.

Shepard, D. 1968. "A two-dimensional interpolation function for irregularly-spaced data". In *Proceedings of the 1968 23rd ACM national conference*, pp. 517–524. ACM.

Sneha, P., and C. G. P. Doss. 2016. "Molecular dynamics: new frontier in personalized medicine". In *Advances in protein chemistry and structural biology*, Volume 102, pp. 181–224. Elsevier.

Thiagarajan, J. J., R. Anirudh, B. Kailkhura, N. Jain, T. Islam, A. Bhatele, J.-S. Yeom, and T. Gamblin. 2018. "PADDLE: Performance Analysis using a Data-driven Learning Environment". In *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 784–793. IEEE.

Tibshirani, R. 2011. "Regression shrinkage and selection via the lasso: a retrospective". *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* vol. 73 (3), pp. 273–282.

"UNAIDS" 2019. "Global HIV & AIDS statistics – 2019 fact sheet".

Van Steenkiste, T., J. van der Herten, I. Couckuyt, and T. Dhaene. 2019. "Data-Efficient Sensitivity Analysis with Surrogate Modeling". In *Uncertainty Modeling for Engineering Applications*, pp. 55–69. Springer.

Willem, L., F. Verelst, J. Bilcke, N. Hens, and P. Beutels. 2017. "Lessons from a decade of individual-based models for infectious disease transmission: a systematic review (2006-2015)". *BMC Infect. Dis.* vol. 17 (1), pp. 612.

Zhang, Z.-Q., D. W. Notermans, G. Sedgewick, W. Cavert, S. Wietgrefe, M. Zupancic, K. Gebhard, K. Henry, L. Boies, Z. Chen et al. 1998. "Kinetics of CD4+ T cell repopulation of lymphoid tissues after treatment of HIV-1 infection". *Proc Natl Acad Sci USA* vol. 95 (3), pp. 1154–1159.

## AUTHOR BIOGRAPHIES

**ANDREW FISHER**, is a graduate student in the Department of Computer Science at Lakehead University, and a visiting scholar at Miami University thanks to the MITACS Globalink Programme. His work is at the intersection of simulation and machine learning, with application to complex problems including biology and social phenomena. His email address is afisher3@lakeheadu.ca.

**BHISMA ADHIKARI**, is a graduate student in the Department of Computer Science & Software Engineering at Miami University. He is passionate about AI. His email address is adhikab@miamioh.edu.

**CHAO ZHAI**, is an undergraduate student in the Department of Computer Science & Software Engineering at Miami University, with research interests in data science. His email address is zhaic@miamioh.edu.

**JOSHUA E. MORGAN**, is a graduate student in the Department of Computer Science & Software Engineering at Miami University. His email address is morgan42@miamioh.edu.

**VIJAY K. MAGO**, Ph.D., is an Associate Professor in the Department of Computer Science at Lakehead University. His research interests include data mining, modelling and simulation. He serves as associate editors for several journals including BMC Medical Informatics & Decision Making and IEEE Access. His email address is vmago@lakeheadu.ca. His website is https://www.datalab.science/.

**PHILIPPE J. GIABBANELLI**, Ph.D., is an Associate Professor in the Department of Computer Science & Software Engineering at Miami University (USA). His research interests include network science, machine learning, and simulation. He currently serves as track chair for the 2020 Spring Simulation Conference, and program chair for the 2020 ACM SIGSIM Principles of Advanced Discrete Simulations (PADS) conference. His email address is giabbapj@miamioh.edu. His website is https://www.dachb.com.