Proceedings of the 2024 Winter Simulation Conference
H. Lam, E. Azar, D. Batur, S. Gao, W. Xie, S. R. Hunter, and M. D. Rossetti, eds

ADVANCED TUTORIAL ON PARATEMPORAL SIMULATION USING TREE EXPANSION

Bernard Zeigler[1,2], Christian Koertje[3], Cole Zanni[4], Sangwon Yoon[4], and Gerardo Dutan[4]

[1]Dept. of Electrical and Computer Eng., University of Arizona, Tucson, AZ, USA
[2]RTSync Corp. Chandler, AZ, USA
[3]Dept. Physics, University of Massachusetts Amherst, Amherst, MA, USA
[4]Dept. of System Science and Industrial Eng., SUNY Binghamton University, Binghamton, NY, USA

ABSTRACT

Stochastic simulations require large amounts of time to generate enough trajectories to attain statistical significance and estimate desired performance indices with satisfactory accuracy. They require search spaces with deep uncertainty arising from inadequate or incomplete information about the system and the outcomes of interest. Paratemporal methods efficiently explore these large search spaces and offer an avenue for speedup when executed in parallel. However, combinatorial explosion of branching arising from multiple choice points presents a major hurdle that must be overcome to implement such techniques. In this advanced tutorial we show how to tackle this scalability problem by applying a systems theory-based framework covering both conventional and newly developed paratemporal tree expansion algorithms for speeding up discrete event system stochastic simulations while preserving the desired accuracy.

1 INTRODUCTION

Climate change mitigation, communication network design, and command and control decision support are examples of complex problems for which stochastic simulations are needed. They require search spaces with deep uncertainty arising from inadequate or incomplete information about the system and the outcomes of interest (Tolk 2022; Davis 2023; Oren et al. 2023). Consequently, stochastic simulations require large amounts of time to generate enough trajectories to attain statistical significance and estimate desired performance indices with satisfactory accuracy (Choi et al. 2014a**;** Choi et al. 2014b; Amaran et al. 2016**;** Liu 2016**;** Tsattalios et al. 2023). Furthermore, simulation models for system engineering analyses present challenges to today's computational technologies. First, questions addressed at the Systems of Systems (SoS) level require large and detailed models to provide sufficient representation of relevant system-to-system interactions of stochastic nature. Second, they also require multiple executions with multiple random seed state initiations to cover the wide range of configurations necessary to obtain statistically significant measurement of performance outcome distributions.

Parallel execution of simulations offers an avenue for the speedup of complex simulations. Exploitation of such parallelization poses many challenges in today's computational technologies (Park and Fishwick 2010; Xu et al. 2015) thus begging for ways to achieve parallelization in more model-centric ways. Paratemporal and other cloning simulation techniques have been introduced that increase opportunities for reuse of state information and parallelism (Lammers et al. 2009; Yoginath et al. 2019; Li et al. 2017). However, *scalability*, the ability to overcome the combinatorial explosion of branching arising from multiple choice points, presents a major hurdle that must be overcome to implement such techniques.

Nutaro et al. (2024) examined the use of tree expansion methods in lieu of the conventional sampling of outcomes when working with the uncertainty inherent in stochastic simulations. Conventional techniques

simulate a large number of randomly sampled trajectories from start to finish in one-at-a-time fashion. Such trajectories can be viewed as paths from an initial state of the stochastic system (the root of the tree) to one of the terminal states (leaves of the tree) in a manner consistent with random sampling. One advantage of tree construction is that states of the model that have been reached at any point—nodes of the tree—can be cloned for reuse, thus avoiding duplication. Branches in the tree from a state correspond to draws of the random variable whose values determine the subsequent course of the model from that state. In particular, tree expansion with breadth-first traversal can significantly speed up the computation required to generate the same sampling outcomes as the one-at-a-time technique. However, the speedup is limited by the exponential growth of the tree with increasing depth. To achieve scalability, Zeigler (2024) showed that merging of states based on homomorphism concepts can reduce tree growth from exponential to polynomial in depth, thus significantly speeding up computation over that possible with cloning alone.

2 PARATEMPORAL SIMULATION OVERVIEW

2.1 General Background

In this tutorial, we explicate paratemporal methods based on tree expansion with node merging that have been shown to be effective in speeding up stochastic discrete event simulations. In such methods, tree expansion starts with one decision point, the root node, and each decision point (node) thereafter creates new branching possibilities. The root node contains the initial state of the stochastic system. When a new node is generated, it holds the information of the current state of the system, and the probability of the state occurring. The state refers to the set of conditions and variables that define the current status of the simulated systems at a specific point in time. The components of a state include: 1) variables and parameters, which capture the dynamic and state properties of the system; 2) probabilistic information which includes probabilities associated with events or outcomes; and 3) branching information, describing the current path of a simulation. A simulation is executed for one possible branch, until another branching point is reached. This process of tree expansion is repeated until all endpoints are reached when the simulation stops. These endpoints are called leaf nodes. The method uses breadth-first traversal to explore all possible paths at the same level, rather than traversing one path of the tree at a time until an end condition is reached at a leaf node. The explorations can be done in parallel to the extent that available processors allow.

2.2 Binary Tree Expansion Example

We will use a binary tree as an illustration of tree-expansion based paratemporal simulation. A binary tree is structured starting with a root node (initial state) and branching off to at most two children nodes, each branch with an associated probability of being taken in a simulation. As the tree expands in depth, conceptually, all the new (children) leaves are attached to the previous parent leaves in parallel. Later we add an important exception to this step when we consider so-called node merging. As the tree expands in a breadth-first manner, two child nodes are generated concurrently at each parent, rather than descending the tree in a single branching path and then backtracking. The product of the probabilities of the branches along any path in the tree gives the probability of that path being taken in a simulation. As the tree grows deeper, the probabilities get much smaller, as there are more possibilities, thus there is a diminished probability of a particular path being taken.

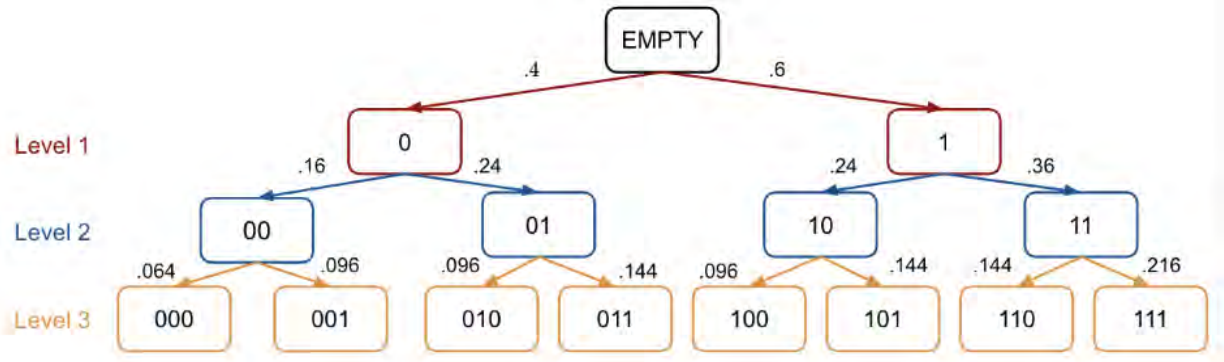

Figure 1: Binary tree illustrating paratemporal methods.

For example, imagine tossing an (unfair) coin three times with tree expanding to represent possible outcomes of each toss. Then the tree in Figure 1 has expanded to depth 3 and has 8 leaves at level 3 that are children of the 4 nodes at level 2. Left and right branches from any node have probabilities .4 and .6 respectively of being taken in a simulation. Nodes are named by the paths taken to reach them from the root (EMPTY). Product probabilities are accumulated as the tree descends. For example, the path 00 has probability 0.16 (.4 * .4) and the extension 000 has probability 0.064 (.16 * .04). Now let each of the leaves have an outcome value associated with it. For example, this could be the number of ones in its name. Then the average of all the outcomes at level 3 in Figure 1 is easily computed to be 1.8 (3 tosses have been made with 0.6 as the probability of each adding a one to the total). In a paratemporal simulation this average is obtained by accumulating all the outcomes weighted by the associated path probabilities.

In contrast, in a typical stochastic simulation, a path down the tree is generated in a depth-first manner, so that for example, three coin tosses are made and the number of ones is obtained as the outcome for that path. This is called a sample of the outcome considered as a random variable. The number of samples taken is determined by the desired statistical significance of the estimate, which is the numerical average of the outcomes produced. So, we see here that the paratemporal tree expansion approach potentially can be both much faster and more accurate than sampling approaches in that the tree yields one (correct) outcome average rather than an estimated average based on a finite number of samples. Since the tree is expanding

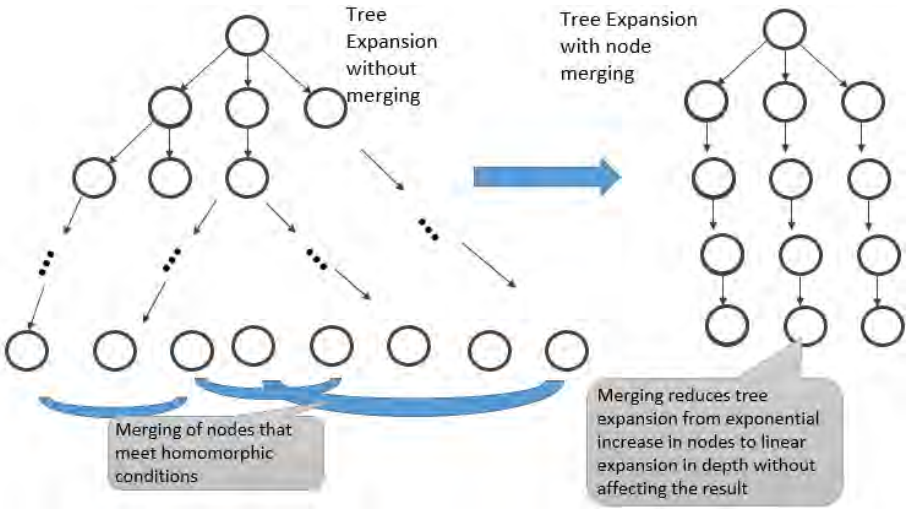

Figure 2: Effect of node merging on paratemporal tree growth.

exponentially, the potential becomes actual if the expansion of the tree can be controlled so that it expands in polynomial time such as constant or linear with depth. Here we explain how node merging can achieve this feat for several examples. Details are available in (Zeigler 2024).

3 HOMOMORPHIC MERGING IN STOCHASTIC SIMULATION TREE EXPANSION

3.1 Motivation

Stochastic simulations require large amounts of time to generate enough trajectories to attain statistical significance and estimate desired performance indices with satisfactory accuracy. In such cases, the computational time for a single model execution may require from a few minutes up to several hours. On the other hand, depending on the problem dimensionality and the irregularity of the response surface, a typical global optimization algorithm may need to evaluate the objective function (and hence *call* the simulation model) thousands of times, in order to converge to a satisfactory solution. Consequently, application of global optimization typically becomes time prohibitive to complex problems such as climate change mitigation, network design, and command and control decision support. Such search spaces are characteristic of problems with deep uncertainty with inadequate or incomplete information about the system and the outcomes of interest (Tolk 2022; Davis 2023). This motivates the need to speed up paratemporal simulations using merging of nodes.

3.2 Homomorphic Merging of Nodes in Tree Expansion

As illustrated on the left side of Figure 2, conventional generation of trajectories can be viewed as traversing paths from an initial state of the stochastic system (the root of the tree) to one of the terminal states, leaves of the tree, in a manner consistent with random sampling. The advantage of tree construction is that states of the model that have been reached at any point – nodes of the tree - can be cloned for reuse thus avoiding duplication. Branches in the tree from a state correspond to draws of the random variable whose values determine the subsequent course of the model from that state.

In particular, tree expansion with breadth first traversal can significantly speed up the computation required to generate the same sampling outcomes as the one-at-a-time technique (Nutaro et al. 2024). However, speedup is limited by the exponential growth of the tree as its depth extends. Zeigler (2024) introduced merging of states based on homomorphism concepts to mitigate against such growth. As illustrated on the right-hand side of Figure 2, we will show by examples that such merging can reduce tree growth from exponential to polynomial in depth thus significantly speeding up computation over that possible without merging.

4 EXAMPLE: TREE EXPANSION FOR CHEMICAL REACTION NETWORKS

4.1 Review of Chemical Reaction Modeling and Simulation

Biochemical reactions underlie all biological activities among micro-organisms such as communication among members of a bacterial population and synchronization of toxin release to attack another population. Networks of such reactions form well-known metabolic pathways within bacterial cells, but also constitute less well-known media for interaction among cells and their internal and external environments. Such reaction networks are composed of large numbers of different molecules in relatively small amounts or

concentrations that require stochastic modeling and simulation approaches with the attendant computational challenges of time and accuracy generally described above (Roy 2019).

4.2 The Gillespie Algorithm

In this tutorial we consider that conventional approaches to simulating cellular biochemical reactions are based on the Gillespie algorithm, and we show how this algorithm can be formulated within the paratemporal framework to achieve faster and more accurate portraits of resulting behavior. The Gillespie algorithm is a relatively simple framework for model construction that can be expressed within the DEVS modeling formalism (Zeigler et al. 2013; Zeigler et al. 2018). In the spirit of a tutorial, we limit the discussion to a relatively simple example (one discussed in the above referenced Wikipedia entry) and leave details to further consideration available in other literature. The Gillespie algorithm, given a set of reactions and their rate equations, generates statistically correct trajectories of the numbers of reagents. The algorithm can be considered as a variant of dynamic Monte Carlo methods and similar to the kinetic Monte Carlo methods (Gillespie 1977).

4.3 Simple Reaction Example

In the simple reaction shown in Figure 3, A and B react reversibly to form an AB dimer, and the AB dimer dissociates into A and B. In the forward reaction, the reaction rate constant for a given single A molecule reacting with a given single B molecule is k_D. In the backward direction, the reaction rate for an AB dimer breaking up is k_B. When considering the reaction involving many molecules, the rate of dimer formation is $k_D * n_A * n_B$. where there are n_A and n_B molecules of type A and B respectively. The rate of dimer disassociation is $k_B * n_{AB}$ where there are n_{AB} molecules of type AB.

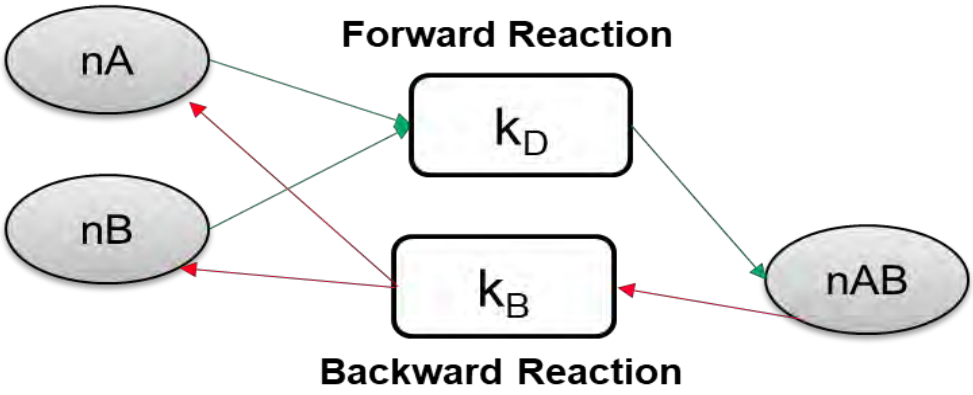

Figure 3: Example chemical reaction with forward and backward reactions.

Rather than considering each reaction as occurring at its own rate, the Gillespie algorithm assumes that the total reaction rate, $R_{TOT} = k_D * n_A * n_B + k_B * n_{AB}$, is the rate at which the reaction evolves. Thus, to advance the simulation, the time of the next event is selected from an exponential distribution with mean, $1/R_{TOT}$. Appendix 1 discusses how a DEVS model formulation elucidates the different possibilities for modeling such a reaction, of which Gillespie is one. Having decided how to compute the advance of time, the algorithm decides which individual reaction will occur at that time using probabilities which are the ratios of the individual rates to the total rate. For the forward choice

$$P(A + B \rightarrow AB) = (k_D * n_A * n_B)/R_{TOT},$$

and for the backward choice

$$P(AB \rightarrow A + B) = k_B * n_{AB}/R_{TOT} = 1 - P(A + B \rightarrow AB).$$

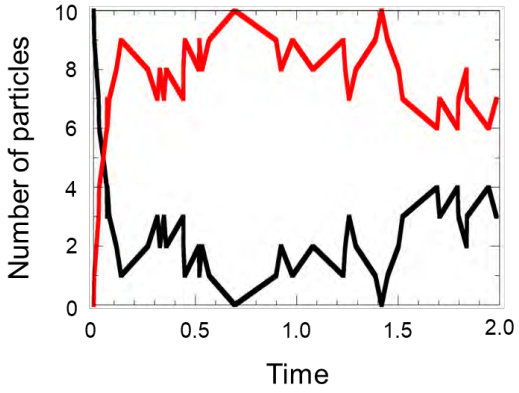

Figure 4: Time evolution of the reaction in Figure 3 of the number of A molecules (black) and AB dimers (red) (this figure is adapted from Wikipedia).

Figure 4, taken from the Gillespie algorithm Wikipedia article, plots a sample trajectory of the Gillespie algorithm where the number of A molecules (black curve) and AB dimers as a function of time starting with $n_A = n_B = 10$ molecules at time $t = 0$. Since every reaction event affects the A and B molecule type in the same way, the number of B molecules is always equal to the number of A molecules and so n_B is not shown in the figure. We see that the number of A (and B) molecules tends to decrease with time trending toward fluctuating around a level of two; while the number of dimers increases and fluctuates around 8. In this simple case, the equilibrium state can be easily calculated using conservation of matter to reduce the number of variables and setting the forward and backward rates to be equal. This results in a quadratic equation confirming that $n_A = 2$ and $n_{AB} = 8$. Due to the small numbers of molecules under study, fluctuations around these values are relatively large. The Gillespie algorithm is used to study systems where these fluctuations are important.

5 PARATEMPORAL TREE EXPANSION IMPLEMENTATION

Figure 5 illustrates how the paratemporal tree expansion works in the case of the Gillespie Algorithm for the Dimer reaction. Each node represents a state of the reaction viz., the number of A types and number of AB types (as before the number of B types is redundant). The root node starts the assumed initial state of the reactants. The tree is binary since there are two branches corresponding to the forward and backward reactions. Accordingly, each branch represents one of these reactions with the associated probability and its effect on the state of the reaction. As illustrated in the tree expansion of Figure 1, as the new branches are created, we keep track of the accumulated probabilities associated with the paths leading to the nodes being added. In this application, the depth of the tree corresponds to the number of reactions that have occurred. Figure 6 plots features of the probability distributions at each depth up to 15 for the two reactants including their averages, minima, and maxima. The standard deviation of the distribution for the A type at each depth is plotted in Figure 6 and its relatively small value suggests that the distribution stays closely clustered around the average as time advances.

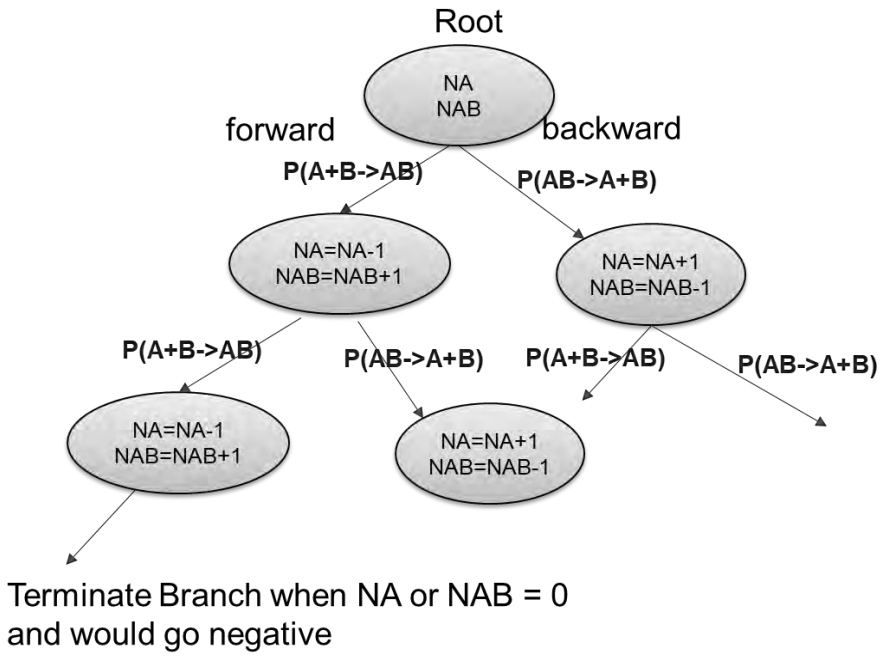

Figure 5: Tree expansion for paratemporal simulation of reaction in Figure 2.

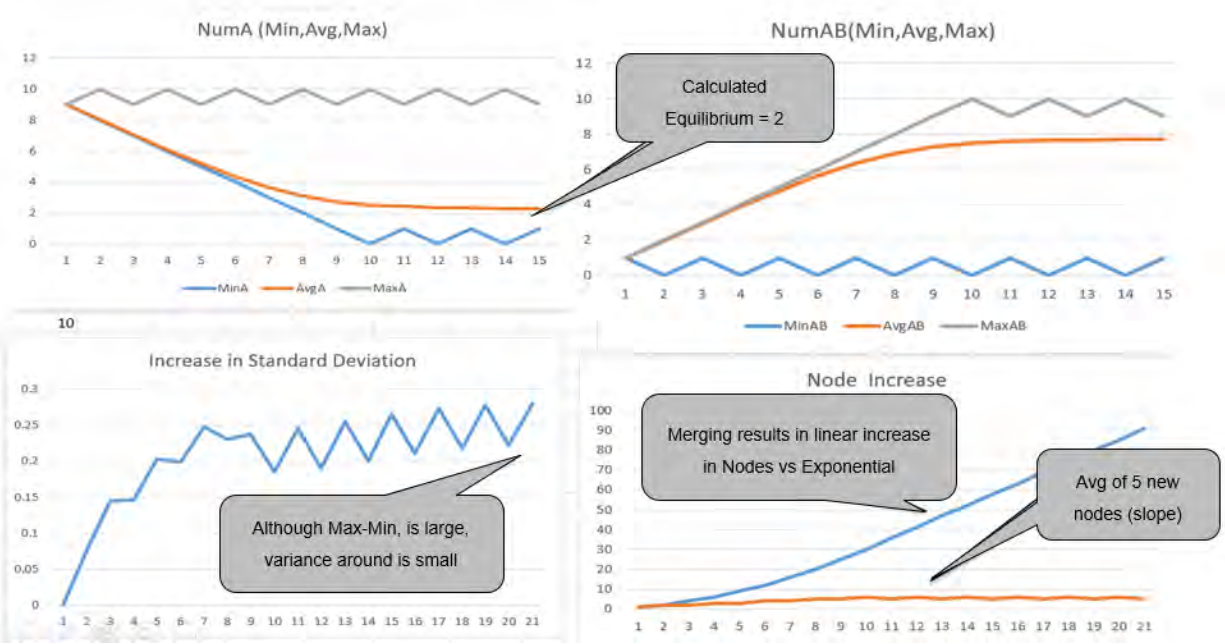

Figure 6: Plots of time course of reaction of Figure 2 generated by paratemporal tree expansion method.

5.1 Merging Nodes to Speedup Execution

As explained above the tree expands exponentially so that successive expansions take longer and longer to generate and eventually memory is exceeded forcing the algorithm to crash – typically this occurs at around depth 22 for the memory available in today's personal computers. However, we can take advantage of node equivalence based on state equality. The state of the chemical reaction system is characterized entirely by

the quantity of A molecules and AB molecules at that stage. Two nodes in the tree at the same level can then be merged if they have the same numbers of A types and AB types irrespective of the history of that node. For instance, a forward reaction followed by a backward reaction will generate the same state if instead a backward reaction were followed by a forward reaction. We can apply merging at each level of tree expansion (breadth first) so that the number of new nodes added is greatly reduced. When two nodes are merged, their path probabilities are added so that the effect on the outcome computation is unchanged. The benefit of such merging is that it drastically limits the tree expansion, changing it from exponential to linear growth (see Figure 6). This greatly reduces the computation time and allows practically unlimited depths to be achieved. Recall that as illustrated in Figure 6, a full depiction of the distribution of reactant numbers is obtained at each depth by this approach as opposed to sampled trajectory estimates obtained in conventional approaches.

6 EXAMPLE: BASEBALL SIMULATION

6.1 Background

Another illustration of Paratemporal tree expansion is to predict baseball game outcomes based on player batting averages. Rather than 0s and 1s being the branching possibilities in the binary tree example, the branches will represent the possible outcomes from each at-bat. These outcomes are an out, single, double, triple, home run, and walk/hit-by-pitch. The walk and hit-by-pitch are grouped together because the advancements of the base runners are the same for both. Note that we will focus only on a batter's appearance at the plate and the configuration of the bases during the progress of the game, while we aggregate all other details into batter average statistics. This (large) simplification allows us to use real player data from Major League Baseball teams to determine the model's probability of the 6 different outcomes for each player. Appendix B gives more details on the implementation of the methodology applied to this example.

6.2 Objective and Results

The objective is to compute the outcomes for all 9! different lineups of the same players to find the top 10% performing ones and assess how much such arrangements matter by comparing the best and worst runs produced. Nine distinct players complete a full baseball lineup and play in the lineup specified order. The depth that the tree traverses will be representative of the batting characteristics of the player in the lineup. The state of each node will encompass the type of hit, current batter's name, number of outs, number of runs, base configuration, probability of current player's hit outcomes, and probability of the branching occurring (product of probabilities). The logic of the tree expansion follows a basic baseball game. A base runner can only advance the number of bases that is achieved for the given hit. For example, if a double is hit, all base runners will advance two bases. The only exception is a walk/hit-by-pitch, in which the batter will then occupy first base. Runners will only advance if the previous base is occupied, forcing them to take the next base. Runs are calculated based on the base configuration and the hit type. When a base runner reaches home, a run is added to the team's total score. Outs are also tracked throughout the simulation. For the baseball game, the equivalence of two nodes is characterized by the base configuration, the number of runs, and the number of outs. For example, if the first two batters hit singles, then they occupy the first and second bases. This is equivalent to the scenario that the first batter hits a double and the second is walked. The merging of nodes in this system is what makes it feasible to explore all 9! batting configurations in a reasonable time.

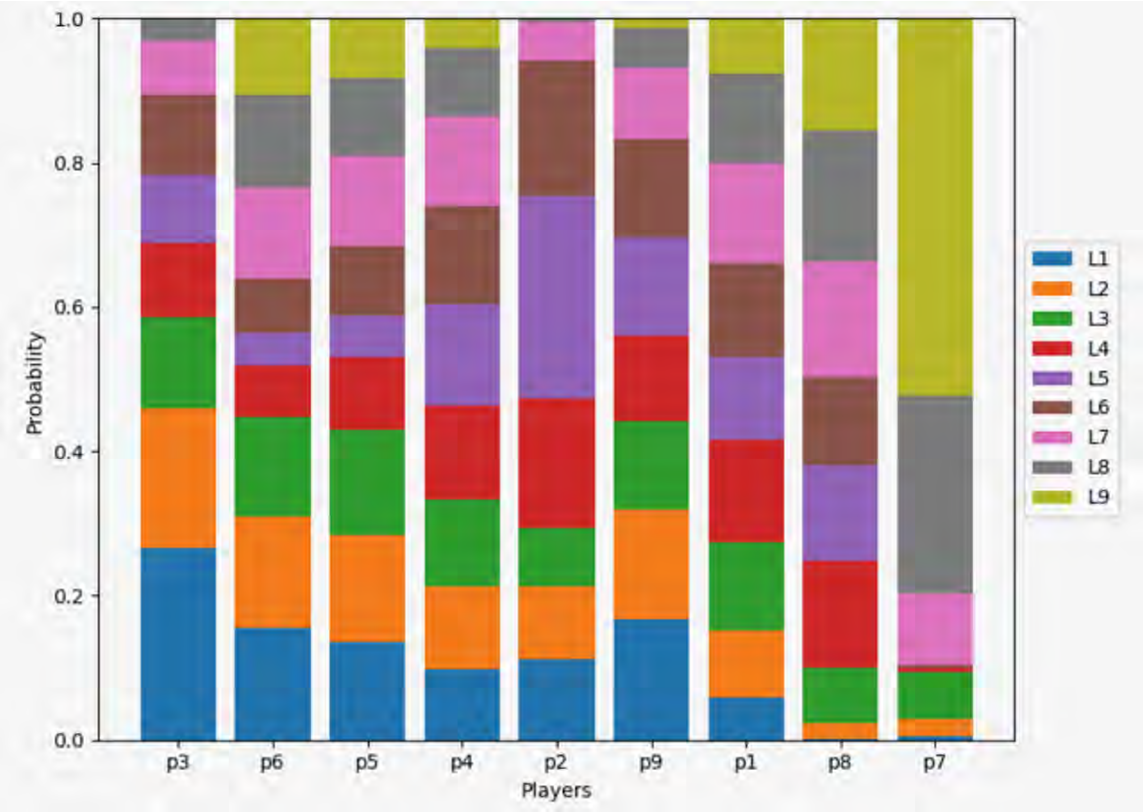

Figure 7: Showing for each player and lineup position pair, the relative number of times that pair occurs in the top 10% performing lineups.

Each lineup is evaluated by expanding a tree to represent enough rotations typically seen in any given baseball game. Figure 7 visualizes the strategy derived for player assignment: prefer to assign players to the position they most frequently occupy in the top decile (shown by the longest patch of the associated color in each column). For example, player p3 is assigned to be first in the lineup (L1) because s/he is found most frequently in the place in the top performing lineups. More useful insights are gleaned from these results but space does not allow going into them here. The full generation and analysis of all 9! lineups is intractable to conventional approaches in that achieving sufficient statistical significance requires sampling millions of simulated baseball games requiring years of computation to perform. The developed paratemporal simulation approach obtained results in under 10 hours.

6.3 Application to Serial Compositions

Many stochastic systems attribute randomness to a continuous variable such as time. Series-Parallel compositions are examples of stochastic models composed of distinct and independent processes that unfold either one after another (in series) or at the same time (in parallel). The probability of successful completion of tasks by such compositions can be evaluated using simulation where the behaviors of the individual systems can be generated and their couplings specified. In Appendix C we discuss an example implementation of systems entirely composed of serial processes where total system success is determined only by all processes succeeding one after another. The discussion illustrates how tree expansion can be applied to such problems, how the approach can be much faster than conventional sampling of trajectories, and how accuracy and computation tradeoffs can be evaluated.

7 CONCLUSIONS

Paratemporal methods based on tree expansion have proven to be effective in efficiently generating the trajectories of stochastic systems. However, combinatorial explosion of branching arising from multiple choice points presents a major hurdle that must be overcome to implement such techniques. Zeigler (2024) tackled this scalability problem by developing a systems theory-based framework covering both conventional and proposed tree expansion algorithms for speeding up discrete event system stochastic simulations while preserving the desired accuracy. This tutorial presented this framework in an informal way and illustrated it with several example applications. We introduced the concept of tree expansion for stochastic simulation with coin tossing followed by an application to a simple formulation of chemical reaction simulation. We then overviewed applications to baseball simulation and to a general class of serial compositions to estimate probability of success of multistage processes. These examples show how merging of nodes in tree expansion is needed to overcome exponential tree branching and can be very effective in producing fast running generation of accurate outcome summaries for stochastic search spaces.

APPENDIX A DEVS FORMULATION OF DIMER REACTION

To formulate DEVS models for the dimer reaction we start with a basic state vector of the form:
(number of As, number of Bs, number of ABs, forward phase, backward phase)

where the forward and backward phases can be {passive, active}.
Then once one or the other (or both) of the reactions reach the active phase, the transition can be defined which moves the appropriate numbers of molecules and returns to the passive phase,

$$\delta(n_A, n_B, n_{AB}, active, passive) = (n_A - 1, n_B - 1, n_{AB} + 1, passive, passive),$$
$$\delta(n_A, n_B, n_{AB}, passive, active) = (n_A + 1, n_B + 1, n_{AB} - 1, passive, passive).$$

The time advance to return to the passive phase could be 0,

$$ta(n_A, n_B, n_{AB}, active, passive) = 0,$$
$$ta(n_A, n_B, n_{AB}, passive, active) = 0.$$

The difference in potential model formulations stems from the mechanism employed to go from passive to active phases: Do we choose forward activation,

$$\delta(n_A, n_B, n_{AB}, passive, passive) = (n_A, n_B, n_{AB}, active, passive),$$

backward activation,

$$\delta(n_A, n_B, n_{AB}, passive, passive) = (n_A, n_B, n_{AB}, passive, active),$$

or both,

$$\delta(n_A, n_B, n_{AB}, passive, passive) = (n_A, n_B, n_{AB}, active, active).$$

Also, the time advance should be associated with such choices

$$ta(n_A, n_B, n_{AB}, passive, passive) = ?$$

The Gillespie algorithm chooses randomly based on computed probabilities which reaction to activate and samples the time advance from an exponential distribution based on the summed reaction rates.

The simpler version of this algorithm used in the tree expansion illustrated above sets the time advance to the mean time of the just mentioned distribution, namely the inverse of the summed reaction rates.

Another choice is to use exponential distributions based on the individual reaction rates of each reaction to sample the time advances of each separately and select the minimum as the time advance. In this case, either or both of the reactions could be activated.

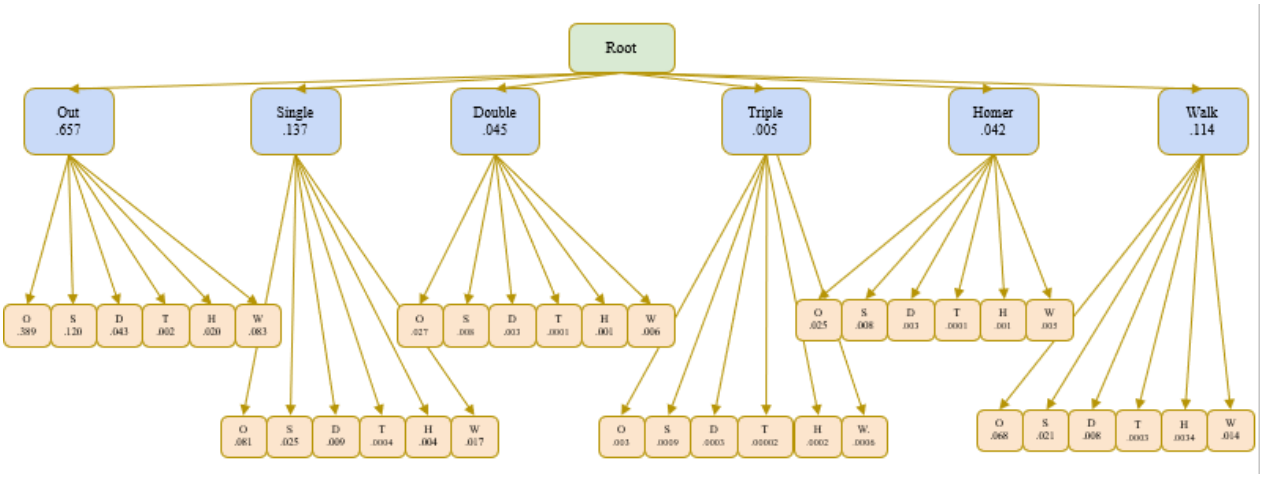

Figure A1: Tree Expansion for Baseball Simulation.

APPENDIX B BASEBALL APPLICATION AND SIMULATION ARCHITECTURE

The tree expansion methodology was applied to simulate baseball outcomes. New nodes were generated with the different outcomes of a baseball at-bat. This includes 6 outcomes: out, single, double, triple, homerun, and walk. Figure A1 illustrates how the tree expands with each consecutive batter. The levels of the expansion are representative of the batters within the lineup. Each node carries the current state of the game, which includes the number of outs, runs, current base configuration, player batting, probability of players outcome occurring, and cumulative product of the entire branches' probabilities. The figure shows the simulation of two batter outcomes, with the first batter being the 1st level (blue nodes) and the second level being the second batter (orange nodes). These probabilities are used to calculate the probability of each branch, or situation, occurring. For example, if the results of the first and second at-bats were both outs, the resulting probability of those two outcomes occurring would be 0.657×0.593. The terminal nodes, or the final nodes of a branching sequence, are used to calculate the main performance metric. The probability and runs of each terminal node are summed together to get the expected runs of the simulated batters in the lineup order. Figure A1 also shows how fast the expansion grows given the 6 outcomes from each node. Homomorphic merging is used to limit this branching by merging nodes that have equivalent game states. This equivalence requires that merged nodes have the same number of runs, outs, and base configurations. The probabilities of the two merged nodes are summed together to represent the probability of the retained node.

The architecture of the simulator is illustrated in FigureA2. The framework is described by an upper level run control and a lower level run control. The upper level run control is comprised of a series of methods that are used to control the expansion of the tree and ensure that new nodes are generated from the most current state of the game. Each new node is created as an instance of a new at-bat, which is simulated

through the methods of the lower level run control. The lower level run control takes the state inputs from the upper level run control and simulated on instance of an at bat to send back to the upper level run control. This frame work is generalized to be used for many different problem scenarios.

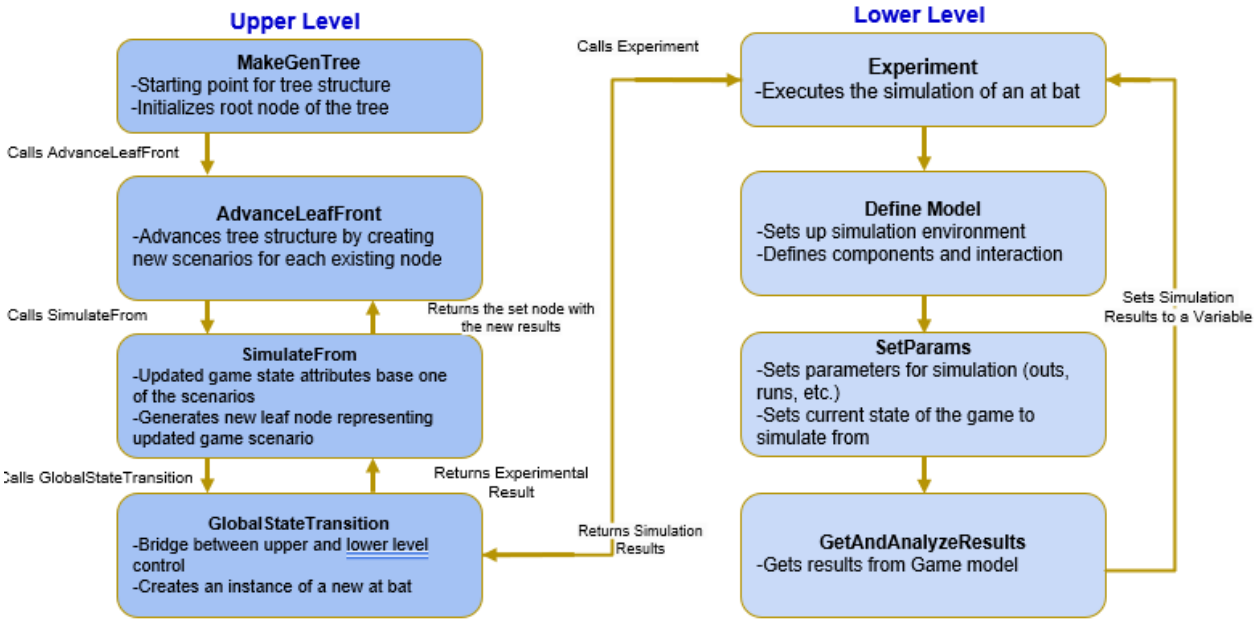

Figure A2: Architecture of Simulator.

APPENDIX C PARATEMPORAL APPROACH TO SERIES-PARALLEL COMPOSITIONS

A large class of stochastic models are characterized by distinct and independent processes that unfold either one after another (in series) or at the same time (in parallel). Here we discuss an example implementation of systems entirely composed of serial processes where total system success is determined only by all processes succeeding one after another. The general process is summarized in Figure A3. We model each

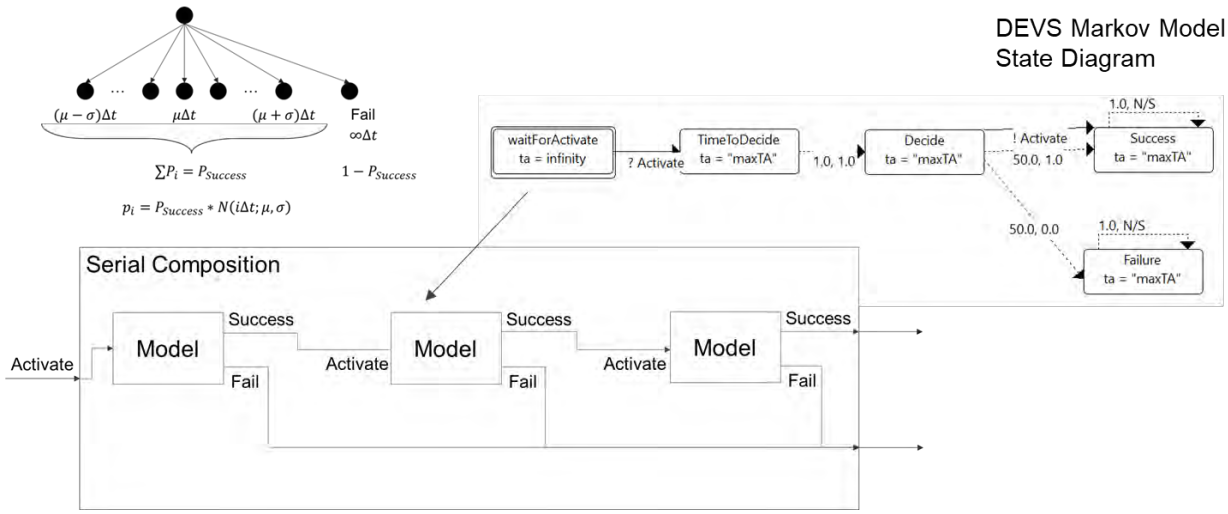

Figure A3:Tree Expansion and models for Serial Composition.

individual process using DEVS Markov process designed in MS4 Me (Zeigler et al. 2013; Zeigler et al. 2018) such that there is a discrete probability associated with success and with failure of the process simulated. We are also able to assign a temporal distribution for modeling the duration it takes each process. In the example modeled below, we choose to ascribe a normal distribution to each process with mean μ and standard deviation σ. In general, these parameters are different for each process. In the upper left portion of Figure A3 we show how the tree expands for each process. Each process expands based on its temporal duration centered on the mean. Shown, we branch out in each direction with 1σ although depending on the interest of the simulation this can be modified. The number of branches is a simulation parameter that can be chosen before execution and is related to the accuracy of the resulting distributions. The probability assigned to each branch is the probability of success weighted by the probability calculated from the normal distribution for a discrete interval of time. Increasing the number of branches (or 'granules') decreases the interval used for calculation. Equivalence between two nodes is determined by the total time it takes in the model to reach that point. This can only be done approximately by establishing a tolerance value. Two nodes merge if $|t_1 - t_2| < \epsilon$ where ϵ is the tolerance. Of course the tolerance can have an impact on the accuracy of the simulation and its choice will depend on the requirements of the model.

Comparing the tree expansion execution times with a version of sampling shows the speed up that we can achieve. To compare equivalent simulation, we look at how long it takes to simulate the same number of distinct trajectories. Figure A4 shows that by sampling, the time to execute increases exponentially whereas tree expansion increases linearly. The linear expansion in this model is due to node merging at each level. The overall system state at each level in the tree (corresponding to each process in the model) only changes by an addition of time to the total model duration. Therefore, we observe lots of overlap between nodes with similar total duration. The total speed up at 250 distinct trajectories is on the order of 10^4.

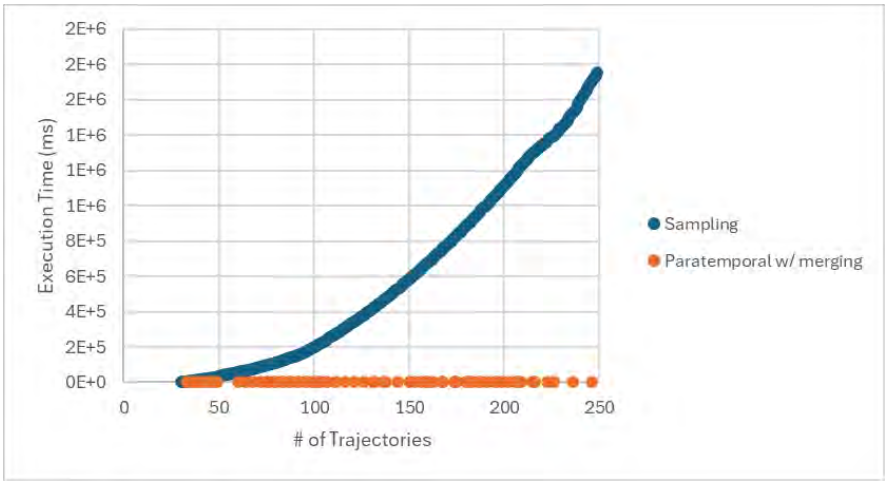

Figure A4: Comparison of execution time between traditional sampling and paratemporal methods with merging.

Approximating a continuous variable in a discrete manner will induce some error. As mentioned in the series model, the number of branches used in the tree decreases the width of time approximated. Figure A5 shows how increasing the number of branches/granules decreases the observed error. The error is calculated

by the absolute difference between the expected duration (which is the sum of the means at each level) to the observed average duration.

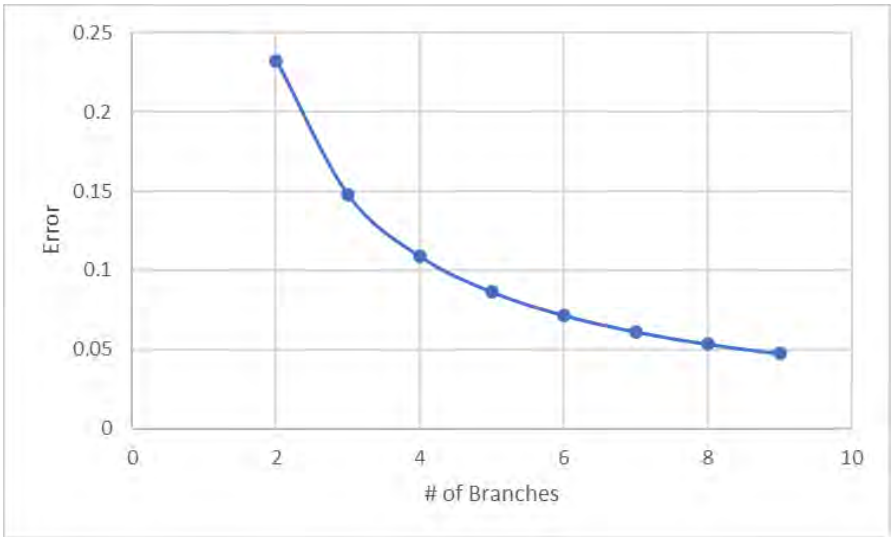

Figure A5: Decrease in error with increase in branches.

REFERENCES

Amaran, S., N. V. Sahinidis, B. Sharda, and S. J. Bury. 2016. "Simulation Optimization: A Review of Algorithms and Applications." *Annals of Operations Research* 240:351–380.

Choi, C., K.-M. Seo, and T. G. Kim. 2014. "DEXSim: An Experimental Environment for Distributed Execution of Replicated Simulators Using a Concept of Single Simulation Multiple Scenarios." *Simulation* 90:355–376.

Choi, C., K.-M. Seo, and T. G. Kim. 2017. "Accelerated Simulation of Discrete Event Dynamic Systems via a Multi-Fidelity Modeling Framework." *Applied Sciences* 7:1056.

Davis, P. K. 2023. "Broad and Selectively Deep: An MRMPM Paradigm for Supporting Analysis." *Information* 14:134.

Gillespie, D. T. 1977. "Exact Stochastic Simulation of Coupled Chemical Reactions." *The Journal of Physical Chemistry* 81(25):2340–2361.

Lammers, C., J. Steinman, M. Valinski, and K. Roth. 2009. "Five-Dimensional Simulation for Advanced Decision Making." In *Proceedings of the SPIE Defense, Security, and Sensing*, April 13th–17th, Orlando, USA, 73480H.

Li, X., W. Cai, and S. J. Turner. 2017. "Cloning Agent-Based Simulation." *ACM Transactions on Modeling and Computer Simulation* 27:15.

Liu, B., S. Koziel, and Q. Zhang. 2016. "A Multi-Fidelity Surrogate-Model-Assisted Evolutionary Algorithm for Computationally Expensive Optimization Problems." *Journal of Computational Science* 12:28–37.

Marchau, V. A. W. J., W. E. Walker, P. J. T. M. Bloemen, and S. W. Popper. 2019. *Decision Making under Deep Uncertainty: From Theory to Practice.* Cham: Springer Nature.

Nutaro, J., B. P. Zeigler, S. Yoginath, C. Zanni, S. Seal, P. Shukla, and C. Koertje. 2024. "Using Simulation Cloning to Sample without Duplication." Working Paper, Oak Ridge National Laboratory, Oak Ridge, TN, USA.

Ören, T., B. P. Zeigler, and A. Tolk. 2023. *Body of Knowledge for Modeling and Simulation: A Handbook by the Society for Modeling and Simulation International.* Berlin/Heidelberg: Springer.

Park, H. and P. A. Fishwick. 2010. "A GPU-Based Application Framework Supporting Fast Discrete-Event Simulation." *Simulation* 86:613–628.

Roy, S., M. A. Islam, D. Barua, and S. K. Das. 2019. "A Scalable Parallel Framework for Multicellular Communication in Bacterial Quorum Sensing." In *Bio-inspired Information and Communication Technologies*, edited by A. Compagnoni, W. Casey, Y. Cai, and B. Mishra. 181-194, Cham: Springer.

Tolk, A. 2022. "Simulation-Based Optimization: Implications of Complex Adaptive Systems and Deep Uncertainty." *Information* 13:469.

Tsattalios, S., T. Ioannis, D. Panagiotis, K. Panagiotis, E. Andreas, and M. Christos. 2023. "Advancing Surrogate-Based Optimization of Time-Expensive Environmental Problems through Adaptive Multi-Model Search." *Environmental Modelling & Software* 162:105639.

Yoginath, S. B., M. Alam, and K. S. Perumalla. 2019. "Energy Conservation Through Cloned Execution of Simulations." In *Proceedings of the 2019 Winter Simulation Conference (WSC)*, December 8th–11th, National Harbor, USA, 2572–2582.

Xu, J., E. Huang, C.-H. Chen, and L. H. Lee. 2015. "Simulation Optimization: A Review and Exploration in the New Era of Cloud Computing and Big Data." *Asia-Pacific Journal of Operational Research* 32:1550019.

Zeigler, B. P. 2024. "Discrete Event Systems Theory for Fast Stochastic Simulation via Tree Expansion." *Systems* 12:80. https://doi.org/10.3390/systems12030080.

Zeigler, B. P., A. Muzy, and E. Kofman. 2018. *Theory of Modeling and Simulation: Discrete Event Iterative System Computational Foundations.* New York: Academic Press.

Zeigler, B. P., C. Seo, and D. Kim. 2013. "DEVS Modeling and Simulation Methodology with MS4 Me Software." In *Proceedings of the Symposium on Theory of Modeling & Simulation—DEVS Integrative M&S Symposium*, April 7th–10th, San Diego, USA.

AUTHOR BIOGRAPHY

BERNARD P. ZEIGLER is Professor Emeritus of Electrical and Computer Engineering at the University of Arizona. He is Fellow of The Society for Modeling and Simulation International (SCS) where he served as President (2002-2004) as well as in other positions. (zeigler@ece.arizona.edu)

CHRISTIAN KOERTJE is a Ph.D. student at the University of Massachusetts Amherst Department of Physics. He is working along with RTSync Corp. in developing complex systems modeling practices and improved simulation algorithms with the goal of making them more commercially available. (ckoertje@umass.edu)

COLE ZANNI is a recent graduate student at Binghamton University, SUNY, specializing in Industrial and Systems Engineering. With a strong academic background and a Bachelor's degree in Industrial and Systems Engineering, his research focuses on modeling and simulation, where he explores innovative solutions to complex problems in military strategy and sports analytics. (cole.zanni1@gmail.com)

SANGWON YOON is a professor in the Industrial Systems Engineering Department at Binghamton University. His research interests are focused in decentralized decision modeling. (yoons@binghamton.edu)

GERARDO DUTAN is a Ph.D student at Binghamton University, SUNY, specializing in Industrial and Systems Engineering. He is working with Dr. Yoon with research in Stochastic System Optimization. (dutangerardo@gmail.com)