Proceedings of the 2017 Winter Simulation Conference
W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, eds.

ANALYZING AND SIMPLIFYING MODEL UNCERTAINTY IN FUZZY COGNITIVE MAPS

Eric A. Lavin
Philippe J. Giabbanelli

Department of Computer Science
Northern Illinois University
1425 W. Lincoln Hwy
DeKalb, IL 60115, USA

ABSTRACT

Fuzzy Cognitive Mapping (FCM) represents the 'mental model' of individuals as a causal network equipped with an inference engine. As individuals may disagree or evidence be insufficient, causal links may be assigned a range rather than one value. When all links have range, the massive search space is a challenge to running simulations. In this paper, we presented, implemented, and evaluated a new approach to identify which ranges are important and simplify models accordingly. Our approach uses a factorial design of experiments, implemented using parallelism to offset its high computational cost. Our implementation (including our new Python library for FCM) is freely available on a third-party repository. Our evaluation on three previously published models shows that our approach can simplify almost half of a model under common settings, and runs within seconds on entry-level hardware for small FCMs. Further research is needed on simplifying the few FCMs having many links.

1 INTRODUCTION

Fuzzy Cognitive Mapping (FCM) is a modeling method that can represent the 'mental model' of individuals by articulating different factors and the dynamics of their interactions. Intuitively, an FCM can be seen as a causal network equipped with an inference engine. While FCM dates back to the late 80s (Kosko 1986) and has been used in a variety of fields (Papageorgiou and Salmeron 2013), it is increasingly popular in participatory modeling specifically. Indeed, modern platforms (e.g., `MentalModeler.com`) now allow individuals to asynchronously and collaboratively develop simulation models by sharing their knowledge into an accessible and standardized format (Gray et al. 2013, Giabbanelli and Crutzen 2014). This is particularly used in ecological modeling, where FCMs allow us to synthesize the different perspectives of the many participating stakeholders (Nyaki et al. 2014, Gray et al. 2015, Douglas et al. 2016), and in health where complex problems may require a large number of experts for each sub-domain (Giabbanelli, Torsney-Weir, and Mago 2012). While FCMs share broadly similar constructs with the modeling technique of System Dynamics (capturing factors and dynamic interactions, use in participatory settings), FCMs are specifically designed to handle situations with uncertainty and vagueness. By building on Fuzzy Logic, FCMs can be seen as a formal tool to manage the imprecision found in real-world problems. For example, participants would assess the strength of a causal link in the FCM using linguistic terms (e.g. 'very high', 'medium') which correspond to fuzzy membership functions. Fuzzy logic would transform their answers into one number for this causal link, which the inference engine uses when updating factors.

However, in real-world problems, we may not be able to assign only one number to a causal link. This may be due to conflicting evidence, a lack of agreement between participants, or the vagueness of what a link may represent. For example, in our FCM of obesity, we found that experts' assessments on one third of the FCM's links had significant variations (Giabbanelli, Torsney-Weir, and Mago 2012). Salmeron extended the formalism of FCMs into Fuzzy Grey Cognitive Maps (FGCMs) by giving a range to all links rather than one number (Salmeron 2010) (Figure 1). While this provides a vehicle to represent vagueness, it also creates a much larger search space in which to run the model: which value should be used for each link in one simulation run? This problem may be avoided altogether by systematically taking the mid-point of the range (Salmeron 2010), but this solution has two issues. First, it brings us back to a normal FCM, thus defeating the purpose of capturing vagueness as a range to start with. Second, this simplification may ignore a lot of the search space. That is, the conclusions made by running the model this way may not be representative of what would have been obtained by running it with the other values included in each link's range. While this simplification can thus lead to erroneous conclusions, the other extreme of running the model for all possible combinations of values would just be infeasible. In this paper, we analyze how to simplify an FGCM while having a minimal impact on limiting the model's outputs.

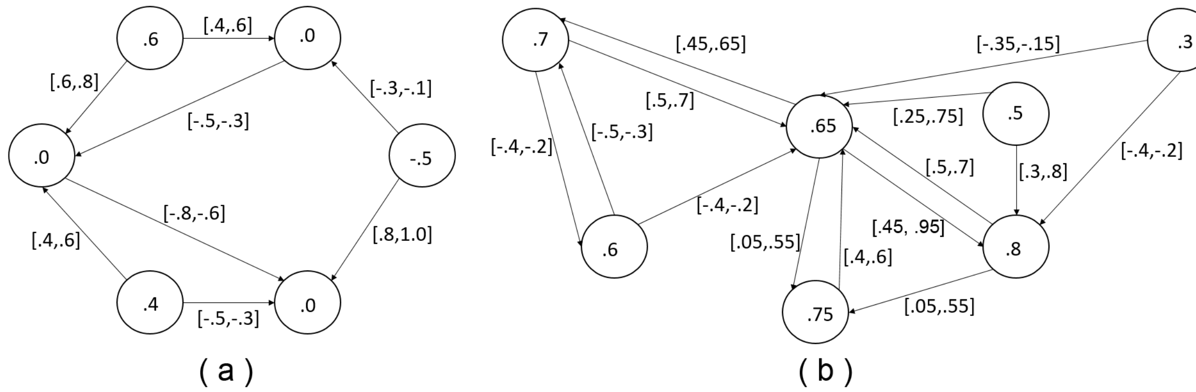

Figure 1: Fuzzy Grey Cognitive Maps for a security system (Salmeron 2015) (a) and a supervisor in radiotherapy treatment planning (Salmeron and Papageorgiou 2012) (b). Nodes show initial values.

Design of Experiments (DoE) has long been used in the field of simulation to address questions of that type (Jain 1990). For example, a 2^k factorial design would inform modelers about how much of their model's variance is due to single parameters, or combinations of these parameters. Parameters that contribute little to the variance (either directly or in combination with others) can then be simplified by being set to one value. In our case, each link is a parameter, and the goal is to identify the links whose range can be replaced by a single number. This simplifies the model after careful analysis, rather than using the same approach for all links regardless of how sensitive they are for the model. Using the DoE approach requires running a model many times. This is unusual for FCMs which are deterministic models and thus only run once. Even when FCMs are used as part of stochastic simulations (e.g. to represent the mental models of populations), the number of runs remains small (Giabbanelli et al. 2014). Consequently, current software packages run FCMs sequentially, which means that running enough of them for a DoE can be prohibitive time-wise. In addition, it is unknown whether FCMs would lend themselves well to simplifications. In theory, one could design an FCM where all links contribute equally to variance, thus our simplification after a DoE would be no better than taking the mid-point of all ranges to start with. Our paper addresses these limitations as follows:

- We develop a new, open source library that runs Fuzzy Cognitive Maps in parallel.
- Using this library, we examine whether Design of Experiments techniques can help to simplify three previously developed Fuzzy Grey Cognitive Maps (which extend FCMs with uncertainty on each link).
- The time required to analyze and simplify the models is examined on different configurations ranging from personal computers to high performance clusters.

The remainder of this paper is organized as follows. In section 2, we provide a technical background on the computations involved in running an FCM, and on the design of experiments. Then, we explain the design of our solution in section 3. This includes our new library for FCMs, using it efficiently to perform simulation runs in parallel over various hardware configurations, and analyzing results to simplify the model. Section 4 provides an experimental evaluation of our approach to simplify three published models, ranging from 8 to 25 links. Section 5 highlights some of the limitations of our approach, and particularly when it comes to simplifying very large models. We conclude by summarizing our current achievements in systematically simplifying small to medium models.

2 BACKGROUND

2.1 The Simulation Approach: Fuzzy Cognitive Maps

A Fuzzy cognitive map (FCM) models the behavior of a system through three key constructs:

(i) nodes, representing concepts of the system such as states or entities. Nodes have a weight in the range [0, 1] indicating the extent to which the concept is present at a simulation step.
(ii) weighted directed links, representing causal relationships. Their weight is from the range [-1, 1] where negative weights indicate that increases in the source node cause a *decrease* in the target node. Conversely, positive weights indicate that increases in the source node cause an *increase* in the target node.
(iii) an inference function, which updates the value of each node based on the weights of both the links going into it and the nodes that these links connect to.

Formally, the number of nodes is denoted by n. The weights of the directed links can be represented as an $n \times n$ adjacency matrix A, where $A_{i,j}$ is the weight of the link from i to j. The value of each concept at step t of the simulation is represented by $V_i(t), i = 1 \dots n$. At each step of the simulation, these values are updated using the following standard equation:

$$V_i(t+1) = f\left(V_i(t) + \sum_{j=1, j \neq i} V_j(t) \times A_{j,i}\right), \tag{1}$$

where f is a clipping function (also known as *transfer* function) ensuring that the values of nodes remain in the [0, 1] range. For example, in an ecological model, a node could stand for the density of fish in a given space, where 0 means no fish and 1 means a maximal density. That value cannot go beyond 1 since it is maximal, and the density cannot be negative either. The clipping function has to be monotonic (to preserve the order of nodes' values) and it is recommended to use a sigmoidal function when modeling planning scenarios (Tsadiras 2008). In this paper, the function we employ is the widely used hyperbolic tangent *tanh* (Groumpos and Stylios 2000, Giabbanelli, Torsney-Weir, and Mago 2012, Mago et al. 2013).

An FCM does not include the concept of time: its steps do not map to physical time (although extensions exist to remedy this limitation). Consequently, an FCM does not run for a time period. Rather, equation 1 is applied until a subset of nodes reaches a stable value. The subset is determined based on the application context. For example, in our previous FCM for obesity, we were interested in the long-term trends for obesity and required this one concept to stabilize in order to stop iterating. Other concepts such as 'food intake' or 'weight discrimination' did not have to stabilize in order to answer the question: how would the level of obesity change in reaction to a new intervention? (Giabbanelli, Torsney-Weir, and Mago 2012) Formally, consider that a subset $S \subseteq V$ needs to stabilize. Then, the simulation will end when:

$$|V_i(t+1) - V_i(t)| \leq \varepsilon, \forall i \in S, \tag{2}$$

where ε is set to a very small positive value. Simulation packages generally include an additional condition whereby the simulation will stop after a set maximum number of steps, in case the condition stated by equation 2 is never met. This additional condition is rarely necessary in practice, as will be illustrated in section 4. Consequently, an FCM is an asymmetrical network of continuous concepts, of which some are required to converge to an equilibrium point or limit cycles. For a broader discussion on methods to improve convergence velocity (e.g., particle swarm optimization), we refer the reader to (Papageorgiou and Salmeron 2013, Napoles et al. 2016).

Since there is no randomness in equations (1) and (2), an FCM does not need repeated simulation runs. The main simulation packages for FCMs, such as `FCM TOOLS` or `MentalModeler.com` thus do not even mention parallelism (Gray et al. 2013, Napoles et al. 2015). While there has been research on parallelism as it relates to FCM, it has mostly been on parallel implementations of techniques used to design FCMs from data (e.g., genetic algorithms (Stach et al. 2007)) rather than on running the FCM itself. Similarly, there have been extensions of the FCM framework which *may* then involve parallelism: these typically propose to design models by combining multiple FCMs either through arbitrary network topologies (Giabbanelli et al. 2014) or via the coordination of a central entity (Stach and Kurgan 2004). However, a parallel implementation was not presented as the algorithms appear to run sequentially.

FCMs are designed to cope with uncertainty, which contributes to their popularity as a participatory modelling technique (Nyaki et al. 2014, Gray et al. 2015, Douglas et al. 2016). Fuzzy Logic is indeed used to compute the weight of each link based on linguistic variables (e.g., ‘very strong’) picked by participants who evaluate its causal strength. However, participants may not agree; or interpret what the link represents in different ways. Thus, there can be significant variations between participants’ assessment of causal strength. In one of our FCMs, significant variations were observed on one third of the links (Giabbanelli, Torsney-Weir, and Mago 2012). To represent these variations and deal with the uncertainty of real-world problems, Salmeron extended the FCMs presented in this section. In particular, he equipped each link with a range as seen in Figure 1. We will refer to this extension as Fuzzy Grey Cognitive Maps, or FGCMs.

2.2 Factorial Design of Experiments

As explained in the introduction, an FGCM may have uncertainty on too many links. Fixing the less important ones would thus make the model more tractable. There are several approaches to assess the importance of different parameters in a simulation model (Jain 1990). At one extreme, one may perform a simple sensitivity analysis which varies one parameter at a time while others are fixed at typical values. This is statistically inefficient and does not account for interactions. At the other extreme, one can generate all possible combinations of parameter values, but exhaustively exploring this search space may be infeasible. A good Design of Experiment (DoE) thus provides information about the contribution of parameters and their interactions, in a way that is feasible given the resource requirements (e.g., computation time). In particular, a 2^k factorial design of experiments reduces the number of levels of each parameter to 2. It is commonly used to determine the relative importance of parameters in a performance study, and readers can refer to (Zhang et al. 2014, Giabbanelli and Crutzen 2013) for examples.

To prepare a 2^k factorial design, one creates a table with all possible combinations of parameter values. Each row defines the setting of a simulation run. The runs are performed, and results are stored in additional columns. For example, in Table 1 we have 8 parameters (i.e. links of an FGCM) so we start with 8 columns and 256 rows for all possible combinations. The first eight rows are shown in Table 1. Simulation results in this example are the values of nodes C_1 and C_2 upon stabilization (equation 2).

Once results are generated, we can calculate the effects: how much of the variance in the results is due to the parameters and their *interactions*. For example, if there are 2 parameters A and B, we would calculate the effects from A, B, and AB (2nd order interaction). While using a 2^k design allows to compute effects up to the k-th order interaction, it is common to stop after the 3rd order if effects become very small (Zhang et al. 2014, Giabbanelli and Crutzen 2013). Several textbooks detail how to calculate effects, and for an updated coverage we refer to chapters 6 and 7 from (Montgomery 2013). While we are not aware of previous work computing effects from a 2^k design in parallel, it is important to understand why it can be done for this paper as k can be large. In short, calculating effects involves (i) representing all parameter

Table 1: Example of a factorial design with 8 links, showing the first 8 combinations out of $2^8 = 256$. Many entries are identical and represented by ... to avoid displaying redundant information.

$C_2 \rightarrow C_1$	$C_5 \rightarrow C_1$	$C_6 \rightarrow C_4$	$C_3 \rightarrow C_1$	$C_3 \rightarrow C_2$	$C_5 \rightarrow C_4$	$C_4 \rightarrow C_2$	$C_6 \rightarrow C_2$	C_1	C_2
-.8	.8	.4	-.5	.4	-.3	-.5	.6	-.9678	.6107
...	...	...	...	...	...	-.5	.8	-.9744	.7472
						-.3	.6	-.9758	.7822
					-.3	-.3	.8	-.9781	.8430
					-.1	-.5	.6	-.9698	.6478
					...	-.5	.8	-.9752	.7662
						-.3	.6	-.9762	.7915
-.8	.8	.4	-.5	.4	-.1	-.3	.8	-.9783	.8488

values as -1 and 1 (thus using a truth table), (ii) generating additional columns for interacting effects by multiplying the respective columns (e.g., the signs for AB are obtained by multiplying the columns for A and B), (iii) multiplying each binary column by results from the simulation runs and adding them. This third step can intuitively be understood as a weighted sum, which qualifies as 'embarrassingly parallel' since computing a weighted sum can be divided into independently computing parts of that sum.

3 METHODS

3.1 New Open-Source Library for Fuzzy Cognitive Maps

The code for our library is publicly available on a third-party research repository at https://osf.io/qyujt/. The library is written for Python 2, and builds on NetworkX (as data structure for the underlying network of an FCM) and NumPy (to compute the matrix operations of equation 1). Note that the matrix operation from equation 1 is already performed in parallel by NumPy (using the Basic Linear Algebra Subroutines) (van der Walt, Colbert, and Varoquaux 2011). Additional parallelism may also be done at the level of the transfer function (f in equation 1), which consists of ensuring that the new values for the concepts are within the [0, 1] interval. We tested whether such parallelism was useful in our situation, by computing the average time to apply the transfer function in parallel or sequentially, depending on the number of concepts. Results of our benchmarking (Figure 3) suggest that parallel processing is *slower* than sequential processing (due to associated overheads) when there are less than 50 concepts, similar up to 100 concepts, and *faster* after 100 concepts. As the case studies in this paper all have less than 50 nodes, we opted for the faster sequential implementation in our library.

3.2 Proposed Process to Simplify Models

3.2.1 Overview

Our proposed process involves five main steps, depicted in Figure 2. In short, we open the file only once, load the FCM, determine how many concepts to stabilize on, and how many links we need to assess for possible simplification. All combinations of binary link values are then generated and the corresponding FCMs are run in parallel, with each computing core running an approximately equal number of FCMs. Rather than aggregating the raw results from each FCM to calculate the effects, each core calculates the effects based on the FCMs that it ran. That is, each core generates the truth table with all factors (i.e. individual links and their interactions), and computes the weighted sums between each factor and each FCM concept to stabilize. The sums are then gathered across the cores so that we know how much of the overall variance in each FCM concept is due to each factor, across all simulation runs. Links whose contribution to variance (either directly or through interactions) is less than a given threshold can be simplified. The next subsection details how experiments are performed in parallel, while the last subsection explains how results are computed.

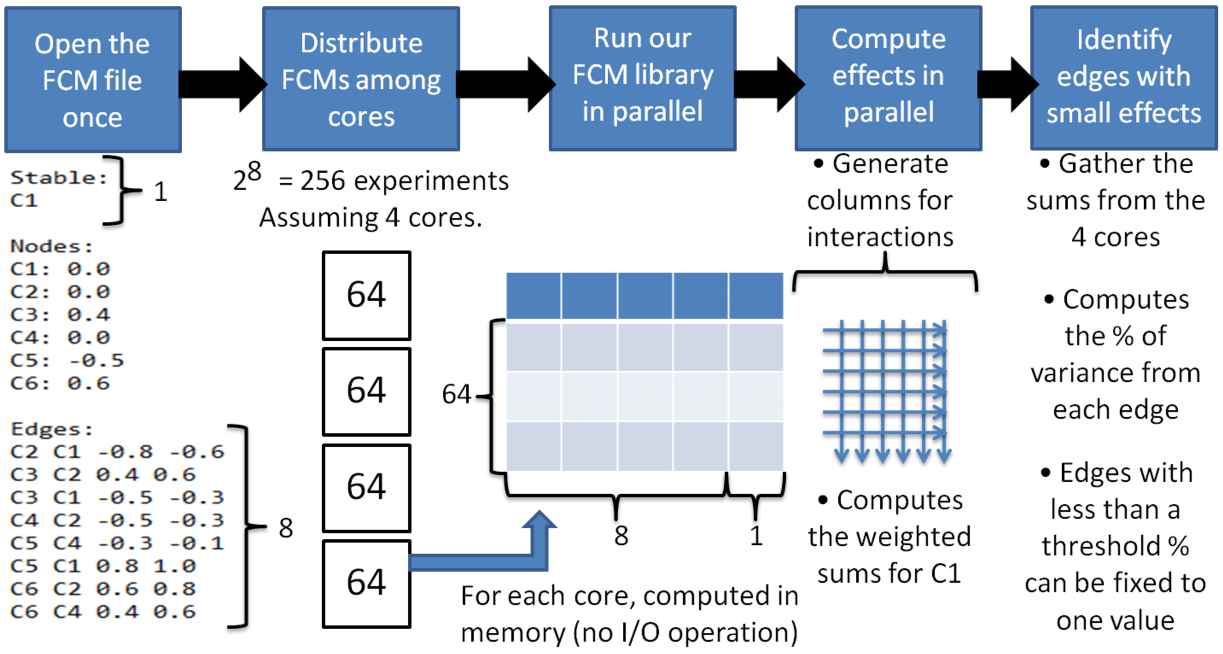
Open the FCM file once
Distribute FCMs among cores
Run our FCM library in parallel
Compute effects in parallel
Identify edges with small effects
Stable:
C1
1
Nodes:
C1: 0.0
C2: 0.0
C3: 0.4
C4: 0.0
C5: -0.5
C6: 0.6
Edges:
C2 C1 -0.8 -0.6
C3 C2 0.4 0.6
C3 C1 -0.5 -0.3
C4 C2 -0.5 -0.3
C5 C4 -0.3 -0.1
C5 C1 0.8 1.0
C6 C2 0.6 0.8
C6 C4 0.4 0.6
8
2^8 = 256 experiments
Assuming 4 cores.
64
64
64
64
64
8
1
For each core, computed in memory (no I/O operation)
• Generate columns for interactions
• Computes the weighted sums for C1
• Gather the sums from the 4 cores
• Computes the % of variance from each edge
• Edges with less than a threshold % can be fixed to one value

Figure 2: General workflow for our approach (top). Each component of the workflow is exemplified using our first case study and assuming a hardware with four cores. When several successive steps are necessary for a component, they are listed as successive bullet points.

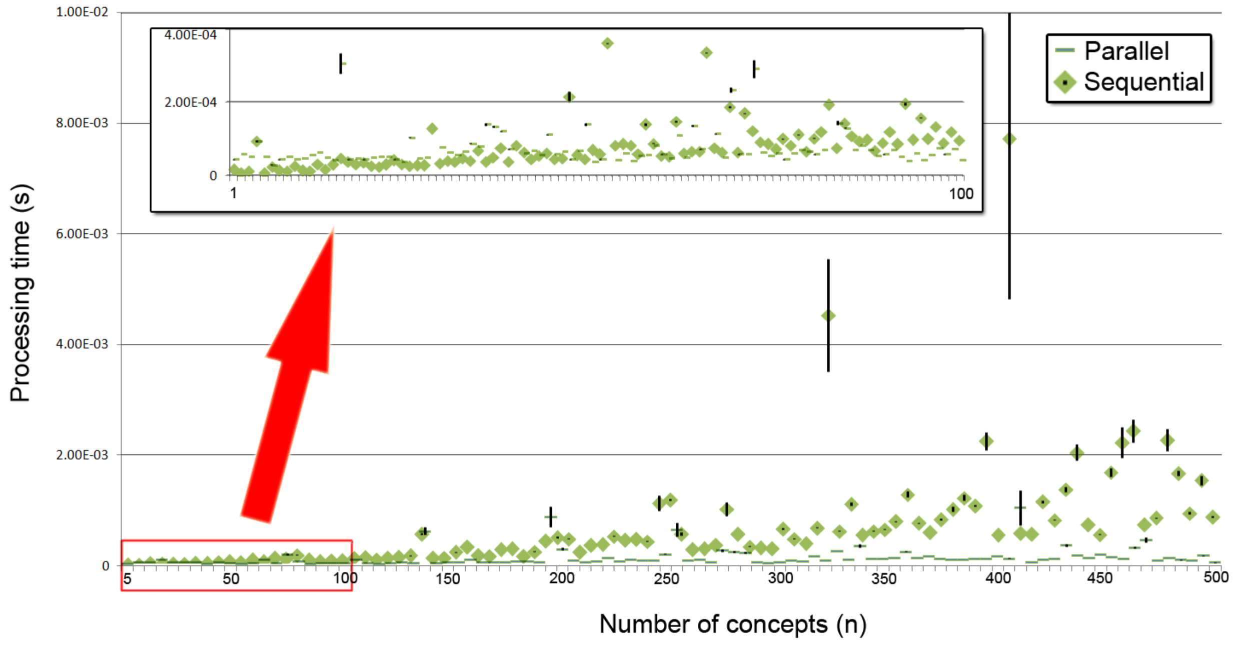

Figure 3: Average and standard deviation of the time to apply a transfer function, either sequentially or in parallel, depending on the number of concepts. The inset shows all number of concepts from 1 to 100 included. The main figure shows concepts from size 5 to 500 by steps of 5. Each data point was computed over 500 repeats. The benchmarking script is available at https://osf.io/qyujt/.

3.2.2 Steps 1–3: Distributing and Running Experiments in Parallel

While one FCM may be considered a relatively small model in terms of the number of links k, using a 2^k factorial design means that the search space grows exponentially with k. It is thus important to efficiently run FCMs in parallel. Our solution defines an FCM in a file listing the concepts, their initial values, the links (with their two possible values), and the concepts that need to stabilize. Reading from this file for each of the 2^k runs would create a significant I/O bottleneck. This is avoided by reading the file once and then distributing the computations among the c cores available on the current machine. The FCMs have the same causal structure and initial value for the nodes: they only differ in the values of their links. Thus, we treat all FCMs as equal[1] by dividing the 2^k experiments into c sets and allocating each set to a core[2].

To further minimize I/O operations, we do not *explicitly* send to each core a list of all experiments in its set. Instead, we only send the ID of the starting experiment, and how many experiments are in the set. A core uses these two numbers to iteratively find the combination of parameter (link) values corresponding to an experiment and generate the next one, until all of them have been performed. To do this, each experiment has an ID (from 0 to $2^k - 1$ included) which encodes the combination of parameter values. To obtain these values, the ID is decoded in binary (over k bits), where the n-th bit specifies whether to use the low (0) or high (1) value for the n-th link. For example, if we have an FCM with 3 links, we perform $2^3 = 8$ experiments numbered from 0 to 7. Experiment 3 would be coded as 011 in binary, stating that the 1st link must be assigned its low value (0), the 2nd its high value (1), and the 3rd its high value (1).

3.2.3 Steps 4–5: Analyzing Experimental Results to Simplify the Model

The results of the simulations performed by one core produce a table similar to Table 1. The simplest approach would be to aggregate all these tables and analyze them (i.e. calculate the effects). However, this would have two significant drawbacks. First, given that we have an exponential time complexity to simulate the combinations of an FCM with k factors, we would also have an exponential space complexity when attempting to store and aggregate all raw results. Examples are provided at https://osf.io/qyujt/ with the ‘intermediate output’ files showing the aggregate tables for k=14 (20Mb file) and k=25 (2.5Gb file). This significant burden on I/O operations would, in turn, slow down the process. Second, aggregating results in a single table with 2^k rows would mean that one core would then analyze the table, which is inefficient and non-scalable. Our solution avoids both drawbacks: we do *not* store and combine raw results. Instead, we use the fact that the analysis can be performed in parallel. That is, the three steps to calculate effects (Section 2.2) are performed on each core based only on its simulation outputs. Each core thus produces a partial weighted sum. These sums are then added on one core, which computes the final results; how much of the variance in each stabilizing FCM concept is produced by each of the links and their interactions.

Once the variance has been analyzed, we have precisely quantified the importance of each link. If a link has an almost negligible contribution to the variance, it means that using a high or low value for that link has a negligible impact on simulation outputs. Such links can thus be set to any value within the range (e.g., the mid-point) which allows for a careful simplification of the model. However, defining the boundary for being ‘negligible’ depends on the simulation setting. The output of one model may be used to continuously control a sensitive device (e.g., medication dosage), while the output of another informs a binary decision (e.g., whether to embark on a given policy or not). Thus, we employ a user-defined threshold T_{var}. A link is simplified if *the sum* of its contributions to variance exceeds T_{var} for at least one stabilizing concept. For example, assume $T_{var} = 5\%$ and three links A, B, C with the following contributions: $A : 91.8\%$, $B : 3.2\%$, $C : 2\%$, $AB : 1\%$, $AC : 1\%$, $BC : 0.5\%$, $ABC : 0.5\%$. The total contributions involving

[1]Some causal values may lead an FCM to converge faster, thus ending a simulation run sooner. However, additional research would be needed to reliably identify such initial settings, and to use that information when distributing the computations. Benefits may be limited as a single FCM runs within milliseconds as discussed in section 4.

[2]We rely on the `multiprocessing` library for Python, which uses *logical cores* to take advantage of hyperthreading. For example, a workstation with 20 physical cores running 2 threads each will be seen as having 40 logical cores.

Table 2: Characteristics of the three case studies.

#links	#nodes	#stabilizing nodes	Description	Reference
8	6	1	Controller for a security system. Concepts include intruder localization accuracy or distance to potential intruder.	(Salmeron 2015)
14	7	2	Supervises radiation therapy. Concepts include the final dose delivered to the patient or the dose prescribed.	(Salmeron and Papageorgiou 2012)
25	15	1	Evaluates the trajectory of individuals in terms of obesity. Involves food intake, depression, or stress.	(Giabbanelli, Torsney-Weir, and Mago 2012)

Table 3: Number and percentage of links that can be set to a single value, depending on the threshold for contributing to variance and whether all or a subset of factors had to stabilize.

Case Studies	Threshold 5% Subset stabilizing	Threshold 5% All stabilizing	Threshold 10% Subset stabilizing	Threshold 10% All stabilizing
1, 8 links	4 (50%)	0 (0%)	5 (62%)	0 (0%)
2, 14 links	6 (42%)	2 (14%)	8 (57%)	2 (14%)
3, 25 links	12 (48%)	11 (44%)	13 (52%)	11 (44%)

each link are: $A+AB+AC+ABC = 94.3\% > T_{var}$ for A, $B+AB+BC+ABC = 5.2\% > T_{var}$ for B, and $C+AC+BC+ABC = 4 < T_{var}$ for C. In this situation, the link C would be simplified by setting its value rather than using a range. We note that our approach produces a very conservative estimate (allowing to confidently set a link's value), as it counts contribution through interactions with the *same* impact as contributions from the link alone.

4 EXPERIMENTAL EVALUATION

The contributions of this paper (as stated in section 1) are threefold: proposing a new method and implementation to simplify models, and evaluating it with respect to (i) how much of a model can be simplified and (ii) how long it takes to perform the computations. The previous section detailed the method and its implementation, thus this section is devoted to the evaluation. The case studies for the evaluation are three models (Table 2) published from 2012 to 2015. They were designed for widely different contexts, and have from 8 to 25 links. The files specifying each case study are available at https://osf.io/qyujt/, using the format shown in Figure 2 (first step). The two smaller models are also shown in Figure 1.

To evaluate how much of each model could be simplified, we used two thresholds: $T_{var} = 5\%$ and $T_{var} = 10\%$. We also examined how to stop a simulation in two ways. First, using the default scenario designed by each model's authors, in which only a designated subset of nodes must stabilize. Second, an extreme setting in which we required *all* nodes to stabilize. This setting makes it much harder to simplify a link's value, because a link can now affect many more outputs other than the designated subset of nodes. This extreme was chosen to assess whether, even in the most draconian situation, we would still be able to simplify a model. Results are shown in Table 3. In the most common setting (column 1: $T_{var} = 5\%$, stabilize the author-designated subset) we observe that about half of each model (42% to 50%) can be simplified. That is, we do not need to use a range for about half of the links and can set their value without significantly changing simulation outcomes. A more lenient setting (column 3 with increasing $T_{var} = 10\%$) allows to simplify 52% to 62% of a model. An extremely strict alternative (column 2 where *all* concepts are potential outcomes) has varied effects: no link can be set in the smaller model (versus half using a subset of nodes as outcomes), while almost half of the links can still be set in the larger model.

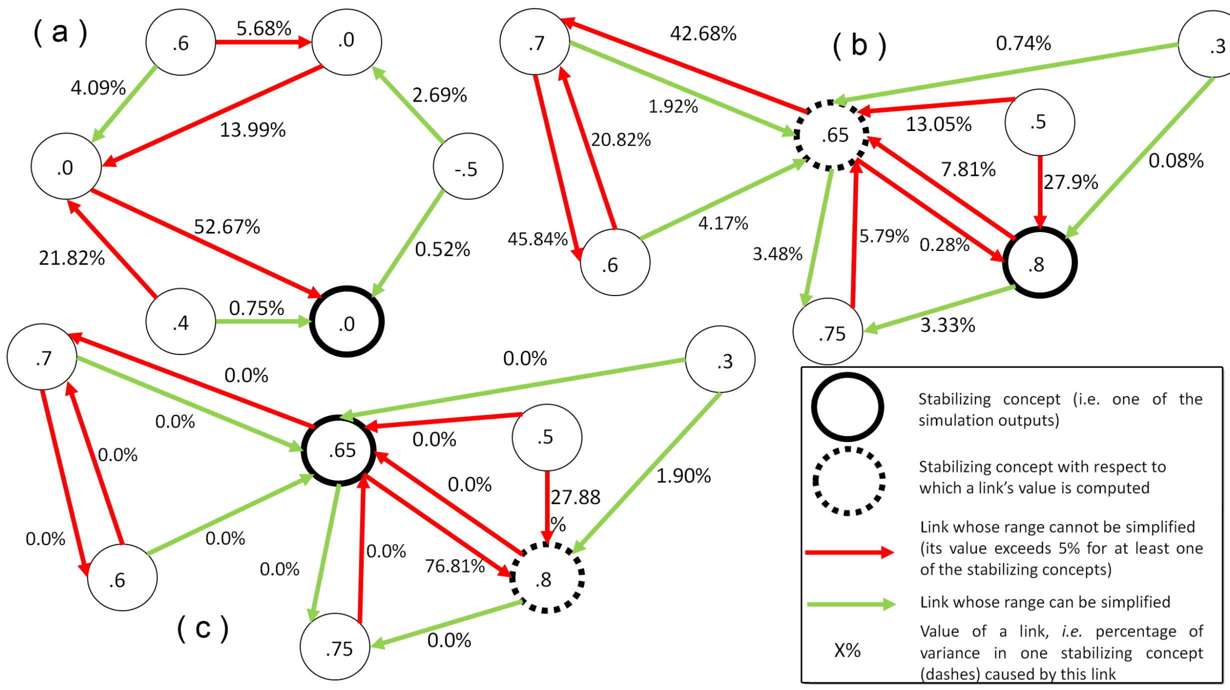
(a)
.6
5.68%
.0
4.09%
2.69%
13.99%
.0
-.5
52.67%
21.82%
0.52%
.4
0.75%
.0
(b)
.7
42.68%
0.74%
.3
1.92%
.65
13.05%
.5
20.82%
7.81%
0.08%
27.9%
45.84%
4.17%
5.79%
0.28%
.6
3.48%
.8
.75
3.33%
(c)
.7
0.0%
0.0%
.3
0.0%
0.0%
.65
.5
0.0%
0.0%
27.88 %
1.90%
0.0%
0.0%
0.0%
0.0%
76.81%
.6
0.0%
.8
.75
0.0%
Stabilizing concept (i.e. one of the simulation outputs)
Stabilizing concept with respect to which a link's value is computed
Link whose range cannot be simplified (its value exceeds 5% for at least one of the stabilizing concepts)
Link whose range can be simplified
X%
Value of a link, *i.e.* percentage of variance in one stabilizing concept (dashes) caused by this link

Figure 4: Results on case studies 1 and 2 with a threshold of 5% and a subset of concepts stabilizing (first result column in Table 3). Values are rounded to the nearest two decimals.

Detailed results for the most common setting are provided in Figure 4, where each link is labeled with its total contribution to variance (including interactions). While we may intuitively expect links directly impacting the stabilizing concept(s) to be more important, Figure 4a exemplifies that it isn't necessarily the case: two of the links directly impacting the one stabilizing concept have a contribution of 0.52% and 0.75% so they can be simplified (colored green). Similarly, we see in Figures 4b-c that links directly impacting one of the stabilizing concepts (dashed) are not necessarily important. This shows that simulating the system as a whole is important to adequately identify the contribution of each link.

Although results suggest that our approach can simplify half of a model under typical requirements, our approach has significant computational costs. To evaluate in which settings these costs may constitute an obstacle to the use of our approach, we computed the time it took to perform the computations on different architectures ranging from an older personal laptop to a modern workstation or a high-performance computer cluster (Table 4). The time for the first two case studies (8 links and 14 links) was computed over 100 repeats, with the criterion that all nodes should stabilized. On average for the smaller case study, it took $0.0238s \pm 0.0243$ on the laptop, $0.0058s \pm 0.0057$ on the workstation, and $0.0029s \pm 0.0141$ on the cluster. For the medium-sized model, it took on average $5.5911s \pm 2.0671$ on the laptop, $0.4615s \pm 0.1962$ on the workstation, and $0.0928s \pm 0.0949$ on the cluster. Each of the individual 100 timings for both cases are available online. Models with up to 14 links can thus be simplified almost instantaneously, even when using entry-level laptops. The largest model with 25 links was run once on the cluster, where it took about 28 hours using only the author-defined subset of nodes to stabilize, and over 260 hours when all nodes had to stabilize. It is thus increasingly infeasible to provide immediate model simplification as the number of links grows beyond 14. Beyond this, a model can still be simplified (at least up to 25 links) but not immediately, and only using specific hardware.

Table 4: Three levels of hardware used to perform experiments.

CPU	Memory	Description (year of purchase)
Intel Core i5-3210M 2.50 GHz 2-core	6Gb 800Mhz (DDR3)	Personal laptop (2012)
2 Intel Xeon E5-2650 v3 2.3 GHz 10-core	32Gb 2133MHz (DDR4)	Professional workstation (2015)
20 nodes, each with 2 Intel X5650 2.66 GHz 6-core	20 nodes, each with 72 GB RAM	High-performance computer cluster (2012)

5 DISCUSSION

We used different hardware configurations to evaluate the time necessary to perform all steps leading to the simplification of model 1 (8 links) and 2 (14 links). The first model could be simplified in 0.02 seconds (on average) using an entry-level laptop, while the second one took 5.59 seconds (on average) with the same laptop. This has important practical implications. It means that facilitators can run a workshop, gather ranges for each link from the participants, and immediately identify the important edges. This, in turn, can be used to guide the conversation with participants on important edges, for example by devoting more time to discussing their possible ranges. Model 3 (25 links) could only be handled using a computing cluster, where it took about 28 hours when considering only one node as output, or over 260 hours when all nodes could be output. This suggests that models between 15 and 25 links can become intractable on entry-level hardware. We thus conclude that our approach is feasible on *typical* model sizes and standard hardware, but becomes intractable as we start handling large FCMs.

While many FCMs tend to be relatively small, there exists a few with a large number of links. For example, a radiotherapy model was proposed with 66 links (Salmeron and Papageorgiou 2012). This would lead to 2^{66} experiments, which is over 70 quintillion experiments. Similarly, our expanded version of the obesity model had 269 links, and would count among the largest FCMs created to date (Drasic and Giabbanelli 2015). Given their already massive number of links, such FCMs could benefit from a simplification. Efficiently simplifying large FCMs should thus be an object of future research. We note that, while small FCMs are typically generated within a facilitated workshop setting, large FCMs can be created over weeks through an asynchronous collaborative process. Such setting does not require the ability to simplify a model using entry-level hardware within seconds. Instead, it would tolerate longer processing times (in the order of days to weeks) and may utilize higher-end hardware such as a computing cluster.

While our approach focused on simplifying the ranges associated to links, there can also be ranges associated with nodes. For example, stakeholders may consider that a variety of settings exist for a given concept, or policymakers may use population distributions rather than an ‘average person’ when initializing a concept. Our approach can also be employed for this setting: instead of generating 2^k experiments for the k links, we would generate 2^{k+n} experiments by also taking into account the n nodes. Results would then show which nodes and links can be simplified. The main consequence is on computational requirements, which in turn impacts the type of hardware that one needs. In model 1, there would be $2^{k+n} = 2^{8+6} = 2^{14}$ experiments, which is feasible on an entry-level laptop within seconds. However, starting with model 2, we would already have $2^{14+7} = 2^{21}$ experiments which may require a computing cluster and days. As mentioned above, this would benefit from research on simplifying larger models.

The endpoints are typically the only information provided about a range (Salmeron and Papageorgiou 2012, Salmeron 2015). Our 2^k factorial design of experiments thus assumes that these endpoints are representative. However, additional data may be available. For example, when each participant has to assign a weight to a link through a questionnaire, the set of questionnaires provides a distribution about the link. If such distributions are heavy-tailed, then the two endpoints of the range may not be representatives and could instead be outliers. As simplifying a model should make use of all available information, future research may explore alternative design of experiments using distributions rather than just endpoints.

6 CONCLUSION

Fuzzy Cognitive Maps (FCMs) can represent the mental model of stakeholders as a causal network equipped with an inference engine. Stakeholders may have widely different views, feel unsure about specific aspects of a complex problem, or wish to capture when the evidence is inconclusive. This can be represented by using a range for each causal link, rather than assigning it a specific weight. The issue then becomes: if all links have a range of values, which ones should we use when running a simulation? Similarly, stakeholders often need to identify the parameters that 'matter' when they design interventions on complex problems. This requires knowing which ranges are unimportant, and which ones strongly impact the model output. Ranges have so far been dealt with by simplifying them to their mid-point (Salmeron 2010), but this systematic simplification does not take into account which ranges matter and which ones do not. In this paper, we presented, implemented, and evaluated a new approach to identify which ranges are important and simplify models accordingly. Our approach uses a 2^k factorial design, where k is the number of links, to evaluate the contribution of each link to variance in the output (i.e., final value of selected nodes in the model). Given the exponential cost of a 2^k factorial design, our implementation uses parallelism not only to run the simulations, but also to analyze the variance. Our evaluation assessed (i) whether our approach can identify unimportant links in previously published models, and (ii) whether our approach is feasible on typical model sizes and standard hardware.

Results from three previously published models show that, under a commonly used setting, almost half of the models can be simplified (42% to 50% of the links contribute to less than 5% of the variance). A more lenient setting allows to simplify an additional 10% of the model (52% to 62% of the links contribute to less than 10% of the variance). Even in the extreme case where the lowest tolerance threshold is used *and* all concepts of the model are considered as possible outputs, some models may be simplified. This setting however exhibits more variability (0% to 44% of the links). We thus conclude that, on previous models, our approach can successfully identify unimportant links. A closer investigation as to which links could be simplified further demonstrated that they could not be straightforwardly identified, for example by assuming that links directly impacting an output would be important to that output's variance.

ACKNOWLEDGMENTS

The authors wish to thank the College of Liberal Arts & Sciences and the Department of Computer Science at Northern Illinois University for financial support. This work was completed in part with resources provided by the Center for Research Computing & Data and the Gaea system at Northern Illinois University. In particular, we are indebted to John Winans for assistance with the Gaea system.

REFERENCES

Douglas, E. M. et al. 2016. "Using Mental-Modelling to Explore How Irrigators in the Murray–Darling Basin Make Water-Use Decisions". *Journal of Hydrology: Regional Studies* 6:1 – 12.

Drasic, L., and P. Giabbanelli. 2015. "Exploring the Interactions Between Physical Well-Being and Obesity". *Canadian Journal of Diabetes* 39:S12–S13.

Giabbanelli, P. et al. 2014. *Modelling the Joint Effect of Social Determinants and Peers on Obesity Among Canadian Adults*, 145–160. Berlin, Heidelberg: Springer Berlin Heidelberg.

Giabbanelli, P., and R. Crutzen. 2013. "An Agent-Based Social Network Model of Binge Drinking Among Dutch Adults". *Journal of Artificial Societies and Social Simulation* 16 (2): 10.

Giabbanelli, P. J., and R. Crutzen. 2014. "Creating Groups with Similar Expected Behavioural Response in Randomized Controlled Trials: a Fuzzy Cognitive Map Approach". *BMC Medical Research Methodology* 14 (1): 130.

Giabbanelli, P. J., T. Torsney-Weir, and V. K. Mago. 2012. "A Fuzzy Cognitive Map of the Psychosocial Determinants of Obesity". *Applied Soft Computing* 12 (12): 3711 – 3724.

Gray, S. et al. 2013. "Mental Modeler: A fuzzy-Logic Cognitive Mapping Modeling Tool for Adaptive Environmental Management". In *Proceedings of the 46th International Conference on Complex Systems*, 963–974.

Gray, S. et al. 2015. "Using Fuzzy Cognitive Mapping as a Participatory Approach to Analyze Change, Preferred States, and Perceived Resilience of Social-Ecological Systems". *Ecol. Soc.* 20 (2): 11.

Groumpos, P. P., and C. D. Stylios. 2000. "Modelling Supervisory Control Systems Using Fuzzy Cognitive Maps". *Chaos, Solitons & Fractals* 11 (13): 329 – 336.

Jain, R. 1990. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley & Sons.

Kosko, B. 1986. "Fuzzy Cognitive Maps". *International Journal of Man-Machine Studies* 24:65–75.

Mago, V. K. et al. 2013. "Analyzing the Impact of Social Factors on Homelessness: a Fuzzy Cognitive Map Approach". *BMC Medical Informatics and Decision Making* 13 (1): 94.

Montgomery, D. C. 2013. *Design and Analysis of Experiments*. 8 ed. John Wiley & Sons.

Napoles, G. et al. 2015. "A Computational Tool for Simulation and Learning of Fuzzy Cognitive Maps". In *2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 1–8.

Napoles, G. et al. 2016. "On the Convergence of Sigmoid Fuzzy Cognitive Maps". *Information Sciences* 349–350:154–171.

Nyaki, A. et al. 2014. "Local-Scale Dynamics and Local Drivers of Bushmeat Trade". *Conservation Biology* 28 (5): 1403–1414.

Papageorgiou, E. I., and J. L. Salmeron. 2013. "A Review of Fuzzy Cognitive Maps Research During the Last Decade". *IEEE Transactions on Fuzzy Systems* 21 (1): 66–79.

Salmeron, J. L. 2010. "Modelling Grey Uncertainty with Fuzzy Grey Cognitive Maps". *Expert Systems with Applications* 37 (12): 7581 – 7588.

Salmeron, J. L. 2015. "A Fuzzy Grey Cognitive Maps-Based Intelligent Security System". In *2015 IEEE International Conference on Grey Systems and Intelligent Services (GSIS)*, 29–32.

Salmeron, J. L., and E. I. Papageorgiou. 2012. "A Fuzzy Grey Cognitive Maps-Based Decision Support System for Radiotherapy Treatment Planning". *Knowledge-Based Systems* 30:151 – 160.

Stach, W. et al. 2007. "Parallel Learning of Large Fuzzy Cognitive Maps". In *2007 International Joint Conference on Neural Networks*, 1584–1589.

Stach, W., and L. Kurgan. 2004. "Parallel Fuzzy Cognitive Maps as a Tool for Modeling Software Development Projects". In *Fuzzy Information, 2004. Processing NAFIPS '04. IEEE Annual Meeting of the*, Volume 1, 28–33.

Tsadiras, A. K. 2008. "Comparing the Inference Capabilities of Binary, Trivalent and Sigmoid Fuzzy Cognitive Maps". *Information Sciences* 178 (20): 3880–3894.

van der Walt, S., S. C. Colbert, and G. Varoquaux. 2011. "The NumPy Array: A Structure for Efficient Numerical Computation". *Computing in Science Engineering* 13 (2): 22–30.

Zhang, D. et al. 2014. "Impact of Different Policies on Unhealthy Dietary Behaviors in an Urban Adult Population: an Agent-Based Simulation Model". *Am. J. Public Health* 104 (7): 1217–1222.

AUTHOR BIOGRAPHIES

ERIC A. LAVIN is a Graduate Research Assistant in the Department of Computer Science at Northern Illinois University (USA). His thesis is at the intersection of Fuzzy Cognitive Maps and High Performance Computing. His email address is ealavin@gmail.com.

PHILIPPE J. GIABBANELLI, Ph.D., is an Assistant Professor in the Department of Computer Science at Northern Illinois University (USA). He received his Ph.D. from Simon Fraser University (Canada), and performed his postdoctoral studies at the University of Cambridge (UK). His research interests include data mining and simulation models. His email address is giabba@cs.niu.edu.