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

CONCURRENT CONVERSATION MODELING AND PARALLEL SIMULATION OF THE NAMING GAME IN SOCIAL NETWORKS

Kalyan S. Perumalla

Discrete Computing Systems Group
Computer Science and Mathematics Division
Oak Ridge National Laboratory
Oak Ridge, TN 37831, USA

ABSTRACT

The Naming Game is an effective self-organization model to understand the emergence of linguistic consensus and to investigate the system dynamics in a variety of phenomena over social networks of autonomous agents. The Naming Game is an effective description for the evolution of consensus despite the absence of any central coordination or specialized initialization even in large-scale networks. While the classical game is effective in description, it was defined with inherently sequential evaluation semantics over the entire network. Here, we develop a new concurrent model as a relaxation of the classical formulation and express it in a discrete event style of evaluation. Further, with the uncovered concurrency that was absent in the classical algorithm, we map the concurrent model to parallel discrete event simulation. Using a prototype implementation, we present an initial parallel performance study on networks containing hundreds of thousands of individuals, with a decrease in simulation time in the best-observed case from 4800 seconds down to 1400 seconds.

1 INTRODUCTION

1.1 Background

The Naming Game (Baronchelli 2016; Lu, Korniss, and Szymanski 2008; Lu, Korniss, and Szymanski 2009; Baronchelli et al. 2006) arises in a variety of contexts, all of which may be categorized as emergent linguistics in a broad sense. Among the earliest identifications is by Steels (1995) where he proposed the linguistic communication capability of humans as an *emergent* phenomenon. He provided a framework by which common words *evolve* over time that convey mutually agreeable meanings and are eventually shared by the speakers and listeners. Consensus on the choice of the words and their associated meanings arises as a collective phenomenon over time, stylized as a series of "conversations" across many agents that repeatedly exercise the candidature of mutually acceptable words for connotations. Steels proposed that a language is an "autonomous adaptive system" for a "self-organizing cultural process." To support this theory, he developed a model of distributed agents that "develop from scratch a vocabulary to identify each other through names and spatial descriptions." Steels developed "dialogs as language games," with dialog structure, dialog iteration, "communicative success," "dialog alteration," and so on, with associated semantics.

It is a factual observation that not only do new words enter any modern language's dictionary over the course of time, but also new words spread and compete to eventually converge from many to a few or single commonly acceptable word (Lass 1997). Baronchelli et al. (2006) provide an excellent first introduction to the concepts (and to several further references to related articles in the literature) on the general evolution of words, languages, meanings, and their competitive dynamics.

In current day, the target domain of emergent, autonomous evolution of common linguistic and naming evolutions now is greatly extended to objects and novel interaction topics in the digital plane of social life. For example, the self-organization of tags to digital objects is a Naming Game that evolves a system of tags in social networks to name and share information objects (such as images in a photo sharing system, or audio files in a music sharing system). Even artificial robotic web agents engage in their own evolution game and/or participate with human agents in the evolution of a common naming enterprise in the networked digital world.

In fact, it is now well understood that it is indeed possible to bootstrap and successfully conclude a naming exercise in a completely autonomous fashion *without* any central coordinator, even with only *localized* communication, and with *zero* initial names, even in a *large-scale* distributed system (see page 5 of Baronchelli et al. (2006) for a proof outline). The Naming Game has been used for this purpose in modeling a variety of consensus formation phenomena in social networks. Lu, Korniss and Szymanski (2008, 2009) investigated the Naming Game in applications such as opinion formation, advertising, and influence of idealist agents. Baronchelli et al. (2006) have investigated the composition of microscopic behaviors into collective behaviors under different proposed variants of the Naming Game. More recently, the evolution of conventions was recently studied using the Naming Game by Franks, Griffiths and Jhumka (2013). Additional complexity of communication mechanisms such as feedback and broadcast has been examined by Baronchelli (2011).

1.2 Game Overview

In its simplest form, the Naming Game is defined on a population of N agents in evolving their language to mutually convey information about objects in their environments. For simplicity, the language evolution is restricted to naming a single object: the subject of discourse for all agents is some single object (such as a physical object like a smart phone, or a work of art like a painting, or slang name for everyday use like soda or pop, or even abstract concept such as cool or hip). Thus, we will assume that the agents are collectively naming some single object, and the vocabulary they utilize is the set of some fixed sized words formed by a string of characters. These words could represent anything such as linguistic words or formatted tags and labels or symbols. Effectively, since the words in the vocabulary may be enumerated, the words may be mapped to integers. The game is thus abstractly defined with some such vocabulary V of $|V|$ integers, with N agents dynamically choosing to communicate with each other using words from V. Each agent is assumed to hold the agent's own individual dictionary. Thus, there are N dictionaries in total in the system. Each dictionary D_i holds zero or more words from the system vocabulary. Thus, every D_i is a subset of V. Moreover, any dictionary can grow or shrink over the course of the game.

The game proceeds as a series of "conversations." Each conversation involves an ordered pair of agents, say, (i,j). The first in the pair is called the speaker, and the latter is the hearer. The speaker picks a random word from his dictionary and "communicates" to the hearer. Upon hearing that word, the hearer checks if he "knows" that word by seeing if it exists in the hearer's dictionary. If it exists, then the conversation is treated as having been successful because both the speaker and hearer have managed to refer to the object with the same name. When this success is detected, both the agents decide to abandon all other words that they held as candidates in their dictionaries. In their place, they both retain only the just-now matched word in their dictionaries. For example, if dictionary of agent i is $D_i=\{A,B,C\}$ and i happens to choose C to speak to j whose dictionary $D_j=\{C,D\}$, then the conversation is a success since C is common to both dictionaries; the two dictionaries drop everything except C to become $D_i=\{C\}$ and $D_j=\{C\}$. Now consider the case when the hearer did not find the spoken word in his dictionary. In this case, the conversation is viewed as a failure; this failed conversation leads to a partial (conditional) learning by the hearer that the spoken word is also one of the possible names. The hearer registers this by adding the spoken word into its own dictionary. The speaker's dictionary is left unchanged. For example, if $D_i=\{A,B,C\}$ and $D_j=\{E,F\}$, and i chose to speak C, then C is added to D_j to give $D_j=\{E,F,C\}$,

and $\boldsymbol{D_i}$ unchanged. This entire speak-match-revise process is repeated by choosing another random pair of agents.

The amazing aspect of the game is that a repeated operation of these rules results in a convergence of the system such that all individuals end up selecting a single common word even if the system starts with absolutely no words in any dictionary (see algorithms in next sections).

While there are many variants that are defined on this basic structure, this by itself leads to interesting dynamics at scale. Some variants include incorporation of "committed agents" that retain a fixed dictionary and do not modify their dictionary in a failed conversation as a hearer. They have been found to accelerate consensus in certain network structures (Lu, Korniss, and Szymanski 2009). Another is an *a priori* limit on the dictionary sizes, mimicking the constant memory bound of agents; otherwise, the upper bound on the dynamic dictionary sizes in the system is a polynomial function over the system population size.

In the remainder of this article, the analogy of spoken language with vocal communications is used for the terminology of the algorithms and results; however, all findings apply equally to other contexts of the Naming Game, such as tagging, convention establishment, and opinion formation.

1.3 Organization

The article is organized as follows. The classical sequentially-defined algorithm of the Naming Game is described in Section 2, and its limitations are highlighted. The proposed new concurrent discrete event model is presented in Section 3 with the parallel execution approach and algorithms. The implementation details of the algorithms in a parallel discrete event simulator are described in Section 4. A comparison of the dynamics of the Naming Game between the classical algorithm and the new concurrent algorithm is presented in Section 5. This is followed in Section 6 by a performance study of the parallel simulation implementation on population sizes from one thousand to quarter million dictionaries scaled from 1 to 16 processor cores. Finally, the findings are summarized and future work is identified in Section 7.

2 CLASSICAL MODEL

2.1 Sequential Algorithm

The classical Naming Game operates on a set of $\boldsymbol{N}$ dictionaries in a sequential fashion, evolving the growth and diminution of the dictionaries over time, until a fixed point is reached by which all dictionaries converge to a consensus single word.

Classical Algorithm

The following steps are executed repeatedly in a loop until consensus is reached (all dictionaries have identical, single word)

1. A random (***S***,***H***) pair of persons is selected from the population
2. If speaker ***S***'s dictionary ***DS*** is empty
    - 2.*a* A (random) word ***R*** is selected from vocabulary ***V***
    - 2.*b* ***R*** is added to ***DS***
3. Speaker ***S*** selects a (random) word ***W*** from own dictionary ***DS***
4. Hearer ***H*** consults own dictionary ***DH*** to verify if ***W*** exists

5.*a* If ***W*** exists in ***DH*** [*Success* of conversation]
    - 5.*a*.1 ***DS*** and ***DH*** are both emptied
    - 5.*a*.2 ***W*** is added to both ***DS*** and ***DH***

5.*b* Else [*Failure* of conversation]
    - 5.*b*.1 ***W*** is added to ***DH***

Figure 1: Classical *Naming Game* algorithmic template.

The classical algorithm with inherently sequential semantics is shown in Figure 1. Initial conditions for dictionaries can be varied: they may start empty or be populated with one or more random words from the vocabulary. A community or connectivity network is used to determine neighborhood structure in selecting a hearer for a speaker.

Note that the classical algorithm operates with a global view of all dictionaries. Every conversation is dependent on previous conversations because the input for one conversation may be the output of previous conversations. Moreover, each conversation may alter the subject dictionaries dramatically: upon a successful conversation, the speaker's dictionary and hearer's dictionary are fully purged, except for the common word that is spoken. This prevents conversations from being processed concurrently.

2.2 Limitations

There are several deviations of the classical algorithm from realistic conditions:

- *Time-spanned Conversations*: The classical algorithm treats conversations as instantaneous. However, typically, some amount of time elapses during conversation. During this time, other pairs may engage in conversations. This simultaneity creates chances for speakers to become hearers even while they are in the process of conversing.
- *Asynchronous, Simultaneous Conversations*: While the classical algorithm treats conversations as occurring sequentially one after the other, in reality many conversations may proceed simultaneously. A simultaneous conversation model also makes it possible for the speaker to encounter a different state of a subsequent hearer's dictionary than it would encounter in the sequential algorithm. Moreover, diffusion of the words may proceed faster due to many concurrent conversations.
- *Larger-scale Populations*: Population sizes can be much larger (nearly billion agents) in modern social networks than considered previously. Thus, simulation speed can become constrained by the inherently sequential processing of the classical algorithm.
- *Scaled Effects*: Dynamics of time-spanned conversations and asynchronous simultaneous conversations are more pronounced when population size increases.

There are several peer-to-peer simulation systems that have been developed in the past few years (Cecin et al. 2006; Naicken et al. 2007; Quinson et al. 2012; Hanai and Shudo 2014; Andelfinger Jünemann, and Hartenstein 2014). However, while they could probably be utilized as a discrete event simulation framework onto which the naming game may be mapped, a new concurrent model for the Naming Game is first needed in order to utilize the peer-to-peer simulators and/or integrate the game operation (as, for example, a realistic timing model for wireless or wired communication channels and social networking applications between agents).

3 CONCURRENT DISCRETE EVENT MODEL

In this section, important factors are identified about the classical algorithm that hinder concurrency and prevent parallel execution. Our new approach is described to introducing concurrency framed in a discrete event style of evolution, and the parallel algorithms for the events are presented.

3.1 Parallel Approach

In the classical algorithm, all operations are implicitly combined into one aggregate. In order to uncover concurrency, the aggregated operations need to be separated. We identify decouple eight different operations that underlie the classical definition:

- Op 1: Speaker selection
- Op 2: Hearer selection
- Op 3: Transmission of word
- Op 4: Receipt of word
- Op 5: Revision of hearer's dictionary
- Op 6: Revision of speaker's dictionary
- Op 7: Inter-conversation time interval
- Op 8: Conversation time period.

First, since the classical algorithm operates with a global view, it is able to generate the speaker-hearer pairs as a random *stream* of pairs. In reality, it is each person who individually determines its hearer dynamically, picking the targets from its neighborhood. Thus, there is no specific speaker selection: all persons are speakers are at varying points in time, and whenever a person initiates a conversation spontaneously, that person will undertake a hearer selection. Thus, operations (Op 1) and (Op 2), while being combined in the classical algorithm, are separated in the concurrent algorithm.

Secondly, in the classical algorithm, there is no explicit operation called the *transmission* of the word from speaker to hearer. The global view of the algorithm implicitly manifests the spoken word from speaker's dictionary to the hearer's dictionary. In reality, the dictionaries are not all necessarily co-located. Communication in real-life happens in a wide variety of mechanisms, most of which occur in a geographically separated fashion. Thus, to uncover concurrency, the transmission operation of spoken word (Op 3) is modeled explicitly. This also provides a modeling hook to introduce more complex models of transmission times, as in integrating the game with a wireless or wired communication network simulation for more detailed and realistically varying time delays in communication. Similarly, receipt of word (Op 4) is the hearer's counterpart missing in the classical algorithm, which we introduce in the concurrent algorithm.

Next, the speaker's and hearer's dictionaries are synchronously updated in a global view of the classical algorithm. However, their updates need to be decoupled in time in order to model the more realistic framework in which the hearer performs the word-matching operation (Op 5) and then informs the speaker of the result who later revises its dictionary accordingly (Op 6). The updates to hearer's and speaker's dictionaries, thus, will in general be separated in time.

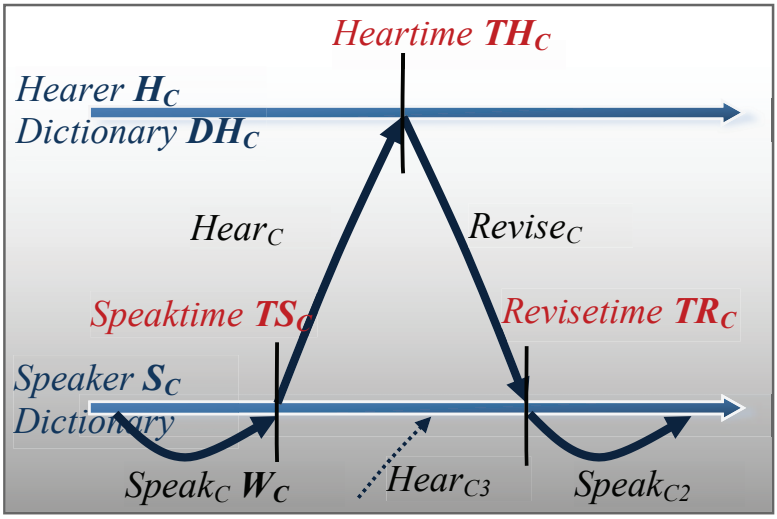

Figure 2: Concurrent asynchronous conversation across speaker and hearer along virtual timelines.

Finally, inter-conversation timing is only implicitly expressed in the classical algorithm as the sequence of conversations. Moreover, the actual time elapsed in the conversation is effectively zero; every conversation is essentially instantaneous. These factors also avoid the complexity of interleaved, simultaneous, asynchronous, and time-spanning conversations. While model complexity is reduced in the classical algorithm, it also quashes concurrency in the model. This complexity of a truly realistic game in fact requires the sophistication of discrete event style of execution, which we employ in the concurrent algorithm by explicitly modeling an inter-conversation time interval (Op 7) and a multi-phased time elapse during conversation (Op 8).

$$
\begin{aligned}
&V = Vocabulary = Set\ of\ \ potential\ words = \{W\} \\
&DI_C = \{W_i\} \subseteq V = Dictionary\ of\ \ person\ I\ in\ conversation\ C \\
&C_C = Conversation_C = \left\langle Speak_C \rightarrow Hear_C \rightarrow Revise_C \right\rangle \\
&Speak_C = (Speaker\ S_C, Hearer\ H_C, Speaktime\ TS_C \rightarrow Heartime\ TH_C, Word\ W_C) \\
&Hear_C = (Speaker\ S_C, Hearer\ H_C, Heartime\ TH_C \rightarrow Revisetime\ TR_C, Word\ W_C) \\
&Revise_C = (Speaker\ S_C, Revisetime\ TR_C, Word\ W_C) \\
&Game = Simulation\ of\ \ conversations\ \{C_C\}\ in\ global\ temporal\ order
\end{aligned}
$$

Equation 1: Definitions of concurrent model elements.

$$
\begin{aligned}
&Hear_C : DH_C \leftarrow DH_C \bigcup \{W_C\},\ or\ \{W_C\} \\
&Revise_C : DS_C \leftarrow DS_C,\ or\ \{W_C\}
\end{aligned}
$$

Equation 2: Effect on dictionaries of hearer/speaker from hearing/revising operations.

3.2 Concurrent, Asynchronous Conversations

Every person is mapped to its own virtual time line. Given a speaker initiating a conversation ***C*** with a chosen hearer, the operation of the conversation in terms of timelines and event dependencies are illustrated in Figure 2. The corresponding definitions of the dictionary sets, operation types, sequences, and simulation composition are shown in Equation 1. The effects of hearing and revision operations on the hearer's dictionary and speaker's dictionary are shown in Equation 2. The arrows depict scheduled events within or across timelines: the *Speak* event is scheduled by a person to itself. The *Hear* event is scheduled from the speaker to hearer, while the *Revise* event is scheduled from the hearer back to the speaker.

3.3 Speaker's Concurrent Role as Hearer

An important issue that is absent in the classical algorithm arises in the parallel algorithm. In the classical algorithm a person is either a speaker or hearer or idle, but a person is never a combination; in particular, a speaker is never a hearer while a conversation is active. In the concurrent algorithm, however, a speaker in an active conversation may simultaneously become a hearer of another conversation. Due to the separation of the speaking operation and dictionary revision operation on the speaker's side of conversation, there is a period during which the speaker may become the hearer of another speaker. For example, this is illustrated in Figure 2 with a potentially incoming $Hear_{C3}$ event intervening on speaker $\boldsymbol{S_C}$ between *Speaktime* $\boldsymbol{TS_C}$ and *Revisetime* $\boldsymbol{TR_C}$. In this period, if there is a match of spoken word, the original speaker who is also now hearing another word may purge its dictionary completely, potentially forgetting its matched word. To resolve this conflict, there are two possibilities. One possibility is to ignore this conflict and let the speaker forget the spoken word. The other is to make the speaker retain a record of the in-progress word, and make the conflicting operations independent of each other on the speaker's dictionary.

3.4 Parallel Algorithm

The full parallel algorithm for discrete event-based operation of concurrent conversations in the new Naming Game is shown in Figure 3. The algorithm is expressed in terms of conversation numbered ***C*** from the speaker's point of view. Initially, for ***C***=0, the logical process to which the speaker is mapped schedules a *Speak* event to jump start the process for all persons. Since the chosen word, the chosen hearer, and the states of dictionaries vary with each conversation, they are all labeled with the conversation subscript ***C***, to make the algorithm elements correspond to a single conversation.

4 IMPLEMENTATION

4.1 Discrete Event Simulation and Logical Processes

The concurrent model is implemented in a parallel discrete event simulation framework. The population is organized into clusters of persons, depending on the underlying community structure of the social network. For example, for simple cliques, the entire population is split into equal sized groups partitioned across processors, typically hosted as one group per logical process (LP), and one logical process per processor core. This can be varied via a flexible person-to-LP mapping visible at all processors. A small-world network is mapped with the first person within each LP being the leader that carries out conversations both within and outside its cluster.

Concurrent Model as Discrete Event Algorithms
[Specified in terms of any conversation C]

Initialization [C=0] for every person S
I1. An exponentially distributed random time TS_0 is selected
I2. An initial "$Speak_0$" event is scheduled to person S at TS_0
Speaker S_C "$Speak_C$" event processing at "$Speaktime_C$" TS_C
S1. If speaker S_C's dictionary DS_C is currently empty
    S1.*a* A (random) word RW is selected from vocabulary V
    S1.*b* RW is added to DS_C
S2. A (random) word W_C is chosen from speaker's dictionary DS_C
S3. A (random) hearer H_C is chosen in network neighbors of S_C
S4. A virtual time increment "*Delay*" dt in future is chosen
S5. A virtual time "*Heartime*" TH_C in future is determined as $TH_C=TS_C+dt$
S6. A "$Hear_C$" event containing word W_C is scheduled to hearer H_C with virtual timestamp TH_C
S7. Word W_C is remembered at S_C as its "*In-Progress Word*" (IW)
Hearer H_C "$Hear_C$" event processing at "$Heartime_C$" TH_C
H1. Hearer H_C consults own dictionary DH_C to verify if W_C exists
H2.*a* If W_C exists in DH_C [*Success* of conversation]
    H2.*a*.1 DH_C is emptied
    H2.*a*.2 W_C is added to DH_C
    H2.*a*.3 If H_C is speaking with an "*In-Progress Word*" IW
        H2.a.3.1 IW is added to DH_C
H2.*b* Else [*Failure* of conversation]
    H2.*b*.1 W_C is added to DH_C
H3. A virtual time "$Revisetime_C$" TR_C in future is determined as $TR_C=TH_C+dt$
H4. A "$Revise_C$" event containing success/failure code is scheduled to speaker S_C with virtual timestamp TR_C
Speaker S_C "$Revise_C$" event processing at "$Revisetime_C$" TR_C
R1. If event success/failure code is *Success*
    R1.*a*.1 DS_C is emptied
    R1.*a*.2 W_C is added to DS_C
R2. The current "*In-Progress Word*" IPW at S_C is reset to none
R3. A future time TS_{C2} is selected from current time TR_C
R4. A new "$Speak_{C+1}$" event is scheduled to S_C at TS_{C+1}

Figure 3: Algorithm of the new concurrent model for asynchronous conversations-based *Naming Game*.

Dictionaries are statically mapped to the LPs and remain mapped to the same processor throughout the simulation. All interactions across dictionaries are only performed via time-stamped event exchanges.

Termination detection is achieved via a single special collector LP in the simulation to which all LPs periodically send summaries of their local dictionaries. The collector LP detects global consensus by compacting all contributed summaries (partial sets) from all LPs and detecting the time at which the global dictionary contains a single word, at the same time when the average and maximum dictionary size is unity. An efficiency enhancement for periodic updates from LPs to the collector LP is to send the summary only if the local dictionary itself show consensus (since local consensus is a necessary, but not sufficient, condition for global consensus). However, in order to allow charting the trajectory of the dictionaries across the processor over the course of the simulation, the periodic updates are sent without this efficiency enhancement.

4.2 Lookahead

The concurrent model provides opportunity to determine directly by observing the minimum time increment added to the three event types. The *Speak* events are scheduled by a person to itself, which in turns means an event sent by an LP to itself; hence *Speak* events need not be constrained by lookahead. In our implementation, we use exponentially distributed virtual time increments to schedule the next *Speak* event at the end of a *Revise* event (step R2 in Figure 3), just as in initialization (step I1). The other two types, *Hear* and *Revise* events, may go across LPs because the hearer person may be mapped to another LP and/or processor. Thus, lookahead ***LA*** is determined as:

$$\begin{aligned} dH_C &= TH_C - TS_C \\ dR_C &= TR_C - TH_C \\ LA_C &= \min(dH_C, dR_C) \\ LA &= \min(LA_C) \forall C \end{aligned}$$

The lookahead is achieved by setting a positive minimum value for $\boldsymbol{dH_C}$ and $\boldsymbol{dR_C}$ at simulation initialization. The effect of variation of this delay on the dynamics of the Naming Game are later studied in Section 5. Using this lookahead value, the parallel simulation is implemented with conservative parallel synchronization protocol. We use "Mattern-style" synchronization (Mattern 1993, Bauer 2009) to overcome the transient event and simultaneous reporting problems in the distributed parallel setting.

4.3 Dictionary Data Structures

A major computational cost lies in the manipulation of dictionaries: adding words, looking up words, and purging. The vocabulary can be very large, as large as ***N***, the population size itself. Yet, the number of words at any given time in any given dictionary can be extremely small, as low as zero, one or a small constant. Since the dictionary may hold any subset of words from the vocabulary, the distribution of words has no specific structure to exploit in an efficient representation of the dictionary. In one extreme, a bit vector of size N bit can be used to represent which words are present or absent from the vocabulary. In the other extreme, an integer vector of variable size can be used to store the integers corresponding to the actual words that are present in the dictionary. Clearly, the bit vector wastes memory for small dictionaries, and the integer vector wastes memory for large dictionaries. We developed a dictionary representation that is capable of both alternative representations, and dynamically switches between the two representations at runtime based on the current state of the dictionary. The compaction and rebalancing can be performed automatically or initiated on demand during simulation. The efficiency of the dictionary becomes quite pronounced for large population sizes.

5 CONCURRENT MODEL COMPARISON

Two major effects arise from adding a realistic concurrency in terms of simultaneous conversations and time-elapsed conversations. The first effect is on dictionary sizes: on the dictionary size that any person individually observes, and on the average dictionary size that the system globally experiences. The second effect is the number of conversations that the system takes to reach consensus. The first effect is an inherent nature of the concurrent model, while the second effect is dependent on the amount of delay used to model the conversation timing characteristics.

Thus, the consensus dynamics depend on the delay employed for communication in the concurrent model. A zero delay corresponds to the unrealistic instantaneous communication of the classical algorithm, while small delays tend to make the concurrent model correspond closely to the classical model. On the other hand, the simultaneity of multiple conversations enabled by concurrency is less intuitive. To understand the effect of the delay on consensus time and dictionary sizes, the dynamics for N=1024 are charted in Figure 4, Figure 5, and Figure 6, showing respectively the average dictionary size, the global dictionary size, and the maximum dictionary size, all tracked during the game across conversations. The sizes are from results averaged over multiple runs (10) of every scenario.

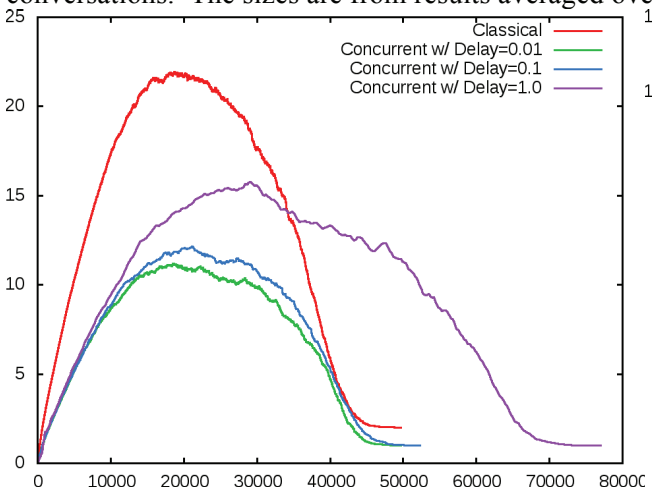

Figure 4: Variation of average dictionary size with number of conversations (population N=1024).

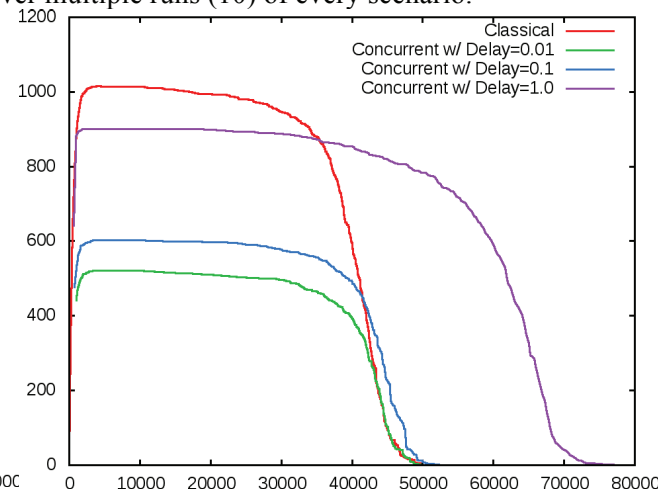

Figure 5: Variation of global dictionary size with number of conversations (population N=1024).

The average dictionary size is a key effect in the dynamics of the Naming Game. The time to consensus may be reached with a relatively larger or smaller average dictionary size, depending on the amount of mixing happening in unit time in the system. Thus, while the classical model predicts an average dictionary size, the concurrent model reduces that size due to more rapid, simultaneous propagation of matching information in the system due to concurrently active conversations that are actively seeking to establish common words. This effect is reflected in Figure 4.

The larger the delay in the concurrent model, the closer is the match of the maximum global dictionary size with that of the classical algorithm. However, the concurrent algorithm's time to consensus is stretched as it takes more conversations to reach consensus. On the other hand, the smaller the delay, the closer is the concurrent model to classical model in terms of time to reach consensus. However, within a given time interval, due to increased simultaneity of conversations under small delay, there is a more thorough mixing (mean field) of dictionaries in the concurrent model compared to the classical model. Hence, the growth of the global dictionary size is more actively limited in the concurrent model. All these effects are reflected in Figure 5.

For the maximum dictionary size manifested in the system, the match between concurrent and classical models is observed to be closer with decreased delay, as expected, in Figure 6.

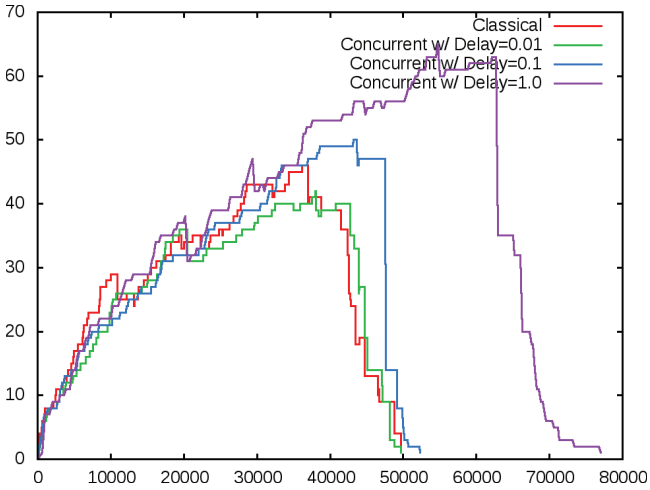

Figure 6: Variation of maximum dictionary size with number of conversations (population size N=1024).

6 PERFORMANCE STUDY

An experimental study for parallel performance has been made with increasing population sizes and number of processor cores. The parallel performance results are obtained from experiments for different population sizes, but keeping the population size the same while the increasing number of processor cores for that population size. The simulator is written in the C++ programming language, using the Message Passing Interface (MPI). All the experiments are performed in a 24-core server with AMD 4x6-core 6174 processors and with 64GB main memory. The number of processors used in the experiments was varied by powers of 2 from 1 up to 16 cores. To minimize perturbation of the results from operating system processes, the number of cores was restricted to 16 instead of using all available 24 cores.

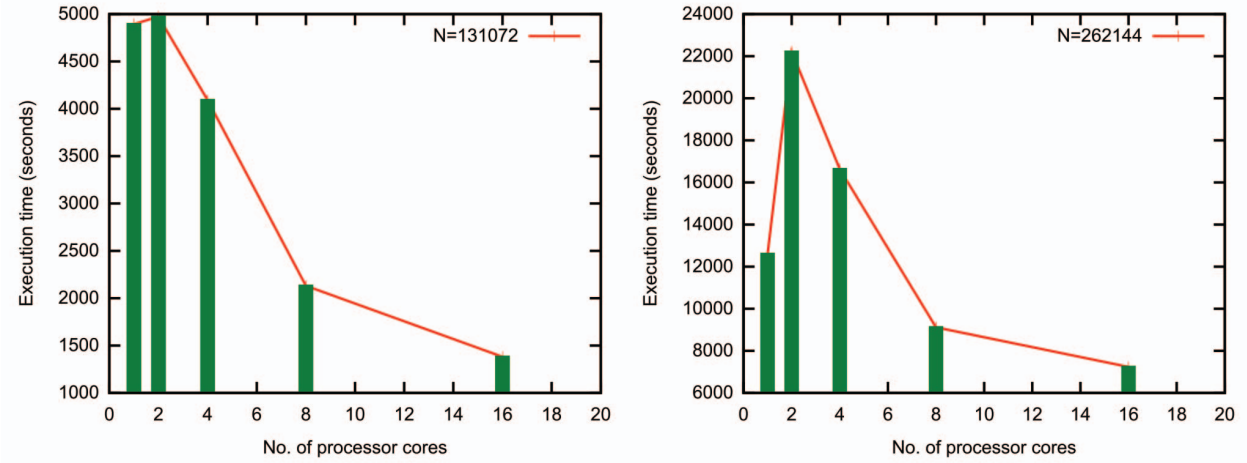

Figure 7: Parallel simulation execution times (N=131,072 and N=262,144).

6.1 Population Sizes and Execution Times

The size N=1024 is the population size on which past work on the Naming Game typically focused for experiments. Since the problem size is small, the parallel processing overheads are relatively significant when moving from sequential to 2 processors. However, the time continually decreases as more processors are added, finally registering speed up after 4 processors. This is as expected, since the dictionary sizes remain small, and hence the per-conversation computation is very fine-grained, making inter-processor event exchange overhead seem relatively large.

As the population increases, the individual dictionary sizes also increase. The maximum dictionary size in the system is theoretically estimated to grow as square root ($\sqrt{N}$). Hence, when the population is increased nearly one order of magnitude, to N=8096, the scaling is improved. With further increases in the population size, improved scaling is observed, as shown in Figure 7 for N=131,072 and N=262,144.

6.2 Conversations to Reach Consensus and Execution Time per Conversation

The number of conversations needed to reach consensus varies with the population size. This metric is charted in Figure 8 for increasing population sizes. The general trend is roughly estimated as $3N\sqrt{N}$ conversations to reach consensus in a population of size N. Because the number of conversations needed to reach consensus varies with the population size, the execution time amortized per conversation varies both with population size as well as the number of processors used in the simulation. These relations are charted from experiments in Figure 9. The amortized cost generally increases with population size but *not proportionately* because the probability of communication with hearers within the same processor increases with larger population sizes. When moving from one processor to two processors, there is an increase in cost because of inter-processor communication; however, thereafter there is a definite and steady decrease with increasing number of processors.

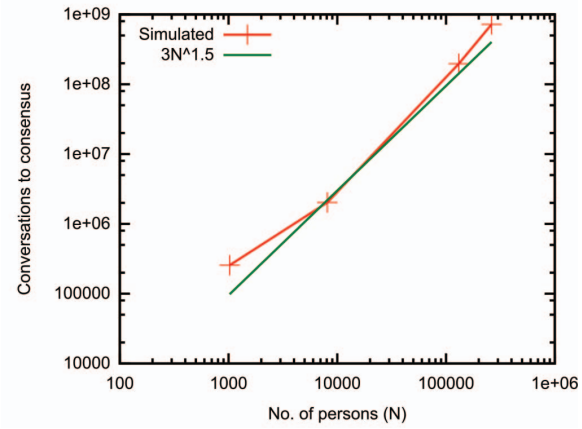

Figure 8: Time to reach consensus for increasing populations.

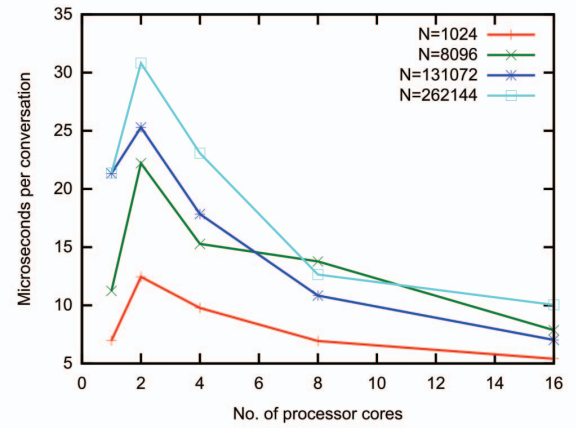

Figure 9: Average execution time per conversation.

7 SUMMARY AND FUTURE WORK

The classical sequential algorithm of the Naming Game is revisited with a view to resolving its limitations. A new concurrent algorithm has been developed and presented in terms of concurrency via asynchronous and simultaneous conversations, non-instantaneous conversations, and discrete event style of execution, in contrast to sequential, instantaneous, and time-stepped style of execution of the classical algorithm. A prototype of the concurrent model has been implemented in a parallel discrete event simulation, and executed on multi-core machines. The model has been scaled to large populations (up to 0.25 million agents); further scaling is currently being investigated via distributed memory processors. A performance study of the scaled execution has been presented. The consensus dynamics have been compared between the sequential and concurrent models, uncovering some phenomena that were missed by the sequential algorithm in larger networks, such as a reduced upper bound on the average dictionary size and the increased time to reach consensus, all due to simultaneity of conversations. This concurrent, discrete event model enables more realistic and larger network phenomena to be modeled than before.

In general, social behavioral modeling efforts need to bear concurrency from the outset, instead of starting with a time-stepped operation as the default premise. Otherwise, this can hurt parallelization. A prime example is the original Schelling Segregation model (Shelling 1971), which is the earliest example

of a behavioral model in which sequential (time-stepped) style is built into the model definition itself. The relaxation for concurrency in the segregation model leads to an even more realistic model. However, such relaxation needs a conflict resolution component, analogous to the conflict resolution of a speaker for hearing-while-speaking. Our concurrent model here serves as an illustrative example to guide in introduction or enhancement of concurrency in social behavioral models.

REFERENCES

Andelfinger, P., K. Jünemann, and H. Hartenstein. 2014. "Parallelism Potentials in Distributed Simulations of Kademlia-based Peer-to-Peer Networks". In *Proceedings of the International ICST Conference on Simulation Tools and Techniques*, 41–50.

Baronchelli, A. 2016. "A Gentle Introduction to the Minimal Naming Game". Belgian Journal of Linguistics 30(1):171–192.

Baronchelli, A., M. Felici, V. Loreto, E. Caglioti, and L. Steels. 2006. "Sharp Transition Towards Shared Vocabularies in Multi-Agent Systems". *Journal of Statistical Mechanics: Theory and Experiment*.

Baronchelli, A. 2011. "Role of Feedback and Broadcasting in the Naming Game". *Physical Review E* 83(4):046103-1–6.

Bauer, D. W. Jr., C. D. Carothers, and A. Holder. 2009. "Scalable Time warp on Blue Gene Supercomputers". In *Proceedings of the Workshop on Principles of Advanced and Distributed Simulation*, 35–44.

Cecin, F. R., C. F. R. Geyer, S. Rabello, and J. L. V. Barbosa. 2006. "A Peer-to-Peer Simulation Technique for Instanced Massively Multiplayer Games". In *Proceedings of the IEEE international symposium on Distributed Simulation and Real-Time Applications*, 43–50.

Franks, H., N. Griffiths, and A. Jhumka. 2013. "Manipulating Convention Emergence using Influencer Agents". *Autonomous Agents and Multi-Agent Systems Journal* 26(3):315-353.

Hanai, M., and K. Shudo. 2014. "Optimistic Parallel Simulation of Very Large-Scale Peer-to-Peer Systems". In *Proceedings of the IEEE/ACM 18th International Symposium on Distributed Simulation and Real Time Applications*, 35–42.

Lass, R. 1997. "Historical Linguistics and Language Change". Cambridge University Press.

Lu, Q., G. Korniss, and B. K. Szymanski. 2008. "Naming Games in Two-dimensional and Small-World-Connected Random Geometric Networks". *Physical Review E* 77(1):016111-1–10.

Lu, Q., G. Korniss, and B. K. Szymanski. 2009. "The Naming Game in Social Networks". *Journal of Economic Interaction and Coordination* 4(2):221–235.

Mattern, F. 1993. "Efficient Algorithms for Distributed Snapshots and Global Virtual Time Approximation". *Journal of Parallel Distributed Computing* 18:423–434.

Naicken, S., B. Livingston, A. Basu, S. Rodhetbhai, I. Wakeman, and D. Chalmers. 2007. "The State of Peer-to-Peer Simulators and Simulations". *Computing and Communications Review* 37(2):95–98.

Quinson, M., C. Rosa, and C. Thiery. 2012. "Parallel Simulation of Peer-to-Peer Systems". *Proceedings of the IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 668–675.

Schelling, T. C. 1971. "Dynamic Models of Segregation". *Journal of Mathematical Sociology* 1(2):143–186.

Steels, L. 1995. "A Self-Organizing Spatial Vocabulary". *Artificial Life* 2:319–332.

AUTHOR BIOGRAPHIES

KALYAN S. PERUMALLA is a Distinguished Research Staff Member and manager in the Computer Science and Mathematics Division at the Oak Ridge National Laboratory, USA, where he leads the Discrete Computing Systems Group. He is also an Adjunct Professor in the School of Computational Sciences and Engineering, Georgia Institute of Technology, USA. His e-mail address is perumallaks@ornl.gov.