Simulating the cryosphere: a 14 billion-cell *Giraph* automata model of Antarctic ice sheet melting

Hai Lan, Research Assistant, Department of Computer Science and Engineering, Tandon School of Engineering, New York University, 370 Jay Street, Floor 12, Brooklyn, NY 11201, USA. Email: hai.lan@nyu.edu.

Paul M. Torrens, Professor, Department of Computer Science and Engineering, Tandon School of Engineering, New York University and Center for Urban Science + Progress, 370 Jay Street, Floor 12, Brooklyn, NY 11201, USA. Email: torrens@nyu.edu.

Abstract

The swelling of the huge troves of data produced for Earth Science phenomena are increasingly placing significant strain on modeling and cyberinfrastructure, with the result that our models are often unable to match pace and parity with the data-sets that are available to feed them. The gap between data and models artificially restrains the pace of insight that might otherwise be advanced if models could add value, through simulation, to available data. In this paper, we suggest a novel pathway that could help to reconcile big data to big models with relative congruity in detail and scope. We introduce a computing solution that makes use of voxel cellular automata to animate ice-melting dynamics with massive detail. We demonstrate the approach using a physical model of ice melting, implemented in a ~14 billion-cell automata lattice, and accelerated using graph-computing on a cloud environment. Our approach offers unprecedented detail and dynamics in simulating Antarctic ice-sheet melting, while also easing some of the burdensome attributes of re-tasking models to fit new scenarios and new data. While our approach is prototypical, it suggests several avenues for further development beyond current limitations of traditional modeling schemes, with promise of extensibility to significant explorative reach.

Keywords: voxel cellular automata, *Giraph*, ice dynamics, cloud computing, Earth Science

"I find it looks the same, but everything has changed" (Reznor *et al.* 2013)

1 Introduction

While mathematical and computer modeling have co-developed alongside data as significant exploratory media for Earth scientists' questions and hypotheses, the relationship between data and models is not always congruous. Indeed, the synergy between models and data can be strained when the size, detail, or volume of data about the Earth's surface incoming to scientists—and the innovation in investigative reach those data carry—outstrips our ability to adjust the computer models that we feed those data to. When computer models serve as the primary medium for inquiry, mismatches between data and computing can slow Earth science's ability to produce insight, which in turn is problematic when the phenomena under study are themselves changing rapidly or suddenly through mechanisms that we must scramble to frame and explain on the ground. Tensions between fast-paced phenomenological change, rapid advances in incoming data, and the commensurate abilities of numerical models and computer modeling and simulation to keep up are of particular concern in the study of cryosphere dynamics in Antarctica. In this paper, we suggest a coupled simulation and computing framework that we argue can grow as newly incoming data grow, and that can make full use of the massive observational scope that now characterizes discovery relative to the cryosphere.

Fantastic new data sets for Antarctica have been introduced over the last decade or so, greatly expanding the potential for us to build new understanding of ice sheet dynamics. These data come from platforms such as Ice, Cloud, and land Elevation Satellite (*ICESat-1*) (Schutz *et al.* 2005), *CryoSat* (Wingham et al. 2006) and *CryoSat-2* (McMillan *et al.* 2014), *IceBridge* (Kurtz and Farrell 2011), and radar instruments placed in situ (Giles *et al.* 2008). The vista that these platforms provide on the cryosphere is massive in scope and developments are set to continue that trend, adding further detail to the observational capabilities. For example, after NASA *ICESat-1* de-orbited in 2010, NASA *IceBridge* was introduced to bridge the gap between NASA *ICESat-1* and NASA *ICESat-2*, which is planned for 2018 (National Aeronautics and Space Administration 2017a). Like *ICESat-1*, *IceBridge* deployed laser altimeters with 15cm vertical resolution (National Aeronautics and Space Administration 2017b).

One can easily foresee a future situation in which entire ice sheets are observed at levels of resolution that could stretch into the tens of centimeters. Estimates on the *Bedmap2* data set place the area of the Antarctic ice sheet at 13.924 million km^2, and its volume at 26.92 million km^3 (Fretwell *et al.* 2013). If we consider ice thickness data from *IceBridge* (Blankenship *et al* 2017), that this affords the satellite a vista of ice sheet height in 400 m x 400 m x 8 m resolution = 1.28 x 10^{-3} km^3 patches over the Antarctic area. The next generation of ICESat (*ICESat-2*) will add further data to this silo, with unprecedented resolution (70 m along-track resolution and 5 cm vertical resolution at 25 km x 25 km spatial scales at less than 1° slope area (Markus *et al* 2017)). In reality, the path over which the satellite traverses the continent, and the high-level products and calculations that can be derived from the data can either increase or decrease that vista. If we assume, roughly, that this corresponds to observational patches of 0.03125 km^3, we will soon need models with lattices of ~861 million voxels to project surface observations through the sub-surface ice. We may actually need many multiples of that number.

In short, as data volumes increase, we need models that can consume those data in all of their “big size”. This means that we need models that can accommodate dynamics across very large lattices of simulation space. This requires significantly more computing than simply inputting big data, as each “piece” of data is exposed to many, many operations in simulation, with the result that large multiples of the input data are required to resolve computational operations. Consider, however, the further complication that we need models that can *get bigger* as new data are incoming, and it is soon apparent that resolving big models to big data becomes a far-from-straightforward task: elementally new approaches to the computing that supports models and simulations are required. To our knowledge, no ice sheet models are even close to representing a level of detail that would accommodate the ~861 million voxels needed to match near-horizon data tapestries for the Antarctic ice sheet.

In this paper, we introduce a preliminary approach to address issues of mismatch between data and simulation capabilities, primarily through computing that can marry the two. Our aims are as follows. We wish to facilitate “big models”, capable of supporting the representation of large volumes of ice, up to and perhaps including whole-continent models at incredibly fine resolutions, ideally matching best-available data to parameterize them. Both models and simulation should be capable of ingesting and poring over the huge detailed data sets currently being provided, or soon to be provided, by observational remote sensing of ice sheet elevation. And, the models should be future-proof relative to current trends in the growth of data size and resolution, with ability to flex to respond nimbly to the sorts of questions that new data will allow us to pose in simulation. To achieve these goals, we introduce a scheme that permits us to (1) build a theoretically-plausible physical model of ice sheet melting, (2) to implement that physical model in massively detailed iconic form with commensurately detailed process dynamics as voxel-based cellular automata (CA), (3) to ease and accelerate the computing of the CA using large-scale graph processing and graph-based computing, and (4) to prove the viability of the concept with a tangible simulation on cloud computing environment.

We will demonstrate a proof of concept simulation of Antarctic ice melting with ~14 billion voxels in a cellular automata simulation framework. This provides a multiplier of *~16 times* the near-horizon best available sheet-wide observation data, projected to the ice sub-surface. The dynamics of the model focus on *ice melting*. While melting is admittedly only part of the story that explains ice sheet dynamics, it is indicative of the volume of interactive capability that our framework can support. We focus our attention on a universal model of melting, rather than optimizing a single specific model that can only be used in a narrow area. However, our model could (and should) be tailored to fit local conditions. We will explain the functioning of the model with synthetic data at finer resolution than currently available data. However, we anchor the model to known meteorological data for the Larsen Ice Shelf in Antarctica. It is hopefully straightforward for the reader to see that real data could be used to benchmark simulation to ice sheet topography data and other knowns that could be provided by remote sensing and observation, and we show how ground truth can be introduced to the model with a straightforward parameterization step.

2 Cellular automata as a foundation for ice sheet modeling

We base our modeling and simulation (and computing) approaches on CA models. CA provide us with a robust foundation for constructing models that are flexible (the ability of the model to bend to fit different formulations) and extensible (the ability of the model to grow in size of representation and volume of computation required to resolve). A typical application of CA in modeling physical phenomena uses the cells of CA lattices as (computational) media over which classic functional or numerical equations can be run as transition rules. Bak *et al.* (1988) introduced the earliest CA models in Earth Science to demonstrate the role of complexity (self-organized criticality in particular) in shaping intricate dynamics of forest fire spread. CA simulations have also been developed to model other process-and-pattern relationships in physical systems, particularly in physical geography where the spatial positioning of cells amid the wax and wane of local-to-global dynamics can be examined, for example, in granular flow (Baxter and Behringer 1990), debris flow (D'Ambrosio *et al.* 2003), avalanches (Kronholm and Birkeland 2005), and crystallization dynamics (Raabe 2002).

CA have found use in glaciology, where the framework provides a useful bridge between (often competing) deterministic and stochastic methods (Fonstad 2006), and where CA may reconcile natural synergies with grid-based data from remotely-sensing platforms (Burrough 2001) or field observation data (Faria *et al.* 2002; Kronholm and Birkeland 2004). CA are of particular use in modeling ice sheet dynamics, where expanses of ice cover are often discretized into lattices, within which numerical simulations are run based on parameters garnered from pixel-based observation from remote sensing devices. For example, Bahr and Rundle (1995) developed a CA model of glacial ice flow using Navier-Stokes equations (Stokes 1845) for fluid flow (similar to lattice-gas automata introduced by Frisch *et al.* (1986)). Brown *et al.* (2010) used a CA model to explore the interplay of mass balance gradient, aspect, avalanching, and ice flow on glaciers in the Rocky Mountains of the United States under different warming scenarios. Ktitarev *et al.* (2002) used CA to explore the microcosm of ice sheets, focusing the CA lattice on problems of ice crystal orientation.

Although CA models have been widely used in diverse scientific research in real world, most of them are not simulated over large lattices, especially in voxel cases. For example, Jjumba and Dragićević (2014; 2016b) introduced a set of innovative voxel-based CA models with an example simulation for querying snow cover (Jjumba and Dragićević 2016a). The number of voxels in their model is not explicitly stated, but their queries suggest that the CA lattice they use has ~25,000 cells in its largest implementation. The vast majority of cellular automata simulations still run on single workstation implementations, limiting the computing capacity of the CA to a few CPU cores. There are some implementations of large-scale CA models—*outside the Earth Sciences*—that take advantage of cloud computing to accelerate the computing required to resolve cell state transitions. Radenski (2013) and Marques *et al.* (2013) demonstrated how the mathematical rule-set for the *Game of Life* (GoL) (Gardner 1970) CA simulation could be run with *MapReduce* on *Amazon Web Services* and Microsoft's *Azure* platform respectively, while Gropp and Lusk (2005) demonstrated acceleration of GoL using *MPI-2*. These examples suggest a promising path that the Earth sciences might investigate to resolve the challenge of building flexible and extensible simulation platforms for future models: accelerating the computing at least removes that bottleneck to growing the simulations to a size that could match data. The GoL is, however, a simple and well-understood "toy" case, in so much as it produces universal computation (Wolfram 1984b) but is not intended to mimic real-world phenomena.

Building *scientific models* (with much more complex state and rule-sets, and more intricate cell-to-lattice relationships via neighborhood functions) on large CA platforms requires some further work. We see two areas, in particular, that could better support applied CA modeling: (1) iterative processing, and (2) high scalability. Further, we argue that graph-based implementations of CA on these schemes afford additional advantages.

3 Graphs and *Giraphs*

In this paper, we contend that traditional array-based CA frameworks are unsuitable candidates for resolving the flexibility and the extensibility required to advance ice sheet modeling to parity with incoming data and the explanatory dynamics that detail might require in simulation. We propose a restructured version of CA, still faithful to the original computational specification of CA as finite state machines with neighborhood polling, but shifted in how CA components are *built*. In particular, we employ *graph theory* to re-build computable CA structures.

Traditional CA models are composed of sets of cells, states, state transition rules, and neighborhoods. How neighbors are derived is left loosely open to interpretation; indeed, the topic of how the local milieu around a CA cell should be considered is a topic of interest in geography, where different schemes have been proposed, including graphs (Semboloni 2000; Torrens and Benenson 2005). In graph theory (Bondy and Murty 1976), a graph (G) is an ordered or unordered pair.

$$G = (V, E) \qquad (1)$$

Graphs (*G*) are posed as a 2-element tuple, containing a set (*V*) of vertices and a set (*E*) of edges. These components are useful in a CA framework, where cells can be expressed as vertices and neighboring relationships between them can be (flexibly) handled by edges (Machì and Mignosi 1993; Wolfram 1984a). Through judging the state of a vertex connected with the target vertex, users can easily establish (contextual) transition rules to update the state of the target vertex. Building CA in graph form offers significant computational *flexibility*. When *framing* model algorithms, nodes and relationships can be flexibly expressed through consideration of vertex value and edge weight to accommodate specification of attributes and relationships. When *implementing* algorithms, modelers can straightforwardly code a single graph-based CA model, once, and at any scale. This would help to reduce the time that a user spends in converting a model into algorithmic form. For example, a model framework/coding schema can easily fit to any scale or scenario by changing the parameters of nodes, the edge link scheme, the vertex value and the edge weight, depending on a specific scale or application context. Indeed, multiple scales can be analyzed during one process, without having to address the necessity of altering the main structure of the program. This can save a model-builder considerable effort that would otherwise be invested in re-developing many different model executors for different cases. Graphs may also simplify CA modeling, especially, in high dimensions. Consider, for example, that modelers no longer need to care about how to represent real-life high dimensional scenarios in CA models. Instead, they can simply change the edge link schema in the graph CA structure to convert models between different dimensions. These attributes of graph-based CA address the first of our self-imposed challenges for ice sheet modeling, providing the flexibility to develop CA models of ice sheet dynamics.

The second of our self-imposed challenges—extensibility (alongside flexibility)—is somewhat more involved to resolve. Simple modeling and implementation of algorithms will not significantly boost CA models' computing capabilities. CA are assemblies of finite state machines: CA states, rules, and cells use-up computing resources as they grow and those facts are so far unavoidable (Asanovic *et al.* 2006). In theory, CA should be massively autonomous, with each cell poring over state information relative to its transition rules, *in parallel*. In practice, this is difficult unless single units are devoted to cells. And even if applied to a general-purpose computing on graphics processor units (GPGPU), additional challenges of passing state data among cells and programming in high dimensional scenarios are known to arise (Gobron *et al* 2011). Nevertheless, a graph-based CA model can exploit *large-scale graph processing frameworks* to significantly increase computational capabilities. In this case, the graph can provide the extensibility in unit count that we need. Moreover, it is possible to provide this extensibility in ways that do not necessarily require that a model-builder modify the model code. This suggests that the graph can adjust (relatively) independently of the model code.

There are a few candidate schemes that could provide large-scale graph processing at the sorts of scales that our many-million-voxel ice sheet example could feasibly address (indeed, we will shortly demonstrate a proof of concept on many-billion voxel CA). The Bulk Synchronous Parallel (BSP) model developed in the 1980s by Valiant (1990) contains many supersteps (discrete units of time over which CA cells exchange state information) where, in the operation process of each superstep, each computing unit is arranged into a certain amount of vertices or edges, which ensures parallel computing. Discrete computing units communicate with one another by message interaction. When computing of a given unit reaches a barrier, the unit will stop until other units complete their message interaction. *Apache Giraph* (Ching 2013) is one important implementation of the BSP model. *Giraph* is an open source software framework for large-scale graph processing, a vertex-centric model, based on *Google*'s *Pregel* (Malewicz *et al.* 2010). Work by Ching (2013) at *Facebook* has shown that *Giraph* can analyze one trillion graph edges using 200 machines in four minutes (Ching, 2013). *Giraph* lies on top of *Apache Hadoop*, so that *Giraph* can be easily deployed and implemented on any *Hadoop Distributed File System* (HDFS) based computing platform (Shvachko *et al.* 2010). *Apache ZooKeeper* (Hunt *et al.* 2010) is an important component to help *Giraph* "workers" (nodes that will perform the computing) coordinate computation within a cluster of worker nodes. When the *Giraph* job is submitted, one worker is elected as a master by *ZooKeeper*. At the same time, the input graph will be loaded and vertices or edges will be partitioned and assigned to workers. After executing compute-functions in code for certain supersteps or certain halt conditions, workers will save the output back to the DFS. In this paper, we argue that this functionality can provide the *extensibility* that we think is necessary to advance computing for ice sheet models. At the same time, large-scale graph processing can address flexibility, as we mentioned. All the while, the perspective from a model-user *coding* a simulation remains largely unchanged. Moreover, *Apache Giraph* can be deployed on cloud platforms such as *Amazon Web Services* (AWS) (Amazon 2018; Wittig and Wittig 2015), *Microsoft Azure* (Microsoft 2018; Wilder 2012), and *Google Compute Engine* (Cohen *et al.* 2015; Google 2018). Those cloud platforms provide Infrastructure as a Service (IaaS), Platform as a Service (PaaS), or Software as a Service (SaaS), with flexibility to design custom systems of computing resources for different computing needs. Virtual instances, applications, and software are provided as requested and users can pay on demand for those services. In other words, users can build a computing system with (effectively) unlimited computing capacities to implement the processing of models in theory (albeit, limited by computing frameworks and budget in real life).

In what follows, we will describe a workflow to optimize traditional CA models for the dual task of flexing relative to large volumes of input data at high resolution, while forging extensibility against likely growth in those data, and simultaneously retaining relative straightforward model-coding capabilities for scientists. These goals are accomplished in three steps (Figure 1). First, we re-structure CA modeling with graph theory. Second, we exploit the *Giraph* computing framework to significantly improve CA computing capabilities. Third, we access cloud computing platforms to gain sufficient computing resources.

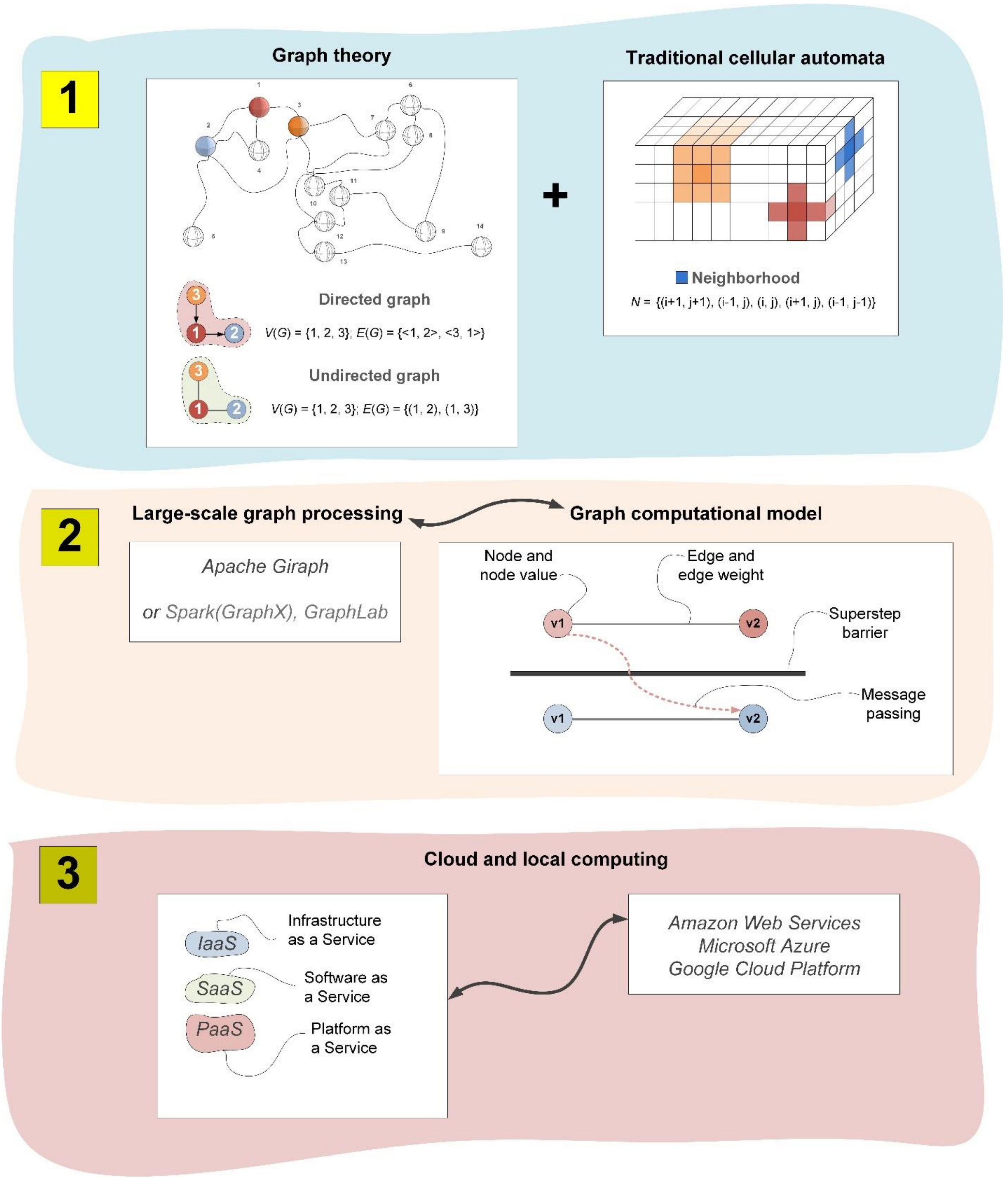

1
Graph theory
Directed graph
V(G) = {1, 2, 3}; E(G) = {<1, 2>, <3, 1>}
Undirected graph
V(G) = {1, 2, 3}; E(G) = {(1, 2), (1, 3)}
+
Traditional cellular automata
Neighborhood
N = {(i+1, j+1), (i-1, j), (i, j), (i+1, j), (i-1, j-1)}
2
Large-scale graph processing
Apache Giraph
or Spark(GraphX), GraphLab
Graph computational model
Node and node value
Edge and edge weight
Superstep barrier
Message passing
3
Cloud and local computing
IaaS
Infrastructure as a Service
SaaS
Software as a Service
PaaS
Platform as a Service
Amazon Web Services
Microsoft Azure
Google Cloud Platform

Figure 1. Work flow of the accelerated CA model and its implementation in cloud computing.

4 Building a *really* big ice melting model

Ice sheets in Antarctica are significant to the Earth sciences across several axes of consideration. Ice dynamics can significantly influence sea level (Shepherd and Wingham 2007), especially in the Southern Ocean (Doake *et al*. 1998), and competing theories of what these effects might be have prompted significant debate for some time (Mercer 1968). Increasing instability of flux ice has been allied to global climate change (Oppenheimer 1998), prompting many to call for enhanced detail in ice sheet simulation to explore what these connections might be (Ktitarev *et al.* 2002). Short-term and medium-term ice sheet dynamics are driven by a variety of forces. We do not fully understand what they might be and how they interplay, and simulation is an important medium for building the science to guide us in finding answers. Traditionally, an array of models based on differential and difference equations from glacier dynamics are used as the basis for modeling ice sheets, focusing on melting (Van der Veen 2013), flux (Kwok and Rothrock 1999), and calving (Benn *et al.* 2007). A more recent trend is a call for the community to develop integrated models that ally these processes, e.g., to treat the interplay among acceleration, calving, and thinning in ice sheets and their surrounding environments (Benn *et al.* 2007). Addressing these connections will require really big, really detailed models to pull off, and it will require detailed data in massive volumes to keep the models on the rails.

Melting/precipitation and ice flow are the two most important components in ice dynamics (Hock 2005). Our starting point in building a model of detailed dynamics in Antarctic ice sheets is Van Der Veen's (2013) glacier thermodynamics theories, which we use as the foundation for a computer model and sets of simulation runs that explore one component—melting—as among the most basic and important elements of ice dynamics. Our aim is to model melting with a rich set of driving factors, including monthly average temperatures, ice albedo, solar radiation, sun zenith angles, and wind velocity. We make use of publically-available climate data collected at Larsen station in 2000 (American Museum of Natural History *et al.* 2002) to provide these data and to parameterize our model in simulation, but we use a synthetic ice sheet for initial conditions. Our rationale for mixing real climate data with synthetic base conditions for ice sheet topography is a function of the meager availability of ice sheet thickness data over very large areas with fine resolution. These data are actively being collected, but are not yet available. Indeed, the overarching premise of our paper is that we should *prepare* models for next-generation data at coverages and resolutions that exceed current capabilities. The synthetic base height data that we use is of the same format as data that will likely be incoming from upcoming missions such as *ICESAT-2*, for example. To assess the feasibility and efficiency of our CA model, here, we show a model on synthetic data at aspirational detail of 1 meter spatial resolution over a coverage of 7.8 km x 3.6 km area and 500 meters depth. This is achieved using ~14 billion automata cells in simulation.

4.1 A physical model of ice sheet melting

Van der Veen (2013, p.145) discusses four analytical assumptions made by Robin (1955) in deriving equations for the thermodynamics of glaciers: (1) basal temperature is lower than the pressure-melting temperature for cold glaciers that are frozen to their bed; (2) through most components of the glacier, diffusion of heat in the horizontal is smaller than that of heat in the vertical, and horizontal temperature gradients are smaller in size than those in the vertical; (3) the ice sheet is in steady state, such that vertical velocity is zero; and (4) advection of heat in the horizontal can be ignored if the horizontal velocity is very small.

Working from these assumptions, we specified a model ice sheet with the following features. We consider a cold glacier; heat exchange between ice and the ground below ice were not be considered (Figure 2). To simplify the model, changes in the melting point due to pressure will not be treated. We represent the whole ice sheet as a composite of one cubic meter units (these can be of two discrete types, either "ice voxels" or "air voxels"; types are not states, rather, the typology determines the parameters of states, rules, and neighborhoods that those CA types operate under). We do not consider any horizontal energy flow between ice voxels because temperature gradients in the horizontal should be relatively small, particularly when ice voxels are laterally symmetric. Furthermore, heat transfer in the horizontal (from the ice sheet edge to its middle locations) should occupy an extremely long time period relative to the transfer among ice voxels in our very large-sized sheet (~14 billion ice voxels). Consequently, we may consider the side of ice voxels as an adiabatic surface. We also specify the model lattice as a steady state ice sheet, such that the redistribution of heat by downward ice flow can be ignored, because the horizontal velocity is either very small or zero. Further, we do not handle internal deformation of the sheet during ice melting. We therefore model the sheet's melting components alone, rather than treating internal ice flow. We consider the influence of the zenith angle of the sun on sheet melting, by representing solar radiation as uniform from the top to the bottom of the CA lattice. We should note that the aforementioned simplifications are for parsimony in our simulation experiments, but that there is nothing limiting in the framing of the model that would prevent the inclusion of omitted functions: the CA framework and our approach to accelerating it in computing is incredibly flexible to specification. The ice-melting framework as represented in our model is illustrated in Figure 3.

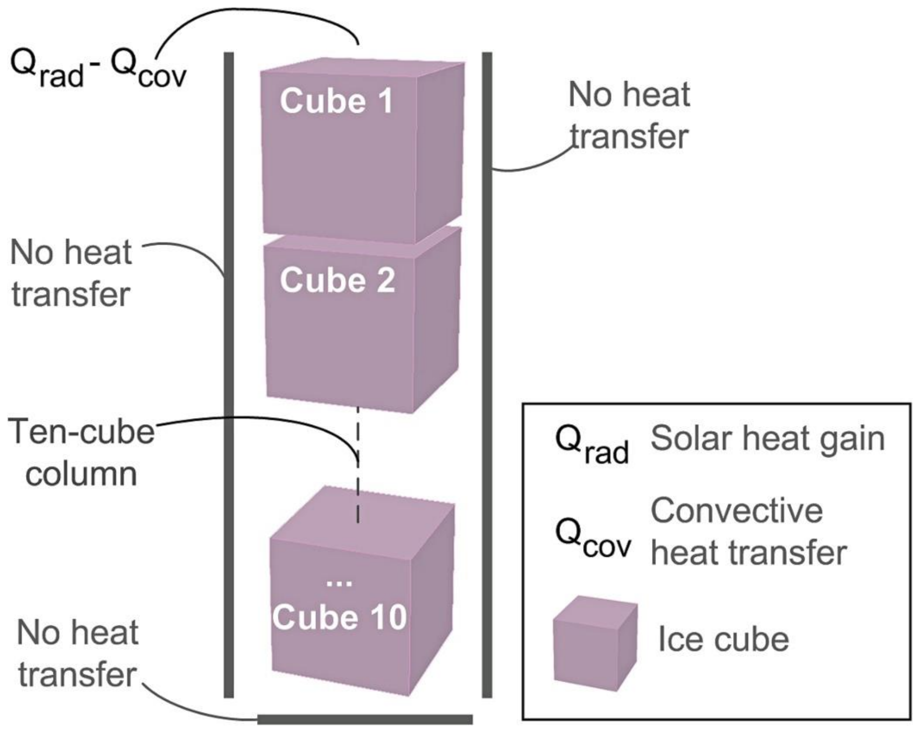


Figure 2. Schematic representation of heat in the ice-melting model.

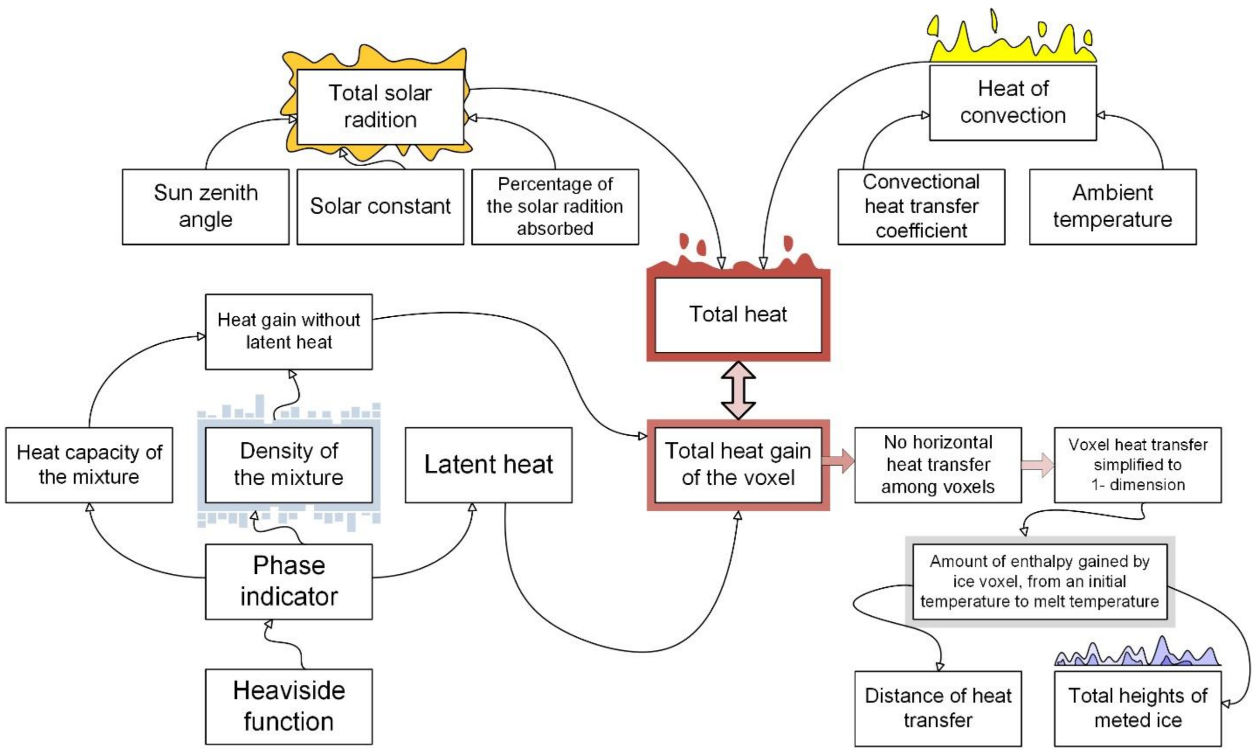

Total solar radition
Heat of convection
Sun zenith angle
Solar constant
Percentage of the solar radition absorbed
Convectional heat transfer coefficient
Ambient temperature
Total heat
Heat gain without latent heat
Heat capacity of the mixture
Density of the mixture
Latent heat
Total heat gain of the voxel
No horizontal heat transfer among voxels
Voxel heat transfer simplified to 1- dimension
Phase indicator
Amount of enthalpy gained by ice voxel, from an initial temperature to melt temperature
Heaviside function
Distance of heat transfer
Total heights of meted ice

Figure 3. Flow chart illustrating the main components of the ice melting physical model.

Also shown in Figure 3, is our inclusion of an energy balance partial differential equation in the model, with phase change treated as below (Weigand 2004, p. 22):

$$\rho C_p \frac{\partial T}{\partial t} = \frac{\partial}{\partial z}\left(k \frac{\partial T}{\partial z} \right) \qquad (2)$$

Above, T denotes the temperature, p denotes the density of the mixture of ice with water in the ice voxel, C_p denotes the specific heat capacity of the mixture, t denotes the time (month), z denotes the vertical position of the ice cube column, and k is the heat transfer coefficient of the mixture.

We may also use a phase indicator (θ) to index the state of the ice voxel as ice/water within a single voxel (Mirjalili *et al.* 2017) (below, L represents the latent heat):

$$k = \theta k_{ice} + (1-\theta) k_{water} \qquad (3)$$

$$C_p = \theta C_{p,ice} + (1-\theta) C_{p,water} + L \frac{d\theta}{dT} \qquad (4)$$

$$\rho = \frac{\theta \rho_{ice} C_{p,ice} + (1-\theta)\rho_{water} C_{p,water}}{\theta C_{p,ice} + (1-\theta) C_{p,water}} \qquad (5)$$

θ may be calculated using a Heaviside function (Abramowitz and Stegun 1972, p. 1020) with changing temperature T_{trans} during heat transfer. x denotes a discrete variable:

$$\theta = Heaviside(\frac{T - T_{trans}}{dT}) \qquad (6)$$

$$Heaviside(x) = \begin{cases} 0 \ , x < 0 \\ 0.5, x = 0 \\ 1 \ , x > 0 \end{cases} \qquad (7)$$

Following Ding *et al* (2017), an estimate of the distance of heat transfer (D_{ht}) can be calculated by comparing the value of the enthalpy that each ice voxel has gained. Based on the enthalphy differential equation of Klotz and Rosenberg (2008, p. 63), enthalphy difference can be accounted for as follows:

$$dH = \frac{\partial H}{\partial T} dT + \frac{\partial H}{\partial P} dP \qquad (8)$$

Above, P donates pressure. We may consider atmospheric pressure relative to the ice sheet surface as relatively stable, so we ignore pressure differences and then simplify our consideration of the enthalpy difference to:

$$dH = C_p dT + L \qquad (9)$$

Above, H denotes the enthalpy, and L is the latent heat of the phase change. The amount of enthalpy accrued to an ice voxel from initial temperature to melting temperature (ΔH_{pc}) is steadfast, which serves as a waypoint value for the distance of heat transfer, such that:

$$D_{ht} = \frac{\Delta H}{\Delta H_{pc}} \qquad (10)$$

Recall that we do not consider heat transfer at the bottom boundary in our conceptualization, but we do treat radiant heat (Q_{rad}) and convective heat (Q_{cov}) at the top boundary. As a result, the two boundaries for closing the model can be specified as:

$$Top: \begin{cases} k \frac{\partial T}{\partial z} = Q_{rad} + Q_{\text{cov}} \\ Q_{rad} = I_0 \sin \vartheta_0 \alpha \\ Q_{\text{cov}} = h_{amb} \left(T_{amb} - T \right) \end{cases} \qquad (11)$$

$$Bottom:\ \frac{\partial T}{\partial z} = 0 \tag{12}$$

Above, h_{amb} denotes the heat transfer coefficient between ice voxels, T_{amb} denotes the ambient temperature, and I_0 denotes the solar constant. Term α follows work by Zhang *et al* (2012) and denotes the ratio (percentage) of the solar radiation absorbed by the Earth to the total solar radiation. ϑ denotes the solar altitude angle distribution function. Term ϑ_0 denotes the daily average solar altitude angle. The value of term ϑ_0 may be calculated as follows:

$$\vartheta = \frac{\vartheta_{\max}}{36} t^2 + \frac{\vartheta_{\max}}{3} t \tag{13}$$

$$\int_0^{12} \vartheta dt = 24\vartheta_0 \tag{14}$$

Equations (13) and (14) offer an approximate method for calculating the value of ϑ_0, by assuming that the solar altitude angle allies with the parabolic profile over a time interval, spanning from 6 a.m. to 6 p.m. Known data suggests that this is a reasonable approximation. The term ϑ_{max} represents the maximum of the solar altitude angle. The value of the solar constant may be considered as 1,353 W/m^2 (Thekaekara 1973). We can therefore consider parameter values for solar radiation as described in Table 1.

Table 1. Parameters for Solar radiation.

name	I_0	ϑ_{max}	a
unit	J/month	degree	-
value	3.506976e^9	20	46%

The heat transfer coefficient (h_{amb}) can be calculated as follows:

$$h_{amb} = \frac{Nu \cdot k_{amb}}{l} \tag{15}$$

$$Nu = 0.664 \mathrm{Re}^{0.5} \mathrm{Pr}^{1/3} \tag{16}$$

$$\mathrm{Re} = \frac{ul}{\upsilon} \tag{17}$$

Above, *Nu* is the Nusselt number (Yunus 2003, p. 358), *Re* is the Reynolds number (Rott 1990, p. 3), *Pr* is the Prandtl number (White and Corfield 2006) (considered as 0.7 for air in Table 3), term *u* represents wind velocity, term *l* indexes the length of the local ice voxel, υ denotes the kinematic viscosity, and k_{amb} represents the heat transfer coefficient of the air.

To start the solution procedure, initial seed conditions should be specified as states for the CA. In particular, we should specify the initial temperature of the ice voxel *column*. Using data for Larsen, we initiate the solving procedure (month zero) with an initial temperature of -14 Celsius (259.15 Kelvin), such that:

$$T_{ice}(x,0) = 259.15K \tag{18}$$

To solve the model, the continuous Equation (2) should be placed in discrete form, which we accomplish using the Lax-Wendroff method (Lax and Wendroff 1960), as follows:

$$\rho C_p \frac{T_z^{t+dt} - 1/2\left(T_{z+dz}^t + T_{z-dz}^t\right)}{dt} = k\frac{T_{z+dz}^t + T_{z-dz}^t - 2T_z^t}{dz^2} \tag{19}$$

$$T_z^{t+dt} = \frac{dt}{dz^2}\frac{k}{\rho C_p}\left(T_{z+dz}^t + T_{z-dz}^t - 2T_z^t\right) + \frac{1}{2}\left(T_{z+dz}^t + T_{z-dz}^t\right) \tag{20}$$

Now, the model may be solved, by dividing one ice voxel into 10 units as shown in Figure 4.

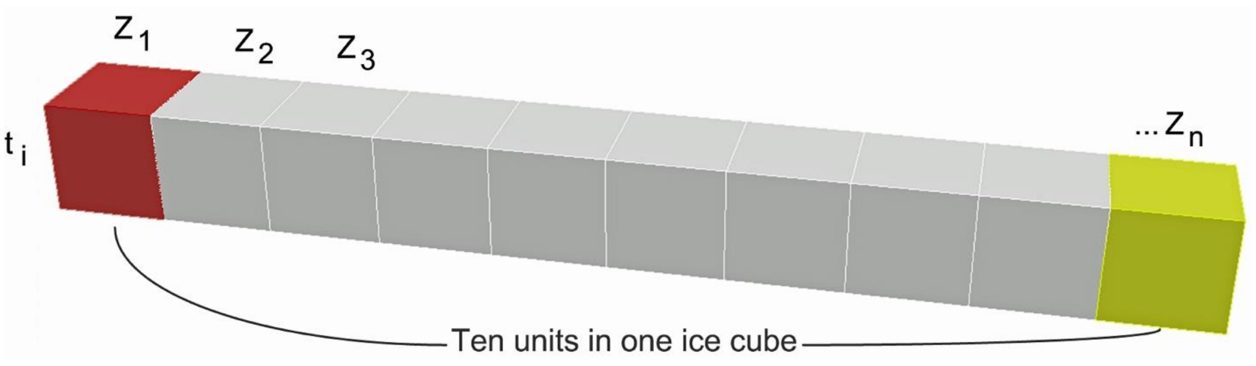


Figure 4. Schematic of discretizing one ice cube.

Using an array of parameters for physical properties, convective heat transfer, solar radiation, and ambient temperature (as specified, for example, in

Table 2 to Table 4), we may calculate the heat transfer coefficient (h_{amb}), as follows:

$$h_{amb} = \frac{Nu \cdot k_{amb}}{l} = 59616\left(J / \left(m^2 \cdot K \cdot month\right)\right) \qquad (21)$$

$$T_{amb} = 273.15 - t_{amb} \qquad (22)$$

$$Q_{\mathrm{cov}} = h_{amb}\left(T_{amb} - T\left(z_1, t\right)\right) \qquad (23)$$

Table 2. Physical property parameters.

name	ρ_{water}	$C_{p,water}$	k_{water}
unit	(kg/m^3)	(J/kg/K)	(J/m/K/month)
value	997	4179	1.5889e^6
name	ρ_{ice}	$C_{p,ice}$	k_{ice}
unit	(kg/m^3)	(J/kg/K)	(J/m/K/month)
value	800	1919	6.2986e^6

Table 3. Parameters for convective heat transfer.

name	u	υ	Pr	k_{amb}	l
unit	m/s	m^2/s	-	(J/m/K/month)	m
value	3	0.913	0.7	57024	1

Table 4. Monthly average ambient temperature.

Jan	Feb	Mar	Apr	May	Jun
-2.93	-4.77	-12.62	-21.01	-20.48	-19.96
Jul	Aug	Sep	Oct	Nov	Dec
-21.04	-30.07	-21.22	-11.14	-6.34	-2.13

According to Equations (13) and (14), the value of ϑ_0 can be calculated as:

$$Q_{rad} = I_0 \sin\theta_0 \alpha = 1.8728e^8 \left(J/\left(m^2 \cdot month\right)\right) \quad (24)$$

And, according to Equations (11) and (12), the boundary conditions can be discretized as follows:

$$k\frac{T_{z_1}^t - T_{amb}^t}{dz} = Q_{rad} + h_{amb}\left(T_{amb}^t - T_{z_1}^t\right) \quad (25)$$

$$T_{z_1}^t = T_{amb}^t + \frac{Q_{rad}}{\left(k/dz + h_{amb}\right)} \quad (26)$$

$$T_{z_{10}}^t = T_{z_9}^t \quad (27)$$

For example, in January, *t* equals 1, *dz* is the spatial interval, and *dz* = 0.1 m in this model, *dt* is the time interval, and *dt* =1 month. So, the enthalpy can be calculated as in

Table 5.

Table 5. Enthalpy rate.

January		January		February		February	
Position	Enthalpy rate	Position	Enthalpy rate	Position	Enthalpy rate	Position	Enthalpy rate
0	1	3.7	0.02	1.1	1	4.8	0.03
0.1	1	3.8	0.02	1.2	1	4.9	0.03
0.2	1	3.9	0.02	1.3	1	5	0.02
0.3	1	4	0.02	1.4	1	5.1	0.02
0.4	1	4.1	0.01	1.5	1	5.2	0.02
0.5	1	4.2	0.01	1.6	1	5.3	0.02
0.6	1	4.3	0.01	1.7	1	5.4	0.02
0.7	1	4.4	0.01	1.8	1	5.5	0.02
0.8	1	4.5	0.01	1.9	1	5.6	0.02
0.9	1	4.6	0.01	2	1	5.7	0.02
1	1	4.7	0.01	2.1	1	5.8	0.02
1.1	0.07	4.8	0.01	2.2	1	5.9	0.02
1.2	0.07	4.9	0.01	2.3	1	6	0.01
1.3	0.07	5	0.01	2.4	0.07	6.1	0.01
1.4	0.07	5.1	0.01	2.5	0.07	6.2	0.01
1.5	0.06	5.2	0.01	2.6	0.07	6.3	0.01
1.6	0.06	5.3	0.01	2.7	0.06	6.4	0.01
1.7	0.06	5.4	0.01	2.8	0.06	6.5	0.01
1.8	0.05	5.5	0.01	2.9	0.06	6.6	0.01
1.9	0.05	5.6	0.01	3	0.06	6.7	0.01
2	0.05	5.7	0	3.1	0.06	6.8	0.01
2.1	0.05	5.8	0	3.2	0.05	6.9	0.01
2.2	0.04	5.9	0	3.3	0.05	7	0.01
2.3	0.04	6	0	3.4	0.05	7.1	0.01

2.4	0.04	6.1	0	3.5	0.05	7.2	0.01
2.5	0.04	6.2	0	3.6	0.05	7.3	0.01
2.6	0.04	6.3	0	3.7	0.04	7.4	0.01
2.7	0.03	6.4	0	3.8	0.04	7.5	0.01
2.8	0.03	6.5	0	3.9	0.04	7.6	0.01
2.9	0.03	6.6	0	4	0.04	7.7	0.01
3	0.03	6.7	0	4.1	0.04	7.8	0.01
3.1	0.03	6.8	0	4.2	0.04	7.9	0.01
3.2	0.03	6.9	0	4.3	0.03	8	0.01
3.3	0.02	7	0	4.4	0.03	8.1	0.01
3.4	0.02	7.1	0	4.5	0.03	8.2	0.01
3.5	0.02	7.2	0	4.6	0.03	8.3	0
3.6	0.02	7.3	0	4.7	0.03	8.4	0

In

Table 5, we consider the physical model in calculating ice melting in January and February. The position term in

Table 5 points to an ID that represents a specific location in the voxel. By dividing one ice voxel into a bundle of ten equal-volume pieces (Figure 4), we can reveal *details* of the model, such as distance of heat transfer and the specific melting place. The maximum value of enthalpy rate is one, which indicates the energy that a single ice voxel needs to absorb in transforming from ice to water in the mixture. For example, in January, the enthalpy rate of locations with ID $\in \{0,1\}$ changes in value to one, which means that those voxels absorb an amount of heat equal to, or over, one unit. In these circumstances, those ice voxels will melt to water and flow. The enthalpy rate can also be considered in terms of liquid phase fractions, which are used to indicate which position of the ice has melted. For example, in Table 5, positions from 1.1 to 5.6 show potential effects of the residual energy. The ice located at those positions has yet to melt, but has been absorbing heat. When the next time step comes, those ice voxels can reach melting status by absorbing less than "one" energy rate. In another words, they melt faster than those ice cubes that have not absorbed any heat. As shown in Table 6, the heat transfer distance for January is 5.7 meters, while for February, it is 7.2 meters.

Table 6. Heat transfer distances.

Month	Jan	Feb	Mar	Apr	May	Jun
Distance	5.7	7.2	8	9.7	8.6	8.8
Month	Jul	Aug	Sep	Oct	Nov	Dec

Distance	9	9.1	9.3	9.4	9.3	9.4

4.2 A corresponding voxel CA model

On the basis of the *physical model*, as described above, we can design a *computational* model in extensible form as a CA, which we will task with animating ice melting dynamics according to the processes outlined in the physical model. Within the lattice, each voxel is represented dynamically, as an individual and autonomous CA. For the simulations discussed in this paper, we specified a synthetic ice sheet as a box lattice, with relatively massive dimensions: 7.8 km × 3.6 km × 500 m, corresponding to 14.04 billion individual voxels. We consider the entire ice sheet as being composed of same-size voxels, each represented at 1 m^3 in volume, which is consistent with the explanatory concepts of the physical model. An example of the significant level of detail that this affords is illustrated in Figure 5. While states are allowed to transition through each voxel in the lattice, we do not allow for cells in the CA ice sheet to move their locations and we do not allow cells to be deleted (although they may change state). We consider two *types* of CA: ice and air. At the initial state of each time slice in simulation, the air above the ice sheet is specified as air voxels. Ice voxels that transition far enough through melting into air (evaporation or flow) will transition to air cells during each time slice. To fit the enthalpy change rate in the physical model, we applied a continuous CA model, using cell state values to represent enthalpy rate. All ice voxels were initialized as one enthalpy rate. By absorbing heat, the rate will reduce until a value of zero, at which point those cells will transition from being ice voxels to being air voxels in the lattice.

The voluminous nature of the CA allows us to characterize sub-volumes of the modeled ice sheet. For example, as shown in the top-down view inset of Figure 5, we may specify columns of the ice sheet and the air above the sheet, together, at resolutions of $^1/_{14,040,000,000}$th *individual* voxels within the lattice. This resolution also allow us to specify the surface conditions of the sheet with high-detail rough and uneven topography, with resolution commensurate to 1 m^3 on the ground. We use synthetic start conditions in the examples that we will show in this paper, but seed conditions could alternatively be specified with real elevation and ice thickness data. The resolution that our CA model offers in simulation matches the best-available-resolution ice sheet topography data that we can expect from near-term remote sensing. Should finer resolutions be required, they could easily be accommodated by simply altering the specification of CA dimensions in the CA model.

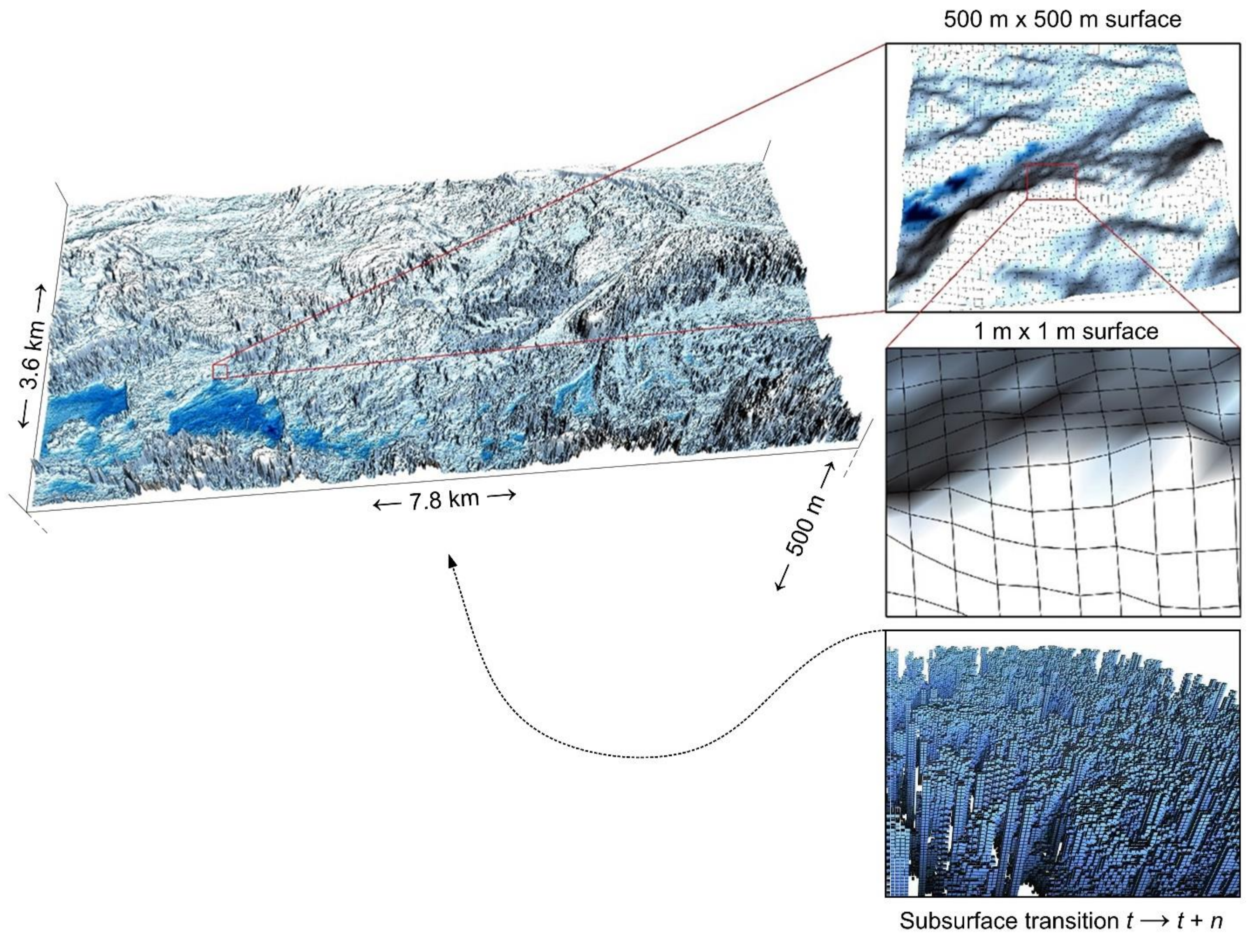


Figure 5. The voxel ice CA lattice implementation of an ice sheet.

Neighborhoods for state exchange are a critical component of CA. According to the ice sheet physical model, target cells are only affected by those cells on top of them in the lattice. However, the number of the cells on the top should vary from month to month as the ice mass changes, and as environmental conditions influence heat exchange. For example, in Table 6, which shows heat transfer distances, *n* equals 6 in January, but equals 8 in February (rounded up to an integer to fit CA units, e.g., spanning unit 1 m^3 ice voxels). These monthly variations can be accommodated in our CA model. Indeed, we may use parameters such as those shown in Table 6 to *vary transition rules temporally as a simulation unfolds*. For example in Table 6, cells in January should abide by the following transition rules:

- If a cell is located at C_1 (see Figure 6 for an illustration), this cell absorbed 0.56 (from 1 to 1.9) energy. So, the state of C_1 should be 0.44 at the end of a single time step.
- If a cell is located at C_3, the cell absorbed 0.23 (from 3 to 3.9) energy. So, the state of C_3 state should be 0.77.
- However, if cell is located at at C_6, it will not be affected by transition in its top cells at all.

In simulation, at each discrete time point that a step begins, the first task is to consult the month that this step "belongs to" by checking the superstep ID (the first January is calculated at superstep

1, but this could be modified to fit other real situations.) After this, the corresponding per-month transition rules are applied. By counting the cell numbers at the top of target cell, the simulator will know how the state of this target cell changes.

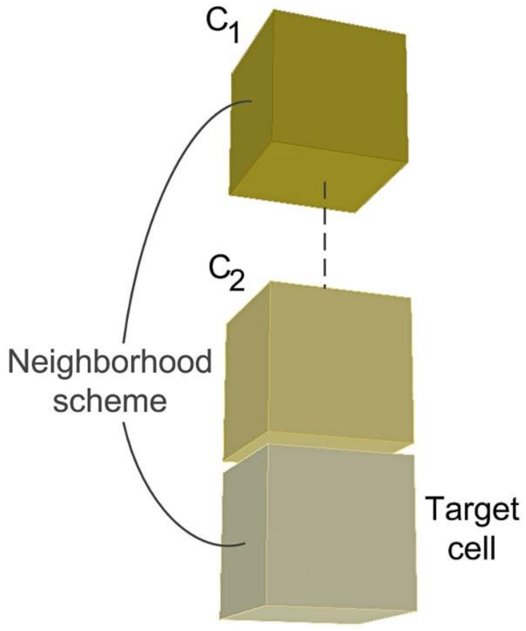


Figure 6. Ice melting CA neighbourhood scheme.

5 Experiments and results

We ran experiments with the CA-based computational representation of the physical model, for a lattice of over 14 billion voxels (where each cell represents a cubic meter as mentioned) (Figure 5). For the simulations that we discuss in this paper, we parameterized the simulation using meteorological seed states derived from Larsen Ice Sheet in Antarctica (American Museum of Natural History *et al.* 2002). The simulation was iterated over a synthetic time in which each CA superstep represented the equivalent of one month of melting dynamics in real-world time. In total, a simulated period of 360 months (thirty years) was run in this simulation, allowing us to peer into the potential future of the ice sheet over the next three (simulated) decades.

We show the implementation of the CA simulation in large-scale graph processing and graph-based computing on *Giraph* with *default (hash-) partitioning* (Lee *et al.* 2012), shown in pseudo-code in Algorithm 1. When the simulation task is submitted to the computing cluster, the simulation user may define the number of supersteps to represent a desired time period, as well as the corresponding relations between supersteps and real simulation time. Initialization is performed before superstep 0. As we will discuss, this step can be relatively straightforward in easing specification and parameterization of the model. As per the computational design of the CA model, different situations (different parameters, different transition rules, different neighborhoods) will explain ice melting in different months. From superstep 1, the simulation will automatically apply the appropriate transition rules, depending on which month constitutes the current superstep.

Algorithm 1 *Ice simulation* on Giraph

```
    class IceSimulation
 1: MaxSuperstep ⟵ user defined simulated time period
 2: function compute(vertex, messages)
 3:   if getSuperstep() == 0  then
 4:     sendMessage to neighbors to detect top surface of ice
 5:     if getSuperstep() <= MaxSuperstep  then
 6:         case Month index of:
 7:         Jan: apply transition rules of January, setVertexValue()
 8:         Feb: apply transition rules of February, setVertexValue()
 9:         Mar: apply transition rules of March, setVertexValue()
10:         Apr: apply transition rules of April, setVertexValue()
            ⋮
11:         Dec: apply transition rules of December, setVertexValue()
10:     else
11:         voteToHalt()
12:     end if
13:   end if
```

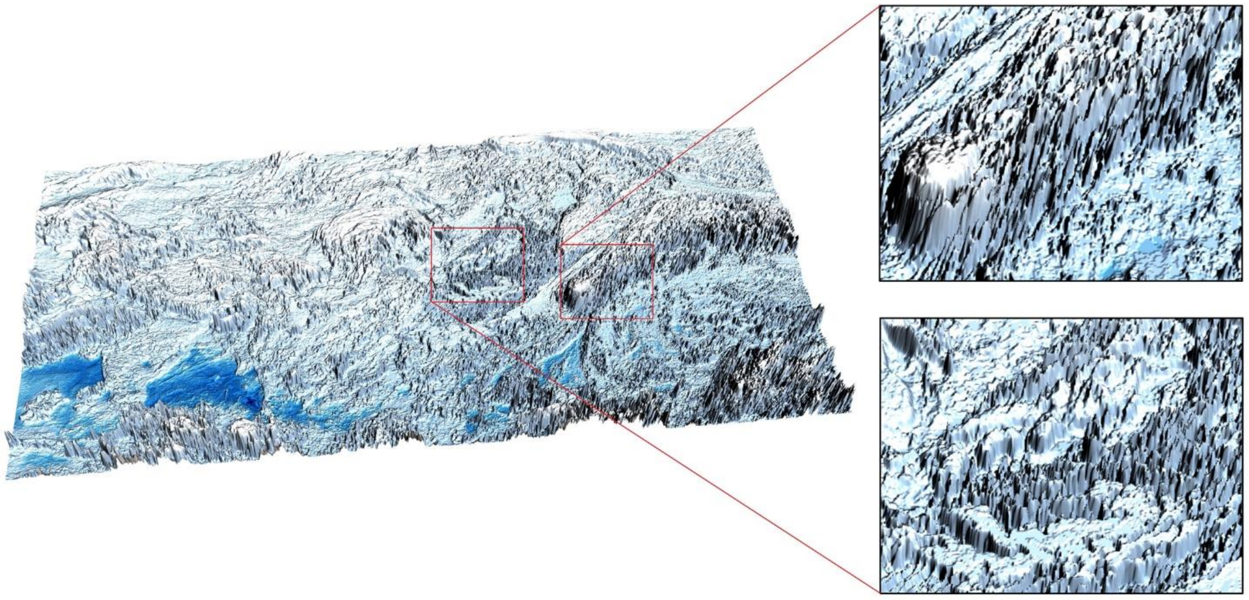
The pseudo code in Algorithm 2 represents the CA simulation with *customized partitioning*. The partitioning operation is performed before supersteps begin. According to the physical model design, there is no horizontal heat transfer between ice voxels. Hence, the neighbors of each cell in the lattice are only located on vertical positions. In other words, the whole CA model could be considered as a bundle of tetragonal prisms. Cells on each tetragonal prism can only evolve inside their own tetragonal prism, rather than influencing other cells in other tetragonal prisms. This is an important point to make, as it suggests some computational schemes that can be injected into the simulation to allow for computing efficiency in ways that ally to known physics. Specifically, because each tetragonal prism is independent, simulation users could perform partitioning for specific numbers of those tetragonal prisms, depending on the computing capacities of each node in any available computing cluster. For example, if there were a total of nine tetragonal prisms to be resolved in simulation, and three corresponding computing nodes, a user could arrange each node with three tetragonal prisms to achieve balanced computing performance.

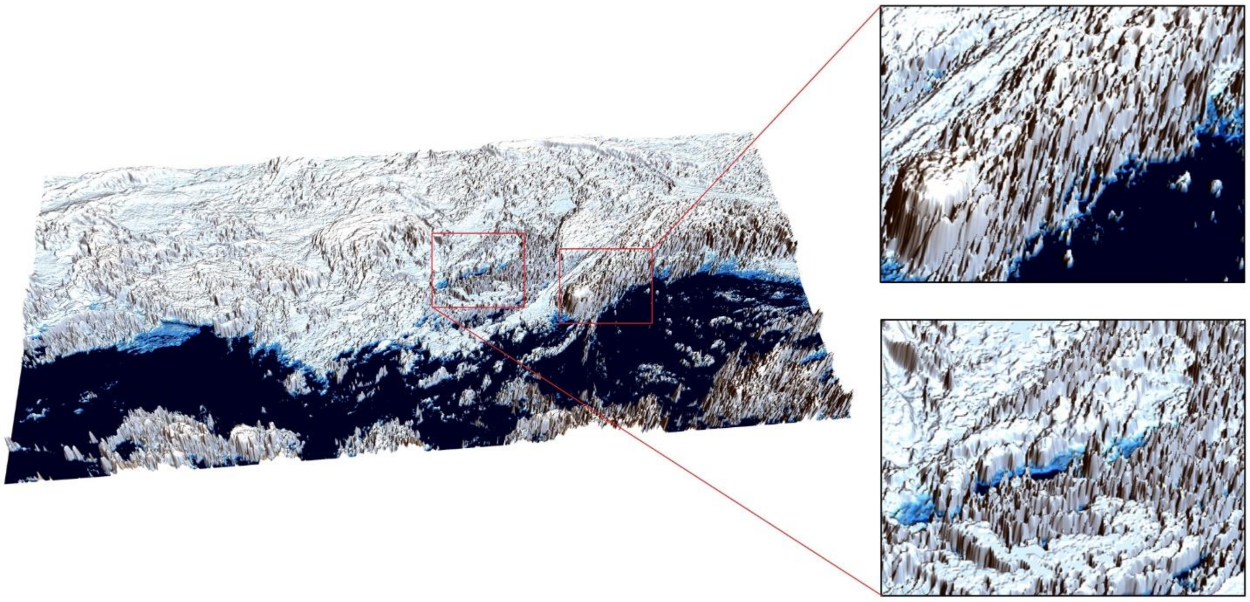
For ice-melting experiments, we deployed the CA simulation on a local *Hadoop* cluster, which can provide twenty computing nodes. A total 4Tb of memory can be used to support in-memory *Giraph* computing across the cluster. In the experiments reported in this paper, a total of 150 workers were used.

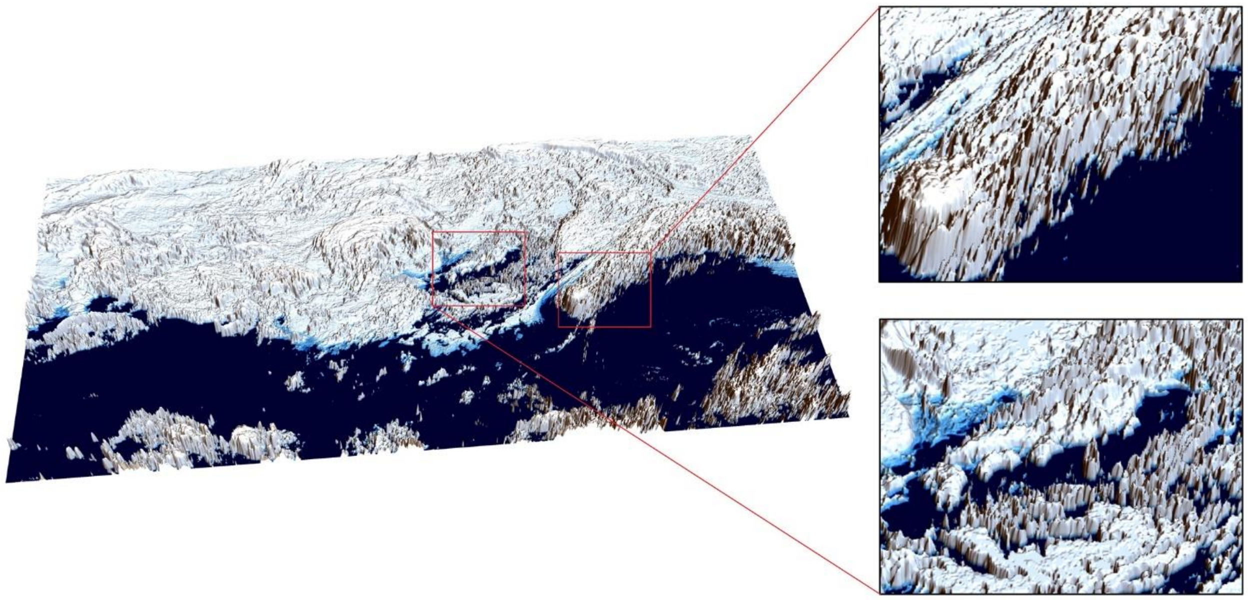
Algorithm 2 *Ice simulation* on Giraph with Customized Partitioning

```
     class IceSimulation
 1:   function getPartition(id, partitionCount)  ▷partitioning for specific number of tetragonal
                                                  prism in this case
 2:  function getWorker(partition)           ▷return worker
 3:  MaxSuperstep ⟵ user defined simulated time period
 4:  function compute (vertex, messages)
 5:   if getSuperstep() == 0  then
 6:     sendMessage to neighbors to detect top surface of ice
 7:     if getSuperstep() <= MaxSuperstep  then
 8:         case Month index of:
 9:         Jan: apply transition rules of January, setVertexValue()
10:         Feb: apply transition rules of February, setVertexValue()
11:         Mar: apply transition rules of March, setVertexValue()
12:         Apr: apply transition rules of April, setVertexValue()
            ⋮
13:         Dec: apply transition rules of December, setVertexValue()
14:    else
15:         voteToHalt()
16:    end if
17:  end if
```

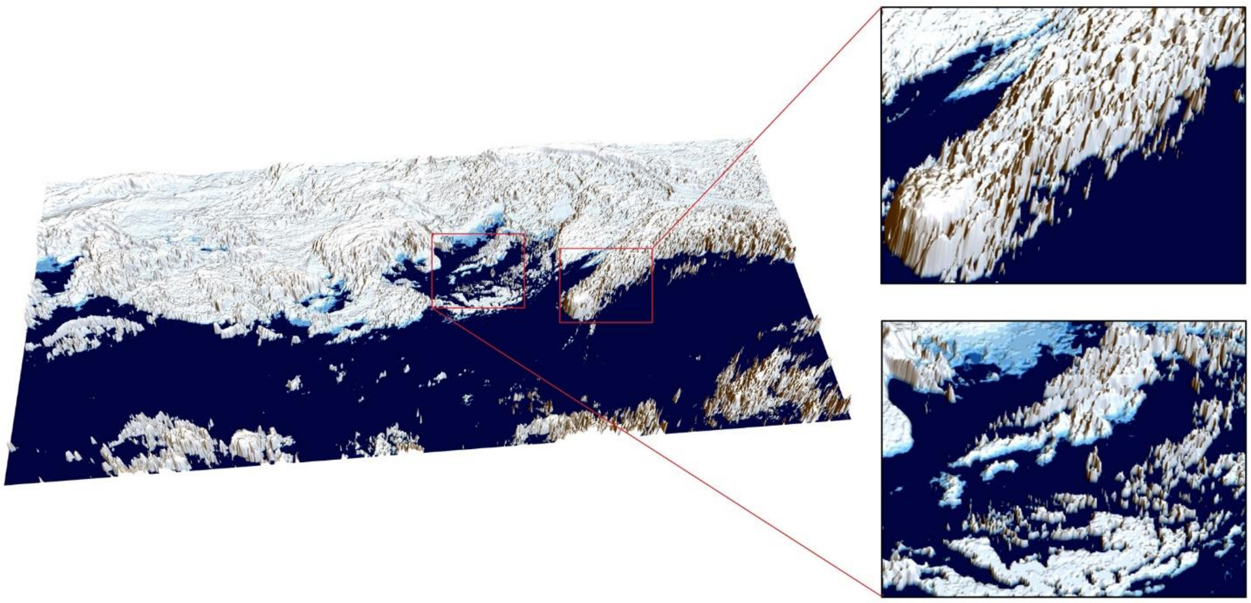
Snapshots of the synthetic ice sheet through the simulation run are illustrated visually in Figure 7, showing how the initial ice sheet progresses through melting dynamics as the simulation runs and as each of the ~14 billion ice and air voxels interact. The appearance of the ice sheet at the 10th, 15th , 20th , 25th and 30th simulated years is shown in Figure 7 (B to F) respectively. From the beginning of the 10th year, seawater (there is no salt water dynamics or ocean wave erosion factors involved in our model. Hence, those parts can also be considered as ice sheet ground) appeared significantly in the melting region on this ice sheet. During the 15th to 30th years, the melting region grew in size and depth. Seawater also appeared in the surrounding crevasse that the ice melting dynamics carved as the CA evolved.


A. Initial conditions

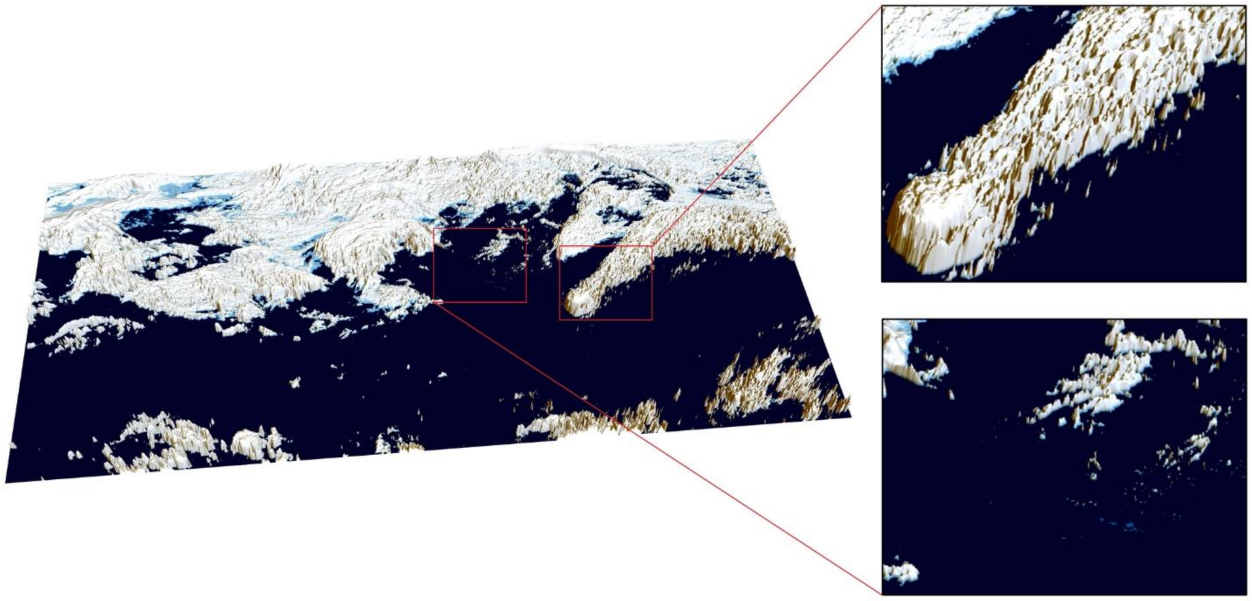

B. After 10 simulated years.



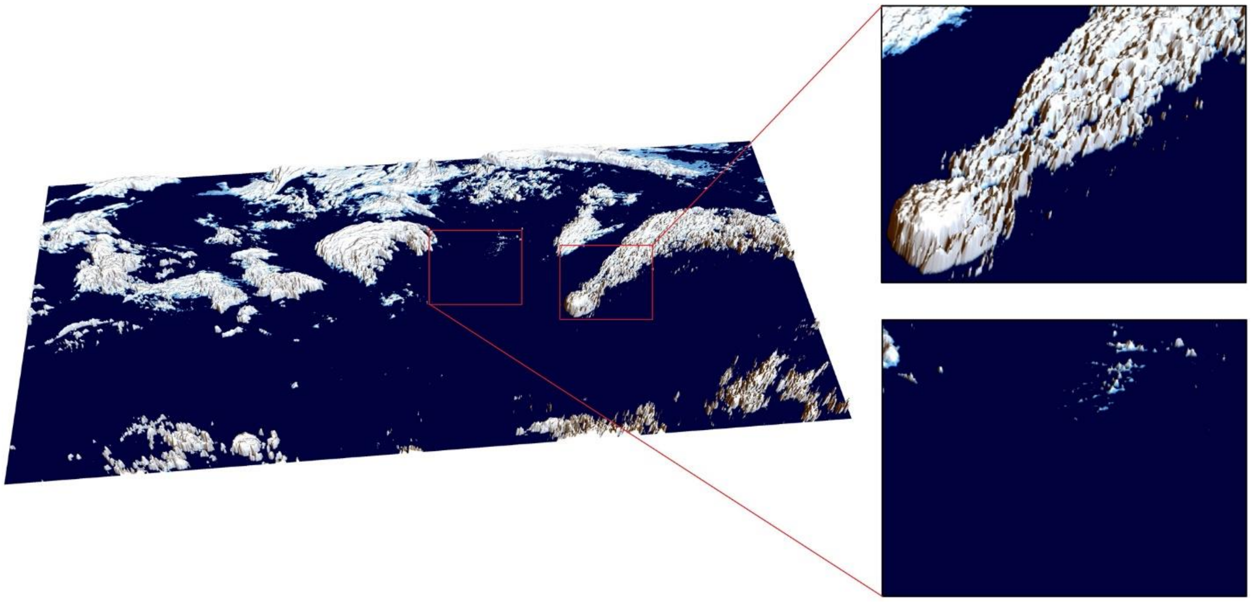
C. After 15 simulated years.



D. After 20 simulated years.



E. After 25 simulated years.



F. At 30 simulated years.

Figure 7. Ice Melting Simulation Rendering Results. (A) Initial state (B) appearance of the ice sheet at the 10th simulated year (C) appearance of the ice sheet at the 15th simulated year (D) appearance of the ice sheet at the 20th simulated year (E) appearance of the ice sheet at the 25th simulated year (F) appearance of the ice sheet at the 30th simulated year.

During simulation, we collected data on processing efficiency using (1) the customized partitioning algorithm (CPA) and (2) with a default hash approach (Figure 8). Using CPA, one single superstep of a simulation run can be processed in about 50s; with default hash-partitioning, processing takes roughly 700% more processing time to resolve a superstep. The ~14 billion-cell model takes roughly five hours to complete over a 30-year run in simulated time. (The default hash-partitioned approach experiment is considered only for performance comparison testing, and we terminated it at the twentieth superstep.)

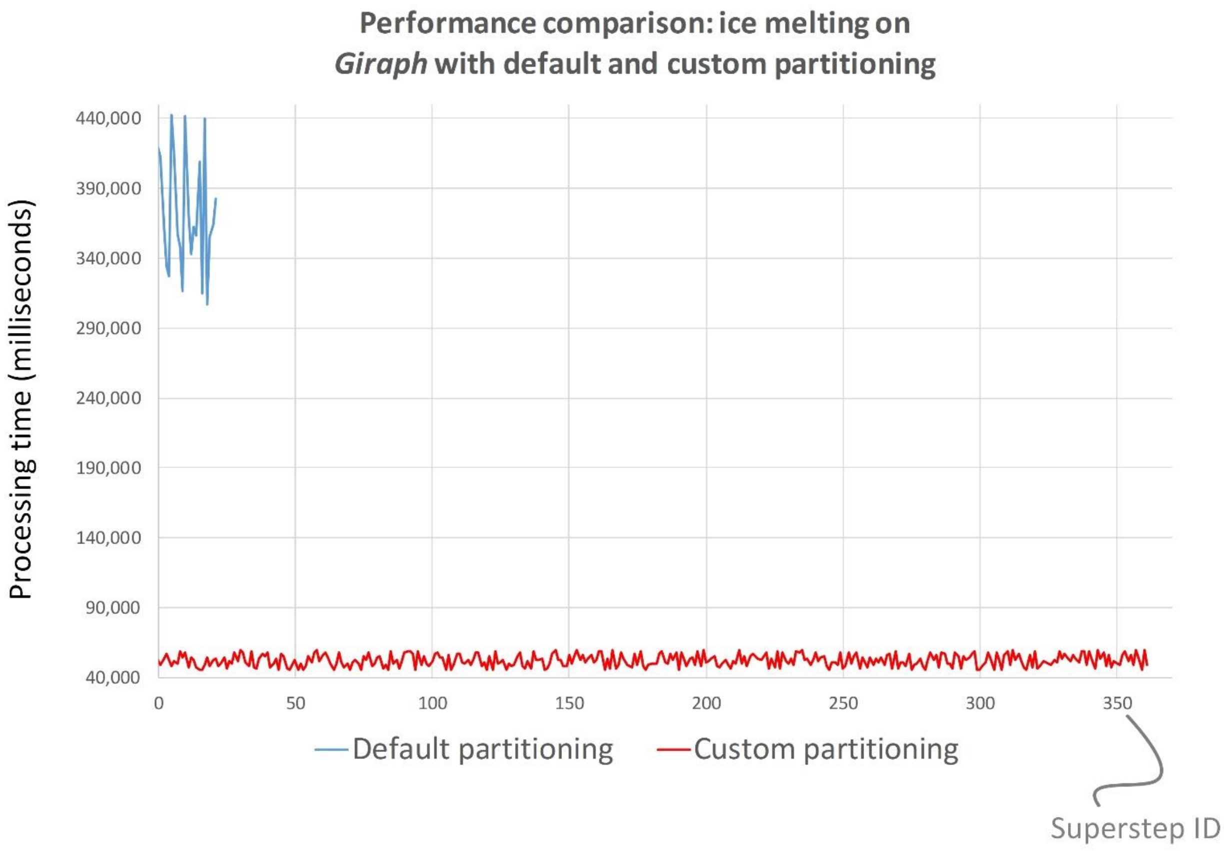


Figure 8. Comparison results of the ice melting simulation, run on a ~14 billion-cell CA lattice, with default hash approach and customized partitioning method.

6 Conclusions

Our intention, in this paper, has been to present novel computing frameworks that can perhaps better position simulation relative to the massive troves of highly detailed surface data that we know to be incoming from next-generation satellites and observation platforms that are tasked on Antarctica. Our approach, in eking out efficiencies to better connect "big data" with "big models" via "big computing" is technological. In particular, we showed how a physical model of ice melting dynamics (1) can be transformed into a massively-detailed cellular automata volume (computer) model, (2) how parameterization of the CA and the transition of its states can be both eased and accelerated with large-scale graph processing, (3) how graph-based computing can accelerate computing of the necessary state transition with varying CA typologies, dynamic neighborhood look-ups, and dynamic transition rules, and (4) how the graph-accelerated CA can be implemented as a simulation on cloud computing environment with tremendously fast resolution times and straightforward setup, and with extensibility to handle future data sizes. Our prototype implementations suggest one possible path for future computing, by which coupled simulation and computing schemes based around CA and graph-computing could provide researchers in applied contexts with more detail and processing power to better represent phenomena of interest to them, and with sufficient resources to begin to build models that can feed on ever-emerging big data resources that might guide them, especially for problems with considerations that span over high dimensions.

Our approach suggests several promising advantages. The first centers on relative simplicity of deployment and use. For example, in the simulations that we discussed in this paper, by simply modifying input files to a simulation run, an entire model can be transformed from 2D into 3D without modifying the structure of the computing functions; only changes to a small number of parameters are required in some cases. The second advantage concerns efficiency. We think we can eke out significantly more efficiency by employing *Giraph*-based large CA models rather than can be considered by running CA on traditional frameworks, and this efficiency can be achieved on both local clusters and cloud platforms. The third advantage is centered on extensibility and scalability. By accessing cloud platforms, researchers can easily gain enough computing resources to support model processing on demand. This, in turn, allows users to "spin-up" resources for a relatively limitless resolution and size CA volume. One could imagine simply adjusting the resolution of a graph-accelerated CA in parameterization to match incoming data feeds with resolution increases of an order of magnitude detail over previous capabilities.

The simulations discussed in this paper are obviously just a first step in these directions and significant improvements could be made to the models and simulation runs that we have shown. Here, they serve to prove the concept and to encourage others in the community to build on the foundation that we have introduced. Obvious extensions of our approach could include the use of real ice thickness data and more detailed meteorological data. An improved physical model of broader processes governing ice dynamics is perhaps sorely needed if the demonstrations that we have shown here are to be considered operational relative to any serious experimental work; our model, in particular, focused rather exclusively on ice melting at the expense of other related processes. Other, computational, improvements will also be the target of future work. In particular, more work remains to be uncovered in the computing performance of the system, particularly in optimizing the input and output functions of the *Giraph* scheme, which would help to compress the size of data and reduce the overhead of large data partitioning, passing, and storage. At the same time, many computing clusters in real life may be made of multiple computing nodes with different performance. Allocating different sub-tasks to those nodes according to the real performance of those nodes may become another effective method to improve the processing performance.

Acknowledgements

The authors gratefully acknowledge the support of NASA Goddard Space Flight Center; Hydrospheric and Biospheric Sciences; Cryospheric Sciences (Grant NNX13AP97G). The authors are grateful for the assistance of Prof. Ge Zhang in the Department of Thermal Engineering at the University of Science and Technology, Beijing, for his advice in building the physical model behind this work. We are also grateful to Prof. Jimmy Lin in the David R. Cheriton School of Computer Science at the University of Waterloo for his advice in building the *Giraph* components of the work.

References

Abramowitz, M., and I. A. Stegun. 1972. *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables*. Mineola, NY: Dover Publications.

Amazon 2018. Amazon Web Services. Last accessed March 8 2018. Available from https://aws.amazon.com.

American Museum of Natural History, Rice University, and Education Development Company Inc. 2002. Monthly average weather reports for Laresen Ice from 1/1/2000 to 12/31/2000. American Museum of Natural History, Last accessed January 1 2017. Available from http://www.amnh.org/education/resources/rfl/web/antarctica/weather/8926/avg.html

Asanovic, K., R. Bodik, B. Catanzaro, J. Gebis, P. Husbands, K. Keutzer, D. Patterson, W. Plishker, J. Shalf, S. Williams, and K. Yelik. 2006. *The Landscape of Parallel Computing Research: A View from Berkeley*. Berkeley, CA: Department of Electrical Engineering and Computer Science, University of California, Berkeley. Technical Report UCB/EECS-2006-183.

Bahr, D. B., and J. B. Rundle. 1995. Theory of lattice Boltzmann simulations of glacier flow. *Journal of Glaciology* 41 (139):634-640.

Bak, P., C. Tang, and K. Wiesenfeld. 1988. Self-organized criticality. *Physical Review A* 38 (1):364.

Baxter, G. W., and R. P. Behringer. 1990. Cellular automata models of granular flow. *Physical Review A* 42 (2):1017-1020.

Benn, D. I., C. R. Warren, and R. H. Mottram. 2007. Calving processes and the dynamics of calving glaciers. *Earth-Science Reviews* 82 (3):143-179.

Blankenship, D. D., S. D. Kempf, D. A. Young, T. G. Richter, D. M. Schroeder, G. Ng, J. S. Greenbaum, T. van Ommen, R. C. Warner, J. L. Roberts, N. W. Young, E. Lemeur, and M. J. Siegert. 2012, updated 2017. IceBridge HiCARS 2 L2 Geolocated Ice Thickness, Version 1. [Indicate subset used]. Boulder, Colorado USA. NASA National Snow and Ice Data Center Distributed Active Archive Center. doi: https://doi.org/10.5067/9EBR2T0VXUDG. [Date Accessed].

Bondy, J. A., and U. S. R. Murty. 1976. *Graph Theory with Applications*. New York: Macmillan.

Brown, J., J. Harper, and N. Humphrey. 2010. Cirque glacier sensitivity to 21st century warming: Sperry Glacier, Rocky Mountains, USA. *Global and Planetary Change* 74 (2):91-98.

Burrough, P. 2001. GIS and geostatistics: essential partners for spatial analysis. *Environmental and ecological statistics* 8 (4):361-377.

Ching, A. 2013. Scaling Apache Giraph to a trillion edges. Facebook, Last accessed August 14, 2013. Available from https://www.facebook.com/notes/facebook-engineering/scaling-apache-giraph-to-a-trillion-edges/10151617006153920?_fb_noscript=1.

Cohen, M., K. Hurley, and P. Newson. 2015. *Google Compute Engine: Manage Secure and Scalable Cloud Computing*. Sebastapool, CA: O'Reilly.

D'Ambrosio, D., S. Di Gregorio, and G. Iovine. 2003. Simulating debris flows through a hexagonal Cellular Automata model: Sciddica S3-hex. *Natural Hazards and Earth System Sciences* 3:545-559.

Ding, B., K. Yang, W. Yang, X. He, Y. Chen, X. Guo, L. Wang, H. Wu, and T. Yao. 2017. Development of a Water and Enthalpy Budget ‐ based Glacier mass balance Model (WEB ‐ GM) and its preliminary validation. *Water Resources Research* 53 (4):3146-3178.

Doake, C. S. M., H. F. J. Corr, H. Rott, P. Skvarca, and N. W. Young. 1998. Breakup and conditions for stability of the northern Larsen Ice Shelf, Antarctica. *Nature* 391 (6669):778-780.

Faria, S. H., D. Ktitarev, and K. Hutter. 2002. Modelling evolution of anisotropy in fabric and texture of polar ice. *Annals of Glaciology* 35 (1):545-551.

Fonstad, M. A. 2006. Cellular automata as analysis and synthesis engines at the geomorphology–ecology interface. *Geomorphology* 77 (3–4):217-234.

Fretwell, P., H. D. Pritchard, D. G. Vaughan, J. L. Bamber, N. E. Barrand, R. Bell, C. Bianchi, R. G. Bingham, D. D. Blankenship, G. Casassa, G. Catania, D. Callens, H. Conway, A. J. Cook, H. F. J. Corr, D. Damaske, V. Damm, F. Ferraccioli, R. Forsberg, S. Fujita, Y. Gim, P. Gogineni, J. A. Griggs, R. C. A. Hindmarsh, P. Holmlund, J. W. Holt, R. W. Jacobel, A. Jenkins, W. Jokat, T. Jordan, E. C. King, J. Kohler, W. Krabill, M. Riger-Kusk, K. A. Langley, G. Leitchenkov, C. Leuschen, B. P. Luyendyk, K. Matsuoka, J. Mouginot, F. O. Nitsche, Y. Nogi, O. A. Nost, S. V. Popov, E. Rignot, D. M. Rippin, A. Rivera, J. Roberts, N. Ross, M. J. Siegert, A. M. Smith, D. Steinhage, M. Studinger, B. Sun, B. K. Tinto, B. C. Welch, D. Wilson, D. A. Young, C. Xiangbin, and A. Zirizzotti. 2013. Bedmap2: improved ice bed, surface and thickness datasets for Antarctica. *The Cryosphere* 7 (1):375-393.

Frisch, U., B. Hasslacher, and Y. Pomeau. 1986. Lattice-gas automata for the Navier-Stokes equation. *Physical Review Letters* 56 (14):1505-1508.

Gardner, M. 1970. The fantastic combinations of John Conway's new solitaire game "Life". *Scientific American* 223 (4):120-123.

Giles, K. A., S. W. Laxon, and A. P. Worby. 2008. Antarctic sea ice elevation from satellite radar altimetry. *Geophysical Research Letters* 35 (3).

Gobron, Stéphane, Arzu Çöltekin, Hervé Bonafos, and Daniel Thalmann. "GPGPU computation and visualization of three-dimensional cellular automata." The Visual Computer 27, no. 1 (2011): 67-81.

Google 2018. Compute Engine: Scalbale, High-Performance Virtual Machines. Last accessed March 8 2018. Available from https://cloud.google.com/compute/.

Gropp, W. D., and E. Lusk. 2005. Using MPI-2: a problem-based approach. In *Proceedings of the 14th European PVM/MPI User's Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface, Sorrento, Italy, September 18-21, 2005*, eds. F. Cappello, T. Herault and J. Dongarra, 12. Berlin: Springer.

Hock, R. 2005. Glacier melt: a review of processes and their modelling. *Progress in physical geography* 29 (3):362-391.

Hunt, P., M. Konar, F. P. Junqueira, and B. Reed. 2010. ZooKeeper: wait-free coordination for Internet-scale systems. In *Proceedings of the USENIX Annual Technical Conference, Boston, MA, June 22-25, 2010*, eds. P. Barham and T. Roscoe, 1-14. Washington D.C.: USENIX.

Jjumba, A., and S. Dragićević. 2016a. A development of spatiotemporal queries to analyze the simulation outcomes from a voxel automata model. *Earth Science Informatics* 9 (3):343-353.

Jjumba, A., and S. Dragićević. 2016b. Towards a voxel-based geographic automata for the simulation of geospatial processes. *ISPRS Journal of Photogrammetry and Remote Sensing* 117:206-216.

Klotz, I. M., and R. M. Rosenberg. 2008. *Chemical Thermodynamics: Basic Concepts and Methods*. New York: John Wiley and Sons.

Kronholm, K., and K. W. Birkeland. 2004. Relating spatial variability to snow stability using a cellular automaton model initialized with field data. *Geophysical Research Letters* 32:L19504.

Kronholm, K., and K. W. Birkeland. 2005. Integrating spatial patterns into a snow avalanche cellular automata model. *Geophysical Research Letters* 32 (19):L19504.1-L19504.4.

Ktitarev, D., G. Gödert, and K. Hutter. 2002. Cellular automaton model for recrystallization, fabric, and texture development in polar ice. *Journal of Geophysical Research: Solid Earth* 107 (B8):EPM 5-1-EPM 5-9.

Kurtz, N. T., and S. L. Farrell. 2011. Large ‐ scale surveys of snow depth on Arctic sea ice from Operation IceBridge. *Geophysical Research Letters* 38 (L20505):1-5.

Kwok, R., and D. A. Rothrock. 1999. Variability of Fram Strait ice flux and North Atlantic oscillation. *Journal of Geophysical Research: Oceans* 104 (C3):5177-5189.

Lax, P., and B. Wendroff. 1960. Systems of conservation laws. *Communications on Pure and Applied Mathematics* 13 (2):217-237.

Lee, K.-H., Y.-J. Lee, H. Choi, Y. D. Chung, and B. Moon. 2012. Parallel data processing with MapReduce: a survey. *AcM sIGMoD Record* 40 (4):11-20.

Machì, A., and F. Mignosi. 1993. Garden of Eden configurations for cellular automata on Cayley graphs of groups. *SIAM Journal on Discrete Mathematics* 6 (1):44-56.

Malewicz, G., M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. 2010. Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, Indianapolis, IN, June 6-10, 2010*, ed. A. Elmagarmid, 135-146. New York: Association for Computing Machinery.

Markus, Thorsten, Tom Neumann, Anthony Martino, Waleed Abdalati, Kelly Brunt, Beata Csatho, Sinead Farrell et al. "The Ice, Cloud, and land Elevation Satellite-2 (ICESat-2): Science requirements, concept, and implementation." Remote sensing of environment 190 (2017): 260-273.

Marques, R., B. Feijo, K. Breitman, T. Gomes, L. Ferracioli, and H. Lopes. 2013. A cloud computing based framework for general 2D and 3D cellular automata simulation. *Advances in Engineering Software* 65 (1):78-89.

McMillan, M., A. Shepherd, A. Sundal, K. Briggs, A. Muir, A. Ridout, A. Hogg, and D. Wingham. 2014. Increased ice losses from Antarctica detected by CryoSat ‐ 2. *Geophysical Research Letters* 41 (11):3899-3905.

Mercer, J. H. 1968. Antarctic ice and Sangamon Sea level. *International Association of Hydrological Sciences Symposium* 79 (1):217–225.

Microsoft 2018. Microsoft Azure. Last accessed March 8 2018. Available from https://azure.microsoft.com.

Mirjalili, S., S. S. Jain, and M. S. Dodd. 2017. Interface-capturing methods for two-phase flows: an overview and recent developments. *Center for Turbulence Research Annual Research Briefs* 1 (1):117-135.

National Aeronautics and Space Administration 2017a. IceBridge Mission Overview. Last accessed March 7 2018. Available from https://www.nasa.gov/mission_pages/icebridge/mission/index.html

National Aeronautics and Space Administration 2017b. IceBridge - Aircraft, Instruments, Satellites. Last accessed March 7 2018. Available from https://www.nasa.gov/mission_pages/icebridge/instruments/index.html

Oppenheimer, M. 1998. Global warming and the stability of the West Antarctic Ice Sheet. *Nature* 393 (6683):325-332.

Raabe, D. 2002. Cellular automata in materials science with particular reference to recrystallization simulation. *Annual Review of Materials Research* 32 (1):53-76.

Radenski, A. 2013. Using MapReduce streaming for distributed Life simulation on the cloud. In *ECAL: 12th European Conference on Artificial Life, September 2-6 2013, Taormina, Italy*, eds. P. Liò, O. Miglino, G. Nicosia, S. Nolfi and M. Pavone, 284-291. Taormina, Italy: European Conference on Artifical Life.

Reznor, M. M., T. Reznor, A. Ross, and R. Sheridan. 2013. *Ice Age*. Welcome Oblivion. Los Angeles, CA: The Null Corporation, Columbia Records.

Robin, G. d. Q. 1955. Ice movement and temperature distribution in glaciers and ice sheets. *Journal of Glaciology* 2 (18):523-532.

Rott, N. 1990. Note on the history of the Reynolds number. *Annual Review of Fluid Mechanics* 22 (1):1-12.

Schutz, B., H. Zwally, C. Shuman, D. Hancock, and J. DiMarzio. 2005. Overview of the ICESat mission. *Geophysical Research Letters* 32 (21):L21S01.

Semboloni, F. 2000. The growth of an urban cluster into a dynamic self-modifying spatial pattern. *Environment and Planning B: Planning & Design* 27 (4):549-564.

Shepherd, A., and D. Wingham. 2007. Recent sea-level contributions of the Antarctic and Greenland Ice Sheets. *Science* 315 (5818):1529-1532.

Shvachko, K., H. Kuang, S. Radia, and R. Chansler. 2010. The Hadoop Distributed File System. In *Proceedings of the 2010 IEEE 26th symposium on Mass storage systems and technologies (MSST), Incline Village, NV, May 3-7, 2010*, ed. E. Miller, 1-10. Washington D.C.: IEEE Computer Society.

Stokes, G. G. 1845. On the friction of fluids in motion, and the equilibrium and motion of elastic solids. *Transactions of the Cambridge Philosophical Society* 8 (1):287-305.

Thekaekara, M. 1973. Solar energy outside the Earth's atmosphere. *Solar Energy* 14 (2):109-127.

Torrens, P. M., and I. Benenson. 2005. Geographic Automata Systems. *International Journal of Geographical Information Science* 19 (4):385-412.

Valiant, L. G. 1990. A bridging model for parallel computation. *Communications of the ACM* 33 (8):103-111.

Van der Veen, C. J. 2013. *Fundamentals of Glacier Dynamics*. Boca Raton, FL: CRC Press.

Weigand, B. 2004. Linear Partial Differential Equations. In *Analytical Methods for Heat Transfer and Fluid Flow Problems*, ed. B. Weigand, 11-42. Berlin, Heidelberg: Springer Berlin Heidelberg.

White, F. M., and I. Corfield. 2006. *Viscous Fluid Flow. Volume 3*. New York: McGraw-Hill.

Wilder, B. 2012. Cloud Architecture Patterns: Using Microsoft Azure. Sebastopol, CA: O'Reilly.

Wingham, D., C. Francis, S. Baker, C. Bouzinac, D. Brockley, R. Cullen, P. de Chateau-Thierry, S. Laxon, U. Mallow, and C. Mavrocordatos. 2006. CryoSat: A mission to determine the fluctuations in Earth’s land and marine ice fields. *Advances in Space Research* 37 (4):841-871.

Wittig, A., and M. Wittig. 2015. *Amazon Web Servies in Action*. Shelter Island, NY: Hanning.

Wolfram, S. 1984a. Cellular automata as models of complexity. *Nature* 311 (5985):419.

Wolfram, S. 1984b. Universality and complexity in cellular automata. *Physica D* 10:1-35.

Yunus, A. C. 2003. *Heat Transfer: A Practical Approach*. New York: MacGraw Hill.

Zhang, R., C. Ke, H. Xie, and S. Bo. 2012. Surface albedo measurements over sea ice in the Arctic Ocean during summer 2010. *Chinese Journal of Polar Research* 24 (3):299-306.